UNIVERSITÄT OSNABRÜCK

**Faculty of**
**Cognitive Science**

# Bachelor Thesis

Romeo: Scripting Environment with interactive Visualizations


**Author:** Simon Danisch
sdanisch@email.de

**Supervisor:** Prof. Dr.-Ing. Elke Pulvermüller

**Co-Reader:** Apl. Prof. Dr. Kai-Christoph Hamborg


**Filing Date:** 01.02.2014

# I Abstract

This bachelor thesis is about writing a simple scripting environment for scientific computing, with focus on visualizations and interaction. Focus on visualization means, that every variable can be inspected and visualized at runtime, ranging from a textual representation to complex 3D scenes. Interaction is achieved by offering simple GUI elements for all parts of the program and the visualizations. All libraries are implemented in Julia and modern OpenGL, to offer high performance, opening the world to scientists who have to work with large datasets. Julia is a novel high-level programming language for scientific computing, promising to match C speed, making it the optimal match for this project. -This section needs more work, and should probably be written in the end

# II Table of Contents

# III List of Figures

# IV List of Tables

# V Listing-Verzeichnis

# VI List of Abbreviations

# 1 Introduction

This Bachelor Thesis is about writing a scripting environment for scientific computing. It aims are to be intuitive to use, with deeply integrated visualizations and easy creations of GUI elements, suitable for large datasets and with a modular, open source core, which is easy to extend. The introduction is structured in the following way. I will start with describing the field of research with its challenges. From the challenges I will identify the ones, that I want to solve in this Bachelor Thesis. Finally this chapter will conclude with how a solution to the problem can look like, how to measure the success and give an outlook on the structure of the entire Bachelor Thesis.

## 1.1 Field of Research

The research field is developing convenient infrastructure for scientific computing. Offering high level software with nice user interfaces for scientific computing is a very difficult task, as state of the art research is a fast moving target. You cannot offer an easy to use software package, for something that hasn't been invented yet. If your research involves creating new algorithms for solving finite element problems, you cannot use the available packages for finite element solvers. What you can do though, is to reuse parts of the visualization and math infrastructure, to speed up development. So instead of writing monolithic software, the common approach is to offer modular libraries, together with a high-level scripting language and editor, to allow the researcher to quickly prototype software, which solves their unique problem. Famous examples are Matlab, Mathematica, R and Python, which have largely different focus areas, but all are commonly used tools by researchers, as they offer the previously mentioned functionality.

All tools have some way to build simple GUIs, visualize data, access a lot of algorithms, and the used languages are high-level, to let the researcher focus mainly on solving the problem.

In addition, research often includes working with huge datasets, or algorithms with a high time complexity. So, in addition to offering a lot of libraries and being easy to use, infrastructure for scientific computing must be really fast. The mentioned tools all have their own way of dealing with this demand, which I will describe in more detail in the chapter Background. As most high-level languages are slow, the common pattern to deal with this is to write performance critical libraries in another, low-level, high performance language. The mentioned tools fall into this class, which means if you want to extend their performance critical core, you will not be able to do so, unless you use a different more complicated language.

In summary, there are two main demands. First, high usability and readily available libraries, and a high performance of the infrastructure, as otherwise working with large datasets becomes infeasible.

## 1.2 Problem

You never get the previous demands in one package, in one language, open source. Divide Problems into more specific sub-topics.

## 1.3 Problem Solutions and Measurements of Success

* Write it in one, open source, high level, high performance language (Julia) * Write it in OpenGL * Open Source * Easy ways of creating GUIs

## 1.4 Outlook

## 1.5 Motivation and Problem Description

Figure 1: different visualizations of f(x,y,z)=sin(x/15)+sin(y/15)+sin(z/15), visualized with Romeo. From left to right: Isosurface, isovalue=0.76, Isosurface, isovalue=0.37, maximum value projection

When one tries to visualize a volume described by f(x,y,z)=sin(x/15f0)+sin(y/15f0)+sin(z/15f0), the limitations of the human mind becomes clear. While a computer doesn't have the creativity of humans yet, the task of visualizing this volume is a no-brainer for a computer (see Figure 1). In science, there are many comparable problems, which are unsolvable without the aid of a computer. For example making sense of one petabyte of data, predicting the folding of a novel protein, simulating black holes and many more. This naturally leads to computers being an essential part of state of the art research. Researchers have unique demands for their tools, which makes writing software for scientific computing an interesting research field. Consider, that researchers mostly do not have a background in computer science, some problems need a lot of computational power and due to the novelty of some problems hand tailored solutions are needed. In addition, high interactivity is a crucial feature, since a lot of research is a creative process which involves exploring

many alternative solutions. This bachelor thesis is about writing a scripting environment with advanced visualization capabilities, while keeping the previous mentioned demands of scientific computing in mind. The resulting demands for the scripting environment are as follows.

