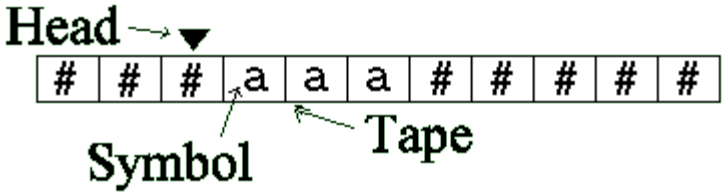
Visual Turing Documentation Contents

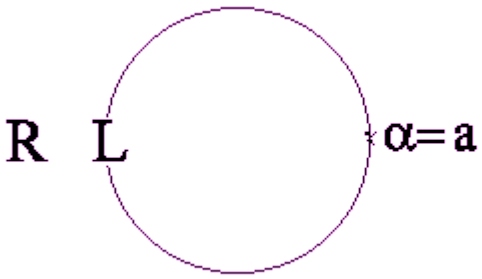
Introduction to Turing Machines and Visual Turing.

What is a Turing Machine ?

A Turing Machine consists of a tape with a left boundary and infinite to the right side, filled with symbols - one symbol at one position - having a head that can read or write symbols or move at left or right. The symbols are the blank "#" and "user-defined" - in Visual Turing you can use letters as symbols.



The way the head moves or writes symbols is determined by his "program", a graph with instructions as nodes - there are some special instructions like **L**-moves the head to left, **R**-moves the head to right, the "symbol-instruction"- writes a mentioned symbol at the current position and the "variable instruction" - writes the value contained by a variable at the current position. The lines link the nodes and determine the execution path. They are entitled with some conditions meaning that, if the condition is true, the execution path will go on that line's way to the destination node. A condition is true if the symbol from the position of the head can be found in any right operand of the condition.
One of the most important thing are the variables. They have the same meaning as in the programming languages- they contain values. The variables can be written on the tape or can be loaded in the following way: when a variable is mentioned in a condition as the left operand- inducing the attribution idea -, if the condition is true, at one moment of the execution, the variable gets the value from the position of the head - the symbols read by the head. The Greek symbols are used for variables.



A simple program that moves once to right then to left while the read symbol is **a.** The alpha variable is loaded with the read symbol

In the expression of the condition can be used the negation - meaning if the read symbol is different from any symbol in the condition the execution goes by there - and loads the variable if there is one as the left operand.
There can be even other machines mentioned as instructions - for example the classical **L#** which scans the tape to the left until it finds the first # it can be placed on the graph as any other instruction and is executed in this way - something like a procedure on function in C or Pascal.
There are many computations that can be made with a Turing machine: addition, substraction, multiplication or string operations like copying or shifting - most of those computations can be found in the examples coming with Visual Turing.

What is Visual Turing ?

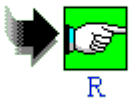
Visual Turing is a tool which helps you to develop Turing Machines. It is supported the entire process of the development, from the designing of the graph- the "program" of the machine, which is made fully visual, to the editing of the tape and running being available the options of the most development tools - running, stepping in, stepping out and over and, a special feature of Visual Turing, stepping back - the reverse execution.
Go to contents to see almost all the things about Visual Turing.

Instructions in Visual Turing

The instructions are represented by little icons. You can meet the classical instructions: L - (a) for moving the head to the left with one position, R -(b) for moving the head to the right with one position, the symbol-instruction - (c) for writing a symbol on the tape, the variable instruction (d) - for writing the value of the variable on the tape and the machine-instruction - (e) for calling another machine to execute its program. I added a supplemental instruction named Nothing - (f), which does nothing and it can be used for combining two arrows to get an AND condition.

