UNIVERSITY OF COPENHAGEN
Department of Computer Science (DIKU)

Project Outside the Course Scope

Closure Plots for Basic Arithmetics

Amr El Sayed
KuId - vwj159

September 8, 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

I am a third year computer science student at the University of Copenhagen, I'm enrolled in the undergraduate (bachelor) part of the education. This report is the final product of a 1-block Project Outside the Course Scope. 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.

2.1 Purpose

The purpose of this document is divided into two purposes:

The first purpose is to describe the UI to Closure Plots for Basic Arithmetics application for instructors and Computer Science students at Department of Computer Science - University of Copenhagen, so they can easily interact with the application.

The second purpose is explain the requirement specifications for Closure Plots for Basic Arithmetics application and explain the steps of implement this project to help the developers to further developing this application.

2.2 Intended Audience and Reading Suggestions

This document 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 technologies.

2.3 Project Scope

The software to be produced is a Closure Plots for Basic Arithmetics, which will be referred to as "CPBA" thorough this document.

CPBA will allow students to learn about the behaviour of basic arithmetic operations in floating point arithmetic with different precision through interaction with UI. The CPBA will also allow the instructors and creative students to modify the variables in arithmetic operations to monitor behaviour of floating point.

The objective of CPBA application is to be 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).

3 Overall Description

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.

3.1 Product Perspective

A simple way to illustrating the behaviour of basic arithmetic operations, that is by someone sitting physically at a computer or with paper and write down the numbers and finish this calculation process. There are tools available, commercial or open source that support arithmetic operations to digits or binary numbers. Many Operating Systems and applications themselves come with calculator support those basic arithmetic operations. But unfortunately, all these tools usually require a person sitting physically at a computer writing down the numbers one by one.

Thus the need for a web based application to be a visualization tool for illustrating the behaviour of basic arithmetic operations.

3.2 Product Features

The software is used mainly to illustrating the behaviour of basic arithmetic operations in floating point arithmetic with different precision of web based software so the user who is maintaining can access it from virtually any where.

This software 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.

4 Analysis

In this section we will be analysing the problem described above. The analysis will be focused on narrowing in special areas of the problem domain that will be central to designing and implementing a working prototype.

[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.

4.1 Different precision

Different precision in floating point is one of the obvious shortcomings when designing the application, if we could not represent those precision with their actual digit length.
We can see the mathematical notation to those precisions as:

- Half-precision is 2^{16}

  The positive numbers for this precision is range between 0 to 65535.

- Single-precision is 2^{32}

  The positive numbers for this precision is range between 0 to 4294967295.

- Double-precision is 2^{64}

  The positive numbers for this precision is range between 0 to 18446744073709551615.

- Quadruple-precision is 2^{128}

  The positive numbers for this precision is range between 0 to 340282366920938463374607431768211455.

But, does programming languages represent those precisions such as the way in which it represented in mathematical notation?
Some programming languages have maximum length of digit with different precisions, and if those digits has overflow the maximum length, then it representing in scientific notation or in rounding format[2]. This scientific notation will require redefining the length of those digit to be able to participation in arithmetic operations, otherwise we will get wrong results.

The below table4.1 showing represent digit length for different precisions to different languages. From this table we can see the similarities and differences in the representation of length for different precisions. This difference will be among the main reasons for choosing the design and implementation of this project.

DP \ PL	JavaScript	Python	GO
Half	65536	65536	65536
Single	4294967296	4294967296	4.294967296e+09
Double	18446744073709552000	18446744073709551616	1.8446744073709552e+19
Quadruple	3.402823669209385e+38	340282366920938463463374607431768211456	3.402823669209385e+38

Table 1: Representation of maximum digit length with different precisions at some programming language

4.2 Viewport Size vs bit Pixel

The browser viewport is the size of the rectangle that a web page fills on your screen. It's basically the size of the browser window, less the toolbars and scrollbars. It's the bit of the screen we are actually using to show the webpage.
Is it possible the CPBA application to illustrating the behaviour of basic arithmetic operations in the viewport size !?

[2]Rounding means making a number simpler but keeping its value close to what it was.

We want to calculate arithmetic operations (e.g. Addition, Subtraction, Multiplication, Division) to integer number with different precision (Half, Single, Double, Quadruple), and to do those calculations we need at least two variables, that is mean if variables X and Y have Single precision, so we will need to represent the result of this arithmetic operations, which will give us $65536^2 = 4294967296$ elements.

This huge number of elements can not be shown in the Viewport while maintaining achieve a clear vision for each element separately. So we need some way to allow us to achieve a clear vision for each element and also allows the user to take advantage of this visualization tool for illustrating the behaviour of basic arithmetic operations.

5 Design

To make this application easily accessible for the students, I decided that it should be web-based: meaning that it should be accessed via internet by using a standard web browser or students can get to the project files from a physical source (e.g. flash memory). This way there would be no need to distribute the program and all of the required packages for it to run, which we would need if i had to make a program that was not web-based. By making the application web-based, I ensure the accessibility of it for most, if not all, users. The solution will not target mobile/tablet users specifically.