No background in computer science puts a strong constraint on the used programming language. It should be high level and easy to learn as researchers should not be forced to waste their time with fixing memory corruptions or obscure language constructs. The entire suite should be open source and written in one language, to allow the researchers to modify every part easily, enabling them to create a solutions which is hand tailored to their particular problem. The two constraints of writing even the core libraries in one language and having also research areas with high demand for computational power means, that in addition to being high-level the chosen language must also be fast. As an example, the high level language Python can be substantially slower than a high-performance language like Fortran. In extreme cases, Python is up to 1155 times slower than Fortran [**?**], which means if a computation needs 1 day with Fortran, it could take 3 years within Python, rendering the problem intractable within Python.

Finally, the scripting environment should offer simple GUI elements like sliders, to enable the researcher a simple way to interact with the parameters in his script.

These considerations lead to the 3 main design choices: Julia, a novel scientific computing language, is chosen as the main language, Modern OpenGL for high performance graphic rendering and writing generic,Reusable open source code for a high amount of customizability. The feature set consists of parametrized scientific visualizations, simple GUIs for the parameters and a simple script editor.

-This section needs more quotation, to integrate it into an ongoing research problem. Also, some formulations have to be rewritten.

## 1.6 Used Technologies

### 1.6.1 Julia

### 1.6.2 OpenGL

## 1.7 Similar Work

### 1.7.1 Other Languages

### 1.7.2 Ipython Notebook

### 1.7.3 Matlab

### 1.7.4 Mathematica

### 1.7.5 Other Graphic acceleration APIs

# 2 Background

# Appendix

# A GUI

A nice Appendix.

## Screenshot

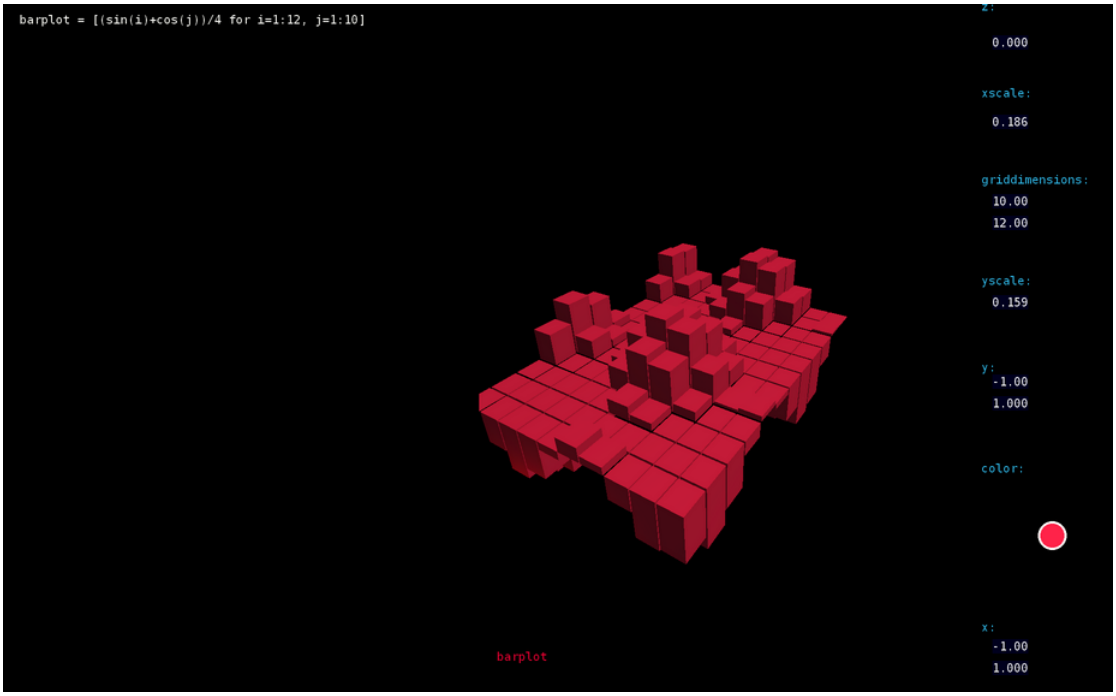


Figure 2: Screenshot of the prototype. Left: evaluated script, middle: visualization of the variable barplot, right: GUI for editing the parameters of the visualization

# Official Statement

I hereby guarantee, that I wrote this thesis and didn't use any other sources and utilities than mentioned.

Date: .....

(Signature)