Prof. Dr. Gerhard Reinelt
Dipl.-Math. Marcus Oswald
Dipl.-Inf. Dino Ahr
Institut für Informatik
Universität Heidelberg
http://www.iwr.uni-heidelberg.de/iwr/comopt/

VisAlg - Visualisation of Algorithms User-Manual

July 11, 2001

Contents

1 The User Interface

The user interface is divided into four sections, shown in 1. There is the menu bar, containing the items **File Edit**, **History**, **Project** and **Help** and various module menus, which are described below, the project window, which is located at the upper left part, the history window at the lower left part, and the working or module window at the right part. The seperator between those two windows can be moved.

1.1 The Menu Bar

1.1.1 The Menu File

The Menu **File** contains the items :

- **New Project**
- **Load Project ...**
- **Save Project ...**
- **Exit**

The item **New Project** resets VisAlg into its initial state. Up to now, there is no confirmation asked, so this item has to be used carefully.

The user can load a project using the item **Load Project ...**. The default project path is ./projects.

Using **Save Project ...** the user is able to store the current project to disk.

Exit will close the application.

1.1.2 The Menu Edit

Up to now, this menu contains just one item **Edit Preferences**. If choosing this item, the preferences dialog will pop up (see 2). There are two tabs in this dialog. The first one, "Play delay time", allows the user to set up the minimum and maximum delay time (in ms) for the "Play" function (see section 1.3). These values are chosen by moving the slider to the left respective to the right side. The second tab, "Undo size", is used to set the number of project states, that are saved in the undo history. These states can be accessed by selecting the "undo" functio (see section 1.3). It is recommended to set this value to "0", since the undo functionality takes up a lot amount of CPU time and memory.

1.1.3 The Menu History

The following items are located in the History menu :

Load project state ... This item loads the project state, that is selected in the History window (see section 1.3).

Save project state ... Saves a project state.

Delete selected project Deletes the selected project state.

