VIMOS SOFTWARE FAMILY

Getting started

17 October 2005

Thank you for your interest in our Vision Inspection and Optical Measurement System (VIMOS). In this manual you will find general information about the main functionality of VIMOS and how it can be used to improve your quality assurance process.

Before going on reading the manual, we kindly ask you to read the following

DISCLAIMER

This documentation is provided for reference purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this documentation, this documentation is provided “as is” without any warranty whatsoever and to the maximum extent permitted, atto-Systems Ltd. and Wolf Systeme AG disclaim all implied warranties, including without limitation the implied warranties of merchantability, non-infringement and fitness for a particular purpose, with respect to the same. Neither atto-Systems Ltd. nor Wolf Systeme AG shall be responsible for any damages, including without limitation, direct, indirect, consequential or incidental damages, arising out of the use of, or otherwise related to, this documentation or any other documentation. Notwithstanding anything to the contrary, nothing contained in this documentation or any other documentation is intended to, nor shall have the effect of, creating any warranties or representations from atto-Systems Ltd., Wolf Systeme AG or any of their suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of this software.
The described software is provided 'as is', without any warranty expressed or implied. No guaranty is given that the software is suitable for any given purpose.

COPYRIGHT

Under the copyright laws, neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of atto-Systems Ltd or Wolf Systeme AG, except in the manner described in the documentation or the applicable licensing agreement governing the use of the software. All rights are reserved. Do not reverse-engineer. Do not modify or distribute without all of the documentation.

© Copyright atto-Systems Ltd. and Wolf Systeme AG

Atto-Systems Ltd.
Sofia, Bulgaria

Wolf Systeme AG
Karlsbad, Germany

All rights reserved.

TRADEMARKS

All trademarks and copyrights mentioned within the documentation are respected. They are the property of their respective owners.

CONVENTIONS USED IN THIS MANUAL

***INFORMATION**. This sign marks section in the manual, which is for information only. You can decide to read or skip this section.*

***ATTENTION**. This sign marks section of the manual, which is particularly important for the general understanding of VIMOS. Please, make sure to read this section before proceeding with reading the manual.*

***TIPS & TRICKS**. This sign marks a Tips & Tricks section. Here you can find some practical advises on using the system or get a more detailed explanation of some features. Reading this section may help you in solving a particular problem or give you some ideas but is not vital for understanding VIMOS.*

***PREMISE**. This sign marks a section, which requires you to do something before proceeding with reading the manual. Usually this is a demo program, you have to run or something similar.*

File	*Menu item*
File > Open	*Sub-menu item or dialog control*
"1.1 About"	*Section reference. If the section is within the current manual no manual name is specified. When the section is within external manual the name of the respective manual is also included.*
`Ctrl+E`	*Hot-key combination. The first part of the combination specifies which system key to use. Possible values are: Ctrl, Alt, Shift. The second part specifies the normal key to be used in the combination. The plus sign means that you should press these keys simultaneously.*

CONTENTS

1. Introduction

1.1. What is VIMOS

VIMOS is a machine-vision system – it uses digital technology to perform visual tasks such as inspection, measurement, quality assurance and control of visual quality parameters. Traditionally, these tasks were performed by trained personnel. However, human inspectors suffer fatigue and their assessments are variable depending upon environmental factors. As an electronic alternative to the human or manual inspection, VIMOS provides consistent performance and 100% accuracy. Typical tasks to be solved using VIMOS are: presence/absence of parts in an assembly, presence/absence of features in a part, counting and sorting parts, precision measurements, 1-D and 2-D Barcode reading, optical character recognition (OCR), etc.

VIMOS runs on intelligent cameras manufactured by Vision Components, Germany, based on DSP processors (CPUs) from Analog Devices and Texas Instruments. The live picture, received by the camera, is displayed on a monitor, connected to the camera. The camera generates a second picture, called "overlay" picture, which is placed over the live picture on the monitor. The overlay usually displays some visual clues of the analysis. VIMOS works on series of pictures, received continuously by the camera, and displays results in the overlay picture. Just connect a mouse or a track-ball or a small keyboard to the camera and you will have a fully operational, stand-alone, user-programmable machine-vision system that suits most of your needs.

A full simulator of the system (with a few hardware-specific exceptions) runs in PC/Windows environment. It provides means to develop and test VIMOS user-programs, or it can be used as a separate PC-based tool for image processing. The term "user-program" and the way VIMOS is programmed are discussed in details in the next sections.

VIMOS can interact with external devices. It is capable of monitoring sensors and controlling machines.

***INFORMATION**. To understand the information, presented in this manual, the user should have some basic computer knowledge and experience with the Windows operating system*

1.2. History in brief

VIMOS started as a system for manual inspection and measurement. The camera looked at objects, simultaneously showing them on the attached monitor. The system displayed different drawings on the overlay picture such as company logo, coordinate systems, rulers, boxes, circles, text-output, etc. Using a mouse connected to the camera the user could move the rulers to different objects on the live picture and the displayed text showed for example the distance between two points or the angle between two lines. VIMOS can still be used in exactly the same way.

The next step was to add image processing functionality – edge-detection, filters, contour following, blobs analysis, etc. The way to expand the system was not "hard-coding" of new algorithms, but adding new commands (tools), which are interpreted by the system. By creating different sequences of commands in the form of user-programs, it is possible to perform unlimited number of image-processing algorithms on the camera.

Further development of the system added tools for communication, jumps and conditional branches inside the user program, text manipulation, and a whole group of new powerful tools, working with lists of points.

The PC Simulator was introduced in the system in order to make developing and testing of user-programs more convenient for the user. It allows development to be done not on the place the system

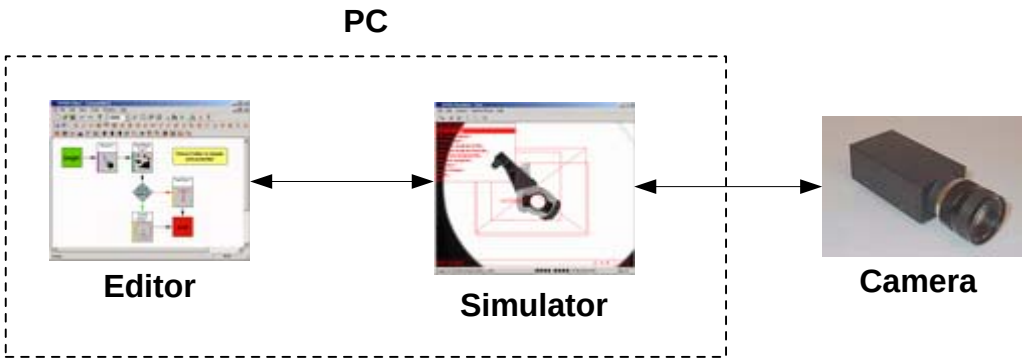
is to be used (usually a dirty and noisy factory environment) but in a comfortable office instead. The user-programs, developed this way, are fully functional and need only be loaded on the camera to run.

Even more convenience was brought to the user by introducing a visual editor in the system. VIMOS tools are represented as icons and the user can create the skeleton of the user-program by simply selecting the tools needed and adding them to the project. Then the tools can be connected in a way representing the program flow, their parameters can be configured and their outputs and inputs can be linked to the inputs and outputs of other tools respectively. To fine-tune the program the user can run it in the Simulator by simply clicking a button in the Editor. Changes made in the Simulator are easily transferred to the Editor and vice versa, thus keeping the user-program up-to-date.

1.3. The VIMOS System structure

The VIMOS System, also referred as the VIMOS Software Family, is comprised of three independent parts – the VIMOS Kernel, the VIMOS Simulator and the VIMOS Editor. The VIMOS Kernel is this part of the system that actually performs the image processing tasks. It runs on the camera and is also a part of the VIMOS Simulator to exactly simulate the camera behavior in the PC/Simulator environment. The VIMOS Simulator greatly enhances the development and tuning of user-programs in PC/Windows environment. It has the whole functionality of the VIMOS Kernel and offers some additional features that allow to use it as a separate PC-based image processing system (e.g. it can receive image information from any PC/Windows compatible image source like an image file, WEB camera, video stream etc.). The VIMOS Simulator is also capable to fully control the camera by means of the PC mouse. The VIMOS Editor brings even a better way for user-programs development. It offers a visual, intuitive approach for completing your specific image processing tasks. There are ways to exchange all information required between the different VIMOS System's parts.

The figure below gives a schematic representation of the VIMOS Software Family structure.



1.4. Basic features

1.4.1. Environment

The VIMOS Kernel can be used completely stand-alone on an intelligent camera with a connected monitor and a mouse. At present the kernel runs on all black-and-white VC-cameras with ADSP CPU type 2 and 3 (VC38, VC61, the cheap model M40, ...) and the high-speed smart cameras with TI CPU (VC2065, VC2038). The term "stand-alone" means that the camera can be used without being connected to a PC. A PC is needed only to load the system on the camera, but the programming of the system (creation of user-programs), the modification of existing user-programs and the running of the system can be done stand-alone. This feature is very useful in factory environments, where it is not possible or unsuitable to place a PC beside the camera.

The PC however greatly facilitates the user-programs development. You can:

- Use the VIMOS Simulator on the PC to create and run user-programs exactly in the way they are created on the camera. Use bitmap files, AVI-movies or PC-cameras as picture sources in the Simulator.
- Use the VIMOS Editor to create and maintain user-programs. Load and run Editor programs into Simulator by one mouse click. Edit tool parameters in Simulator and update corresponding user-program in the Editor by one hot key.
- Export programs created entirely in the Simulator or on the camera to the Editor.
- Use the Simulator to transfer executable files, user-programs, images and other system files from the PC to the camera and vice versa.

VIMOS is programmed as hardware-independent as possible. The Simulator on the PC uses the same source code that runs on the camera. Further VIMOS development is possible by porting the system to other platforms.

1.4.2. Functions and resources

VIMOS is a user-programmable system, i.e. to perform a particular task, it needs to be programmed. The programming means creating a sequence of tools (commands), called user-program. VIMOS has a good and fast growing list of tools that you can use in your programs. See sections "*2.2 User-programs"* and "*2.4 Overview of VIMOS tools"* in this manual for details.

VIMOS works with the following types of data:

- points (x/y coordinates)
- floating-point numbers
- 16 and 32 bit integers
- angles
- text strings
- point-lists (buffers of points)

These types refer to input arguments and results of user-program tools.

VIMOS operation is controlled by a graphical interface with menus and dialogs. A single mouse is sufficient to access all VIMOS functions, including the creation and modification of user-programs.

VIMOS supports various types of files, which are stored in the flash memory of the camera or the hard disk of the PC:

- user-program files
- calibration files
- statistics-data files
- image files
- text (string) files
- point-list files.

VIMOS has built-in functions for image inspection, which do not require the creation of user-program and can be accessed directly via menus/dialogs:

- Take picture, save picture into DRAM
- Picture binarization
- Histogram in rectangle
- Line profile
- Display of pixel values.

VIMOS can be calibrated for real-world measurements and X/Y distortions.

VIMOS supports digital I/O (PLC) lines. Use these lines for triggered picture taking in pipeline inspection of machine parts or to integrate VIMOS in more complex industrial control systems.

VIMOS has interfaces to the V&C IO-box with more PLC-inputs and outputs.

VIMOS functionality can be extended through custom program-modules (exec-call) with extensive parameter transfer. These can be viewed as VIMOS plug-ins.

VIMOS has statistics-counters that can be saved in the power independent memory to survive a power-down.

2. Basic philosophy of VIMOS

VIMOS has everything to run in stand-alone mode on an intelligent camera. When a mouse or other pointing device is connected to the serial port of the camera, the system is controlled using a graphical user interface (GUI) with menus and dialogs. At present, the GUI is displayed on a single-color overlay picture, but soon the system will utilize the multi-color overlay of the new smart cameras from Vision Components.

The PC-software allows controlling the VIMOS Kernel on the camera by the PC-mouse, so it is not always necessary to have a mouse connected directly to the camera.

The VIMOS Kernel runs in two modes:

- **Edit Mode** - control and configure the system, including the creation or modification of user-programs.
- **Run Mode** - execute the user-program.

The VIMOS Simulator is used to create and test user-programs in PC/Windows environment, which are then executed by the VIMOS Kernel on the camera. It is the bridge between the PC environment and the embedded camera environment. The Simulator contains the VIMOS Kernel and also has additional features that allow full control of the camera directly from the PC. Since the same kernel runs on the camera and on the PC you have exact simulation of the camera behaviour. A few differences exist due to the camera specific hardware. A typical difference is the image acquisition. The PC lacks an image sensor so you have several options for the image input:

- Sequence of still image files
- AVI-movie
- Any Video for Windows (VfW) compatible video source such as cheap USB-cameras or more sophisticated video capture cards.

The VIMOS Simulator can also be converted into a complete PC-based image-processing program, compatible with the VIMOS Kernel.

The VIMOS Editor is another convenient way to develop your image processing software. The user-program is graphically represented as a flowchart where you can easily track the program execution flow and the logical links between the different program parts.

