Kandidatarbete vår 2016

Dennis Jönsson, 87062238337
Jonathan Skårstedt, 8510252912

Data structure and algorithm design and visualization tool

Background

In recent years many investments as well as public and political actions have taken place in order to strengthen and broaden the field of computer science in higher education, as well as to introduce computer science to lower levels of education. More and more people are coming in contact with programming through their jobs or educational program, and the public awareness and ability to program a computer is becoming increasingly important. This has not only increased the popularity of programming in education among common students within computer science, as well within minority groups, but has also brought on the need for more efficient and easy to understand tools for bringing programming to the masses.

Project description

A lot of new software, like Scratch or Appinventor, with the aim to visualise code flow and introduce beginners to programming have been emerging lately. However, mapping complex data structures and their manipulations to visual representations is lacking both attention and concrete implementations. This kind of research is something we believe programming education would greatly benefit from.

We want to create a tool which helps programmers better visualize data structures and how they are being manipulated by algorithms. The idea is to create a visualization tool which, given a small enough program written in a common language such as Java or Python, creates a graphic representation of specified data structures and how they are being manipulated in different steps of an algorithm.

The tool will consist of three parts:

- Generic markup language which specifies what is to be graphically visualized.
- Web based visualization tools, which given the markup languages, draws a correct representation of the data structures and manipulations, specified in the markup language.

- Third party libraries for common programming languages which, by using annotations, helps the programmer specify what data structures to visualize, and which also produces a markup language that can be read by the visualization module.

We want to start by focusing on a subset of the Java programming language, namely primitive types and arrays. We choose this more primitive representation as it will ease the problem of communicating the data structures to our visual representation. The markup will most likely be generated from an AST of the Java code. That is, there will be a module for generating the AST and then generating the markup from the AST within the third party library.

The final product might make it easier for users to visualise complex behaviours in the algorithms they are developing. By simply annotating the data structures of interest in code, a user will be able to view how the execution of the code affects those data structures, and more easily determine whether an algorithm is working according to its specification.

Literature

Arne Ranta: *Implementing Programming Languages. An Introductions to Compilers and Interpreters*
WBGL, js