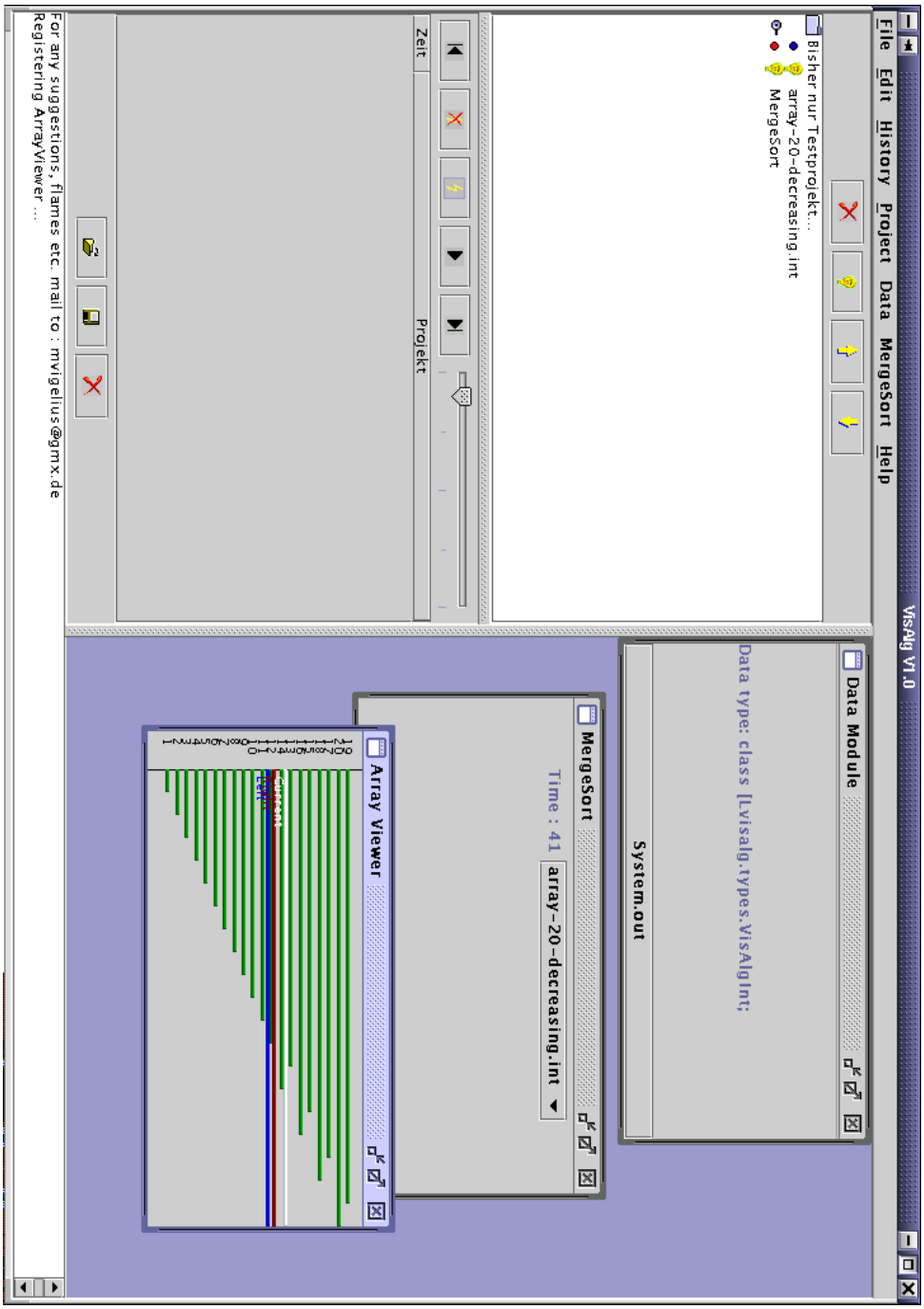

Figure 1: The Main Window

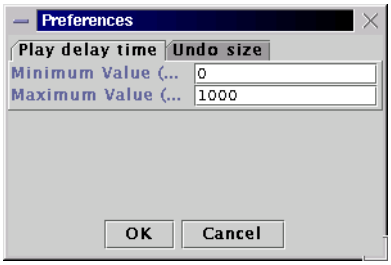

Figure 2: The Preferences Dialog

Load previous project state Loads the project state, that is previous to the selected one.

Load next project state Loads the project state, that is next to the selected one.

Step project Lets all active modules perform one step.

Undo If the undo history size is set, this item undos the last step.

1.1.4 The Menu Project

The Project menu consists of these items :

Add module ... This item lets a dialog pop up, where the user can choose one module out of a list of all available modules. This module will then be added to the module list in the Project window (see section 1.2).

Delete module ... Deletes the selected module. The user will be asked to confirm the deletion.

Toggle module on/off The selected module will be set either "active" or "inactive".

Module up The selected module will be moved up in the module list.

Module down The selected module will be moved down in the module list.

1.1.5 The Menu Help

Two items are located in this menu. The first one, **Info** gives out some version information into on the standard output, the second one **Manual ...** starts the VisAlg help system.

1.1.6 The Module Menus

Depending on the loaded modules, there might be various module menus. These can be used to access some module functions.

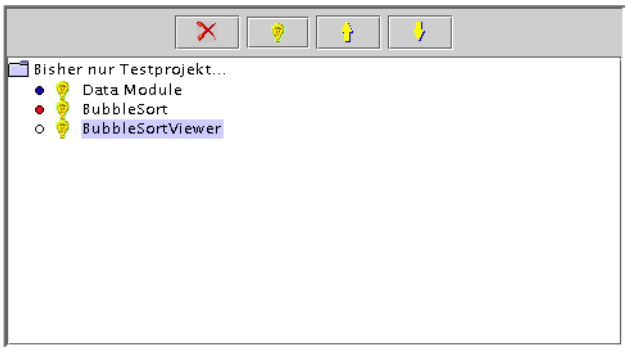

Figure 3: The Project Window

1.2 The Project Window

The Project window is shown in figure 3. At the top part of the window, there are four icons, each of them representing an item in the Project menu (see section 1.1.4). The red cross deletes the selected module, the light bulb toggles its state to "active" or "inactive" and the arrows move the selected module up or down.