Since this project should be web-based application, so i need to use HTML and CSS, because they are two of the core technologies for building Web pages, but to build the functionality of this project i can choose between many programming languages, but only JavaScript is the only ones that fit the CPBA requirement, because users can get access to CPBA via internet or offline if they get the project like physical source.

The above section (Analysis) I've discussed some of the problems that will confront us in the design and implementation of this project, therefore in the below subsection, i will representation of these problems and explain my ideas for the design and implementation of the solution to these problems.

5.1 Arithmetic operations & Different precision

The user need a simple way to illustrating the behaviour of basic arithmetic operations (Addition, Subtraction, Multiplication, Division) with different precision (Half, Single, Double, Quadruple), therefore the project should give the possibility for user to select and choice between different arithmetic operations and different precision. For this reason, i decided the front-end design of CPBA should include two <select> tags, which create a drop-down list, and thereby I collect user input from the selected option.

5.2 Mathematical notation

where have been chosen javascript as primary programming language to the functionality of this project, because it will provide access possibility of user via online or offline, but does Javascript represent precisions such as the way in which it represent in mathematical notation. Unfortunately, The answer is No, because the maximum integer can be represent of JavaScript is $2^{31} - 1$, or 2147483647, otherwise the integer will be represent in scientific notation. That's why I decided to make CPBA is limited to precisions Half and Single only.

5.3 Elements

To represent precisions Half and Single, that will give huge number of elements, which need to be shown in viewport, but to represent all those elements at once will not give the possibility to users to see and illustrating the behaviour of basic arithmetic operations, therefore in CPBA I need to summarize the those elements to miniature form.
This form differs from precision to other precision in the number of stages. In other words for example in addition operation for precision Half, we have 2^{16} element for variable X and 2^{16} for variable Y, so the sum of those two variables is 2^{32} elements.
I decide to show 256 elements of this result at each stage, where each element represents 4 edges from the real results as shows in figure 1.

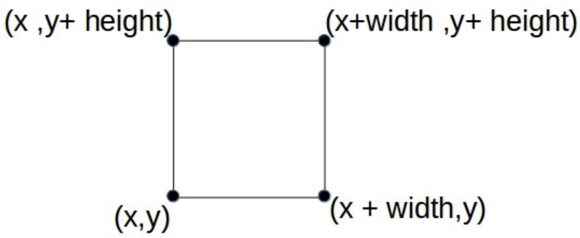

Figure 1: summarize form shows how each element represents in CPBA

The precision Half will have four stages. At the first stage width and height will be equal to 4095, at the second stage will be equal to 255, at the third stage will be equal to 15, and at the fourth (last stage) will be be equal to 0.

The below examples is Illustrative examples to prove of concept in addition operation of half precision at first stage form:

X = 4096, Y=53248

Leftmost bottom:	4096 + 53248	= 57344	,57344 is less then 2^{16}	Correct
Rightmost bottom:	(4096 + 4095) + 53248	= 61439	,61439 is less then 2^{16}	Correct
Leftmost upper:	4096 + (53248 + 4095)	= 61439	,61439 is less then 2^{16}	Correct
Rightmost upper:	(4096 + 4095) + (53248 + 4095)	= 65534	,65534 is less then 2^{16}	Correct

There are 4 correct and 0 Overflow, therefor i will show single Correct element represent this above range.

X = 4096, Y=57344

Leftmost bottom:	4096 + 57344	= 61440	,61440 is less then 2^{16}	Correct
Rightmost bottom:	(4096 + 4095) + 57344	= 65535	,65535 is less then 2^{16}	Correct
Leftmost upper:	4096 + (57344 + 4095)	= 65535	,65535 is less then 2^{16}	Correct
Rightmost upper:	(4096 + 4095) + (57344 + 4095)	= 69630	,69630 is more then 2^{16}	Overflow

There are 3 correct and 1 Overflow, therefor i will show single Mix element represent this above range.

X = 4096, Y=61440

Leftmost bottom:	4096 + 61440	= 65536	,65536 is more then 2^{16}	Overflow
Rightmost bottom:	(4096 + 4095) + 61440	= 69631	,69631 is more then 2^{16}	Overflow
Leftmost upper:	4096 + (61440 + 4095)	= 69631	,69631 is more then 2^{16}	Overflow
Rightmost upper:	(4096 + 4095) + (61440 + 4095)	= 73726	,73726 is more then 2^{16}	Overflow

There are 0 correct and 4 Overflow, therefor i will show single Overflow element represent this above range.

5.4 Forward, Back & Reset

Wrtie something about back and reset buttons

6 Implementation

In this section, I will explain the steps involved to implement CPBA application. The Implementation steps is split up into two different parts:

- UI
- Back-end

7 Testing

8 Conclusion

Footnotes

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