The instructions in Visual Turing

They can be combined in the graph-programs of the machines. Any machine has a start instruction - the instruction which is executed first. That instruction has a left arrow like in the figure below



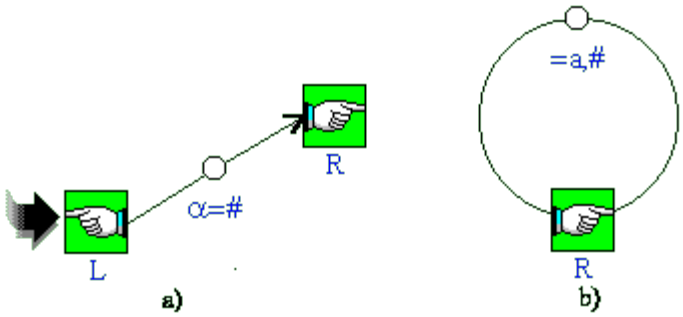
The start instruction marking

In the process of the debugging it is useful to have breakpoints on some instructions. Use the context menu or the properties dialog to do this. An instruction with breakpoint looks like the figure below. Don't forget that the breakpoints stop the program only in visual debug mode.

An instruction with breakpoint

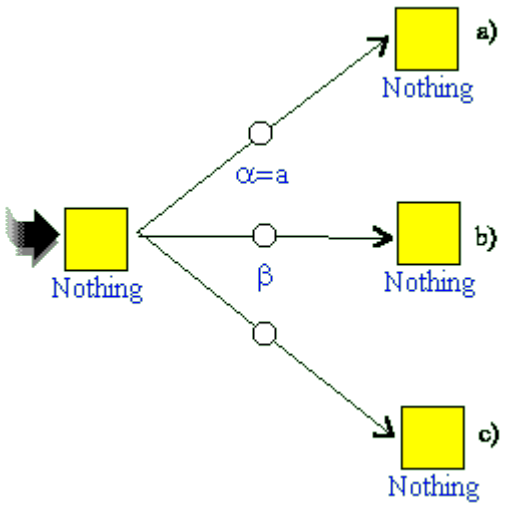
Arrows and conditions in Visual Turing

The arrows in Visual Turing are just like in the original Turing machines. They link two instructions. For the ease of the editing, there are two kind of arrows: normal arrows - a) linking two different instructions in the graph and self arrows which link the instruction to itself. The distinction was made because of the different forms of the arrows.



The two types of arrows in Visual Turing

The role played by the arrows is to be paths of execution. At one moment of the execution, the program will follow one path or another by choosing the arrow with the true condition. A condition is true if the the symbols read by the head is equal with one of the symbols or variables from the right side of the condition. If that path is chosen, the variable acting the left operand - if any - is loaded with that symbol read by the head, and the instruction pointed by the arrow becomes the current one.
The order of the evaluation of the conditions follows their classification. There are normal conditions - like a) with some symbols mentioned in the right side, having the maximum priority, loading variables conditions - like b) which loads the variables with the symbol read by the head and else conditions.- like c) with minimum priority. Note that this order is specific to Visual Turing, because it seemed to me to be very natural.