The main part of the Project window is the list of loaded modules. The uppermost entry represents the project itself, the branches stand for the modules. Data modules have a blue circle icon, algorithm modules a red one and viewer modules are visualised by a white circle. Submodules are located in another branch connected to the associated module itself. The light bulb on the left side shows, if the module is active or not.

Modules can be selected by clicking on the project tree. Using the Shift or the Control key, multiple selections are possible. By right clicking on a selection, a popup menu appears, that shows the entries in the Project menu (see section 1.1.4). By double-clicking on a module, the associated window pops up in the Module Window (see section 1.4). The window appears by default in the upper left corner, so it might be possible, that new windows overlap old ones, which then are not visible anymore.

1.3 The History Window

The History window can be seen in figure 4. At the top part, there are five icons and a slider. The "CD-Player" icons can be used to browse through the list of saved project states, or to start "playing". If the user presses the play button, VisAlg will continously step all active modules. The delay time between each step can be chosen by using the slider. Also, the user can invoke a single step by pressing the lightning button or undo the last step by pressing the crossed lightning button.

The main part shows a list of all saved project states. By double-clicking on the list, the comment can be changed for this state. If the user clicks one time, an entry will be selected, allowing the user to apply the operations described in the section 1.1.3. This menu can also be received by right clicking on an entry.

There are three items located at the bottom part of the window. They allow the user to save the current project state, to load the selected project state or to delete the selected project state.

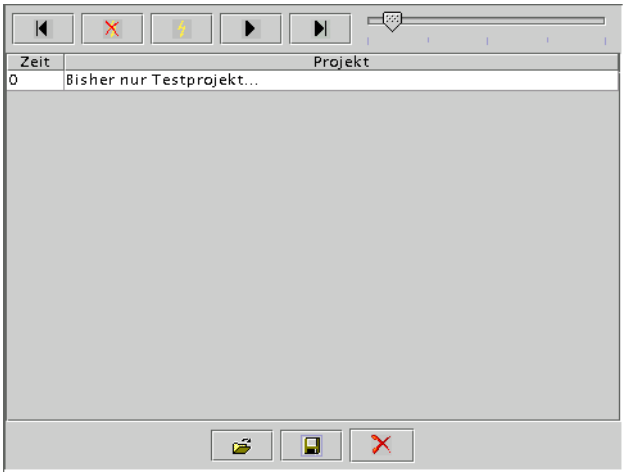

Figure 4: The History Window

1.4 The Module Desktop

The Module desktop takes the right part of the main window. All module windows are located in this part. The user can close, move or resize each window. VisAlg remembers the position of each window, so it will be re-opened at the last location.

2 Modules

This section describes the available modules and their usage.

2.1 Data Module

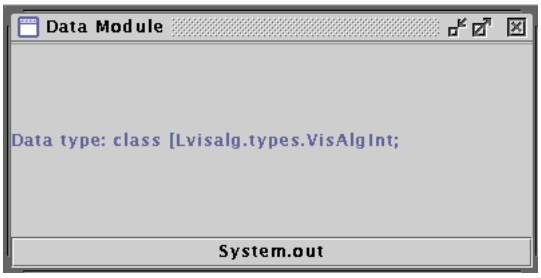

Figure 5: The Data Module

2.1.1 Concept

There are two possibilities of how data, that should be handled by VisAlg, are available :

- It concerns data in a random format, like an ASCII file containing numbers or text.
- It concerns data, that have already been processed by VisAlg, and were then stored to disk.

Every new data, that we want to process using VisAlg, has to be "Introduced" to VisAlg first. Achieving the best flexibility on handling of data, there is an own converter for every format of input data. A converter is a class, written in Java (or made available to Java), that has to implement a specified Interface Converter, and is available, to read a certain data format. For deeper understanding of how converters work, have a look at the *Programmers Manual*.

In order to tell VisAlg, which converter to use for a given file, there are two alternatives :

- The classname of the converter is specified in the first line of the input file using an exclamation mark, e.g. `!visalg.converter.ArrayOfIntConverter`.
- VisAlg recognizes, if no converter is specified, and asks the user to enter the class name of the desired converter.