2.1. Embedded environment

The VIMOS Kernel on the camera is an embedded system. It runs on a computer with a specific operating system and less hardware resources, compared to a PC. The different camera models offer different hardware configurations. Be careful to select a model, which has enough resources for your VIMOS application(s).

The critical camera resources are:

- CPU speed (MHz)
- Size of DRAM memory
- Type of camera sensor

Use models with faster CPU and progressive-scan sensor for real-time tasks. Select a camera with TI CPU if system performance is critical. Cheaper ADSP cameras can be used for applications with less speed/memory requirements.

One important part of the camera is its flash PROM. It plays the role of the hard disk of the PC, but has some peculiarities. The file system of the camera writes each new file into the next free area above all existing files. The flash area, where a previous copy of the file resides, can’t be used until a pack operation of the whole flash is performed. After continuous work with the flash, comes a moment when the flash becomes full.

***INFORMATION**. This problem is caused by the electrical properties of the flash PROM chip. The flash memory is organized in sectors of 64K bytes. To overwrite one byte in the flash, it is necessary to erase the whole sector the byte belongs to.*

VIMOS solves the problem in the following way. When started on ADSP camera, it displays information about the free flash space and the total flash size. Before exit VIMOS prompts for a flash pack operation. If the flash is about to be full (for example less than 64K bytes remain free), exit VIMOS by packing the flash and restart VIMOS. The flash can also be packed by a Simulator function or by the “pk” shell command of the camera if you use a terminal program for communication with the camera. Another possibility is to use automatic flash packing on exit, set by the “General-purpose configuration” dialog of the VIMOS kernel. You may also perform dynamic packing during the execution of the user-program by the “Delete and pack” tool.

***INFORMATION**. Packing is not necessary when VIMOS runs on TI camera equipped with a multimedia card (MMC) and the MMC is the default drive for the VIMOS files.*

Hardware configurations of some camera models:

Model	CPU	Flash	DRAM	Multimedia card (MMC)
VC38	ADSP2181	2MB	8MB	-
VC61	ADSP2181	2MB	8MB	-
VCM40	ADSP2181	512K	8MB	-
VC2038	TMS62C11 – 150 MHz	2MB	16MB	8MB

Another important aspect of the camera operation is the way it communicates with the external world. The camera has one serial port, which may be connected to a PC or other device. Connect the camera to a PC and use the PC software to develop VIMOS applications. Use the Simulator to transfer VIMOS system files, user-programs and other files between the PC and the camera. Use the Simulator to run VIMOS on the camera and to simulate the operation of an external mouse.

***INFORMATION**. The operating system of the camera has a command shell. Use a terminal like PROCOMM or HyperTerminal to communicate with the shell. The terminal programs provide file transfer to/from the camera as well.*

Connect a Microsoft-compatible mouse to the camera serial port and run VIMOS in stand-alone mode. You have access to all VIMOS resources. Remember that using a mouse for data input without a keyboard has some limitations when entering numbers or text. For details about mouse operation see section *“2. Mouse interface”* in the *“Using the VIMOS kernel”* manual.

You can connect different external devices to the serial port of the camera to control the VIMOS application (V&C IO-box for example). Before the connection, while you have a mouse, you need to:

- Create/load a user-program that communicates with the external device.
- Configure VIMOS I/O settings for the external non-mouse device.
- Save the user-program to save I/O settings selected in the previous step.

Remember that once you have connected a non-mouse device, you can’t control VIMOS but you can just use the system in run mode to do some job. Connect a mouse or use the PC Simulator to regain control of VIMOS by a mouse. When started, VIMOS always checks for a connected mouse and ignores other I/O settings if it detects one.

2.2. User-programs

VIMOS is an open, user-programmable system for machine-vision applications. It does not support a fixed number of hard-coded complex algorithms, but interprets a sequence of relatively simple yet powerful commands (tools). The tools are arranged in a sequence, called "user-program". Different

sets of tools can be combined into separate user-programs in such a way, that each user program performs a complex image-processing algorithm.

In order to cooperate the tools have input arguments and output results. The basic argument/result types are integer numbers, floating point numbers, 2D points, angles and text strings. The result(s) of one tool can be passed to the input(s) of other tool(s).

To create a user-program:

- Add tools to the program.
- Link the results of given tools to the input arguments of other tools. The results of one tool may be linked to an arbitrary tool, which is executed next.

The system is open, which means it can be easily extended with new tools.

The user-program is created by visual means, i.e. by selecting tools from a tool-list menu, placing tools on the screen (for tools with graphical representation), dragging tools' representation with the mouse, etc. Menus and dialogs are used for tool configuration and system control.

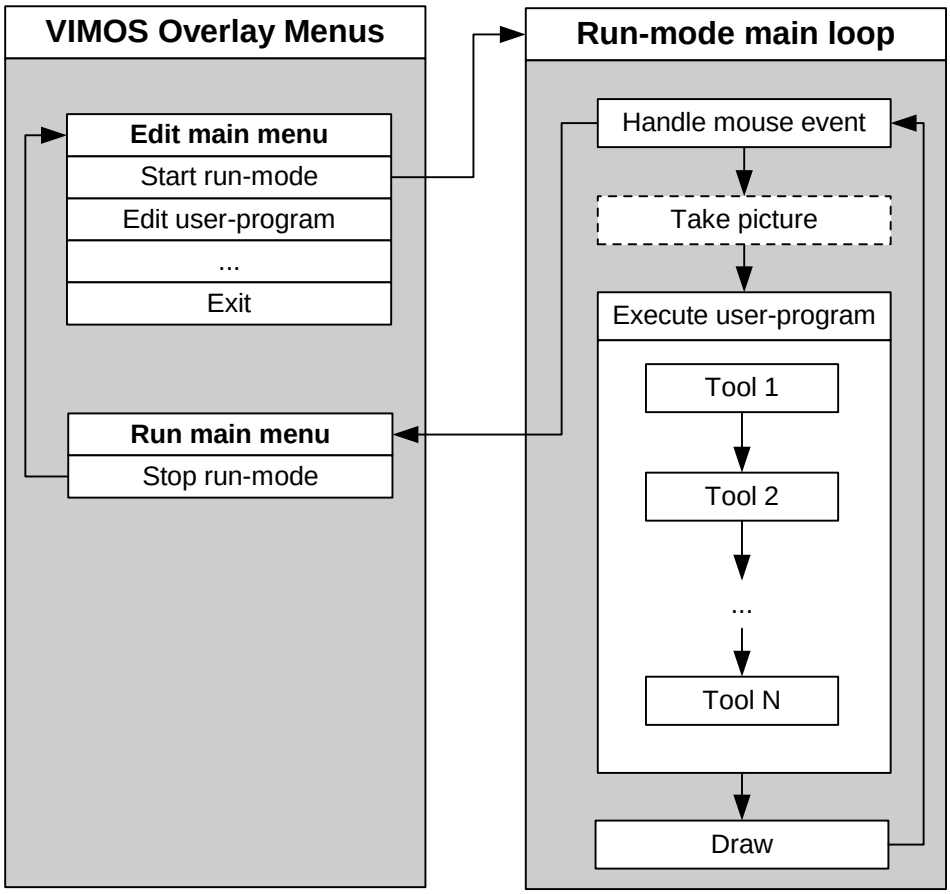
Conditional branches and jumps can be used inside the user program. General-purpose counters can be used to construct FOR and WHILE-type loops.

2.3. Main system loop

VIMOS executes each user-program repeatedly in a loop, called the "main system loop". Usually each loop pass takes a new picture from the video-input (automatically or by a special tool), executes the tools in the user-program and displays the results in the overlay. In other words the main loop applies the image-processing algorithm, coded in the user-program, on the current camera picture. It is possible to execute tools in parallel with the picture-taking process. The main loop also manipulates the pointing device and activates the GUI to properly respond to the operator's commands.

It depends on the speed of the camera and the complexity of the user-program, but usually multiple loop iterations are completed within a second.

The figure below shows a simplified diagram of the VIMOS Kernel structure and the flow of execution.



2.4. Overview of VIMOS tools

The VIMOS tools (also referred as commands or user-program elements) are divided into several groups. Here is a brief description of the main tool groups:

- **Image processing tools**. All tools that read or modify image pixels belong to this group. Typical representatives are edge-detection, 3x3 operator, contour following, blob analysis, OCR, bar-code reader, normalized correlation.
- **Graphics and calculation tools**. This group includes tools that combine drawing and calculations. These tools do not work directly on images, but use results of image processing tools. Most of these tools display some diagram on the overlay.
- **I/O tools**. This group includes tools for communication through PLC-lines and the serial interface.
- **Conditional branch and jump commands**. These commands make jumps and conditional branches within the user-program by IF-ELSE-ENDIF and GOTO / LABEL operators.
- **Statistical tools**. A set of counters can be used for statistical purposes. Tools for operations on counters belong here - reset, increment, decrement, save to and restore counters from flash memory.
- **String tools**. This group contains tools for text manipulation. A reserved memory buffer in VIMOS is used to store text tool results (bar code, OCR). The text results may be displayed on the overlay or sent via RS232.
- **Point-list tools**. A reserved memory buffer in VIMOS is used to store sequences of points. Additional data could be associated with each point. Many tools store results or read arguments from the point-list buffer.

- **GUI tools**. Tools, which display GUI objects like window, button, check-box, etc, and process mouse or touch-screen commands, entered by the user. Used to communicate with the user-program during its execution. Present on PC and TI cameras only.
- **Other tools**. Tools, which don't fit into any of the above groups, are placed here - execution of external modules, timer and pause tools, dynamical loading of user-programs, etc.

2.5. Input/output

VIMOS can read data from the following camera sources:

- Video input - the camera stores sensor data into DRAM images
- Serial port
- Ethernet port
- PLC inputs (4 digital lines)
- Flash memory

VIMOS can write data to the following camera destinations:

- Overlay and background pictures on the monitor
- Serial port
- Ethernet port
- PLC outputs (4 digital lines)
- Flash memory

3. Suggested way to become familiar with VIMOS

In this section you will have the opportunity to see how VIMOS works on a few predefined user-programs. You will get acquainted with the two ways to develop your user-programs – using the VIMOS Kernel and the VIMOS Editor. In these first exercises you are not expected to load and run your user-programs on a camera. However this important final step will be mentioned for completeness.

ATTENTION. *It is very important to understand at the very beginning that the VIMOS System is mainly designed to run on a camera. All of its functionality is available in a stand-alone mode on the camera using only a mouse connected to the serial port of the camera.*

The VIMOS Simulator is a tool that exactly simulates (with a few differences) the VIMOS Kernel on the camera and lets you work as if you work on the camera by using the more convenient PC environment instead. The VIMOS Simulator can also be used as a separate PC based image-processing system.

The VIMOS Editor is another tool which main application is to enhance the process of user-program development.

The recommended user-program design flow is VIMOS Editor > VIMOS Simulator > VIMOS Kernel on the camera. This way you will always keep your information up to date and will always support maximum compatibility with different camera models. If for some reason you are not able to use the VIMOS Editor you can alternatively use the following design flow – VIMOS Simulator > VIMOS Kernel on the camera or simply work directly on the camera.

You must also know that there is a way to export user-programs created directly on the camera or in the Simulator to the Editor. Nevertheless this practice is generally not recommended.

No matter how you developed your user program the last step you usually do is to load and run this program on one of the cameras supported by the VIMOS System.

Section "*3.1 Installation and working with VIMOS programs"* gives you a brief description of how to install and operate the VIMOS applications and demo programs. In case you need more detailed information, please refer to the *"Installation Guide"* manual.

Section *"3.2 Simple demo-program"* shows you how to start a simple demo program just to see how VIMOS looks and works.

In section "*3.3 Create your first VIMOS program"* you will be guided step-by-step through the process of developing the same simple user-program using the VIMOS Kernel (the PC Simulator in this case).

In section "*3.4 Create a user-program in the VIMOS Editor"* the same user-program will be developed with the means of the VIMOS Editor.

At the end of this section you will have a general understanding on what VIMOS can do and you will be able to start using it on your own.

ATTENTION. *To be able to run the demo programs you must have the demo programs, supplied with your VIMOS distribution, installed on your hard disk. For instructions on how to install the demo programs refer to the "Installation Guide". After installing the demo programs a new program group, called "VIMOS Demos" will be created within the main VIMOS program group.*

Although there are many different ways to get familiar with VIMOS, we recommend following the steps below. This will ensure that by the end you will have the basic skills needed to freely interact with the Editor, Simulator and the camera itself and will be ready to start developing the solution to your particular problem:

- ✓ Install VIMOS software (see *"Installation Guide"*)

- ✓ Get familiar with loading and starting VIMOS applications (*"Getting started" 3.1*)
- ✓ Load and run simple demo-program in Editor/Simulator (*"Getting started" 3.2*)
- ✓ Create your first user-program with the Simulator (*"Getting started" 3.3*)
- ✓ Create the same user-program with Editor (*"Getting started" 3.4*)
- ✓ Read the manuals *"Tools' Descriptions"*, *"VIMOS programming"*, *"Using the VIMOS kernel"*, etc.
- ✓ Run demo programs for tools/group of tools - *"Examples"*

3.1. Installation and working with VIMOS programs

Setup VIMOS software as directed by the installation guide. You will see two Windows programs installed on your computer – Editor and Simulator, and a number of data files. You have a 30-days trial period, during which you can use the programs without registration. You may run demo user-programs or create and test your own VIMOS applications without a camera.

