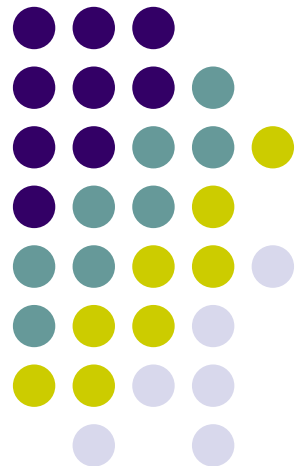
IML

Image Manipulation Program

Steven Chaitoff
Eric Hu
Cindy Liao
Zach van Schouwen



Motivation

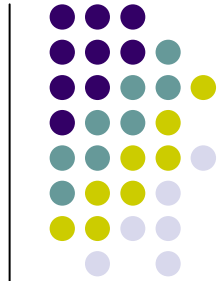
- Batch image processing can be tedious using graphical programs
- Command line image manipulation programs like ImageMagick have limited flexibility especially when it comes to file specification
- Wouldn’t it be great if we can write a program to specify both the manipulation we want and which files to process?

Overview

- IML is designed for easy image manipulation and batch processing
- Constructs for
  - Getting and manipulating pixel data
  - Opening and saving images
  - Flow control and math operations
- Allows for complex image transformations and effects



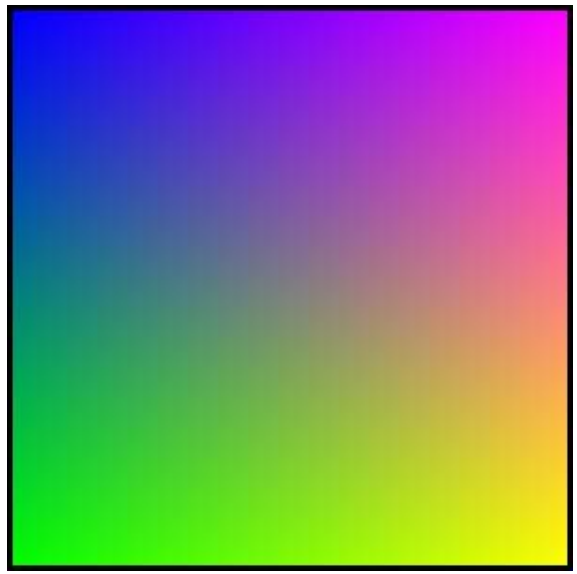

Overview



- Uses C like syntax so it is easy to pick up by programmers
- Constructs unique to IML are kept simple and unambiguous
  - File system interaction limited to “save” and “open” commands
  - Uses duck typing principle instead of explicit typecasting

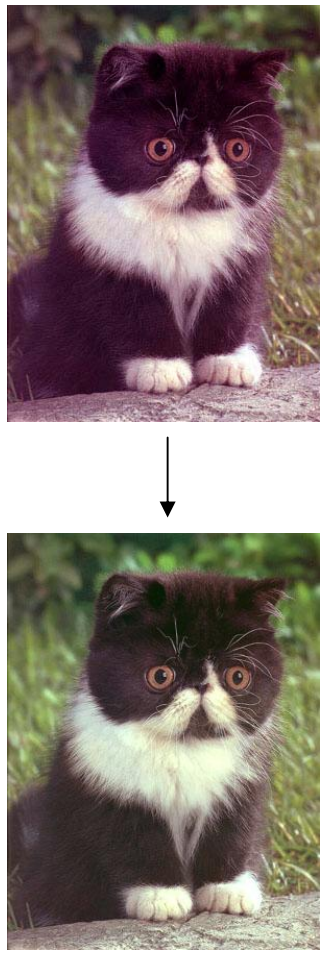
A Simple Introduction

```
/*
* spectrum.iml
* generates a color spectrum
*/

function main() {
   Pixel s[300][300];
   Image spectrum;
   spectrum = s;

   Int i;   Int j;
   for (i=3; i < cols spectrum-3; i = i + 1) {
      for (j=3; j < rows spectrum-3; j = j + 1) {
        red   spectrum[i][j] = 255 * (i - 3) / (cols spectrum - 6);
        green spectrum[i][j] = 255 * (j - 3) / (rows spectrum - 6);
        blue  spectrum[i][j] =-255 * (j - 3) / (rows spectrum - 6) + 255;
        alpha spectrum[i][j] = 255;
      }
   }
   spectrum save "testing/images/spectrum.png";
   print ("spectrum saved");
}
```