Up to now, there are two converters, which are able to construct either an array or a matrix out of a sequal of integer numbers, seperated by commas. These converters are called *ArrayOfIntConverter* respective *MatrixOfIntConverter*. There is another converter available, that converts a sequence of double numbers into a matrix, called *MatrixOfDoubleConverter*.

2.1.2 The Data Module Window

If the window of the Data Module (see figure 5) is opened, the main menu bar contains a menu **Data** with the items textbfNew Data ..., textbfLoad Data ..., textbfSave Data and **Open Viewer ...**. The bottom part of the window shows the type of the data, that are currently loaded. There also is a button textbfSystem.out, that writes the data into the standard output.

2.1.3 Functionality of the Data Module

By selecting textbfNew Data ..., the user can choose a file (in the directory ./import, by default), containing the data, that are to be imported. This function has to be used, if data should be processed by VisAlg for the first time. If no converter is explicitly given in the input file, the user will be asked, to enter the file name of the converter.

Example given : If one wants to read in the file *integer-test.int*, one will be asked, to specify the converter. The associated converter for this case is given in the file *ArrayOfIntConverter.class*, which can be found in the path *./src/visalg/converter*.

If the data has been read in once, it can be stored using an internal binary format, by selecting textbfSave Data .... These data are stored in the directory ./data by default and can be reloaded by selecting textbfLoad Data ....

If the user chooses **Open Viewer ...**, a new dialog pops up, which allows the user to open a viewer module, that is capable of showing the current data.

2.2 Module for Selection Sort

2.2.1 SelectionSort Module

This module represents the Selection Sort algorithm. The associated window for this module shows, at which step the algorithm is. It lets the user also choose, which data module should be used for this algorithm.

2.2.2 Selection Sort Viewer

This module looks for the currently active Selection Sort module and visualises the algorithm.

2.2.3 Background Information

This module implements the SelectionSortAlgorithm. The SelectionSort algorithm is suitable for the sorting of big data sets, in that the swapping of values takes a lot of time, but not the comparing.

Selection Sort looks for the smallest element in the set and swaps it with the first one. After that, the smallest element of the not yet sorted set is swapped with the second one. It is necessary for that, to use two interlocked loops. The outer one provides the position of the element, that is to be swapped next, the inner one looks for the smallest element of the unsorted data set.

Due to the high number of comparisons, the runtime of this algorithm is $O(N^2)$. The swapping commands, however, take $O(N)$ memory, so that this algorithm is suitable for complex data sets.

2.3 Module for Bubble Sort

This module implements the Bubblesort algorithm as a VisAlg module. Bubblesort is a simple, but not very efficient ($O(N^2)$) sorting algorithm. Starting with the first two values, two number are compared and swapped if necessary. After one swapping, the algorithm starts at the beginning until no swappings are done anymore.

Because after the first pass the biggest number is at its place, it does not have to be compared again. The same is after the second pass and so on. The worst case scenario for Bubblesort is, if the numbers are sorted exactly opposite. In this case, the algorithm takes $O(n^2)$ time. The name of this algorithm results of the fact, that the numbers ascend like bubbles.

2.4 Module for Merge Sort

This module implements the "Merge Sort" algorith. MergeSort provides the possibility of sorting data sets, that are much bigger, than the available workspace. The set of data is divide in two or more seperate sets, that are sorted by a random algorithm. If all sets have been sorted, they have to be merged again. For this, the first two elements of each set are compared, and the smaller one will be written into the target file. If the end of one set has been reached, the other set has just to be appended on the target file. For this algorithm, it is just necessary to keep two datas in the workspace.

The average as well as the maximum time of the MergeSort algorithm is $O(nlogn)$.

2.5 Module for Quick Sort

This module implements the QuickSort algorithm. QuickSort is one of the fastest sorting algorithms. The runtime in the worst case is $O(N^2)$, in the best case $O(NlogN)$. The average runtime is hardly worse than the best one.

The set is divided in two subsets. This is done by choosing one Pivot element and copying all elements, that are smaller or equal this Pivot element into the left subset. After that, there are in the right subset just elements, that are bigger than the Pivot element.

The Pivot element can be chosen randomly at the beginning. If one chooses the element at the right border, then one can let run two pointers from left to right respective from right to left. If

the left pointer has found an element, that is bigger than the Pivot element and the right pointer has found one, that is smaller, these elements are swapped.