3.1.1. VIMOS operation

Before you proceed with the next section, you need some basic knowledge how to operate with the mouse and configure VIMOS by the GUI interface.

Basic mouse commands:

- Right click opens the main menus in run or in edit mode.
- Right click closes current menu or cancels selected option.
- Left click activates selected menu line or dialog control.
- Up/down mouse movements select previous/next menu line or dialog control.

The VIMOS Kernel has two main menus: edit menu and run menu. Right click opens the menu of the current system mode. Second right click closes the menu. Here are some basic VIMOS-configuration commands initiated through these menus:

- **Run mode main menu.** Enter edit mode by choosing **Stop run-mode** menu line.
- **Edit mode main menu.** Enter run mode by choosing **Start run-mode** menu line. Load user-program by choosing **Load user-program from file...** menu line. Save user-program in a file by choosing **Save user-program to file...** menu line.

Refer to the manual *"Using the VIMOS kernel"* for more details.

3.1.2. Loading and running VIMOS applications

After you have done all setup described above, you are ready to use the VIMOS software. Here are some basic instructions how to create and run the VIMOS applications.

- ♦ Use the Editor, the Simulator or the camera to create and/or modify user-programs (best choice for creation is Editor). Note that you can transfer a user-program from Editor to Simulator.
- ♦ More important functions of the Simulator program:
  - ❑ Press the **Camera Simulation** button  to start VIMOS simulation (captures Windows mouse input).
  - ❑ Press the **Start camera VIMOS** button  to start VIMOS Kernel on the camera and use the PC mouse to control it (captures Windows mouse input).
  - ❑ Use **Camera Files Functions** button  to open the file-transfer dialog. Here you can transfer files between PC and camera flash and get camera pictures.

- Transfer and run programs, opened in the Editor, directly in the Simulator.
- Run user-program in the Simulator or on the camera by:
  - Enter edit mode
  - Load the user-program file from main edit menu
  - Enter run mode to start the program
- Export user-program from Editor to Simulator in two ways:
  - Press the **Run**  button in the Editor to load and start the current user-program in the Simulator.
  - Right-click on a tool in the Editor to open its context menu. Selecting **Edit in Simulator** menu line will load the program in the Simulator and start VIMOS simulation in edit mode. Note that tools can't be inserted/deleted in this mode, only arguments can be modified. Press **F6** to update the program back in the Editor.
- Export user-program from Editor to camera. There is no direct link between the Editor and the camera. Load the Editor program into the Simulator (see previous item). Enter edit mode, save current program into a file and transfer the file to the camera using the file-transfer dialog of the Simulator (same as next item).
- Export user-program from Simulator to camera. Enter edit mode of Simulator. Save current user-program into one of the predefined user-program files, `up0.vm` for example. Open the file-transfer dialog of the Simulator and send `up0.vm` to the camera. Start VIMOS on the camera, enter edit mode and load `up0.vm`. Enter run mode to start the program.
- Export user-program from camera to Simulator. Start VIMOS on the camera, enter edit mode and save current program to file `up0.vm` for example. Exit VIMOS on the camera. Start Simulator and open the file-transfer dialog. Read the flash file `up0.vm` to a PC file with the same name by the **Get** button. Exit the file-transfer dialog and start VIMOS simulation. Enter edit mode and load `up0.vm`. Enter run mode to start the program.
- Export user-program from Simulator to Editor. Start VIMOS simulation, enter edit mode and load the user-program. Export the program to MPR file by the File > Export MPR… menu option. Start Editor and import the MPR file by **File > Import**.

***ATTENTION**. The Simulator captures the mouse during camera simulation and mouse simulation (toolbar buttons*  *and*  *). Then you will see that the mouse pointer has disappeared. This is done because the Simulator needs to receive all mouse signals exclusively and it cannot share the mouse with Windows. This is necessary in order to achieve exact camera simulation.*

*You can temporarily release the mouse by pressing **`Alt+M`**. Then you can use the mouse in Windows but not in the camera simulation. To recapture the mouse press **`Alt+M`** again.*

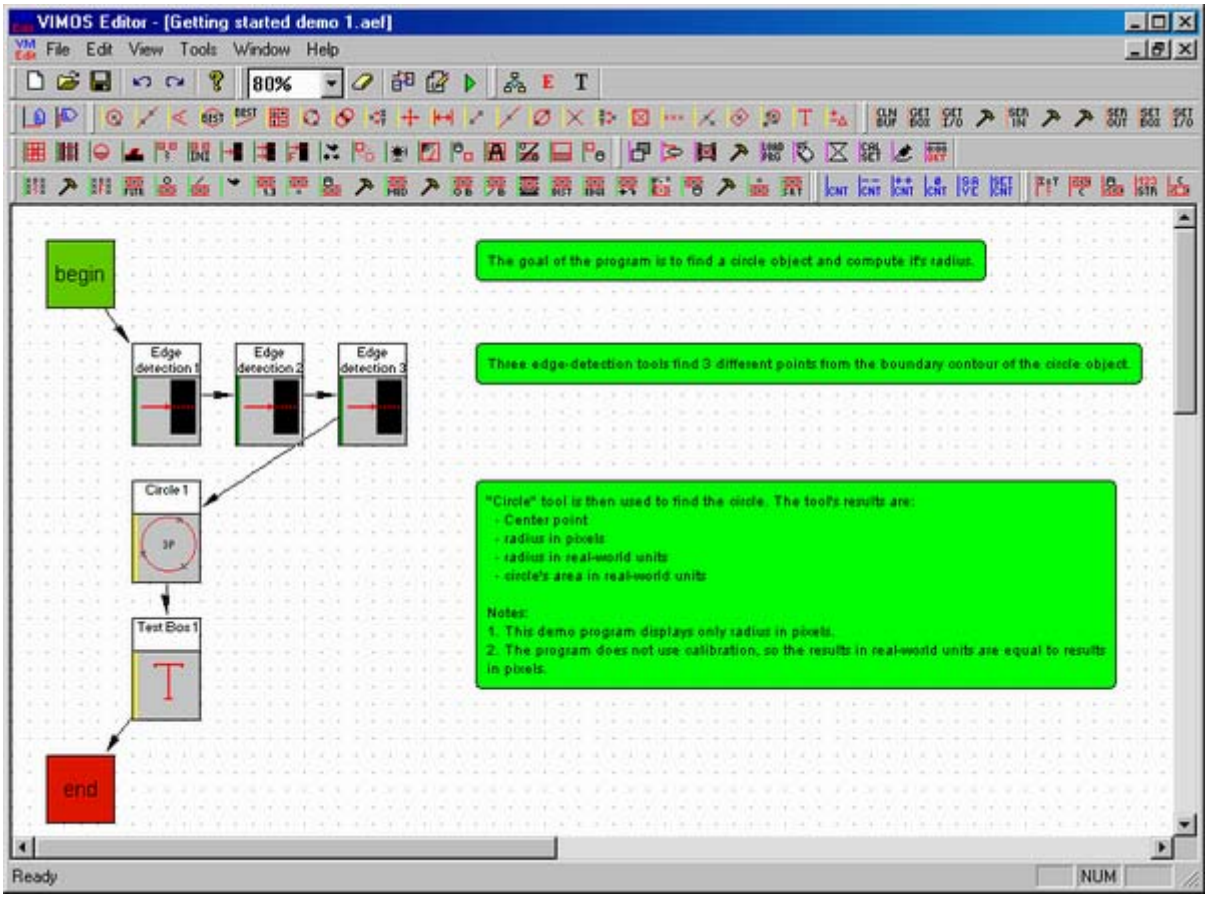
3.2. Simple demo-program

Welcome to the first VIMOS user-program, you can see running. It works on an image with several black objects, placed on white background. The purpose of the program is to find the radius of a circle object.

This is the first of a set of demo user-programs so it is the simplest both in what it does and in what you have to do in order to start it. Actually, it is so simple, that you only need to click a couple of times and watch the result. All steps to run the simple demo program are explained below.

Step 1 – start the VIMOS Editor and load the demo program

First you need to start the VIMOS Editor and load the demo program. To do so you can either use the short-cut **Simple demo** provided in the **VIMOS Demos** program group or start the Editor and open the file `Getting started demo 1.aef` from the corresponding demo folder (usually …`\vimos\demo programs\getting started demo 1` but may vary in different installations). You should now be able to see the diagram of the user-program in the Editor.



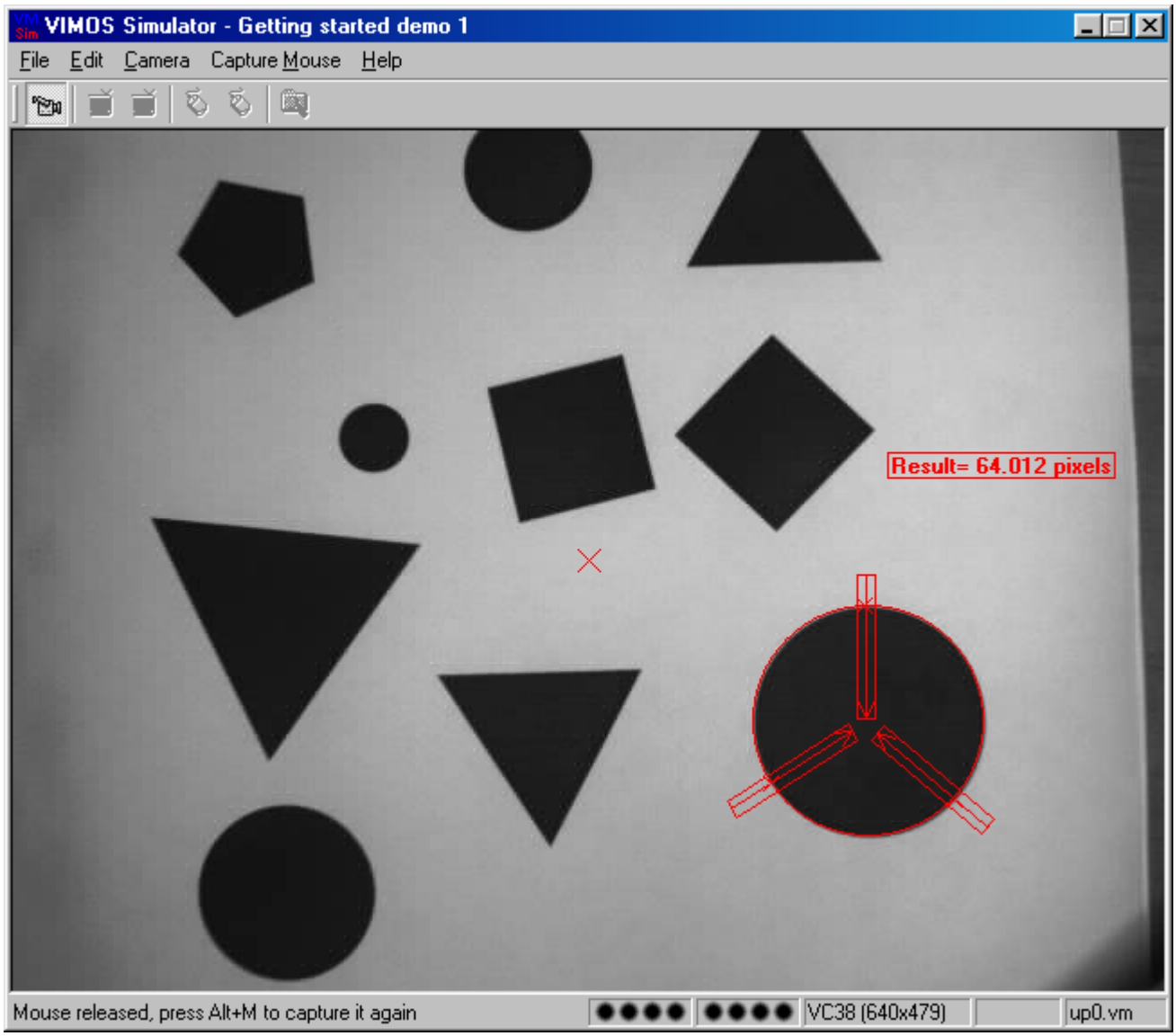
The program contains 3 *Edge-detection* tools, placed in a circle. They find points in the image, where essential change in the pixel intensity occurs - from white to black or vice versa. The edge-detection tools find 3 points, lying on the boundary of the circle.

The next tool in the user-program is a *Circle* tool, which finds a circle, specified by 3 points. The result points of the 3 edge-detection tools are passed (linked) to the input arguments of the circle tool. The tool finds the center point and the radius of the circle.

The last tool in the user-program is a *Text-box* tool, which displays the calculated radius. The *Radius 1* result of the circle tool is linked to the input argument of the text-box.

Step 2 – start the demo program simulation in the VIMOS Simulator

To see what actually the program does you need to start it in the VIMOS Simulator. To do so, just press the Run button  on the Editor's toolbar. This should start the Simulator, load the user-program in it and start VIMOS in run mode. What you see now is the result of the program execution.



