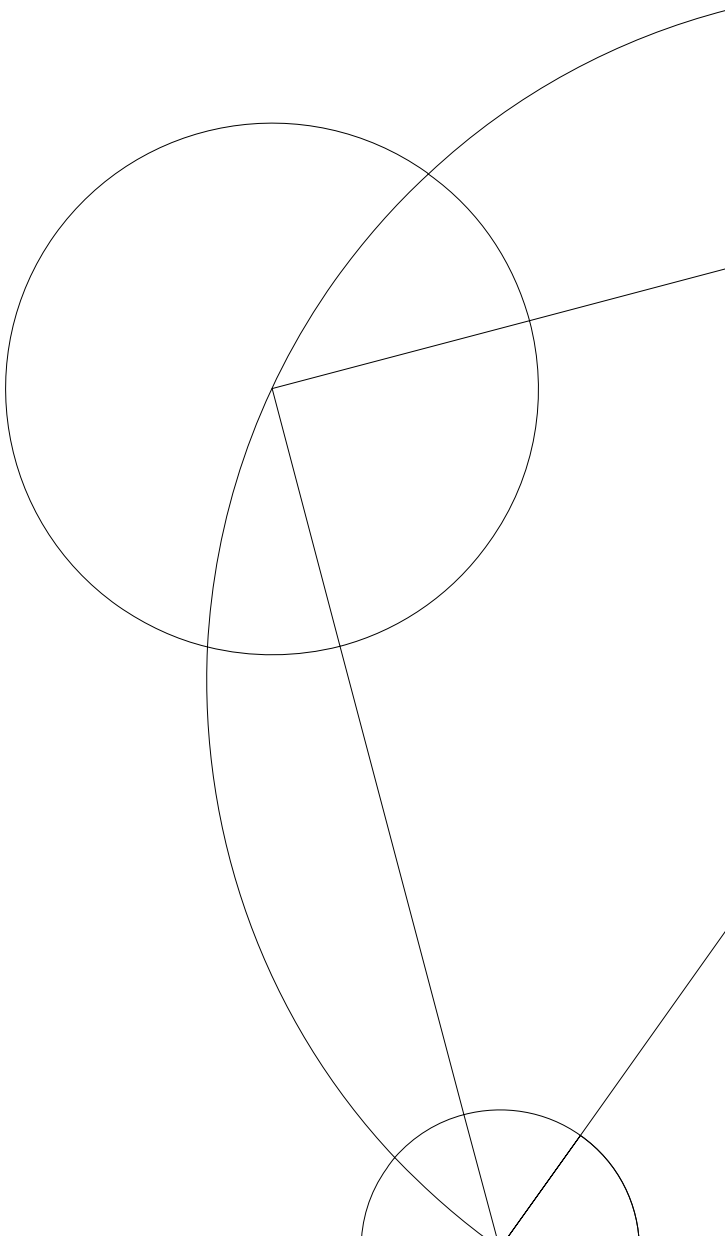
UNIVERSITY OF COPENHAGEN
Department of Computer Science (DIKU)

Project Outside the Course Scope

Closure Plots for Basic Arithmetics

Amr El Sayed
KuId - vwj159

August 22, 2016



Closure Plots for Basic Arithmetics

DIKU - Project Outside the Course Scope

Students Names	:	Amr El Sayed - VWJ159
University	:	University of Copenhagen
Institution	:	Department of Computer Science (DIKU)
General Supervisor	:	Michael Kirkedal Thmosen
Practical Supervisor	:	Oleksandr Shturmov
Period	:	Block 5
Year	:	2016
Pages	:	
Github	:	`https://github.com/Amr116/ClosurePlots`

Certificate

This is to certify that the work contained in the thesis entitled "Closure Plots for Basic Arithmetics" by Amr El Sayed has been carried out under our supervision and that this work has not been submitted elsewhere.

General Supervisor
Michael Kirkedal Thmosen
m.kirkedal@di.ku.dk
Department of Computer Science (DIKU)
University of Copenhagen

Practical Supervisor
Oleksandr Shturmov
oleks@di.ku.dk
Department of Computer Science (DIKU)
University of Copenhagen

1 Abstract

Resume

Contents

2 Introduction

This paper covers the development process of Closure Plots for Basic Arithmetics on those aspects of floating-point that have a direct impact on designers of computer systems, and is intended for readers who have studied Computer Science or similar, or are simply interested in the field of IT. Readers are assumed to have basic programming knowledge and some familiarity with web development.

The purpose of this project is to create a visualization tool for illustrating the behaviour of basic arithmetic operations (e.g., Addition, Subtraction, Multiplication, Division) in floating point arithmetic with different precision (e.g., Half, Single, Double, Quadruple).

Floating-point arithmetic is considered an esoteric subject by many people. This is rather surprising because floating-point is ubiquitous in computer systems. Almost every language has a floating-point datatype, computers from PCs to supercomputers have floating-point accelerators, most compilers will be called upon to compile floating-point algorithms from time to time, and virtually every operating system must respond to floating-point exceptions such as Overflow, Undeflow, Infinity and Inexpressible.

In computing[1], floating point is the formulaic representation that approximates a **real number**, that include all the rational numbers, such as the integer -5 and the fraction 4/3, and all the irrational numbers, such as $\sqrt{2}$ (1.41421356..., the square root of 2, an irrational algebraic number). Included within the irrationals are the transcendental numbers, such as π (3.14159265...), so as to support a trade-off between range and precision. A number is, in general, represented approximately to a fixed number of significant digits (the significand) and scaled using an exponent in some fixed base.

The most popular code for representing real numbers is called the IEEE Floating-Point Standard, that have three basic components: the sign, the exponent, and the mantissa. The mantissa is composed of the fraction and an implicit leading digit. The exponent base (2) is implicit and need not be stored.

This project is based on the principle of float in the floating point, which is derived from the fact that there is no fixed number of digits before and after the decimal point; that is, the decimal point can float, therefore the implementing of this project focuses on create a visualization tool for illustrating the behaviour of basic arithmetic operations (Addition, Subtraction, Multiplication, Division) to integer number with different precision (Half, Single, Double, Quadruple), and also create the infrastructure that will allow for future development to representing different code for floating point eg. IEEE Floating-Point Standard or something else on this approach.

This report is divided into four parts (Analysis, Design ,Implementation and Testing).
For the analysis part, I will discuss the requirements that are needed to implement the project and compare to some of the technology, which can achieve the goals of this project, and to show the strengths and weaknesses of this technology.

For the design part, I will introduce tools and techniques that have been chosen to im-

[1]Computing is any goal-oriented activity requiring, benefiting from, or creating a mathematical sequence of steps known as an algorithm — e.g. through computers. Computing includes designing, developing and building hardware and software systems, processing, structuring, and managing various kinds of information, doing scientific research on and with computers, making computer systems behave intelligently, and creating and using communications and entertainment media.

plement the project and to clarify whether it is possible to achieve all the objectives of this project through the chosen technologies.

For the implementation part, I will explain the method of execution to achieve the specification of this project based on the design part, and i will explain each function and method separately.

For the test part, I will an investigation conducted to provide with information about the quality of the product (application) with the intent of finding bugs (errors or other defects), and It would achieve the benefit of the development in the future.

3 Analysis

4 Design

5 Implementation

6 Future Development

Footnotes

- http://steve.hollasch.net/cgindex/coding/ieeefloat.html