The two pointers will meet at an element, that is equal or bigger than the Pivot element. This element is swapped with the Pivot element, and is chosen as the new Pivot element. QuickSort is then called for each, the right subset and the left subset.

QuickSort is a recursive algorithm. This implementation, however, is not recursive. A non-recursive algorithm must use a stack in order to remember the subsets, that are to be sorted. This implementation does not use a stack either, and is by that not a real QuickSort algorithm ! It does treat the left subsets correctly, but will set the right border of right subsets to the end of the data set and by that perform more steps than necessary. Additionally, the end condition is not correctly. It can, however, demonstrate the partition algorithm.

Because the implementation of the algorithm is quite "spagetti-code" like, a basic-like description can be found in the file "QuickSort.basic".

2.6 The Matrix Tool

After activating the MatrixEditor in the Matrix-Tool window by clicking on the button that reads "Editor" you can be sure that you are looking at what this documentation deals with.

At first you can choose the MatrixEditor mode. You do that by choosing a task in the upper ComboBox. There are five modes:

1. Exchange rows I and j (default)
2. Move row I
3. Exchange columns k and l
4. Move column k
5. Modify entry (I,k)

Way to proceed:

In mode (1) and (3) you choose two rows or columns. You can do that by inserting the row or column numbers into the TextFields, by using the buttons or by clicking the mouse on the desired row or column position in the "Matrix" window. If you use the buttons or the mouse you have to fix the first row or column with the button "Select" after you have chosen it. If you use the TextField you just press RETURN after your choice. The rectangle around the selected row or column will switch from blue to red now. After you chose the second row or column the same way you can exchange your rows or columns with the button "Exchange". The Button "Back" will deselect the rows or columns if you have selected something.

In mode (2) and (4) you choose one row or one column. Like described before the navigation can be controlled by the TextField, the buttons or the mouse. Click on the buttons "^" and"v" and the selected row will move up and down in mode (2). Or in mode (4) the buttons "¡" and "¿" will move the selected columns left and right. With the button "OK" you accept the transformation, with the button "back" you abort it.

In mode (5) you choose one single entry of the matrix. Once again you can navigate with the TextFields, with the buttons or with the mouse. In the TextField named "value(I,k)" the value of the chosen entry is displayed. After you selected the entry with the button "Select" you can insert a new value. But because there can be just one data type for each matrix, the new value must have the type that is defined in the header row of the MatrixEditor window.

Figure 6: Types

Type	size	range
byte	8 Bit	-128 bis 127
short	16 Bit	-32.768 bis 32.767
int	32 Bit	-2.147.483.648 bis 2.147.483.647
long	64 Bit	-9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807
float		1.4E-45 bis 3.4E+38

If the new value changes the old value range of the matrix the matrix window is painted again with the new colors. With the button "OK" you accept the new value, with the button "back" you abort it. With the "¿¿" button you can skip to the next entry.

Further features:

- After a valid operation was accepted with "OK", you can undo the operation with the button "Undo"
- With the button "Redo" you can redo the operation

Closing the MatrixEditor window:

- The button "Cancel" closes the MatrixEditor and aborts all changes made to the matrix
- The button "Done" closes MatrixEditor and accepts all changes made to the matrix.

3 Examples

3.1 Selection Sort

The initial situation is VisAlg directly after the start or after selecting textbfProject - textbfNew Project.

Since by default no module is loaded at the beginning, we have to add the required modules *Data Module*, *SelectionSort* and *SelectionSortViewer* first. Have a look at the section 1.1.3, to see, how this can be done.

In the first place, we need some data, therefore we activate the data module by double clicking its entry in the project tree. A window will pop up in the module desktop, which we might to resize or move. We can, for example, load the data set *integer-test.int* (see section 2.1.3). By pressing the System.out button, we can check, if this is a suitable data set.

After that, we activate the SelectionSort algorithm module by doubleclicking its entry in the project tree. In the associated window, we have to select the data module first, in order to tell the algorithm, that we want it to process this data.

At last, we open the SelectionSortViewer.

We can now by pressing either the step or the play button let the algorithm proceed (see section 1.3).

Index