Notes:

- The demo program works on a sequence of images provided with the demo program.
- The measured radius is in pixels. We can get the real radius of the circle after system calibration for real-world measurement units (see section *"6.2.2.4 Calibrate Dialog"* in manual *"Using the VIMOS Kernel"*).
- This program yields correct results for images, where the position of the circle changes in certain limits.

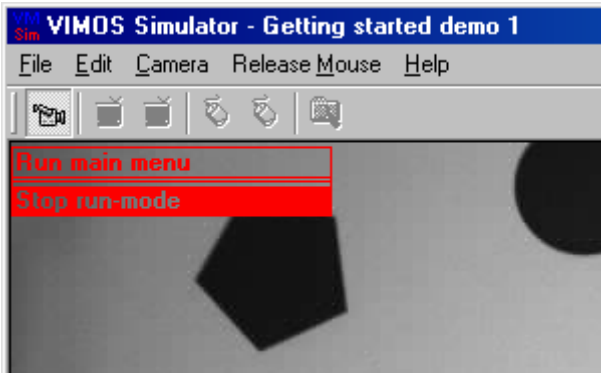
As you have probably noticed the Windows mouse cursor disappeared. This is because the VIMOS Simulator captured the mouse when it was started in run mode by the VIMOS Editor. This is done to exactly simulate the behavior of the VIMOS Kernel where the user interface does not use a mouse pointer.

The mouse will be automatically released when you exit the camera simulation mode. An alternative way to release the mouse is to press the **`Alt+M`** key combination. If you release the mouse by **`Alt+M`** while the VIMOS Simulator is in camera simulation mode you will lose your mouse control on the simulation. To re-enable your mouse control press **`Alt+M`** again to capture the mouse and be able to control the simulation again.

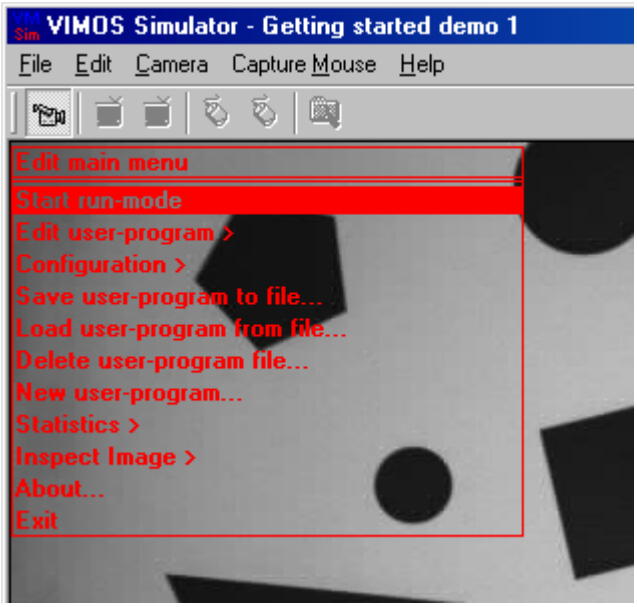
***ATTENTION**. If you experience difficulties working with the mouse, please read the section "3.1.1. VIMOS Operation" once again. For more detailed information about how to use mouse with VIMOS please read the manual "Using the Simulator".*

Step 3 – stop the simulation and exit the VIMOS Simulator

To stop the simulation process, right click the mouse first. This will open the run mode main menu.



The only available option **Stop run-mode** is selected. Left click the mouse to activate this option. The user-program simulation now stops and you are in edit mode. Right click the mouse again to activate the edit mode main menu.



Select the **Exit** option by moving the mouse up or down and left click to activate it. When you exit the camera simulation mode the VIMOS Simulator releases the mouse and the mouse pointer appears on the screen. Now the VIMOS Simulator can be controlled as any other Windows application and can be closed by **File > Exit** menu command.

3.3. Create your first VIMOS program

TIPS & TRICKS *In this section you will learn how to start VIMOS Simulator, add tools to a new user-program, configure them and test the newly created user-program.*

In this section you will be guided step by step in the process of creating a user-program by means of the VIMOS Kernel. The VIMOS Simulator will be used in this case, but exactly the same actions apply if you work directly on the camera. You will create and simulate the same user-program that you have seen in section "*3.2 Simple demo-program*".

Step 1 – prepare a workspace for your new project

Before you start creating a new program in the VIMOS Simulator environment some preparations should be done.

***INFORMATION**. On the camera all your user-programs, as well as some other important VIMOS information, are stored as files. Since the Simulator is actually a VIMOS Kernel that runs on a PC it uses the same means of storing information. The main difference, however, is that on the camera you have on your disposal only one place to store files, while on the PC you can have as much folders as you like. We recommend, however, that you place all files for a specific project in a designated folder, including all image files, the Editor file(s), etc. By doing so, when the time comes to transfer your user-program on a camera, you will have all files you need conveniently stored in one single folder and will minimize the chance of making mistake by forgetting to load an important file like, for example, the calibration set your user-program uses.*

Create a folder named `Simulator Demo` anywhere you wish. This folder will be your workspace and will hold all information required by your program. Now you must prepare the images your user-program will process. We have supplied 5 image files for your convenience. They are located in the `Getting started demo 1\Images` folder in the VIMOS demo folder. Please copy the folder `Images` to your workspace. Now you are ready to start the VIMOS Simulator and create your new program.

Step 2 – start the VIMOS Simulator

Start the VIMOS Simulator using the shortcut in the VIMOS program group. If your Simulator is still in a trial period click the **OK** button to close the info screen. What you see now depends on what you have previously done with the VIMOS Simulator.

***INFORMATION**. Upon exit the VIMOS Simulator writes information in Windows registry. When started again the VIMOS Simulator will restore the previous status upon exit. This means that your last workspace will be selected and your last user-program will be loaded. The last set of image files will also be loaded so you could continue your work from the point where you stopped.*

If you have just done the exercise described in section 3.2 you should see a screen similar to the one in Step 2 of that exercise. In case that you have loaded other user-program(s) you may see a different screen. In any case you can see the name of the workspace written in the window's title bar and the corresponding set of image files is loaded.

Step 3 – assign your project workspace

From the menu select **File > Change Workspace**. This will bring up the **Choose Workspace** dialog. Go in `Simulator Demo` folder you have created in step 1. Press **Open** button without changing the **File name**. When you successfully change the workspace you will see its name in the window's title bar. Now you should see no image in the simulation screen and the background of the screen should be black.

Step 4 – specify your camera model

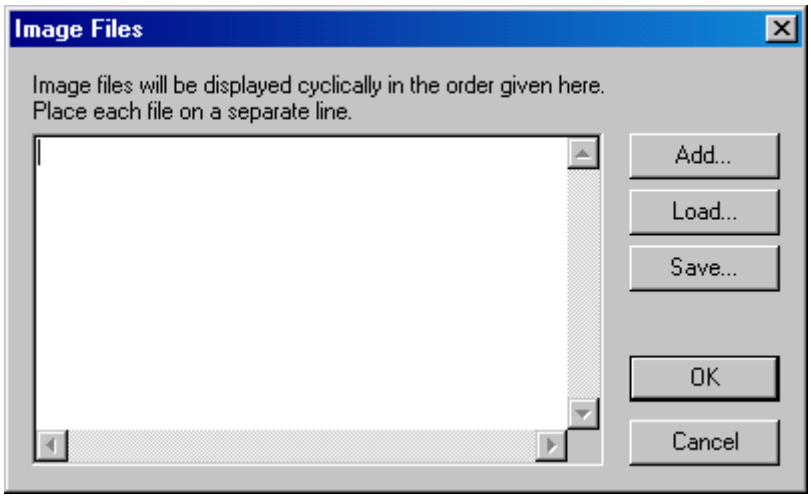
Next thing to do is to select the camera model. Although most VC cameras are very similar they still have some differences in the size of the image, the number of VIMOS tools supported, etc.

***ATTENTION**. To ensure best results and compatibility we strongly recommend developing your user-program in Simulator or in Editor using the same camera model as the one you plan to use in production.*

From the menu select **Camera > Simulation Options**. This will bring up the **Simulation Options** dialog, where you can select the camera model you wish to use. Please, select **VC38** as the camera model for this demo program and confirm with the **OK** button.

Step 5 - specify the set of image files to work on

Now you have to select the images for the demo program to work on. Select the menu option **Camera > Image source**. This will open the **Image Source** dialog with the option **Image file(s)** selected by default. Click the **Setup** button against **Image file(s)**.



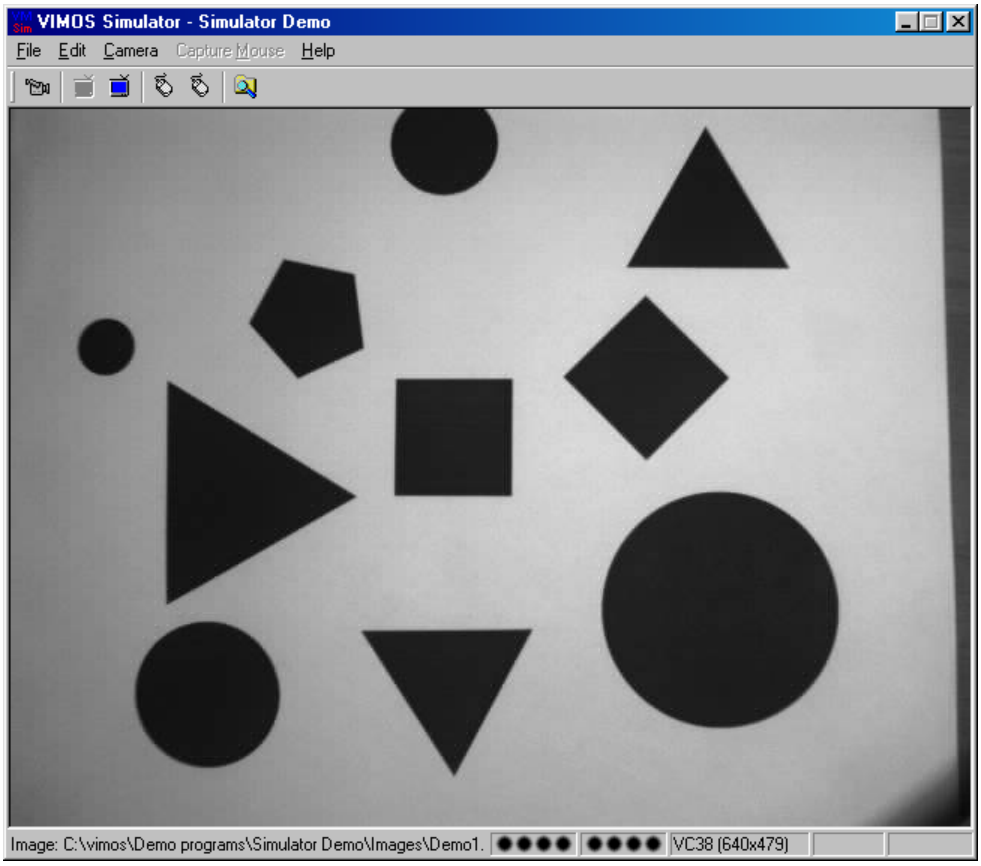
Press the **Add** button in **Image Files** dialog to open the file selection dialog. Go to the `Images` directory you created in step 1. Select all 5 image files and click the **Open** button.

***TIPS & TRICKS**. Hold down **`Shift`** key to select multiple files.*

Press **OK** button to close **Image Files** dialog.

When you are back in **Image Source** dialog make sure **Image file(s)** radio button is checked. Confirm your choice by clicking the sequence of **OK** buttons. The images will be processed in the order they are listed in the **Image Files** dialog. When the last image is processed the system will start processing the list from the first image again. After closing the **Image Source** dialog the first image from the list is displayed as background image in the VIMOS Simulator.

Your screen now should be like this one:



Step 6 – enter camera simulation mode

Now you are ready to start developing your user-program. To do so you have to enter the camera simulation mode by selecting **Camera > Camera Simulation** menu option or by pressing the **Camera Simulation** button  on the VIMOS Simulator toolbar. When you do so, VIMOS starts cycling through the images in the image list, i.e. it starts in run mode. Since there are no tools in your user-program yet, no image processing is actually carried out. The Windows mouse cursor disappears since the Simulator captures the mouse.