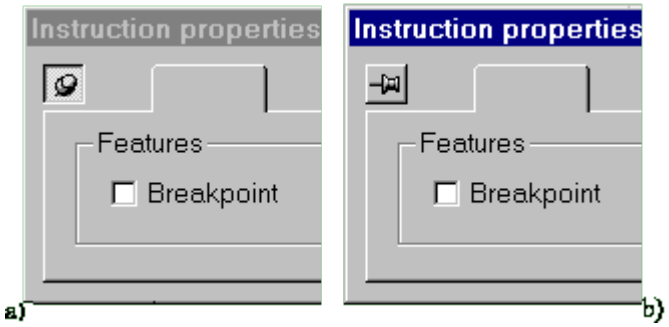


Priorities in evaluating the possible paths

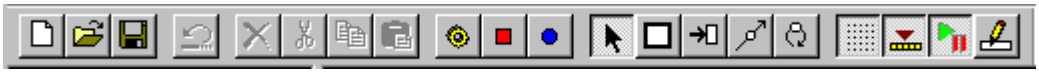
THE GRAPH EDITOR

Introduction

Visual Turing has a powerful editor for the "graph-program" that supports multiple levels of UNDO and the classical operations CUT-COPY-PASTE. Any operation you made in the editor - moving or inserting instructions or arrows -has a visual feed-back. You can multiple select the objects in the editor for moving or "clipboarding" them. Any instruction, arrow, machine etc. has a context menu and a properties dialog which allow you to quick-edit the attributes of the specified object. You can get the context-menu by clicking the right button over the desired object. For the properties, select the **Properties** option from the context menu or double-click the object if it is instruction or arrow. When you use the properties dialog you can fix this on the frame (a) with a nail or let it be free (b). If the dialog is fixed, the losing of the focus doesn't close it - something like in Open Motif. The dialog has instant effect on the edited object.



All the operations for editing the graph are available on the main toolbar. You can see what does every button by when you "fly" over it with the mouse. A longer explanation of the command is shown on the status bar.



For the ease of the editing process, you can toggle a grid - use the **View**/**Grid** menu option

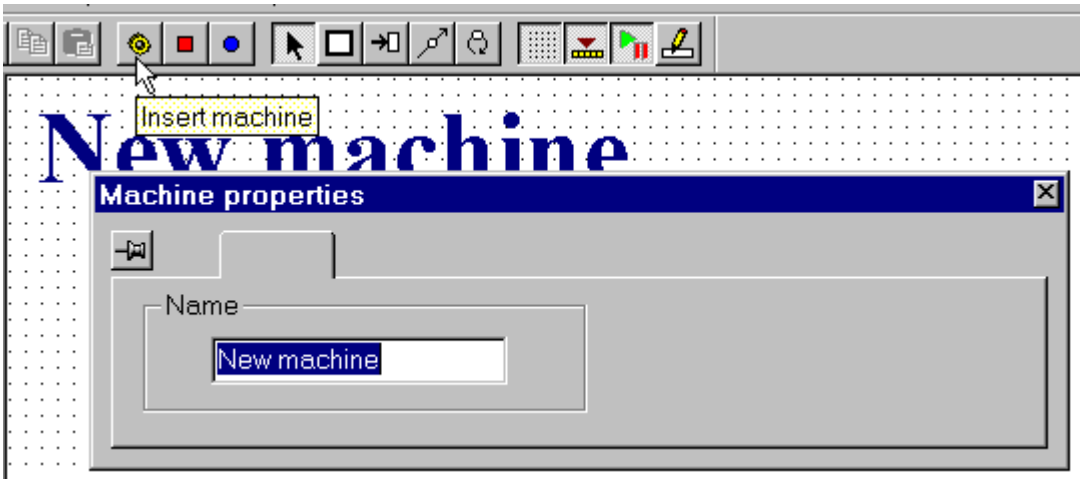
Creating a new project

When you start Visual Turing environment, a default project is created by default. You can also create a new project using the **File**/**New** menu option or pressing the **New** button on the main toolbar. This project contains a single machine named "Main"- you can change the name later, using the properties dialog of this procedure and the symbol which is common to all Turing machines: # the blank symbol. Note that you cannot

delete this symbol during the development time, neither the main machine inserted by default, so you must place your graph there.

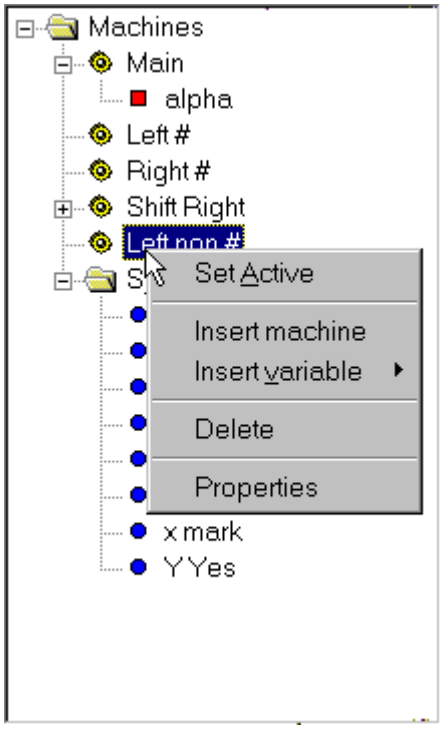
Inserting a new machine

In Turing programming, there is allowed to use compound machines as instructions in the graph-program- something like procedures. Visual Turing supports these constructions. You can insert a procedure by selecting the Tools/Insert Machine menu option or a button from the main toolbar - see the figure below.



