FotoScript

Matthew Raibert
mjr2101@columbia.edu

Norman Yung
ny2009@columbia.edu

James Kenneth Mooney
jkm2017@columbia.edu

Randall Q Li
rql1@columbia.edu

December 22, 2004

Contents

Chapter 1

Introduction: The White Paper

This chapter introduces the reader to the concept of Fotoscript. It also attempts to provide motivation. This section was written before Fotoscript had been implemented, and so, many of our design decisions and compromises have cause the details to be altered. The main ideas of Fotoscript remain the same.

1.1 Introduction

There are many tools for creating and editing images. Almost none of those tools allows the user to write scripts. Most of them rely on a graphical interface that presents to the user a virtual canvas; she is then expected to pull down some menus and make some gestures until the result seems to look like what she expected. Whereas this is the familiar way to interact with images it is error prone, inexact and not particularly powerful. In this paper we propose to create a simple, robust, relatively familiar language for scripting image manipulations on a computer.

1.2 Why Not GUI?

It may appear that the logical way to interact with an image is to use a graphical interface, perhaps with a mouse approximating the way a painter uses her brush. This comes from the paradigm of computing which dictates that the way we interact with computers ought to simulate traditional tasks—indeed, what could be more traditional than a painter with a brush. This kind of interface is certainly familiar and intuitive but it puts a number of limitations on what the user can reasonably do. We get the most significant advantage from using a computer by using it for those tasks that we humans are not particularly good at. So by limiting the user to using gestures and estimation—which we can do just fine without a computer—GUI oriented programs necessarily limit the users ability to achieve clear, unambiguous and consistent results. As an alternative, we propose to create a language for the user to specify what she has in mind in a clear and simple way that allows for precision and repetition.

1.3 What Software Already Exists?

One example of a language for scripting image manipulations is the old LOGO family of languages. These languages allow the user to direct a “turtle” around a canvas and it leaves a trail as it goes. It can detect the color at its current position and use user programmed logic to decide how to react. We will probably take some ideas from it but LOGO was designed primarily to teach children how to program. It focuses on visualizing common programming tasks and is not very powerful as an image editor.

Most image manipulation programs such as MS Paint and Adobe Photoshop focus on the graphical interface. We will be getting ideas for functionality for our language primarily from these programs. Because these programs are familiar to many users, we will try to base our function names on naming conventions used by such programs.

1.4 What Functionality Will Fotoscript Include?

There should be a set of basic operations such as resize, stretch, rotate, brightness, contrast etc. These are good cantidates for operators, for example division ’/’ and multiplication ’*’ might signify adjusting the image size.

Then a second set of operations could be like LOGO: the program maintains a ’sprite’ which consists of a point of focus and a heading. The user directs the sprite to move and to draw or perform other operations on the small portion of the image at the point of focus. The user could have a movements in mind and program them specifically, or he could program a slightly smarter sprite that would find places with some set of characteristics and perform its operations at those positions.

A third set of operations could include what are called “filters” in Photoshop. Some examples of these filters might include the ability to overlay one image on another, the ability to change the color of all pixels in a certain color range, or the ability to cause the image to conform to a user defined histogram. The user should also be able to design his own filters by combining the simpler filters and perhaps even using the sprite.

1.5 What Will Fotoscript Look Like?

Fotoscript will be a concise scripting language with an image object as a basic type. The user will be able to designate any section of an image as itself an image object and perform any operation that can be performed on the whole.

There will be about half a dozen operators for the most basic operations; some possible examples (a and b are both variables of the image type):

- a / 2; //resize the image by a half
- a @ 90; //rotate the image counterclockwise by 90 degrees
- a + b; //concatenate two images vertically
- a — b; //concatenate two images horizontally

1.6 How Will Fotoscript Be Implemented?

Our focus will be on designing a regular language that simplifies common image manipulation tasks. To that end we have decided to use existing libraries to perform many of those tasks rather than implement the algorithms ourselves. We will write our compiler in C and use the GIMP library for implementations of image manipulation tasks. As an alternate implementation style we might write our compiler in Java and output C code that has calls to the GIMP API. It could then be translated into machine language by the C compiler.

1.7 Summary

Fotoscript will increase the productivity of graphic designers by allowing them to better leverage the power of computers. It will certainly not seek to replace the canvas and brush approach to image manipulation, but to complement it. Fotoscript's concise syntax will simplify image manipulation, even when working with a hundred images at once.

Chapter 2

A Tutorial

This chapter will introduce the would-be scripter to Fotoscript, it is not a complete description of the language; instead it introduces each of the key concepts of Fotoscript with one example.

2.1 Introduction

Fotoscript is a simple scripting language for editing images. The scripter must first open one or more images, then she must perform some operations, probably iterating through each opened image, and then she may save the result to disk.

2.2 Opening An Image For Editing

The scripter must first open one or more images for editing. This is usually done at the beginning of a script and is achieved with the `open` function. For example the command:

```
open ‘‘foo.jpg’’
```

opens the image in the local directory called “foo.jpg”. The scripter can also open a directory with similar syntax. The command:

```
open ‘‘bar’’
```

opens all the images in a local subdirectory called “bar”.

Note:

When more than one image is opened at once, their order in the internal data structure is arbitrary and unknown to the scripter. Addressing more than one image is dealt with by `foreach` described below.

2.3 Performing Operations

Operations can be either control-flow such as if-statements or functions.

2.3.1 Functions

The scripter might like to resize the image (for the moment I assume that only one image is opened at once). To achieve this the scripter can use the `resize` function:

```
resize 1024 768}
```

The image is specified implicitly so this command will resize the image to a size of 1024x768 pixels.

