# Teaching Interactive Computer Science

## Group 7

Erik Skogby
Alexander Kjellén
Björn Delin
Jan-Erik Bredahl
Joakim Israelsson

**Who are the users and what problem does the system solve for them?**

For beginners in computer science and programming, it can be hard to understand how various data structures and algorithms really work. To help people improve their understanding of these things, we shall develop a visualization tool to help demonstrate algorithms and data structures visually. The main users are students studying computer science and are beginning to learn programming, and others who want a deeper understanding of how some algorithms and data structures work.

Furthermore, since it is open source software it is also possible for teachers and other professionals to extend the program by adding more data structures and/or algorithms. So, if a teacher wishes to visually demonstrate a data structure that is not available, then he can create the visualization himself. Note that the main users for this particular feature are advanced programmers.

**The main uses of the system**

What this program does is visually demonstrating algorithms, when using a particular data structure. The user will be able to load a data structure and the accompanying algorithms and functions from a menu. The data structure will then be visually represented on the screen, in its current state. The user can then run the functions belonging to the data structure. The visualization will then be animated to display how the function works, how objects interact with each other and how objects are created and destroyed. The objects will move around, numbers will be updated and pointer arrows will change target.

For example, Bob is a 19 year old computer science student in college. He has to understand how the quicksort algorithm works in order to be able to complete his homework which is due tomorrow. He has read the literature and has gotten himself a clue on how it works, but is not ready to implement it himself yet. His teacher gave them a link to this software tool and said that they could check out the array data structure if they had trouble understanding how the sorting algorithm works. So Bob downloads the program, starts it up and loads the array data structure. He is asked by the program to fill in some info about the array, such as size and content. He chooses to fill the array with twenty randomly generated numbers. At this point, he is shown a list of available functions including, but not limited to, insert, delete, mergesort and quicksort. First he fills the array with 20 random numbers using a supplied function. Then he chooses to run the quicksort function, which does not take any parameters. The visualization of the array now begins animating. Pointer arrows move around and numbers are swapped as the array is sorted. First he views the whole sorting once. Then he restarts the animation, but this time he pause it and steps through it, while following the algorithm in his book. After using the tool and the book together like this for a couple of times give him the understanding of the quicksort algorithm that he needs.

It is possible for advanced users, such as computer science teachers, to use this program as a presentation tool while teaching students how data structures and algorithms work. Here is another example:

Walter is a CS teacher that wants to explain to the class how a binary tree works. He brings his computer to the class but goes over the structure on the whiteboard first to explain how it works, and what it can be used for. Now he connects his computer to the projector, starts the program and loads the Binary Tree. The structure gets drawn up so that all students can see how it looks like in a graphical way. He calls the function “insert” and in the animation window objects begin to move around, displaying how pointers get redirected and values changed. When inserting another value into the tree he chooses to step through the animation and for each step explain what is happening. Then Walter goes over a few other functions such as find and delete in the same way.

**The context/environment in which the system is to be used**

This software is targeted at an academic environment. It will be used on computers running Java. It will be run as a regular application on the user's computer. It is not meant to be run on a server or something else.

**The scope of the system**

This tool is not intended to replace teachers or literature, but to complement them. Therefore we will not include extensive documentation on the theory of these data structures. We will however have some documentation on how each function works.

| ***Topic*** | ***In*** | ***In if possible*** | ***Out*** |
|---|---|---|---|
| Show objects and how they relate to each other visually. | X | | |
| Animate this visualization when functions are executed on it. | X | | |
| Ability to control the animation speed, pausing it, reversing it, etc. | | X | |
| Show the algorithm code. | | X | |
| Code editing in the tool. | | | X |
| Manipulate visualized data structures through supplied functions. | X | | |
| Manipulate visualized data structures directly, without using supplied functions. | | | X |
| Automatic positioning of objects in the visualization. | | X | |
| Extract code documentation from data structure source files. | | X | |
| Make our own compiler or interpreter. | | | X |
| Ability to zoom in or out in the visualization. | | X | |

| ***Topic*** | ***In*** | ***In if possible*** | ***Out*** |
|---|---|---|---|
| Ability to add new graphical symbols. | | | X |

**The main factors that need to be taken in to account when designing and building the system**

- Many of the users of our program do not know how the data structures works. However, they will most probably have someone to ask who knows them well.
- The visualization has to work in a pedagogic and clear way so that it does not confuse people more than it helps them.
- The program should be able to run on almost any computer with Java installed.
- An error in an animation might cause students to learn the structure incorrectly.

**Technologies & Risks**

The technology we will use is primarily Java version 1.5. The newest one is 1.6, but it is not supported on all platforms, most notably the Mac. For the custom graphics part, we will use Java2D. We are fairly confident that Java2D is up to the task, performance wise. Still, there is a slight chance that Java2D might not be able to handle the graphic complexity we'll throw at it. In this case, we will simply reduce the graphical complexity of our software.

Another risk is that we have a pretty vague idea on how complex our project really is. This gives us a fairly high risk of underestimating the complexity. Our strategy here is to have a priority on each feature, and if the project gets too complex, start dumping features until we can handle it.

There is also a pedagogical risk here. Our visualizations need to help more than they confuse. If they don't, then there is no reason to do this software. Our estimate is that there is a low to medium chance of this happening. Since this is simply not an option, we will have to test the system on people unfamiliar with the subject during the development cycle. We might also contact some HCI (Human-Computer Interaction) people to help us on this one.

For the project to be useful, we need to have a certain amount of data structures and algorithms to show. One risk is that we might not have enough to show that people will find useful. In this case, we need to increase the complexity of the software, which might conflict with the risk above, which the complexity can get out of hand. Our strategy here is to minimize this risk by making it as easy as possible for us to add new content. This will be a high priority when designing the system.