The cursor is over the button which inserts a new machine. See the properties dialog for the machine

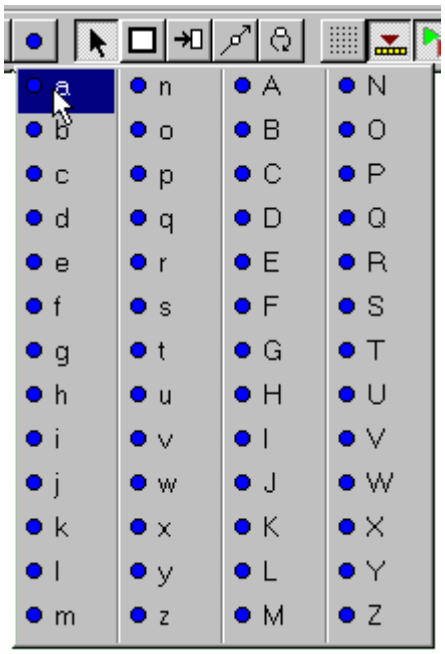
The machines can be selected from a special panel - a tree view- which shows the procedures and the variables assigned connected to it -see the figure below.



Look at the context menu for a machine.

Inserting a new symbol

The symbols can be inserted using the Tools/Insert symbol menu option or the corresponding button on the main toolbar- see the figure below.



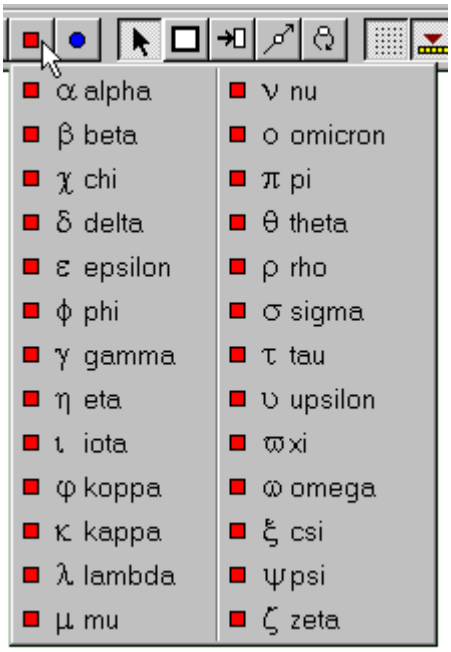
The symbols are inserted by pressing the button with a little blue circle

The symbols can be written or read to/from the tape and represent the "currency" on the Turing machine. In Visual Turing, the symbols that can be used are 'a' to 'z' and 'A' to 'Z'. A little innovation in Visual Turing is the "nicknaming" of the symbols. Using the properties dialog, you can assign a nickname to a symbol which appears at the bottom of the respective symbol-instruction.

Inserting a new variable

You can use the **Tools**/**Insert variable** menu option or the button on the main toolbar to do this - see the figure below.

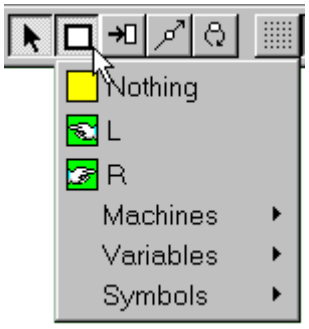
The Turing programs and Visual Turing use Greek symbols for the variables. For a easier use of them, in Visual Turing the names of the Greek symbols are written with the symbol. The variables are used to retain symbols. The loading of a variable is made by placing it as the left operand in a condition on an arrow- see Modifying the condition of the arrow



The little red square represents variables

Inserting a new instruction

In Turing programming, the essential element is the instruction. See Instructions in Visual Turing for more information.
You can insert a new instruction using the **Tools/Insert instruction** menu option or the button from the main toolbar - see the figure below.