2.3.2 Control-Flow

The scripter might wish to perform a test before performing a function. For example, the scripter might want only to resize an image if it is wider than 1024 pixels. This is achieved using an if statment as follows:

```
if (width > 1024) then {
  resize 1024 768
}
```

2.4 The Foreach Loop

Fotoscripts power lies in its ability to operate on multiple images. Imagine that the scripter has opened three images. The way to perform operations on all of them is with `foreach`.

2.4.1 My First Foreach Loop

Given the code constructed above, we can imagine wanting to resize each of the loaded images. This is achieved by placing the above code into a `foreach` loop as follows:

```
foreach () {
  resize 1024 768
}
```

2.4.2 Tests In The Foreach Loop

Combining the ideas from above, we see that the scripter could test each image before resizing it. Hence we can write:

```
foreach () {
  if (width > 1024) then {
    resize 1024 768
  }
}
```

Which resizes each image if it is wider than 1024 pixels.

Built In Tests In Foreach

The foreach loop also allows the user to create an equivalent construction as follows:

```
foreach (width > 1024) {
  resize 1024 768
}
```

Which exhibits identical behavior, but is a bit neater.

Note:

Using a `foreach` loop before opening any images is legal code but will have no result.

2.5 Saving The Results

Finally, the scripter probably wants not to have done all this in vain. So she needs to save her creations. This is achieved using `save` or `saveall`. These commands are simple, `save` simply saves the current image to disk and `saveall` saves them all to disk at once. Putting a `save` command at the end of a foreach loop or a `saveall` command just after then end of a `foreach` loop will both cause all of the images currently opened for editing to be written back out to disk.

Chapter 3

The Language Reference Manual

Here is a description of the various features of FotoScript and how to use them. This manual is based in part on The C Programming Language by Brian W. Kernighan and Dennis M. Ritchie:

3.1 Tokens

There are five classes of tokens in FotoScript: identifiers, keywords, literals, separators and numerical operators. Blanks, tabs, and comments are ignored except as they separate tokens.

3.1.1 Comments

As an experiment, we have chosen to implement comments in an unorthodox way. All upper-case letters will be ignored by FotoScript with the exception of those within string literals. A user will be able to write comments in all caps anywhere in the code. Because this is unusual, and possibly really annoying, it is subject to change.

Note:

It is important to remember that all punctuation is still considered active code, even when surrounded by upper-case letters. This is a potential pitfall.

3.1.2 Identifiers

Identifiers are made up of lower-case letters and digits. The first character must be a letter; upper-case letters are ignored by fotoscript and cannot, therefore, be used in identifiers.

3.1.3 Keywords

Fotoscript reserves several keywords:

```
concat depth format else foreach height if modified new open
save saveall set setdepth setformat setheight setmodified setname
setwidth while name width
```

3.1.4 Functions

Fotoscript also has several functions for manipulating images: `blur crop enhance flip flop grayscale negate resize rotate`

Note:

This is a subset of the functions that we envision Fotoscript implementing. The Java version of the Imagemagick image manipulation library is currently incomplete. We will add more functionality to Fotoscript as it becomes available; we have implemented the language in such a way that this should be trivial.

3.1.5 Literals

There are two kinds of literals: string literals and number literals.

string literals

A string literal is a string of characters surrounded by double quotes.

number literals

A number literal is a string of digits which represent a non-negative decimal integer. A user may use operators to create negative numbers as well as fractional numbers.

3.1.6 Separators

Separators include curly braces and newline characters. The curly braces, '{' and '}', designate blocks of code. The newline character separates expressions.

3.2 Types

In Fotoscript there are three classes of objects: images, numbers and strings. Variables can be either numbers or strings and are implicitly typed when they are created. For this reason, when it is created, each variable must either be assigned the value of a literal or that of an already typed variable; it takes its type from whence its initial value is taken.

3.2.1 The Image Class

Fotoscript has a native class of objects for storing all the data related to an image. This object has the following fields:

Field Name	Brief Description
`name`	the relative path name of the file
`width`	the image width in pixels
`height`	the image height in pixels
`depth`	the image color depth in number of available colors
`format`	the file format for the image to be saved to disk
`modified`	set to one if the image has been modified, zero otherwise

The Current Image

For convenience, FotoScript has a special pointer to an image object; it signifies the current image. It is assigned to point to each of the available images in turn as foreach iterates through them.

The Image Array

Because it may be useful to load and edit several images at once, there is a built-in array of image objects. When a new image object is loaded or created, it is appended to this array. The images in this array can then be accessed by `foreach` as described below.

3.2.2 The Number Class

FotoScript does not distinguish between different classes of number. A number variable can contain a reasonable range of positive and negative floating point numbers. Only non-negative integer values can be represented by string literals, but there are some operators that allow non-integer number literals to behave as one might expect. Here is a table of number operators; those with highest precedence are higher; each precedence level groups left to right:

Operators
unary and binary `.`
unary `-` and `+`
`*`, `/` and `%`
binary `-` and `+`
`==`, `!=`, `<`, `<=`, `>`, `>=`

The Unary - Operator

The unary - operator negates the number that follows it. Because their functions are similar, it does not need to be strongly distinguished from the binary - operator; if there is only one sensible operand, the unary version is used; if there are two, the binary version is used; no sensible operands is an error. For example: the string `-25` returns a negative number with magnitude twenty-five.

The Binary - Operator

The binary - operator has the same function as the unary version except that after the number following the operator is negated, it is added to to the number preceeding the operator. For example: the string `10-25` returns a negative number with magnitude fifteen.

The Unary . Operator

The unary . operator divides the number that follows it by 2^l where l is the length of the shortest string representing the number in decimal form. Because their functions are similar, it does not need to be strongly distinguished from the binary . operator; if there is only one sensible operand, the unary version is used; if there are two, the binary version is used; no sensible operands is an error. For example: the string `.25` returns a number with the value twenty-five hundreths.

The Binary . Operator

The binary . operator has the same function as the unary version except that after the number following the operator is divided as described above, it is added to to the number preceeding the operator. For example: the string `10.25` returns a number with the value ten and twenty-five hundreths.