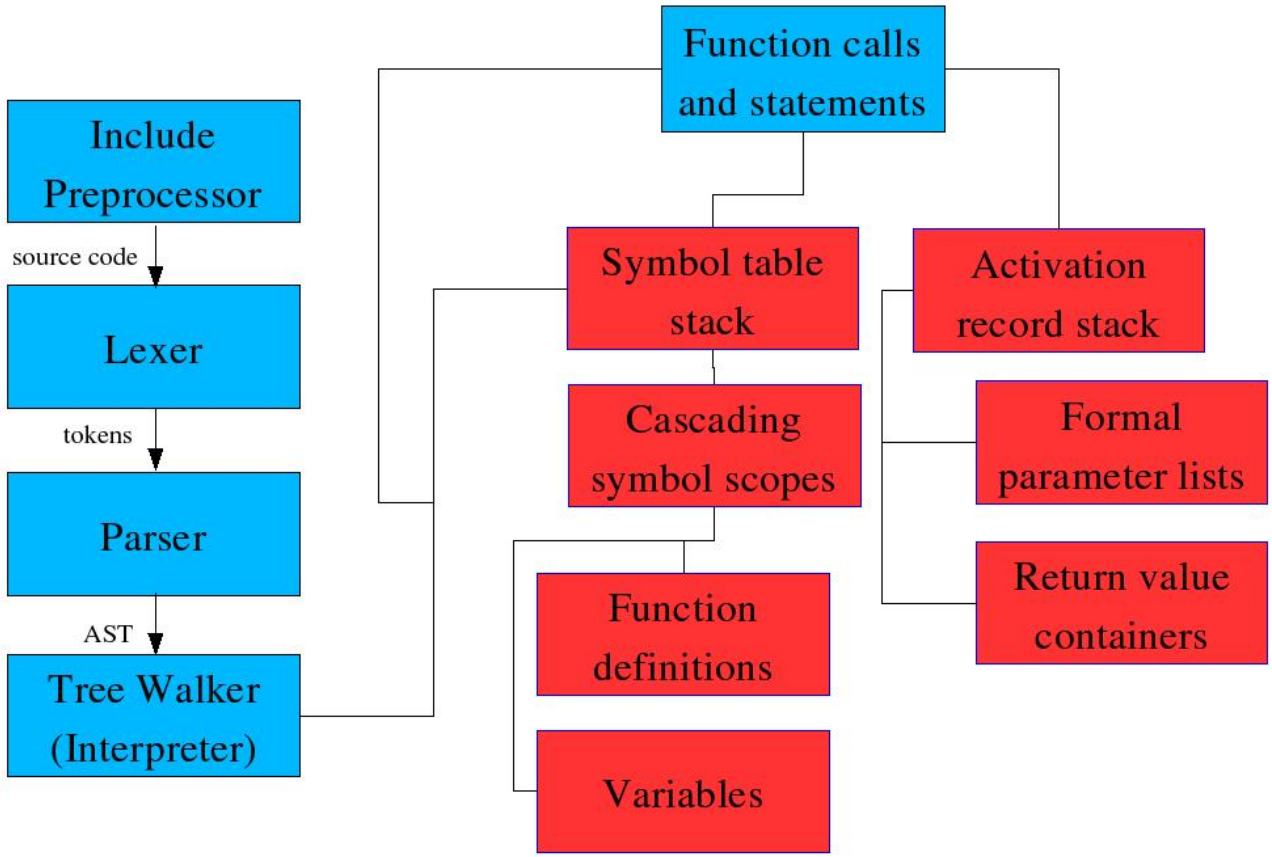


A Simple Introduction

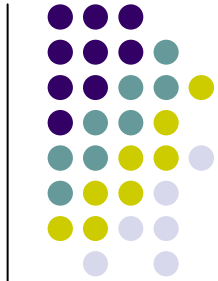
```
/*
* color_correct.iml
* modifies green channel of an image and saves a copy
*/
function color_correct(Image img, Int grn) {
   Int i;   Int j;   Int temp;
   for ( ; i < cols img; i = i + 1) {
      for (j=0; j < rows img; j = j + 1) {
         if ((temp = green img[i][j]) + grn <= 255) {
            green img[i][j] = temp + grn;
         }
                  else { green img[i][j] = 255; }
      }
   }
   return img;
}

function main() {
   Image kitten;
   "kitten.jpg" open kitten;
   color_correct(kitten, 25) save "kitten_corrected.jpg";
   print ("image saved");
}
```



Interpreter Structure



Lessons Learned



- Organization and Design
  - Start early
  - Use Interfaces
- Testing
  - Write test code for someone else’s section
- Working Together
  - Use Subversion