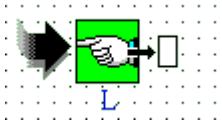


Inserting a new instruction from the toolbar

When you press the left button of the mouse, the instruction appear and you can drag it anywhere you want. When you release the button, the instruction will be effectively inserted in the graph.

Inserting a new branch

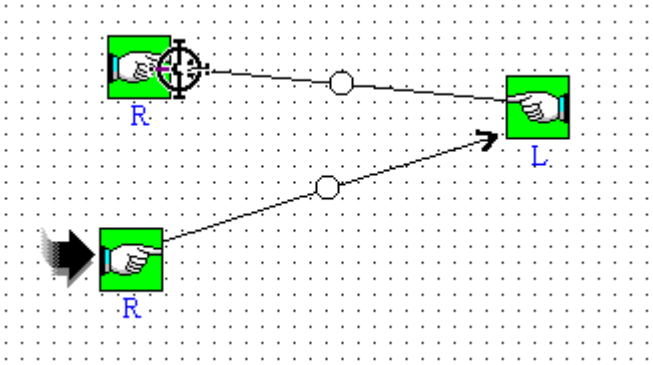
A very useful feature of the editor is the branch tool. By selecting **Tools/Insert branch** or the button on the toolbar, you can extend the graph-program by adding a branch. What is a branch? It is simply an arrow that links the selected instruction to another. The inserting of the arrow and the new instruction is made in just a step. Choose the "destination" instruction and fly over the "source" instruction. The shape of the cursor suggests the direction of the branch - if any - like in the figure below. When you click, the branch appears.



Look at the shape of the cursor

Inserting a new arrow

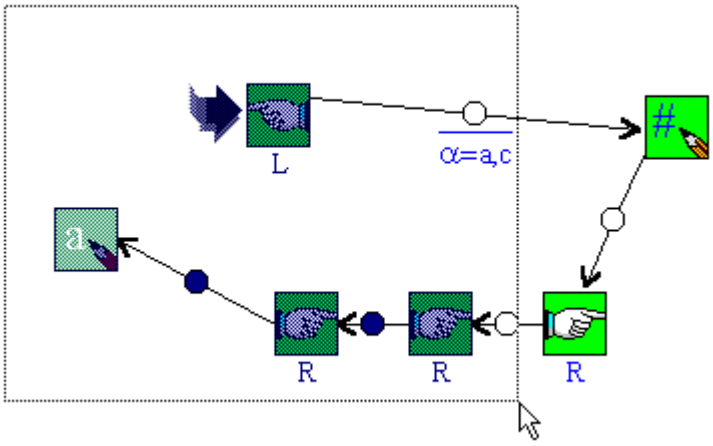
Using the **Tools/Insert arrow** or **Tools/Insert self arrow** menu options or the toolbar buttons you can insert arrows to link the instructions - the nodes in the graph. See Arrows and conditions for more information about the arrows.
After you select an inserting arrow command, you can fly over the instructions to select the source instruction. When the cursor becomes a target you can click the left button and you select that instruction as source. You can drag the arrow to another position for the self arrows or to the second instruction for the normal arrows. When you click again, the inserting is finished.



See the shape of the cursor when the arrow is linked with the destination instruction

Selecting objects

When you work with the instructions and the arrows, you may need to move, delete or cut/copy/paste some objects. Before any operation you have to select the objects you want. The graph editor supports multiple selection that can be done by tracking a rectangle with the mouse - see the figure below



Selecting multiple objects

When you select more instructions, only the arrows between them are selected. You can deselect on object by clicking it with CTRL key or SHIFT key pressed down.