Note:

The . operator is usually called “the decimal point” and it behaves as expected. Cascading decimal points is an error. For example: the string `10.25.36` is an error.

The Binary + Operator

The binary + operator adds its two operands together. If there is only one sensible operand it will have no function; no sensible operands is an error. For example: the string `10+25` returns a number with the value thirty-five.

The Binary * Operator

The binary * operator multiplies its two operands together. Fewer than two sensible operands is an error. For example: the string `10*25` returns a number with the value two-hundred-and-fifty.

The Binary / Operator

The binary / operator divides the operand preceeding it by the operand that follows it. Fewer than two sensible operands is an error. For example: the string `10/25` returns a number with the value ten twenty-fifths.

The Binary % Operator

The binary % operator divides the operand preceeding it by the operand that follows it and returns the remainder. Fewer that two sensible operands is an error. Non-integer operands is an error. For example: the string `10%25` returns a number with the value five.

The == Operator

The == operator compares its two operands and returns zero if they are equal and one otherwise. Fewer than two sensible operands is an error. For example: the string `10==25` returns one.

the != Operator

The != operator compares its two operands and returns one if they are equal and zero otherwise. Fewer than two sensible operands is an error. For example: the string `10!=25` returns zero.

The <Operator

The <operator compares its two operands and returns zero if the latter is greater than the former and one otherwise. Fewer than two sensible operands is an error. For example: the string `10<25` returns zero.

The <= Operator

The <= operator compares its two operands and returns zero if the latter is greater than or equal to the former and one otherwise. Fewer than two sensible operands is an error. For example: the string `10<=25` returns zero.

The >Operator

The >operator compares its two operands and returns zero if the latter is less than the former and one otherwise. Fewer than two sensible operands is an error. For example: the string `10>25` returns one.

The >= Operator

The >= operator compares its two operands and returns zero if the latter is less than or equal to the former and one otherwise. Fewer than two sensible operands is an error. For example: the string `10>=25` returns one.

3.3 Syntax

Fotoscript syntax is very simple. Each line of code can be of one of two types: flow control and functions.

3.3.1 Function Syntax

Function syntax is simple. The function name is written followed by some number of identifiers and it is terminated by a newline character. Some examples of functions are the following keywords:

`set` $variable\ value$
sets $variable$ to $value$; variables have three types: number, string and image; the literals for each type are non-ambiguous so the variable types are implicitly defined; $value$ can either be a literal or a variable that has already been implicitly typed by setting it with a literal. Setting a variable that has already been typed to a value of the wrong type is an error.

open

`open` $name$
opens a file or directory called $name$; if $name$ is not a readable file or directory it is an error.

new

`new` $name$
creates a new image for editing called $name$; note that it does not write the image to disk unless `save` is called.

save

`save`
writes the current image out to disk.

3.3.2 Flow Control

Special keywords control the flow of the program. Here are descriptions of each of these keywords and their syntax. In all cases $test$ can be omitted and will, by default, evaluate to true; sections surrounded by square brackets are optional.

foreach

`foreach` ($test$) then$\{\ newline\ statements\ newline\}\ newline$

For each member of the image array, set the current image to that member and perform $statements$ if $test$ is true.

if

`if` ($test1$) `then`{ $newline$ $statements1$ $newline$} [`else if` ($test$) `then`{ $newline$ $statements2$ $newline$}] [`else` { $newline$ $statements3$ $newline$}] $newline$

Perform $statements1$ if $test1$ is true[, $statements2$ otherwise if $test2$ is true][, $statements3$ otherwise]. The `else if` section may be repeated any number of times or omitted; the `else` section may be included exactly once or omitted.

while

`while` ($test$) then{ $newline$ $statements$ $newline$} $newline$

If $test$ is true, then perform $statements$; repeat.

Chapter 4

Code Listing

4.1 The Grammar

Written by James Kenneth Mooney

```
class FotoLexer extends Lexer;

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

NL : ('\n' | "\r\n" | '\r') { newline(); };

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

COMMENT : ('A'..'Z')+ { $setType(Token.SKIP); };

PLUS : '+';
DASH : '-' (NUMBER { $setType(NUMBER); })?; // for negative numbers
STAR : '*';
MOD : '%';
SLASH : '/';
LPREN : '(';
RPREN : ')';
LBRACE : '{';
RBRACE : '}';
COMMA : ',';
EQ : "==";
NE : "!=";
```

```
GE : ">=";
LE : "<=";
GT : '>';
LT : '<';
NOT : '!';
AND : "&&";
OR : "||";

ID options { testLiterals = true; } : 'a'..'z' ('_' | 'a'..'z' | '0'..'9')*;

NUMBER : ('0'..'9')+ ('.' ('0'..'9')*)? | '.' ('0'..'9')+;

STRING : '"'! (~('"'))* '"'!;

class FotoParser extends Parser;

options {
k = 2;
buildAST = true;
exportVocab = FotoScript;
}

tokens {
ROOT;
}

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

statement
   : assignment {System.out.println("assignment");}
   | foreach {System.out.println("foreach");}
   | while_stmt {System.out.println("while");}
   | if_stmt {System.out.println("if");}
//   | function {System.out.println("functin declaration");}
//   | ID arguments NL! {System.out.println("user definied function call");}
   | func_call NL! {System.out.println("built in function call");}
   | NL! {System.out.println("nl");}
   ;

assignment : "set"^ ID expression NL! | "sets"^ ID (STRING|ret_func_str) NL!;

while_stmt : "while"^ LPREN! logic RPREN! then_stmt NL!;

foreach : "foreach"^ LPREN!
```

```
(logic | { #foreach = #(#foreach, #([NUMBER, "1"])); } )
RPREN! then_stmt
//LBRACE! NL! (statement)* RBRACE! NL!
;

if_stmt : "if"^ LPREN! logic RPREN! then_stmt (NL! | else_stmt);

then_stmt: "then"^ LBRACE! NL! (statement)* RBRACE!;

else_stmt : "else"^ if_stmt | "else"^ LBRACE! NL! (statement)* RBRACE! NL!;

expression : expr2 (PLUS^ expr2 | DASH^ expr2)*;

expr2 : expr3 (STAR^ expr3 | SLASH^ expr3 | MOD^ expr3)*;

expr3 : LPREN! expression RPREN! | expr_atom;

logic : logic2 (AND^ logic2 | OR^ logic2)*;

logic2 : logic3 (EQ^ logic3 | NE^ logic3 | GT^ logic3 | LT^ logic3 | GE^ logi

logic3 : (NOT^)? logic4;

logic4 : LPREN! logic RPREN! | expression | "equals"^ str_atom str_atom;

expr_atom : ID | NUMBER | ret_func_expr;

str_atom : ID | STRING | ret_func_str;

//function : "function"^ LPREN! ID (ID (COMMA! ID)*)? RPREN! LBRACE! NL! (sta

//arguments : ((ID | STRING | NUMBER) (COMMA! (ID | STRING | NUMBER))* )?;

func_call
: "resize"^ expression expression
| "greyscale"^
| "rotate"^ expression
| "blur"^ expression expression
| "crop"^ expression expression expression expression
| "flip"^
| "flop"^
| "enhance"^
| "negate"^
| "open"^ str_atom
| "save"^
| "saveall"^
```

```
| "setwidth"^ expression
| "setheight"^ expression
| "setdepth"^ expression
| "setformat"^ str_atom
| "setname"^ str_atom
| "setmodified"^ logic
;

ret_func_expr
: "width"^
| "height"^
| "depth"^
| "modified"^
;

ret_func_str
: "format"^
| "name"^
| "concat"^ str_atom str_atom
;
```

