3D Graphics Module

Ramesh Srigiriraju
Computer Systems Lab 2006-07

January 18, 2007

Abstract

The purpose of this research project is to write a program that allows the user to graph functions of two variables. The program will incorporate elements of 3D graphics by allowing the user to rotate, shrink, and stretch the graphs. The program will be included in the student Intranet as a module and will thus include elements of modular design. The program will also include modules that allow the user to perform operations on matrices, although this module is not necessary for the 3D graphics part of the project.
This project is worth doing because it allows researchers to compare the graphics capabilities of Java with those of other languages. The groups who will be interested in the results are TJ students who need a program to graph functions and researchers who want to compare the graphics capabilities of Java with those of other languages. My method of research involves the use of linear algebra (specifically, matrices) to program graphical operations.

1 Introduction

1.1 Scope of Study

The scope of this program is to use homogeneous coordinates and matrices to perform various operations on the graphs, such as stretches and rotations. The part of the function to be graphed will be determined by a set of bounds inputted by the user. After graphing the function, the user can calculate

numeric derivatives or integrals of the function and find the intersection between two graphs. In order to input the function, the program will create a binary expression tree and substitute in values of the independent variables to determine the value of the function at various points. This project will also include a matrix module, which will allow the user to perform operations on matrices (such as matrix multiplication, row reduction, finding the inverse of the matrix, finding the determinant, transposing the matrix, etc). This module will create separate matrix editing windows whenever the user decides to edit a matrix. When closed, the matrix editing windows will return the values that were inputted by the user.

1.2 Purpose/Subject

The purpose of this research project is to write a program that allows the user to graph functions of two variables. The subjects of this project include 3D graphics and modular design. This project will help researchers compare the grahics capabilities of Java with those of other languages such as Python, C, and OpenGL.

2 Background

Previous projects concerning this area of research include The Investigation of Graphics in the Processing Language by J. Trent, CityBlock Project: Multi-perspective Panoramas of City Blocks by M. Levoy, and TJForge Iodine for the modular programming component. The 3D graphics projects seemed to use rotation matrices, such as the 2D matrix [[cos(a) -sin(a)] [sin(a) cos(a)]], to rotate graphs by an angle a (Levoy, Trent). Other sources specific to Java programming suggested the use of the format xB instead of Ax for linear transformations, where both the column vector and the matrix get transposed. The purpose of this was to take advantage of the way arrays are stored in Java and to reduce errors (Ameraal). Iodine used HTML to program in the modules. Possible state-of-the art programs could be MatLab or other computer algebra systems or even the 3D-graphing feature of the TI-89.

3 Development

3.1 Development Plan

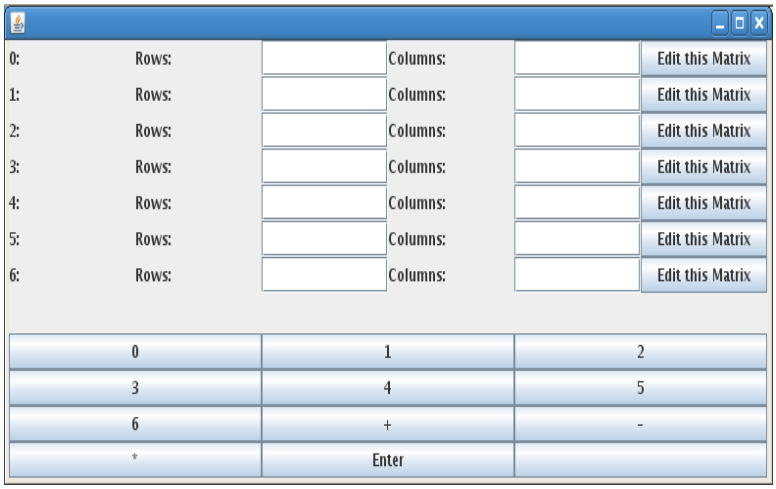
My project uses the staged delivery development process, since I have a different plan for each quarter. Every quarter, I have a specific version in mind that has specific functionality. For the first quarter, I planned to just implement a regular calculator module to make sure my recursive evaluation algorithms were functioning properly. These recursive algorithms would be used again for my graphing calculator, since the equations would be read in and stored in binary expression trees before graphing. During second quarter, I planned to implement a matrix editing module since the 3D graphics component requires the use of matrices. For third quarter, I plan to actually implement my graphing module.