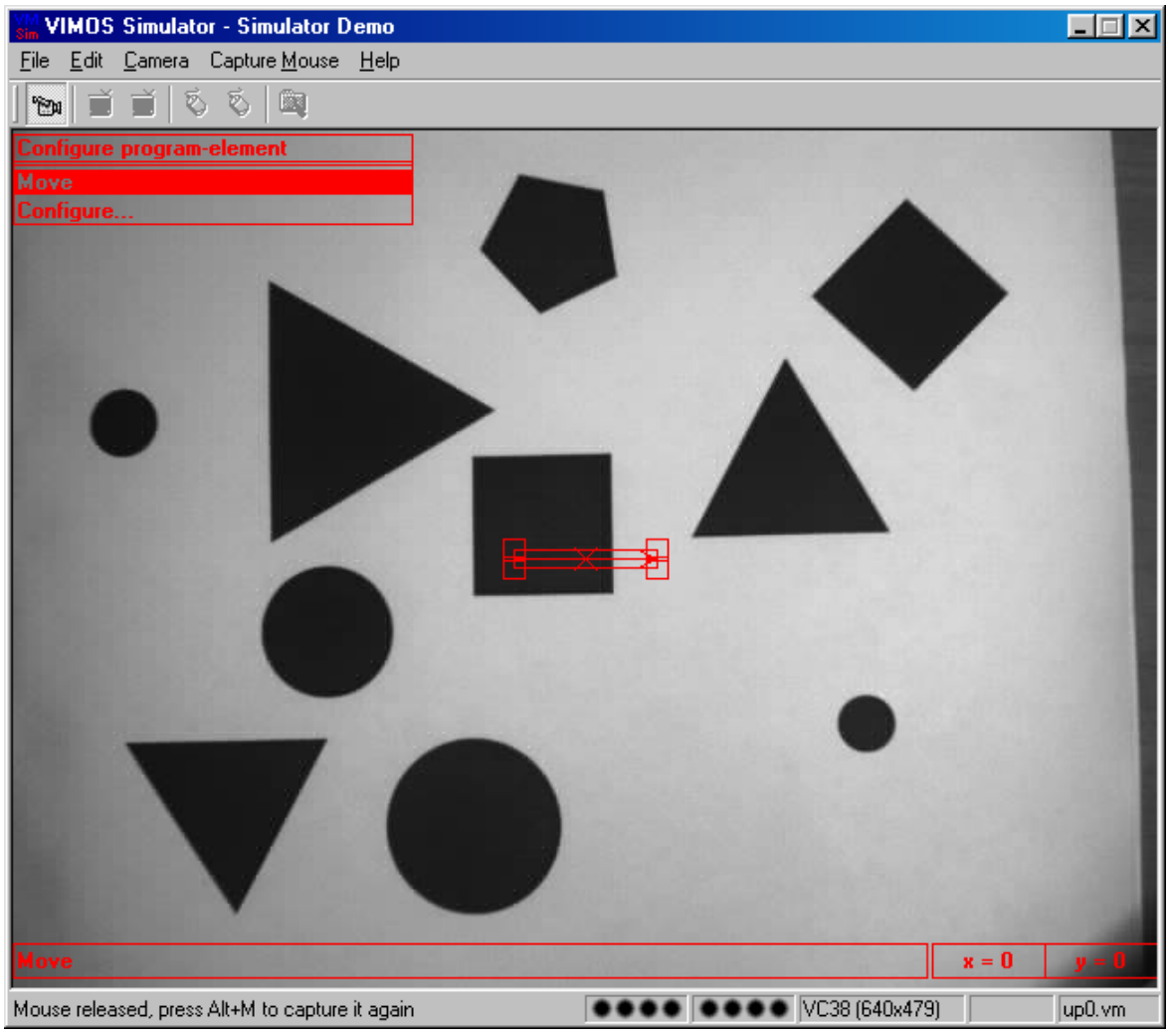
To start editing the user-program you have to exit the run-mode. Press the right mouse button to activate the run mode main menu. The only available option **Stop run-mode** is selected. Left click to exit run mode and activate edit mode. The image cycling stops and you are now in edit mode.

***ATTENTION**. If you experience difficulties working with the mouse, please read the section "3.1.1 VIMOS operation" once again. For more detailed information about how to use mouse with VIMOS please read the manual "Using the VIMOS Kernel".*

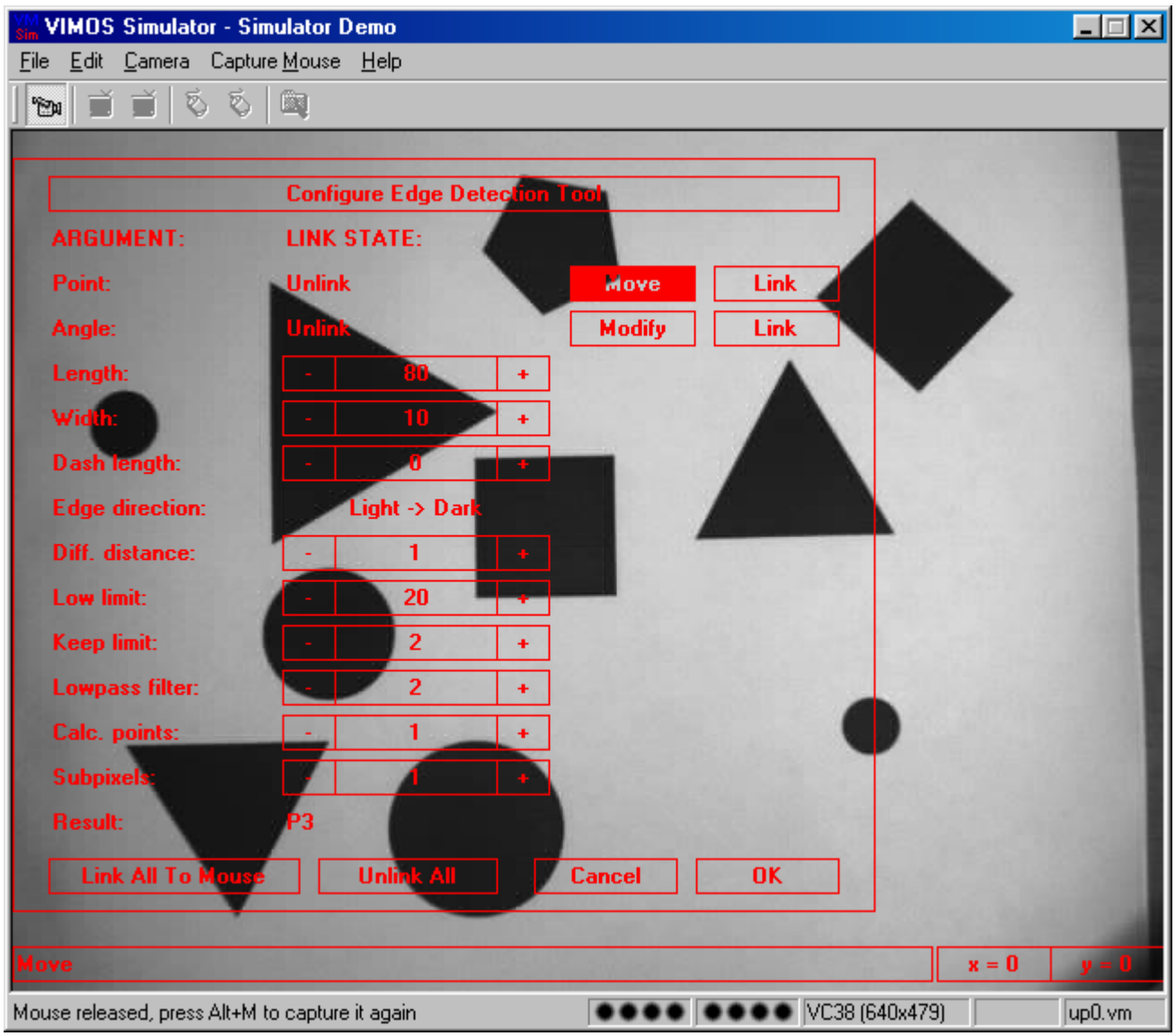
Please note that after you stopped run mode the background image you see may be any of the 5 images you have specified. It depends on when exactly you stopped the simulation process. This is not a problem for the current exercise since any of the circles will allow you to adjust the image processing tools.

Step 7 – add the necessary program elements

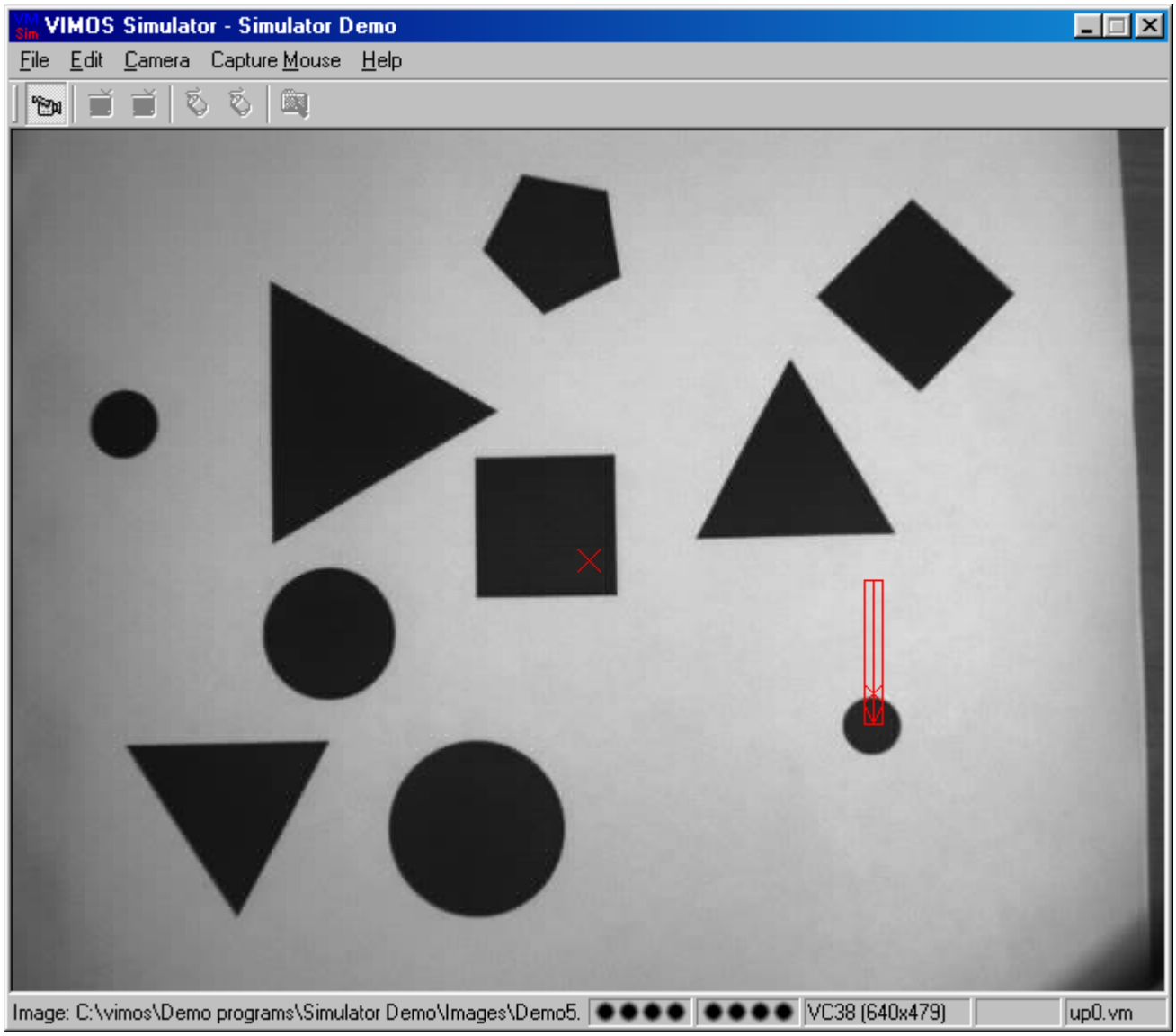
Press the right mouse button to open the edit mode main menu. By moving the mouse up or down select the **Edit user-program** option by left click and then successively select **Add program-element > Image-processing tools > Edge detection tool**. Now your screen should look similar to this one:



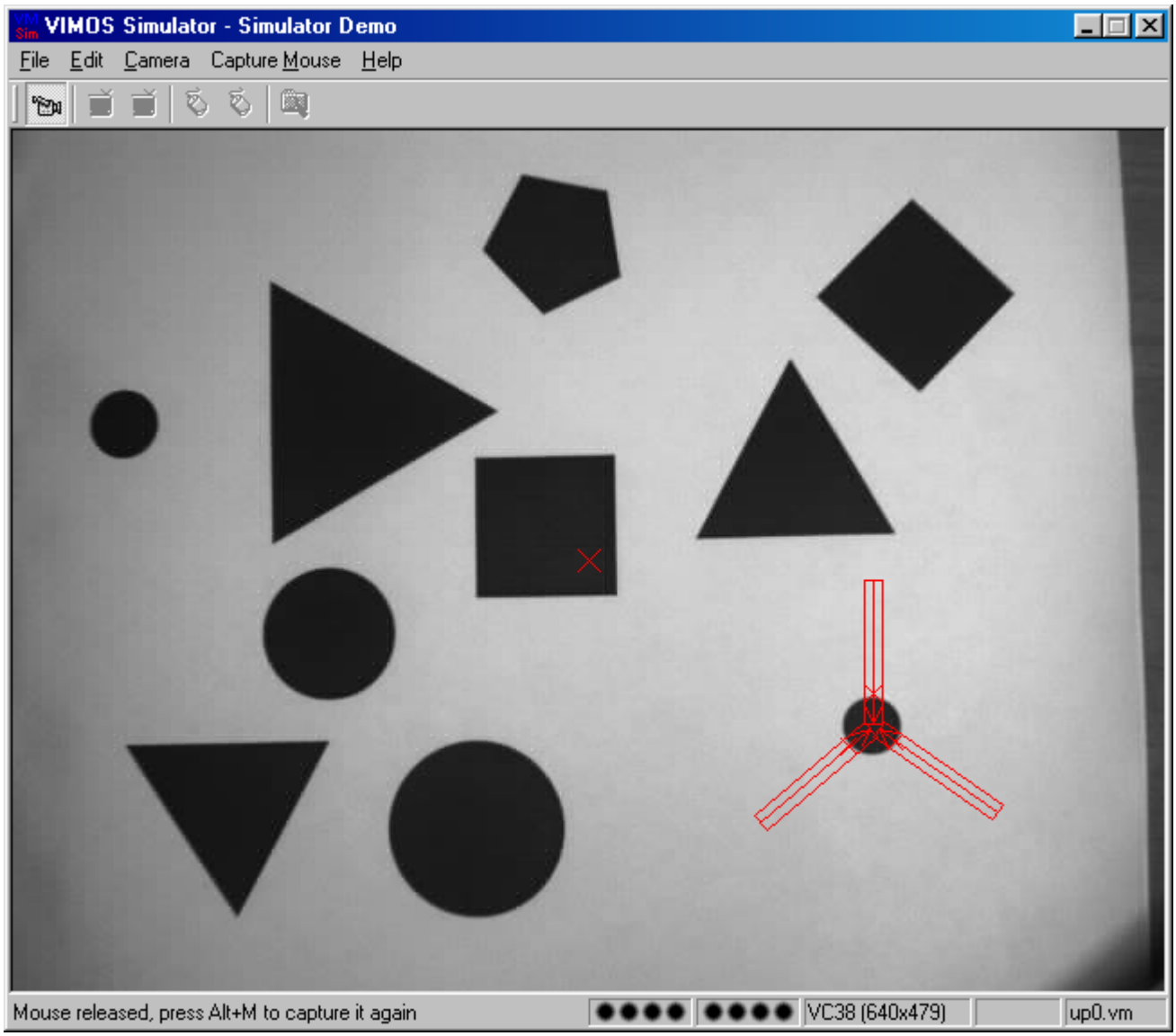
You can move the newly added tool by selecting **Move** from the menu and pressing the left mouse button. Alternatively, you can select **Configure…**, which will open the **Configure Edge Detection Tool** dialog.



For now you will use only two options from the configuration dialog – **Point > Move** and **Angle > Modify**. When you select **Point > Move** you can change the tool's position and place it over the object it is supposed to process. When you select **Angle > Modify** you can change the rotation angle of the tool by moving the mouse up or down. While doing this pay attention to the direction of the arrow, because this will be the direction in which the edge will be searched for. By combining **Point > Move** and **Angle > Modify** make the rectangle vertical, with the arrow pointing down and place it so that its lower end is close to the center of the circle as shown on the picture below:

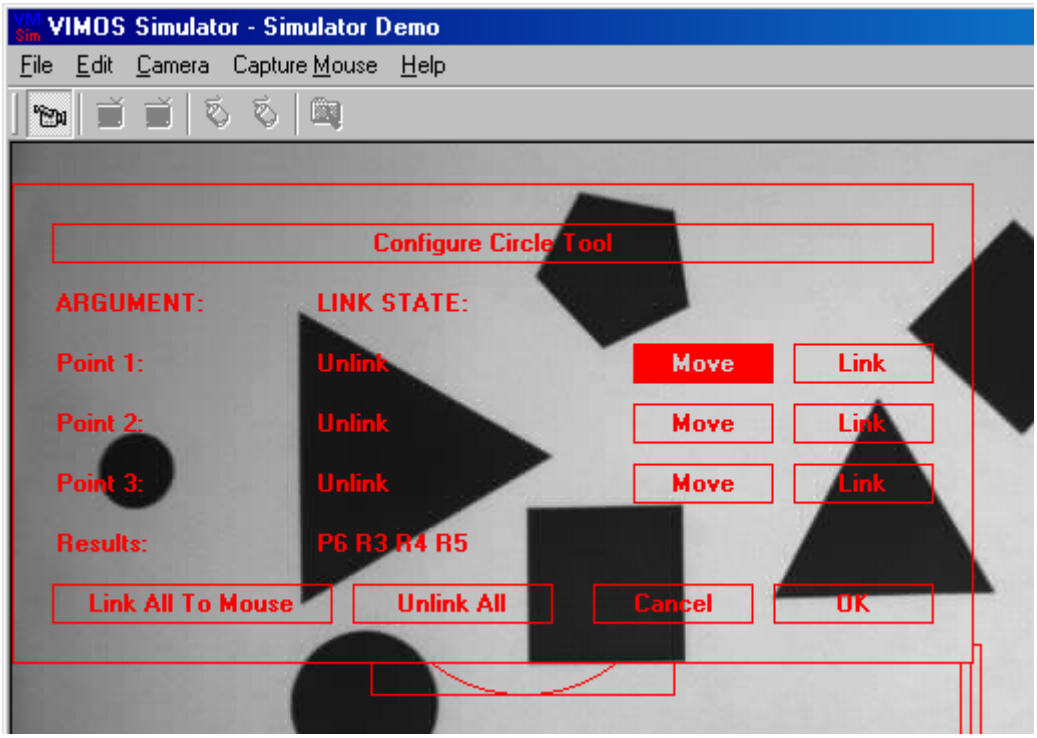


Repeat the above actions to add and modify two more edge-detection tools. After you finish this, your screen should look similar to the one below.

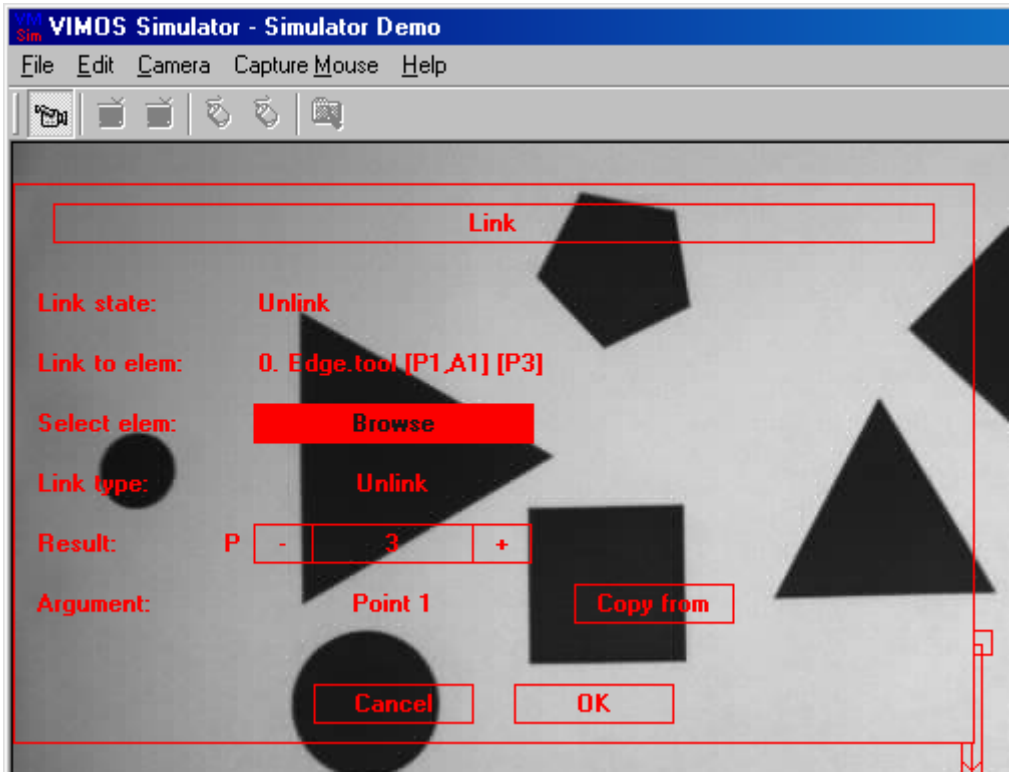


Now let's add one more tool – Circle Tool. It calculates a circle by 3 points. The results from the tool are: Circle's center point, radius in pixels, radius in real world units and circle's area in real world units.

To add the Circle Tool select the **Edit user-program** option from the edit mode main menu and then successively select **Add program-element > Graphics & Calculations > Circle tool**. Now select **Configure…** and you should see the **Configure Circle Tool** dialog box. Here you can configure the points, which are used to compute the circle. For the purpose of our demo we will link each of these points to the results of the 3 edge detection tools.

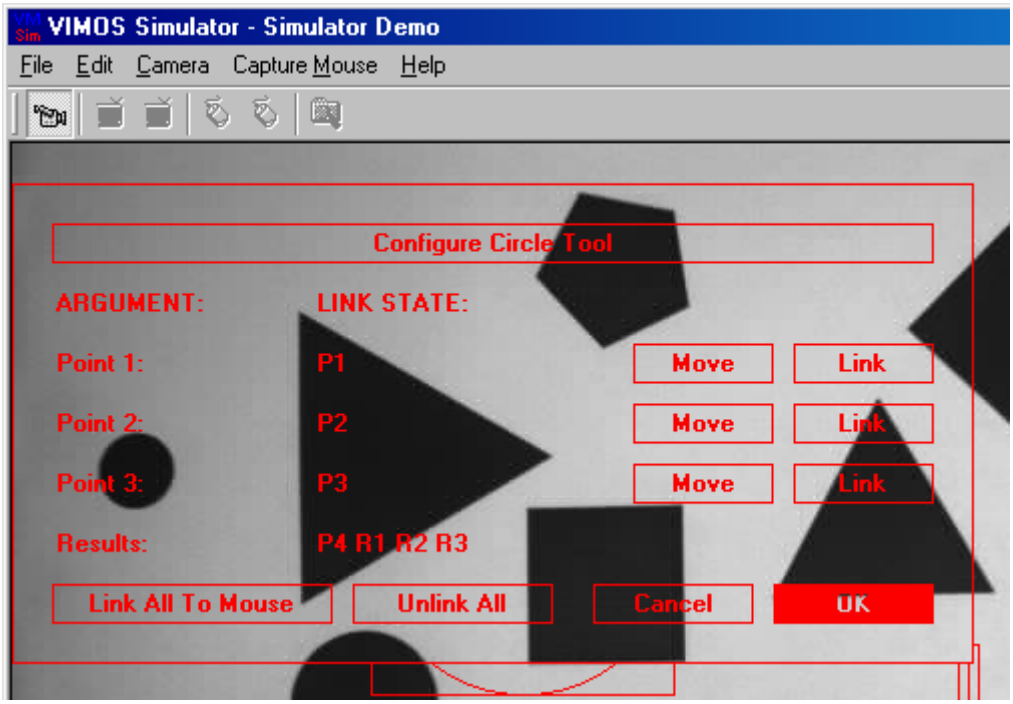


To link the first point select **Point 1 > Link** by moving the mouse up or down and pressing the left mouse button. In the next dialog – **Link** dialog – you have to select the program-element, which will provide the input value to the circle tool. For Point 1 this will be Edge Detection tool 1 which is referenced in the user program as program element 0.

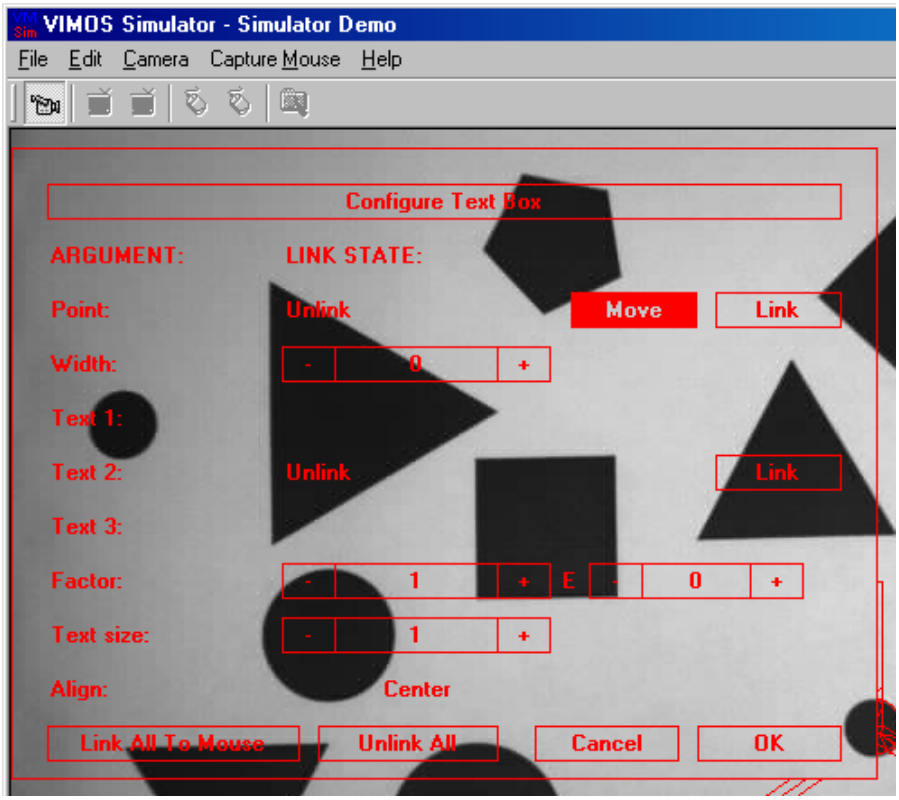


If the program element you intend to use is not selected by default you have to select it by clicking **Browse** and then selecting the correct program element from the list. When you select a program element the first of its results is displayed by default in the **Result** field. Since the edge detection tool returns only one result you cannot change the result number. To actually link the input argument with the result of the edge detection tool you have to change the **Link type** from **Unlink** to **Link to result**. If you now press **OK** your changes will be applied. You have to repeat these steps for the remaining two