4.2 The Walker

Written by Norman Yung with substantial contributions by James Mooney and Randall Li

```
{
import magick.*;
import java.util.*;
import java.io.*;
}

class FotoWalker extends TreeParser;
options {
    importVocab = FotoScript;
}

{
Vector imageVector = new Vector();
Hashtable userVars = new Hashtable();
static int imageNum = -1; //index of last added image
static int currNum = -1; //index of current image to process
}

program : #(ROOT (statement)*)
```

```
;

statement
: file_ops
| image_ops
| control_flow
| setvars
;

image_ops
{
Double a, b, c, d;
String n;
}
: #("resize" a=expr b=expr {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.scaleImage((int)a.doubleValue(), (int)b.doubleValue());
})
| #("rotate" c=expr {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.rotateImage(c.doubleValue());
currImage.setModified(1);
})
| #("blur" c=expr d=expr {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.blurImage(c.doubleValue(), d.doubleValue());
currImage.setModified(1);
})
| #("crop" a=expr b=expr c=expr d=expr {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.cropImage(new java.awt.Rectangle((int)a.doubleValue(), (int)b.doubl
currImage.setModified(1);
})
| #("flip" {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.flipImage();
currImage.setModified(1);
})
| #("flop" {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.flopImage();
currImage.setModified(1);
})
| #("enhance" {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.enhanceImage();
```

```
currImage.setModified(1);
})
| #("negate" {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.negateImage();
currImage.setModified(1);
})
| #("greyscale" {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.greyImage();
currImage.setModified(1);
})
| #("setwidth" a=expr {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.setWidth((int) a.doubleValue());
currImage.setModified(1);
})
| #("setheight" a=expr {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.setHeight((int) a.doubleValue());
currImage.setModified(1);
})
| #("setdepth" a=expr {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.setDepth((int)a.doubleValue());
currImage.setModified(1);
})
| #("setformat" n=string_type {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.setFormat(n);
currImage.setModified(1);
})
| #("setname" n=string_type {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.setName(n);
currImage.setModified(1);
})
| #("setmodified" a=expr {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.setModified(a.doubleValue());
})
;

control_flow {
Double a;
}
```

```
: #("if" a=if_cond:expr {
AST thenpart = if_cond.getNextSibling();
AST elsepart = thenpart.getNextSibling();
if (a.doubleValue() != 0) {
statement(thenpart);
}
else if (elsepart != null) {
statement(elsepart);
}
else {
// do nothing
}

})
| #("while" a=while_cond:expr {
AST thenpart = while_cond.getNextSibling();
while (a.doubleValue() != 0) {
statement(thenpart);
a=expr(while_cond);
}
})
| #(f:"foreach" {
AST foreach_cond = f.getFirstChild();
AST thenpart = foreach_cond.getNextSibling();
for (currNum = 0; currNum <= imageNum; currNum++) {
a=expr(foreach_cond);
if (a.doubleValue() != 0) {
statement(thenpart);
}
}
currNum = imageNum;
})
| #("then" (statement)* )
;

file_ops {
String a;
}
: #("open" a=string_type {
File f = new File(a);
if (f.isDirectory()) {
File[] files = f.listFiles();
for (int i = 0; i < files.length; i++) {
if (FotoImage.checkFormat(files[i].getPath())) {
try {
imageVector.add(new FotoImage(new ImageInfo(files[i].getPath())));
```

```
if (currNum == imageNum) { // notwithin a foreach loop
currNum++;
}
imageNum++;
System.out.println(imageNum + "    " + files[i].getParent() + "   > " + files
} catch (MagickException e) {System.err.println(e);}
} // end if checkFormat()
else {
System.out.println("file extension of [" + files[i].getPath() + "] not recogn
}
} //end forloop over directory
}
else if (f.isFile()) {
if (FotoImage.checkFormat(a)) {
try {
imageVector.add(new FotoImage(new ImageInfo(a)));
if (currNum == imageNum) { // notwithin a foreach loop
currNum++;
}
imageNum++;
System.out.println(imageNum + "    [" + f.getParent() + "]    " + a);