Deleting objects

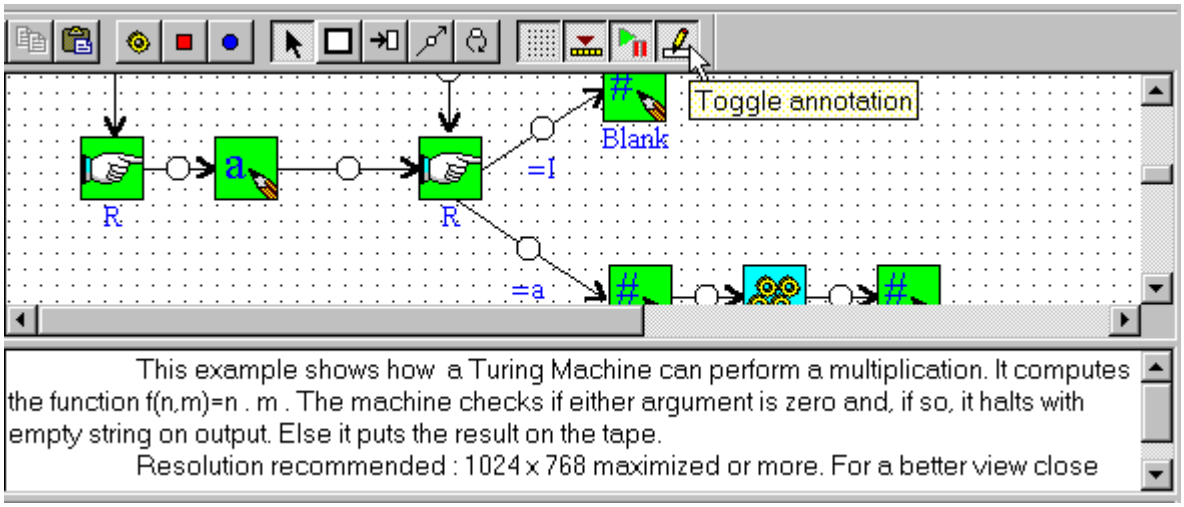
After you have selected some objects, you may delete them by selecting the **Edit/Delete** menu option or the button on the toolbar. Notice that all the edit operations - delete, cut/copy/paste - are available in the context menu of the current machine, that you can get with a right-click on the background. When you delete some instructions, all the arrows that link them are deleted.

Cut/Copy/Paste

In Visual Turing you may work with the clipboard to increase your efficiency. After you have selected some instructions, you can put them in the clipboard. Note that only the arrows having both the source and the destination instructions selected are placed in the clipboard. You may paste the contents of the clipboard any time after this.

Annotating a machine

In Visual Turing you can make annotation on the machines. You can toggle the annotation for a machine and edit any time the content of the annotation.