3.2 Testing Requirements

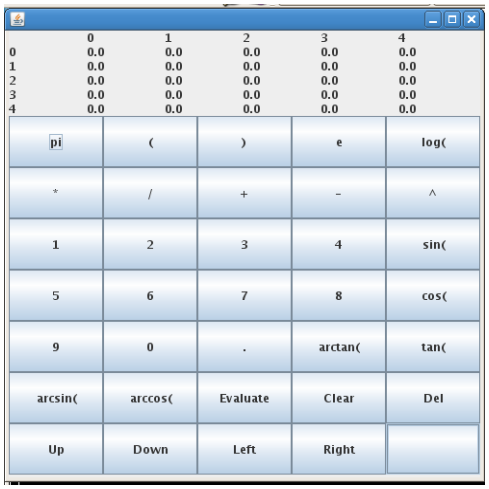
The implementation of the binary expression trees was pretty straightforward, so my first criteria for determining success involved the actual parsing of strings. I had to make sure that the listeners associated with the "Enter" button followed the correct order of operations and created binary expression trees based on that order of operations. To test the accuracy of my program, I used the TI-83 evaluation algorithm as a standard. My second criteria was to make sure the matrix editing panel and the matrix operations panel interacted correctly so that matrices could be inputted without losing data.

3.3 Runtime Process

During the runtime of my second quarter version, the user starts out with a matrix operations window.



After typing in the number of rows and number of columns for a certain matrix, the user then presses the "Edit this Matrix!" button. This causes a matrix editor screen to pop up.



The user must then use the buttons on the panel to type in expressions or numbers into the cells. If an expression is inputted, the "Evaluate" button will replace it with the corresponding numerical value. The "up," "down," "left," and "right" buttons move the cursor. When this window is closed, the program returns the values that were typed in and stores them in the matrix, allowing the user to perform operations on it. This can be done on the matrix operations screen. The user must use the buttons on the bottom

of the panel to type in matrix equations, and when the "Enter" button is pressed, the resultant matrix is displayed.

4 Results, Conclusion, and Discussion

So far, I have managed to create a working binary expression trees class that can handle logarithmic functions, exponential functions, trigonometric operations, inverse trigonometric operations, and regular arithmetic operations. The trees for non-arithmetic operations only have one subtree since they only take one argument. To test my binary expression tree class, I created a simple calculator that takes in a string containing an expression and evaluates it. I also created a class that parsed the string and broke it up based on an order of operations that I determined. My algorithms work so far, except for the fact that my string parser can't handle nested functions. It assumes that the closing parenthesis for a function is the first closing parenthesis it encounters, which is obviously not true for nested functions.
My matrix editor met with less success, since the various classes didn't interact the right way. I tried to get the program to just switch back and forth between the matrix editor and the matrix operations frame by putting a new GUI on top of the old GUI. However, adding another GUI onto a JFrame didn't make the previous GUI go away, so I had to create a separate JFrame for my matrix editor. My matrix operations frame was easier to program, since I used a structure that was similar to my calculator module. I had the program read in expressions off a JLabel, parse the string based on an order of operations, and then perform operations on the matrices. Since I haven't programmed a binary expression tree class for my matrix editor yet, it can only perform simple matrix operations now.

References

[1] M. Levoy, "CityBlock Project: Multi-perspective Panoramas of City Blocks," 2006.

[2] J. Trent, "The Investigation of Graphics in the Processing Language," 2006.

[3] L. Ameraal, Computer Graphics for Java Programmers, 1998.