} catch (MagickException e) {System.err.println(e);}
} // end if checkFormat()
else {
System.out.println("file extension of [" + a + "] not recognized");
}
}
else { // file or directory does not exist
System.out.println("file or directory: " + a + " does not exist");
}
}) //finish "open"
| #("save" {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
currImage.saveImage();
})
| #("saveall" {
for (int i = 0; i < imageVector.size(); i++) {
FotoImage currImage = (FotoImage) imageVector.get(i);
currImage.saveImage();
}
})
;

setvars {
String a;
Double b;
```

```
String s;
}
: #("set" a=fresh_id b=expr {
// System.out.println("######################" + value.getText());
System.out.println("putting " + a + " with [" + b + "] as the value");
userVars.put(a, b);
})
| #("sets" a=fresh_id s=string_type {
// System.out.println("######################" + value.getText());
System.out.println("putting " + a + " with [" + s + "] as the value");
userVars.put(a, s);

})
;


expr returns [Double r] {
Double a, b;
String n, m;
r = new Double(0);
}
: #(PLUS a=expr b=expr { r = new Double(a.doubleValue()+ b.doubleValue()); })
| #(DASH a=expr b=expr { r = new Double(a.doubleValue()- b.doubleValue()); })
| #(STAR a=expr b=expr { r = new Double(a.doubleValue()* b.doubleValue()); })
| #(MOD a=expr b=expr { r = new Double((int)a.doubleValue()% (int)b.doubleVal
| #(SLASH a=expr b=expr { r = new Double(a.doubleValue()/ b.doubleValue()); }
| #(EQ  a=expr  b=expr {
if (a.doubleValue()== b.doubleValue()) r = new Double(1);
else r = new Double(0);
})
| #(NE  a=expr  b=expr {
if (a.doubleValue()!= b.doubleValue()) r = new Double(1);
else r = new Double(0);
})
| #(GE  a=expr  b=expr {
if (a.doubleValue()>= b.doubleValue()) r = new Double(1);
else r = new Double(0);
})
| #(LE  a=expr  b=expr {
if (a.doubleValue()<= b.doubleValue()) r = new Double(1);
else r = new Double(0);
})
| #(GT  a=expr  b=expr {
if (a.doubleValue() > b.doubleValue()) r = new Double(1);
else r = new Double(0);
})
```

```
| #(LT  a=expr  b=expr {
if (a.doubleValue() < b.doubleValue()) r = new Double(1);
else r = new Double(0);
})
| #(NOT  a=expr {
if (a.doubleValue()== 0) r = new Double(1);
else r = new Double(0);
})
| #(AND  a=expr  b=expr {
if (a.doubleValue()== 0 || b.doubleValue() == 0) r = new Double(0);
else r = new Double(1);
})
| #(OR  a=expr  b=expr {
if (a.doubleValue()== 0 && b.doubleValue() == 0) r = new Double(0);
else r = new Double(1);
})
| #(NUMBER { r = new Double(#NUMBER.getText());} )
| #("true" {r = new Double(1);})
| #("false" {r = new Double(0);})
| #("equals" n=string_type m=string_type {
if (n.compareTo(m) == 0) {
r = new Double(1);
}
else {
r = new Double(0);
}
})
// ------------ IMAGE INFO TYPES ----------------
| #("width" {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
r = new Double(currImage.getWidth());
})
| #("height" {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
r = new Double(currImage.getHeight());
})
| #("depth" {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
r = new Double(currImage.getDepth());
})
| #("modified" {
FotoImage currImage = (FotoImage) imageVector.get(currNum);
r = new Double(currImage.getModified());
})
| #(ID {
System.out.println("parsing " + #ID.getText() + " as an ID");
```

```
n=#ID.getText();
if (userVars.containsKey(n)) {
try {
r = new Double(userVars.get(n).toString());
} catch (NumberFormatException e) {
System.err.println("error: cannot parse " + n + " as a.doubleValue()number");
System.exit(1);
}
}
else {
System.err.println("error: " + n + " is not defined [expr]");
System.exit(1);
}
})
;


string_type returns [String r]
{
String a, b;
r = "";
}
: #("format" {
FotoImage currImage = (FotoImage) imageVector.get(imageNum);
r = currImage.getExtension();
})
| #("name" {
FotoImage currImage = (FotoImage) imageVector.get(imageNum);
r = currImage.getName();
})
| #("concat" a=string_type b=string_type {
r = a + b;
})
| #(STRING { r = #STRING.getText(); })
| #(ID {
a=#ID.getText();
if (userVars.containsKey(a)) {
r = userVars.get(a).toString();
}
else {
System.err.println("error: " + a + " is not defined [str]");
System.exit(1);
}
})
;
```

```
fresh_id returns [String r]
{
r = "";
}
: #(ID {
r=#ID.getText();
})
;
```