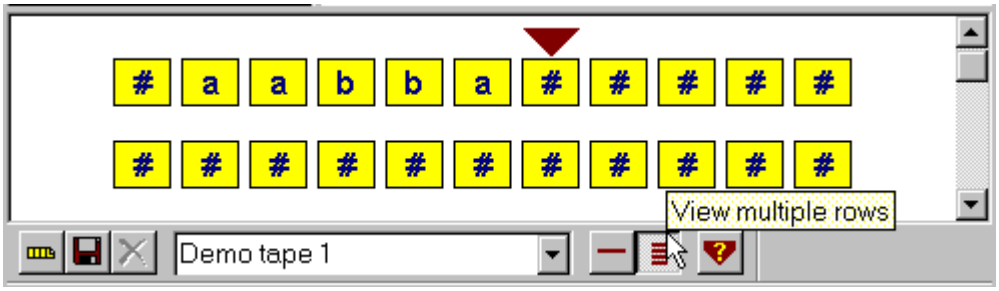


Use the button under the cursor to toggle annotation

THE TAPE EDITOR

The tape in Visual Turing

There is a special panel for the tape in Visual Turing as you can see below

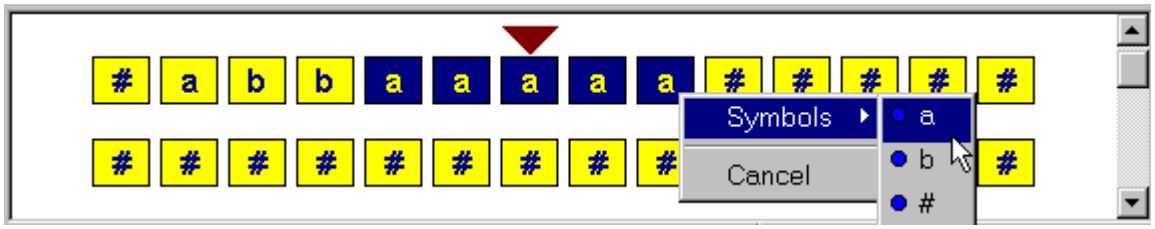


The tape panel

In Visual Turing you can see the tape as a single long row - the classical view - or as a matrix, with multiple lines. In that image, the tape is displayed on multiple rows. You can use the two buttons from the tape toolbar to switch the view- look again at the image !

Modifying the tape

You can edit the contents of the tape in the tape panel. With the mouse you can make multiple selections of tape cells and then change the contents of them by their context menu as you can see below.



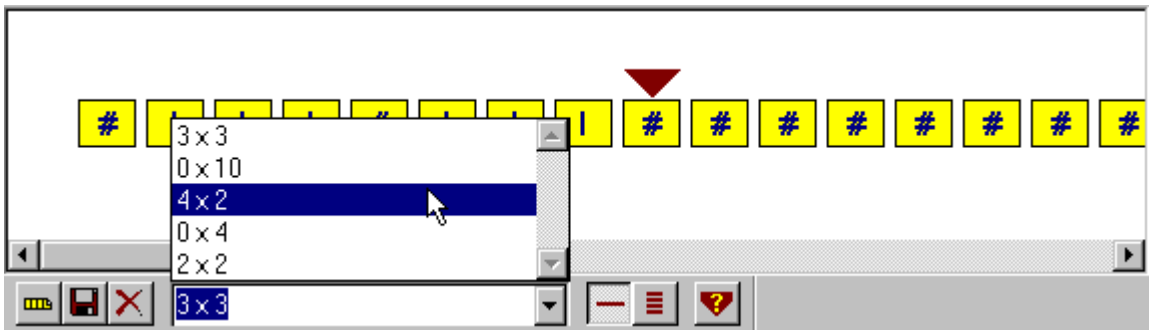
Use the context menu to modify easy the cells of the tape

You can use the keyboard to edit the tape. When a cell is selected, use the arrows keys to move between cells. If you hold the SHIFT key you can make multiple selections. When you press the key corresponding to a symbol, the selection is changed to that symbol. To erase the content of the selection press the DEL key. To move the head at the selected cell press ENTER.

Saving the tape

In Visual Turing you can save certain configurations of the tape that can be loaded later. There are three buttons on the tape toolbar that help you to save the tape with the name typed in the edit ofthe combo box, to delete the tape with the name in the same edit and

to create a new tape. The tapes can be loaded when the combo box drops down. They are saved in the project file.



Selecting a saved tape

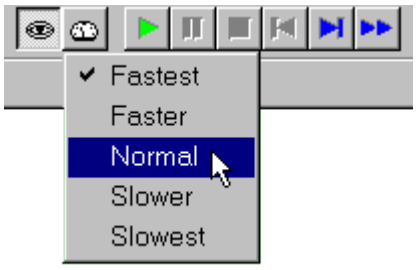
RUNNING AND DEBUGGING

Running a program

After you have "introduced" your graph-program , you can "play" your machine using the multiple facilities included in Visual Turing. Use the play toolbar that you can see below to do this