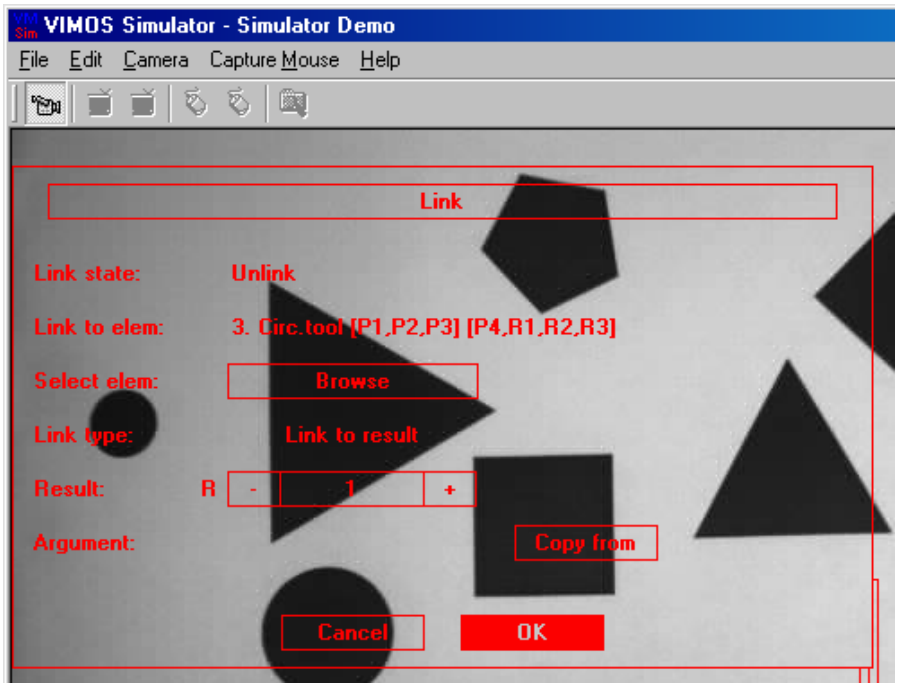
arguments. You have successfully configured the circle tool when **Point 1** is linked to **P1**, **Point 2** is linked to **P2** and **Point 3** is linked to **P3**. Finally the **Configure Circle Tool** dialog must look like the one below:



Now we need to add something to show us the result of the calculations. We want to see only the radius of the circle in pixels. What we need is a *Text Box*. Select the **Edit user-program** option and then successively **Add program-element > Graphics & Calculations > Text Box**. By selecting **Move** from the menu or **Point > Move** from the dialog you should be able to place the text box somewhere on the screen, where it will be well visible. Now you have to configure its properties. Open the **Configure Text Box** dialog.



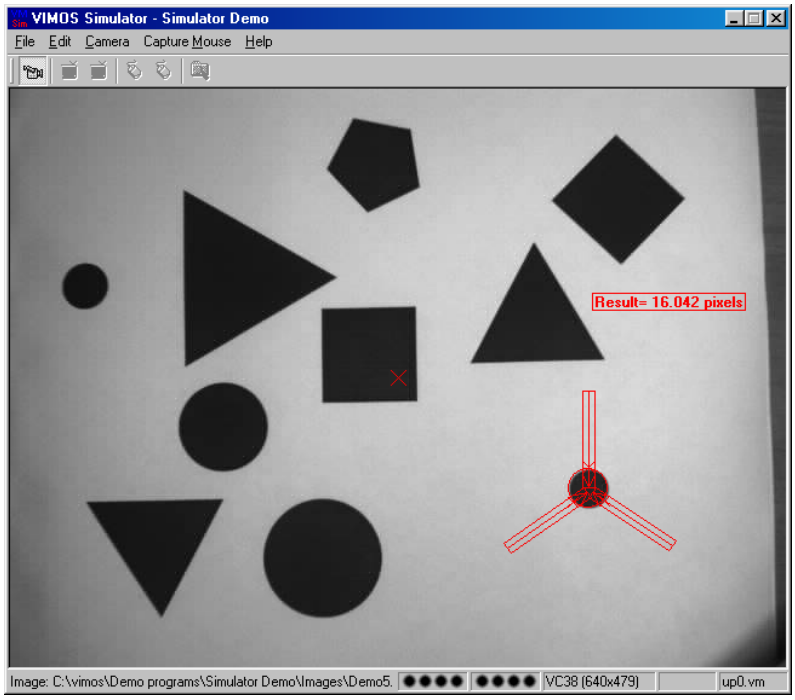
By moving the mouse up or down you can select the text field of the **Text 1** and **Text 3** parameters. When **Text 1** or **Text 3** is selected you can repeatedly press the left mouse button to change the actual text assigned to the corresponding parameter. The actual text is selected from a predefined set of possible strings like “Result=”, “Angle=” etc. We suggest that you select “Result=” as text to be displayed before the result itself (**Text 1**). Also you may select “pixels” as **Text 3**, which text will be displayed after the actual result value. Then you have to link **Text 2** with the result of the Circle Tool. By browsing the program-elements you can select Circle tool as **Link to element** and then select **R1** as result to be displayed. Link type must be **Link to result.**



Apply the **Link** dialog settings by pressing the **OK** button.

Step 8 – start user-program simulation

If your screen is now similar to the one below then you are ready to start simulating your first VIMOS program.



To see it working you have to press the right mouse button and then select **Start run-mode.** You should now see the background image change and with every change the text displayed should change too, showing you the radius of the circle found. If you don't have a correct result for each circle due to the positions of the edge detector tools you can reposition them by entering edit mode again and selecting **Edit user-program > Configure program element** from the edit mode main menu. Then in the browse window you must select the tool you wish to reposition and to change its position by clicking the **Move** menu option.

Step 9 – stop user-program simulation and store your program to file

To stop the user-program execution, right click to open the run mode main menu. The only available option **Stop run mode** is selected by default. Left click to change from run to edit mode. Now you are in edit mode and the user-program simulation is stopped. Right click again to activate the edit mode main menu. Select the **Save user program to file …** option and leave the default file name – `up0.vm`. Confirm the operation by clicking the **OK** button. Now your user-program is saved to the `up0.vm` file and can be later edited or simulated again.

Step 10 – send the user program to the camera and run it there

This step is for information only. You are not expected to work with the camera yet. However this is to remind you that VIMOS is mainly designed to run in the camera environment.

Step 11 – exit the VIMOS Simulator

Right click to open the edit mode main menu. Select the **Exit** option by moving the mouse up or down and left click to activate it. When you exit the camera simulation mode the VIMOS Simulator releases the mouse and the mouse pointer appears on the screen. Now the VIMOS Simulator can be controlled as any Windows application and can be closed by **File > Exit** menu command.

3.4. Create a user-program in the VIMOS Editor

PREMISE *In this section you will learn how to start VIMOS Editor, add tools to a new user-program, configure them and run the newly created user-program in the Simulator. You will also fine-tune your program in the Simulator environment and will transfer the changes back to the Editor.*

In this section you will be guided step by step in the process of creating a user-program by means of the VIMOS Editor. You will create and simulate the same user-program that you have seen in section 3.2 and you have created in section 3.3. Finally you will learn how the VIMOS Editor and the VIMOS Simulator interact with each other in order to fine-tune a user-program.

Step 1 – create a workspace for your project

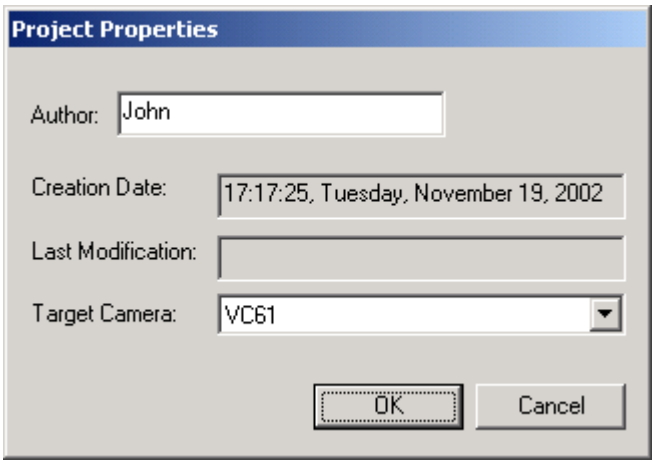
As we have already mentioned in section *"3.3 Create your first VIMOS program"* it is a good idea to have a separate workspace for each project. Create a new folder named `Editor demo` anywhere you wish. This will be the workspace for your Editor program.

Step 2 – prepare the images for the simulation step

While you work with the Editor you do not need images to process. However you will need image files when you decide to simulate and fine-tune your program. We have supplied 5 image files for your convenience. They are located in the `Getting started demo 1\Images` folder in the VIMOS demo folder. Please copy the folder `Images` from `Getting started demo 1` to your workspace. Now you are ready to start the VIMOS Editor and create your new program.

Step 3 – start the VIMOS Editor and create a new program file

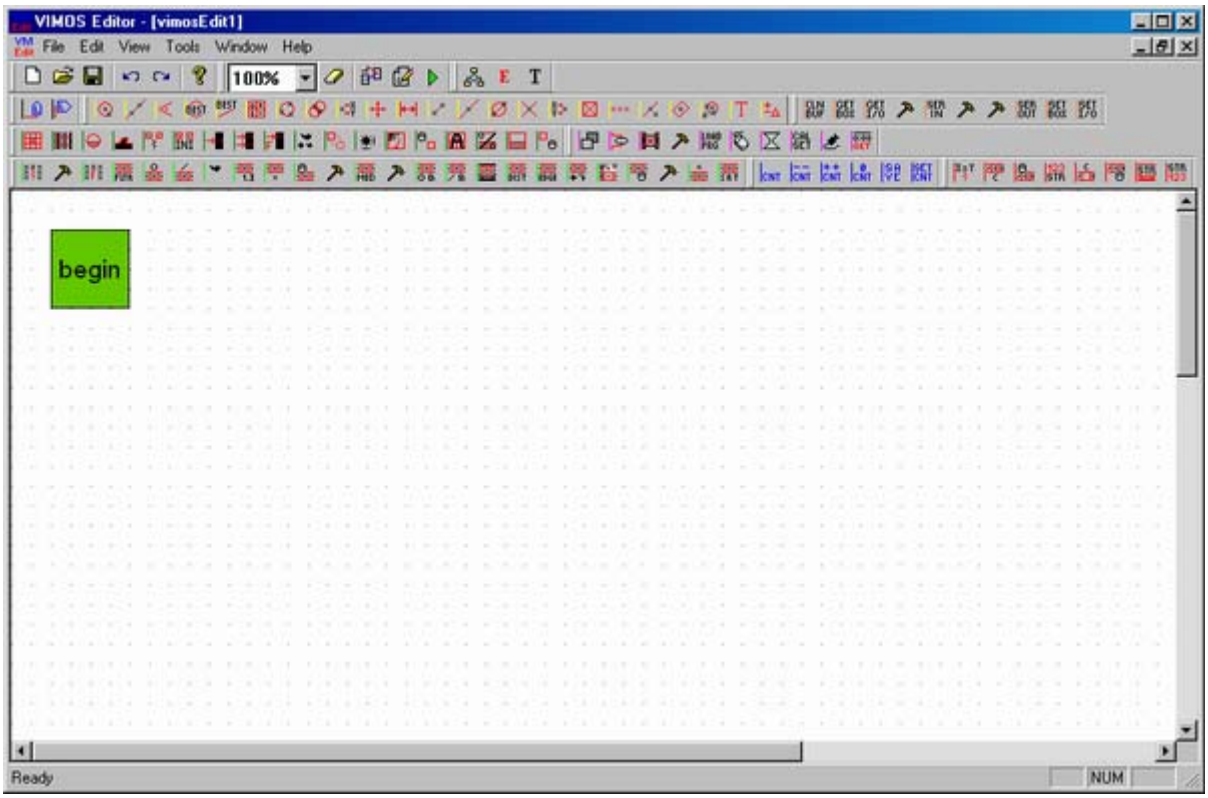
Start the Editor using the shortcut in the VIMOS programs group. If the Editor is still in a trial period press the **OK** button to close the info screen. Now select **File > New** from the menu or press **New** button  on the toolbar. The **Project Properties** dialog will appear.



Change the **Author** field if you wish and select **VC38** as the camera model for this exercise.