4.3 The Image Class

Written by Norman Yung

```
import magick.*;

public class FotoImage {
    private MagickImage image;

    private boolean modified;

    FotoImage(ImageInfo i) {
try {
    image = new MagickImage(i);
} catch (Exception e) {
    System.err.println(e);
}
modified = false;
    }

    void scaleImage(int a, int b) {
try {
    image = image.scaleImage(a, b);
} catch (Exception e) {
    System.err.println(e);
}
    }

    void rotateImage(double c) {
try {
    image = image.rotateImage(c);
} catch (Exception e) {
System.err.println(e);
}
    }
```

```
    void blurImage(double c, double d) {
try {
    image = image.blurImage(c, d);
} catch (Exception e) {
    System.err.println(e);
}
    }

    void cropImage(java.awt.Rectangle r) {
try {
    image = image.cropImage(r);
} catch (Exception e) {
    System.err.println(e);
}
    }

    void flipImage() {
try {
    image = image.flipImage();
} catch (Exception e) {
    System.err.println(e);
}
    }

    void flopImage() {
try {
    image = image.flopImage();
} catch (Exception e) {
    System.err.println(e);
}
    }

    void enhanceImage() {
try {
    image = image.enhanceImage();
} catch (Exception e) {
    System.err.println(e);
}
    }

    void negateImage() {
try {
    image.negateImage(0);
} catch (Exception e) {
    System.err.println(e);
}
```

```
    }

    void greyImage() {
try {
    image.setGrayscale();
} catch (Exception e) {
    System.err.println(e);
}
    }

    void setDepth(int a) {
try {
    image.setNumberColors(a);
} catch (Exception e) {
    System.err.println(e);
}
    }

    void setWidth(int a) {
try {
    image = image.scaleImage(a, getHeight());
} catch (Exception e) {
    System.err.println(e);
}
    }

    void setHeight(int a) {
try {
    image = image.scaleImage(getWidth(), a);
} catch (Exception e) {
    System.err.println(e);
}
    }

    void setName(String n) {
try {
    image.setFileName(n + "." + getExtension());
} catch (Exception e) {
    System.err.println(e);
}
    }

    void setFormat(String n) {
if (checkFormat("blablabla." + n)) {
    try {
image.setFileName(getName() + "." + n);
```

```
    } catch (Exception e) {
System.err.println(e);
    }
} else {
    System.out.println("format: [" + n + "] not recognized");
}
    }

    void setModified(double a) {
if (a == 0) {
    modified = false;
} else {
    modified = true;
}
    }

    void saveImage() {
try {
    image.writeImage(new ImageInfo());
} catch (Exception e) {
    System.err.println(e);
}
    }

    // ---------------- accessor ----------------
    int getHeight() {
try {
    return (int) image.getDimension().getHeight();
} catch (Exception e) {
    System.err.println(e);
    return 0;
}
    }

    int getWidth() {
try {
    return (int) image.getDimension().getWidth();
} catch (Exception e) {
    System.err.println(e);
    return 0;
}
    }

    String getName() {
String r = null;
try {
```

```
    String temp = image.getFileName();
    int ext_pos = temp.lastIndexOf(".");
    r = temp.substring(0, ext_pos);

} catch (Exception e) {
    System.err.println(e);
}
return r;
    }

    String getExtension() {
String r = null;
try {
    String temp = image.getFileName();
    int ext_pos = temp.lastIndexOf(".");
    r = temp.substring(ext_pos + 1);

} catch (Exception e) {
    System.err.println(e);
}
return r;
    }

    int getDepth() {
int r = 0;
try {
    r = image.getNumberColors();
} catch (Exception e) {
    System.err.println(e);
}
return r;
    }

    int getModified() {
if (modified) {
    return 1;
}
return 0;
    }

    MagickImage getImage() {
return image;
    }

    static boolean checkFormat(String filename) {
String temp = filename;
```

```
int ext_pos = temp.lastIndexOf(".");
String r = temp.substring(ext_pos + 1);
if (r.compareTo("gif") == 0 || r.compareTo("jpg") == 0
    || r.compareTo("bmp") == 0) {
    return true;
} else {
    return false;
}
    }
}
```

4.4 The Starter Class

Written by Norman Yung

```
import java.io.FileInputStream;
import antlr.CommonAST;

public class fotoscript {
public static void main(String args[]) {
try {
FileInputStream input = new FileInputStream(args[0]);

// Create the lexer and parser and feed them the input
FotoLexer lexer = new FotoLexer(input);
FotoParser parser = new FotoParser(lexer);
parser.program(); // "program" is the main rule in the parser

// Get the AST from the parser
CommonAST parseTree = (CommonAST) parser.getAST();

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

} catch (Exception e) {
System.err.println("Exception: " + e);
}
}
}
```

