

EZGraphs

A graphs and charts generating language

Language Proposal

02/07/07

Team:

Vincent Dobrev vd2006@columbia.edu
Edlira Kumbarce ek2248@columbia.edu



COMS W4115: Programming Languages and Translators
Prof. Stephen A. Edwards
Spring 2007



Background

Information graphics are visual representations of data, knowledge and information. They are used anywhere information needs to be explained simply and quickly, especially in mathematics, statistics and computer science where they aid in the process of developing and communicating conceptual information and scientific data. The first information graphics date back to early humans and their cave paintings, and later can be seen in the design of maps. The first *data* graphs appear in William Playfair's book, *The Commercial and Political Atlas*, in 1786.

Why Graphs?

Graphs such as bar and pie charts communicate visually data that, if presented in words or in tables, could appear cumbersome and difficult to understand. All sorts of complicated information can be presented using graphs, ranging from a simple curve representing the relationship of temperature over time to a bubble and line hierarchical graph depicting the HTML structure of a website. Because of the intrinsic simplicity of graphs in presenting scientific information, they are introduced to children in schools early on.

Introduction

Our objective is to design a language that can be used to create graphs. Although there are various tools available for generating graphs, most of these tools are not very intuitive and require a certain amount of learning before they can be used. EZGraphs aims in becoming the simplest way for any user to define a graph by having them specify the graph's characteristics in commands very similar to how they would verbally describe a graph using English.

Simple Syntax

The target users for this language are those with no prior programming experience and middle-school children. Therefore, the syntax of our language will be as close to English as possible, minimizing the use of special characters where they're not necessary. Most keywords in the language aim to be intuitive. Each would be the first thing that comes to mind when thinking about the specific object or action it corresponds to. This makes the language syntax very easy to learn and memorize. Moreover, the language itself provides for writing script files that are organized, with clear, distinctive statement blocks defining each component of a graph, and simple names that "make sense" for functions that are to be called on a component. To make it as effortless as possible to use, EZGraphs is intentionally devised in a way that enables users to write very high-level scripts that design the graphs without having to worry about any low-level details of how this is actually accomplished.



Organized

Because our language pivots around the concept of looking at a graph as made up of components, it is designed so that the user can write separate blocks of code to define the characteristics of each graph component. EZGraphs tends to be object-oriented and therefore allows for more organized code. Aside from non-programmers and young students, this language can be used by beginner programmers since it can help them learn concepts of OOP. As they write scripts in our language, these users can learn about notions such as Inheritance and Composition by declaring that an Axis IS A Line and a Graph HAS A Bar.

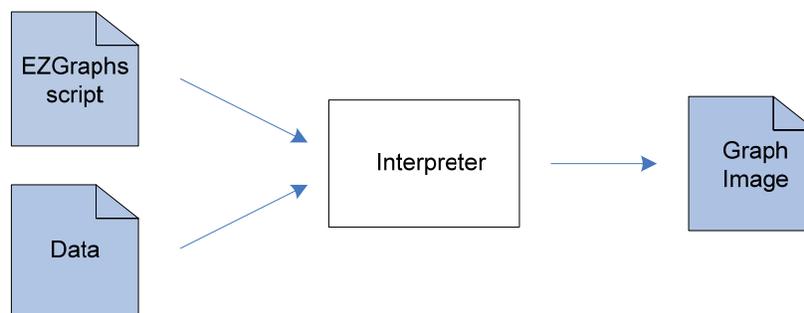
Easy Debugging

One of the reasons we are using ANTLR when designing our language is the nice and informative error messages it gives. It makes a world of difference when learning a new language or trying to use a new application. EZGraphs will have non-cryptic and informative error messages that will make it easy to use and debug.

Portable

Because it is based on Java, EZGraphs will be very portable and can be used on machines running Windows, Linux or Mac OS. It will constitute of a few classes which could very easily be put in a single JAR file. The tool can then be used on any computer that has Java Runtime Environment installed.

High-Level Structural Diagram





Example Syntax

```
define stats as Graph {
    type = barH                //options: barH, barV, line, point, pie
    grid = on                  //options: on, off
}

define canv as Canvas {
    color = Color(255,213,170)
    width = 300                //in pixels
    height = 200
}

define xAxis as Line {
    thickness = thin          //options: thin, normal, thick
    color = Color(lightblue)
    step = 5                  //step for the tick marks
    start = 0                 //beginning x-value
    end = 50                  //ending x-value
}

define yAxis as Line {
    thickness = xAxis.thickness
    color = xAxis.color
    step = 20
    start = 0                 //beginning y-value
    end = 200                 //ending y-value
}

define title as Label {
    font = times              //times, arial, courier
    fontsize = 16
    text = "Class statistics"
    position = 24,30          //coordinates of upper-left corner of
                              //label, relative to canvas upper-left
}

//...some more components defined, such as a legend

myGraph.add(myCanvas)
myGraph.add(xAxis)

myGraph.scale(75%)           //make it smaller
myGraph.section(x1,y1,x2,y2) //get just a rectangular piece of it
                              //(n/a for pie graphs)

myGraph.render(data)
```



Summary

EZGraphs is a simple and easy to learn language intended to facilitate the process of creating graphs for people with very little or no programming background. We believe it is also very suitable for an academic environment as it is simple and intuitive enough for children to use and it might also help teach basic programming concepts. EZGraphs has a more organized structure than Ploticus and provides more design options than the graphs wizard in Microsoft Excel. It strives to be user-friendly with clear, unadorned syntax rules and informative error messages for easy debugging, so that anyone writing EZGraphs scripts finds the experience pleasant, effortless, just plain EZ.
