All That Matrix Language Design Proposal

Stefanie Zhou (sz2475)

Introduction

All That Matrix (ATM) is a programming language targeted at matrix manipulations with emphasize on concise syntax and lightweight compiler. Applications of matrix are very common across scientific fields. In statistics, matrices are used for probability calculations. In computer graphics, matrices are used to project and transform images. And you won't get through a lecture of linear algebra without encountering matrices.

While there are many applications that involve matrix computations, ATM was designed for image processing and filtering. Instead of performing complex calculations on each pixel of an image to get a certain effect one wants, black and white for example, ATM can produce the same result using compact matrix operations, making the computation simpler and code easier to read.

Another feature of ATM is that the built-in types, operators, and keywords are kept to a minimal set, with Matrix being the fundamental data type. Color and Image make up the rest of the built-in types for easy image processing purposes. Other applications of matrix can later be constructed either in the form of standard library or customized classed written by programmers.

Built-in data types

Integer, Double

Integers and doubles are defined in the conventional way with basic arithmetic operators including addition, subtraction, multiplication, division, greater than, less than, equals, greater than or equal to, and less than or equal to.

Color

Standard RGB colors for image processing.

Matrix

Matrices are defined with square brackets with vertical bar separating the rows and spaces separating the columns. The built-in operators for matrix include scaling, transpose, inverse, etc.

Image

Image is made up of a collection of matrices and its own RGB channel. Built-in functions for reading in an image file and converting it to the Image object and vice versa will come in handy.

Reserved

[ ] | = // + - * / ; ( ) == :=

Keywords

if else return print

Sample Code

```
// define a 3 by 2 matrix
Matrix myMatrix = [1 2 | 3 4 | 5 6];

// define a function called demoFunction
Matrix demoFunction (Matrix x, Matrix y) {
         // check if x and y are the same size
        if (x := y){
                return Matrix[x[0] | y[0] ];
        }
        else {
                return Matrix[0];
        }
}

// read in an image with built-in function read
Image myImage = read (“sameImage.jpg”);

// give myImage a soft blur with softBlur, which can be either a built-in or user-defined function
Image newImage  = softBlur(myImage)

// output the image
write(newImage , ‘blurry_effect.jpg’)
```