4.5 The Test Suite

Written by Randall Q Li

```
import java.io.*;
```

```
public class CreateCode {
    static int padding;
    static int filenumber;
    static String leaf[];
    static String conditional[][];
    static String stringConditional[][];
    static String conditionOperator[];
    static String equalsOperator[];
    static String operator[];
    static String file[];
    static String leafMod[];
    static String controlFlow[][];
    static String controlFlowElse[][];
    static String outFileName;
    static String refOutFileName;
    static boolean appendtoRefFile;
    public static void main(String[] args) throws IOException {
initialize();
shortness();
longness();
    }
    static void longness(){
String val[], val2[], temp[];
temp=  copy(leaf);
int count;

//for(j=0;j<)
val2 = temp;
val2 = new String[temp.length*temp.length];
count=0;
for(int i=0;i<temp.length;i++){
    for(int j=0;j< temp.length;j++){
val2[count] = temp[i] + "\n" + temp[j];
count++;
    }
}
//printStringArray(val2);

printoutArray(val2);
val = createIf(val2);
printoutArray(val);

val2 = new String[file.length*val.length];
count=0;
for (int i = 0; i < file.length; i++) {
    for(int j=0;j< val.length;j++){
```

```
val2[count] = file[i] + "\n" + val[j];
count++;
    }
}
printoutArray(val2);
//val = append("\nabc", val);
//printoutArray(val);
//val = prefix("abc\n", val);
//val = copy(createIf(val));
val = createIfElse(temp, temp);
val2 = new String[file.length*val.length];
count=0;
for (int i = 0; i < file.length; i++) {
    for(int j=0;j< val.length;j++){
val2[count] = file[i] + "\n" + val[j];
count++;
    }
}
printoutArray(val);
printoutArray(val2);

//printStringArray(val);

    }
    static void shortness(){
String temp[],temp2[];
temp= new String[2];
temp[0]=leaf[1];
temp[1]=leaf[2];
printoutArray(temp);
//for(int i=0;i<temp.length;i++){
temp2=prefix("if(modified) then {\n\t",temp);
temp2=append("\n}",temp2);
printoutArray(temp2);
temp2=prefix("foreach()then{\n\t",temp);
temp2=append("\n}",temp2);
printoutArray(temp2);
//}


    }
    static String[] copy(String[] original) {
String value[] = new String[original.length];
for (int i = 0; i < value.length; i++) {
    value[i] = original[i];
}
```

```
return value;
    }
    static void printStringArray(String[] value) {
for (int i = 0; i < value.length; i++) {
    System.out.println(value[i]+"\n##\n");
}
    }
    static String[] prefix(String value, String[] array) {
String temp[];
temp = new String[array.length];
for (int i = 0; i < array.length; i++) {
    temp[i] = value + array[i];
}
return temp;
    }
    static String[] append(String value, String[] array) {
String temp[];
temp = new String[array.length];
for (int i = 0; i < array.length; i++) {
    temp[i] = array[i] + value;
}
return temp;
    }
    static String[] replace(String oldString, String newString, String[] arra
//static String replace(String oldString, String newString, String
// array){
String temp;
String value[], preOld;
int i;
value = new String[array.length];
for (int j = 0; j < array.length; j++) {
    value[j] = "";
    temp = array[j];
    i = temp.indexOf(oldString);
    while (i != -1) {
preOld = temp.substring(0, i);
temp = temp.substring(i + oldString.length());
value[j] = value[j] + preOld + newString;
i = temp.indexOf(oldString);
    }
    value[j] = value[j] + temp;
}
return value;
    }
    static String[] createIf(String[] statement) {
String value[];
```

```
int current = 0;
statement = replace("\n", "\n\t", statement);
//System.out.println(controlFlow.length*leaf.length*conditional.length*condit
value = new String[controlFlow.length
   * statement.length
   * (conditional.length
      * (conditionOperator.length + equalsOperator.length) + stringConditiona
      * equalsOperator.length)];
for (int i = 0; i < controlFlow.length; i++) {
    for (int j = 0; j < statement.length; j++) {
for (int k = 0; k < stringConditional.length; k++) {
    for (int l = 0; l < equalsOperator.length; l++) {
value[current] = controlFlow[i][0];
value[current] = value[current] +"equals "+ stringConditional[k][0];
value[current] = value[current] + " ";
value[current] = value[current] + stringConditional[k][1];
value[current] = value[current] + controlFlow[i][1];
value[current] = value[current] + "\n\t" + statement[j]
    + "\n";
value[current] = value[current] + controlFlow[i][2];
current++;
    }
}
for (int k = 0; k < conditional.length; k++) {
    for (int l = 0; l < equalsOperator.length; l++) {
value[current] = controlFlow[i][0];
value[current] = value[current] + conditional[k][0];
value[current] = value[current] + equalsOperator[l];
value[current] = value[current] + conditional[k][1];
value[current] = value[current] + controlFlow[i][1];
value[current] = value[current] + "\n\t" + statement[j]
    + "\n";
value[current] = value[current] + controlFlow[i][2];
current++;
    }
    //if((!conditional[k][0].equals(conditional[0][0]))&&(!conditional[k][0].
    for (int l = 0; l < conditionOperator.length; l++) {
value[current] = controlFlow[i][0];
value[current] = value[current] + conditional[k][0];
value[current] = value[current] + conditionOperator[l];
value[current] = value[current] + conditional[k][1];
value[current] = value[current] + controlFlow[i][1];
value[current] = value[current] + "\n\t" + statement[j]
    + "\n";
value[current] = value[current] + controlFlow[i][2];
current++;
```

```
    }
    //}
}
    }
}
return value;
    }
    static String[] createIfElse(String[] statement, String[] statement2) {
String value[];
int current = 0;
statement = replace("\n", "\n\t", statement);
//System.out.println(controlFlow.length*leaf.length*conditional.length*conditi
value = new String[controlFlowElse.length
   * statement.length
   * statement2.length
   * (conditional.length
      * (conditionOperator.length + equalsOperator.length) + stringConditiona
      * equalsOperator.length)];
for (int i = 0; i < controlFlowElse.length; i++) {
    for (int j = 0; j < statement.length; j++) {
for (int p = 0; p < statement2.length; p++) {
    for (int k = 0; k < stringConditional.length; k++) {
for (int l = 0; l < equalsOperator.length; l++) {
    value[current] = controlFlow[i][0];