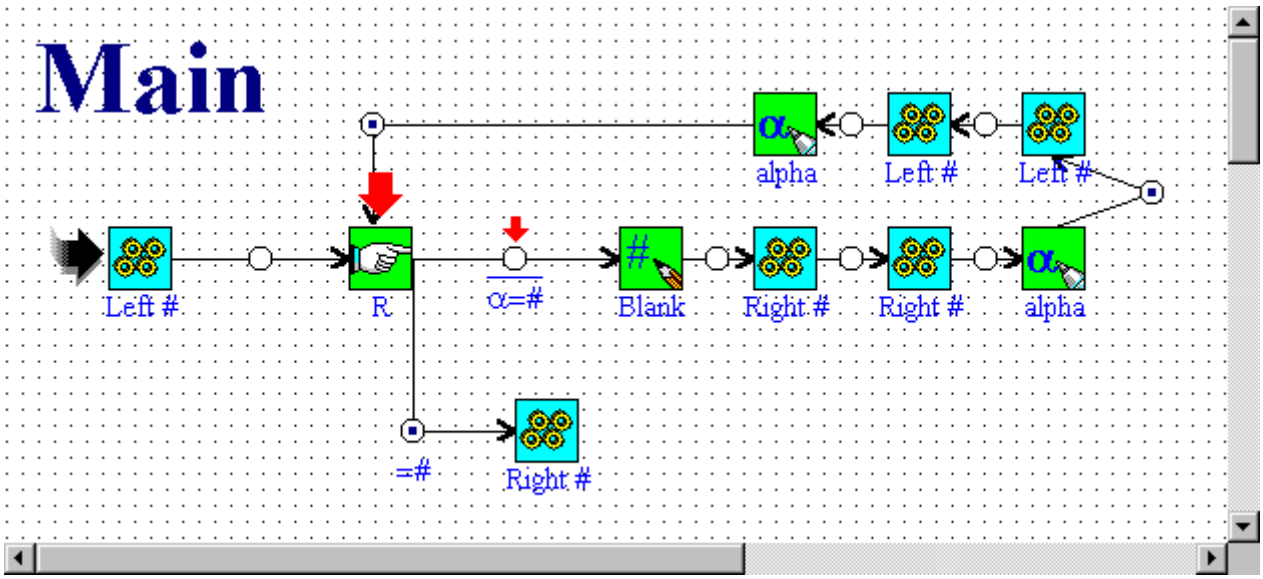
The playbar

The buttons are just like in a VCR and their semantic is very clear. When you run a machine, you have some options that make the debugging easier. The first is the dynamic change of the speed. There are 5 speeds you can choose to run your machine.



The speed menu

For the debugging is available the "visual debug" mode that you can toggle using the "eye" button on the playbar. When the machine is in visual debug mode, every move of the program is displayed on the screen and the special debug options are available: the breakpoints and the reverse execution.
When you watch a machine running, you can see the track of the execution - see the figure below.

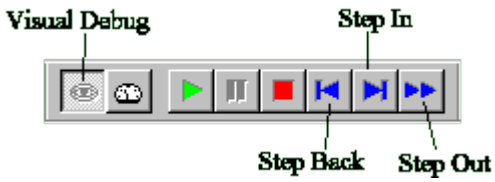


Running a machine

The big red arrow shows the current instruction in the execution track and the little red arrow shows the path to the next instruction. If there is no little red arrow, the current machine will terminate - that mean if it is a submachine it will be simply terminated and the execution will go back where it called that machine or it will end the execution if that machine is the main one.

Debugging a program

After you developed the graph-program, you may debug your "code" structures using multiple features of debugging featured in Visual Turing. It is recommended to be in the visual debug mode when you debug a machine. See the picture below and remark the useful buttons for debugging.



The debug options

When you debugging a program, you can use breakpoints on instructions. You can put a breakpoint on a instruction using the context menu of that instruction. The breakpoint pauses the execution just before the instruction is to be executed. Note that this option is available just in visual debug mode.
With the **Step In** option you can make one step in execution. This feature is available both in visual debug mode and the normal mode.
The **Step Out** option has two "applications": when the execution arrow is over a submachine, you press this button and the machine will be executed without changing the view to that machine - this is stepping over the machines. The other case is when you are in a machine and you want to execute the program until the machine finishes - that is stepping out. This feature is available both in visual debug mode and the normal mode.
The **Step Back** option allows you to make reverse steps of the execution - this is "reverse execution". It is available just in visual debug mode.