    value[current] = value[current] +"equals "+ stringConditional[k][0];
    value[current] = value[current] + " ";
    value[current] = value[current] + stringConditional[k][1];
    value[current] = value[current] + controlFlow[i][1];
    value[current] = value[current] + "\n\t"
+ statement[j] + "\n";
    value[current] = value[current] + controlFlow[i][2];
    current++;
}
    }
    for (int k = 0; k < conditional.length; k++) {
for (int l = 0; l < equalsOperator.length; l++) {
    value[current] = controlFlow[i][0];
    value[current] = value[current] + conditional[k][0];
    value[current] = value[current] + equalsOperator[l];
    value[current] = value[current] + conditional[k][1];
    value[current] = value[current] + controlFlow[i][1];
    value[current] = value[current] + "\n\t"
+ statement[j] + "\n";
    value[current] = value[current] + controlFlow[i][2];
    current++;
}
```

```
//if((!conditional[k][0].equals(conditional[0][0]))&&(!conditional[k][0].equa
for (int l = 0; l < conditionOperator.length; l++) {
    value[current] = controlFlowElse[i][0];
    value[current] = value[current] + conditional[k][0];
    value[current] = value[current]
+ conditionOperator[l];
    value[current] = value[current] + conditional[k][1];
    value[current] = value[current]
+ controlFlowElse[i][1];
    value[current] = value[current] + "\n\t"
+ statement[j] + "\n";
    value[current] = value[current]
+ controlFlowElse[i][2];
    value[current] = value[current] + "\n\t"
+ statement2[p] + "\n";
    value[current] = value[current]
+ controlFlowElse[i][3];
    current++;
}
//}
    }
}
    }
}
return value;
    }
    static void zeroOut(String[][] value) {
//String value[][]=new String[9][9];
for (int i = 0; i < value.length; i++) {
    for (int j = 0; j < value[i].length; j++) {
value[i][j] = "0";
    }
}
    }
    static void initialize() {
outFileName = "testcode";
refOutFileName = "refereance";
padding = 0;
appendtoRefFile = false;
conditionOperator = new String[4];
conditionOperator[0] = "<";
conditionOperator[1] = "<=";
conditionOperator[2] = ">";
conditionOperator[3] = ">=";
equalsOperator = new String[2];
equalsOperator[0] = "==";
```

```
equalsOperator[1] = "!=";
operator = new String[4];
operator[0] = "/";
operator[1] = "*";
operator[2] = "-";
operator[3] = "+";
stringConditional = new String[2][2];
//attribute
stringConditional[0][0] = "name";
stringConditional[1][0] = "format";
//value
stringConditional[0][1] = "\"bar\"";
stringConditional[1][1] = "\"jpg\"";
conditional = new String[4][2];
//attribute
conditional[0][0] = "height";
conditional[1][0] = "width";
conditional[2][0] = "depth";
conditional[3][0] = "modified";
//value
conditional[0][1] = "50";
conditional[1][1] = "40";
conditional[2][1] = "3";
conditional[3][1] = "0";
//conditional[0][2]="4";
//conditional[1][2]="gif";
//conditional[2][2]="35";
//conditional[3][2]="45";
//conditional[4][2]="bar";
//conditional[5][2]="1";
leaf = new String[18];//[3];
//attribute
leaf[0] = "setname \"bar\"";
leaf[1] = "setformat \"jpg\"";
leaf[2] = "setheight 50";
leaf[3] = "setwidth 40";
leaf[4] = "setdepth 3";
leaf[5] = "setmodified true";
leaf[6] = "resize 50 30";
leaf[7] = "flip";
leaf[8] = "flop";
leaf[9] = "negate";
leaf[10] = "save";
leaf[11] = "saveall";
leaf[12] = "enhance";
leaf[13] = "greyscale";
```

```
leaf[14] = "rotate 90";
leaf[15] = "blur 3 4";
leaf[16] = "crop 5 5 35 25";
leaf[17] = "open \"foobar.jpg\"";

/*
 * values leaf[0][1]="3"; leaf[1][1]="jpg"; leaf[2][1]="50";
 * leaf[3][1]="40"; leaf[4][1]="bar"; leaf[5][1]="0"; leaf[0][2]="4";
 * leaf[1][2]="gif"; leaf[2][2]="35"; leaf[3][2]="45"; leaf[4][2]="bar";
 * leaf[5][2]="1";
 */
file = new String[3];
//attribute
file[0] = "open \"foo\"";
file[1] = "open \"foo.jp\"";
file[2] = "open \"foo.gif\"";
/*
 * file[1][0] = "open"; //values file[0][1] = "foo"; file[1][1] = "foo";
 * file[0][2] = "foo.jpg"; file[1][2] = "foo.jpg";
 */
leafMod = new String[2];
leafMod[0] = "";
leafMod[1] = "set ";
controlFlow = new String[2][3];
controlFlow[0][0] = "if(";
controlFlow[1][0] = "foreach(";
controlFlow[0][1] = ")then{";
controlFlow[1][1] = ")then{";
controlFlow[0][2] = "}";
controlFlow[1][2] = "}";
controlFlowElse = new String[1][4];
controlFlowElse[0][0] = "if(";
//controlFlowElse[1][0] = "foreach(";
controlFlowElse[0][1] = ")then{";
//controlFlowElse[1][1] = "){";
controlFlowElse[0][2] = "}else{";
//controlFlowElse[1][2] = "}else{";
controlFlowElse[0][3] = "}";
//controlFlowElse[1][3] = "}";
    }
    static void printoutArray(String[] value) {
for (int i = 0; i < value.length; i++) {
    printout(value[i]);
}
    }
    static void printout(String value) {
```

```
//prints value out to a file of name and increments counter
// ie testfile001.txt
FileOutputStream out; // declare a file output object
PrintStream p1; // declare a print stream object
FileOutputStream refOut; // declare a file output object
PrintStream p2; // declare a print stream object
String newOutFileName = outFileName + pad(filenumber);
try {
    // Create a new file output stream
    refOut = new FileOutputStream(refOutFileName + ".foto",
  appendtoRefFile);
    appendtoRefFile = true;
    // Connect print stream to the output stream
    p2 = new PrintStream(refOut);
    //out = new FileOutputStream(newOutFileName + ".foto");
    //p1 = new PrintStream(out);
    //p1.println(value);
    //System.out.println("XXXXXX IN FILE " + outFileName.toUpperCase()+ " XXX
    System.out.println("XXXXXXXXXXXXXXXXXXXXXXXX");
    System.out.println(value);
    //p2.println("XXXXXX IN FILE " + outFileName.toUpperCase()+ " XXXXX");
    p2.println("XXXXXXXXXXXXXXXXXXXXXXXX");
    p2.println(value);
    filenumber++;
    //p1.close();
    p2.close();
} catch (Exception e) {
    System.err.println("Error writing to file");
}
    }
    static String pad(int r) {
String value = "";
int j = r;
for (int i = 0; i < padding - 1; i++) {
    if (j < 10) {
value = value + "0";
    } else {
//value=value+"0";
j = j / 10;
    }
}
return value + r;
    }
}
/*
 * DataInputStream in = new DataInputStream(new FileInputStream(inFileName));
```

```
*
* String unit=new String("blah");
*
* unit = in.readLine(); size=Integer.parseInt(unit); in.close();
*/
```