***ATTENTION**. To ensure best results and compatibility we strongly recommend developing your user-program in Simulator or in Editor using the same camera model as the one you plan to use in production.*

Confirm with the **OK** button. Now you have an empty program file and your screen should look like this one:



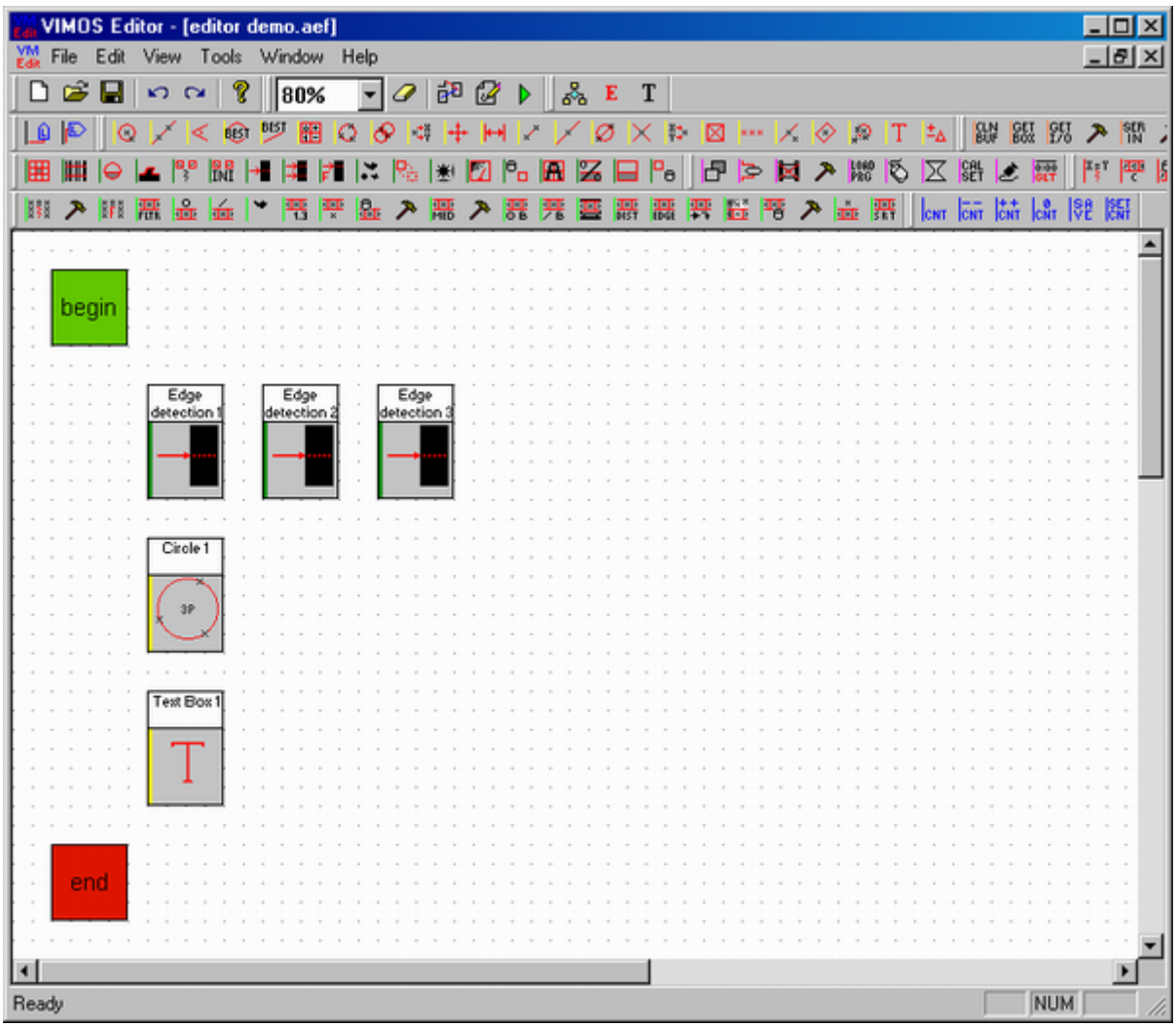
The *Begin* statement is only present to show where you user-program starts. The *Begin* statement is a special tool in the Editor only. It is not needed in the camera or in the Simulator. Select **File > Save as…** from the menu, go to your workspace directory (you have created it in step 1) and save the file as `editor demo.aef`. Now you can see the name of your program appear on the title bar of your screen.

Step 4 – add image-processing tools to your program

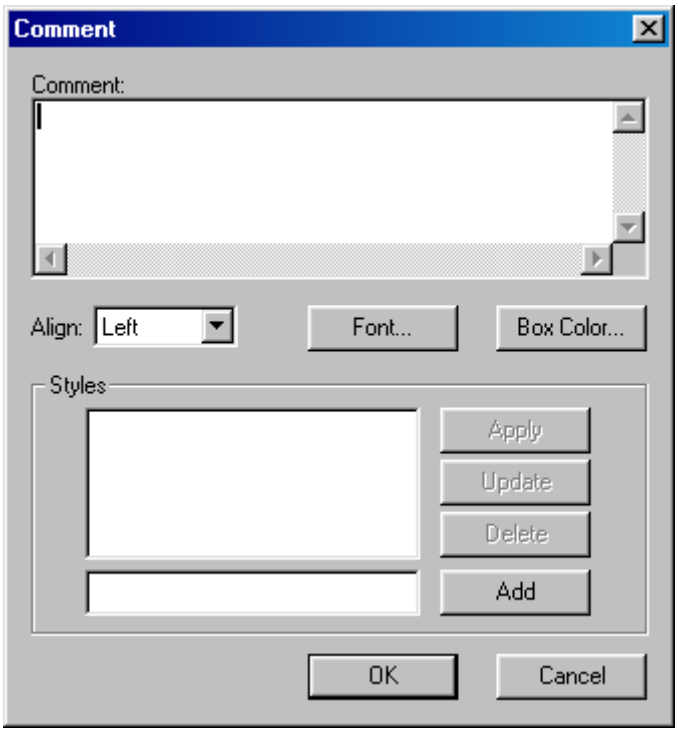
You can add tools by selecting them from the **Tools** menu or by pressing the appropriate tool button on the toolbar. When you select the tool to add the mouse cursor will change to a cross. Now you

must select the position of the tool on the diagram. The actual adding of the tool is done by selecting its location and left clicking the mouse. You move any tool at any time by dragging it with the mouse.

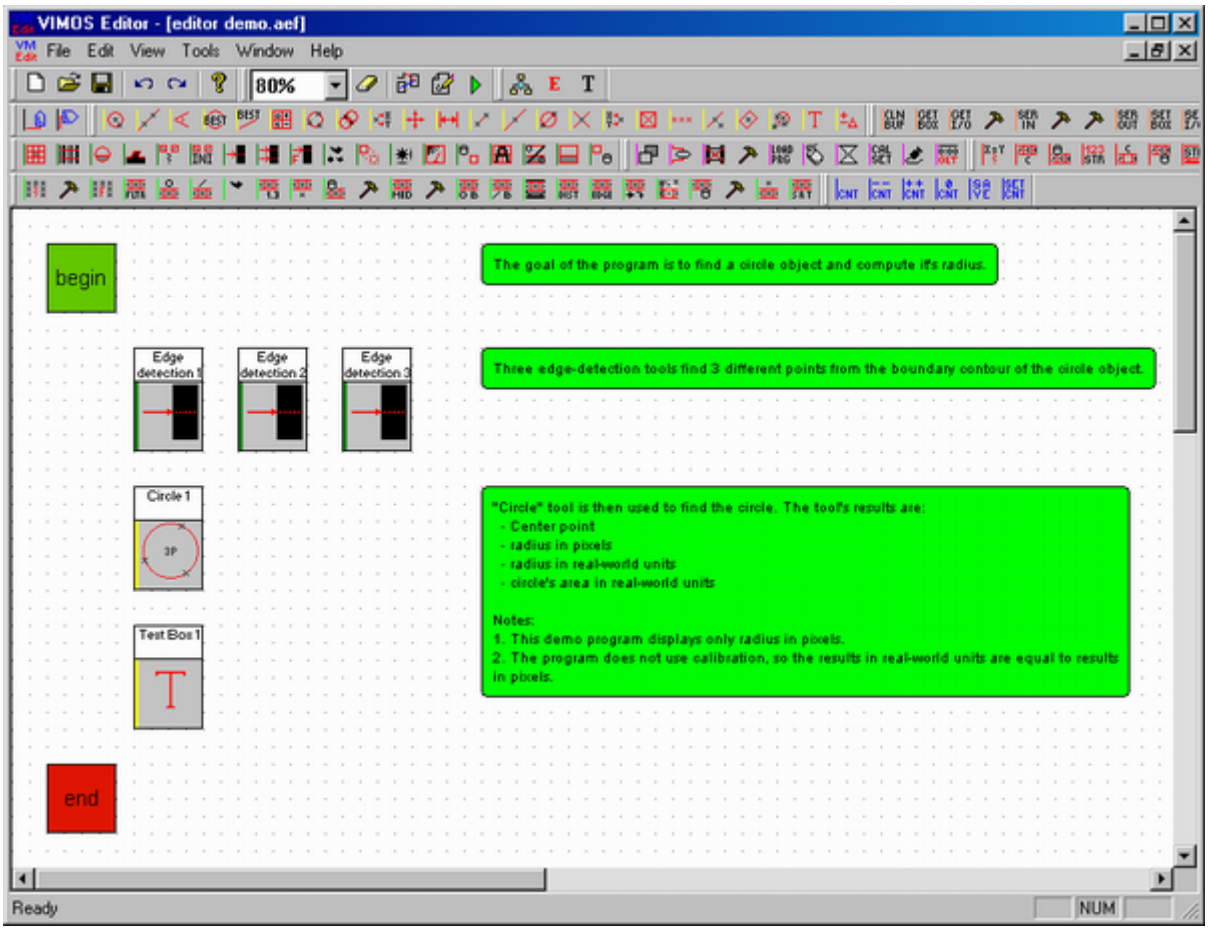
Add 3 *Edge detection* tools by menu selection **Tools > Image processing tools > Edge detection** or by pressing the **Edge detection** tool button on the toolbar. Add a *Circle* tool by menu selection **Tools > Graphics & calculation > Circle** or by pressing the **Circle** tool button on the toolbar. Add a *Text box* tool by menu selection **Tools > Graphics & calculation > Text box** or by pressing the **Text box** tool button on the toolbar. Finally you must add an *End* statement which is a special tool available only in VIMOS Editor to show where your program ends. Select **Tools > End** or press the **End** statement button on the toolbar. You can arrange the tools on the diagram as you prefer. However it is a good practice to structure your programs according to their functionality and to add comments for future reference. The following picture shows an example of tools arrangement.



Comments are added exactly the same way. Select **Tools > Comment** or press the **Comment** button on the toolbar. Select the position of the comment and left click the mouse. The **Comment** dialog will appear.

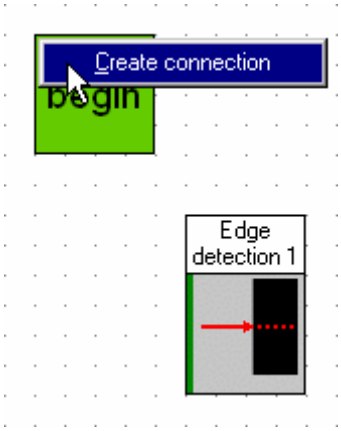


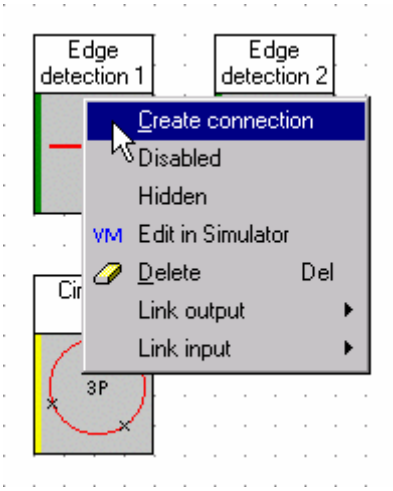
Fill in the **Comment** text field and edit the other properties if you wish to change the visual properties of the comment box. Confirm with the **OK** button. If it is necessary, move your comment by dragging with the mouse. An example screen is given below.



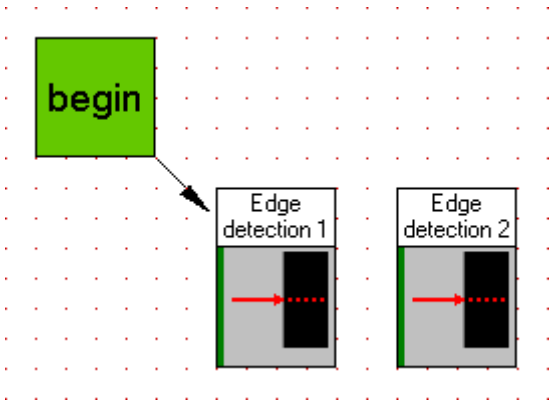
Step 5 – define the program flow sequence

Now you must define the sequence in which your tools will be executed. To do this, position the mouse pointer over the *Begin* icon and right click. A context sensitive menu will open where you must select the **Create connection** option.

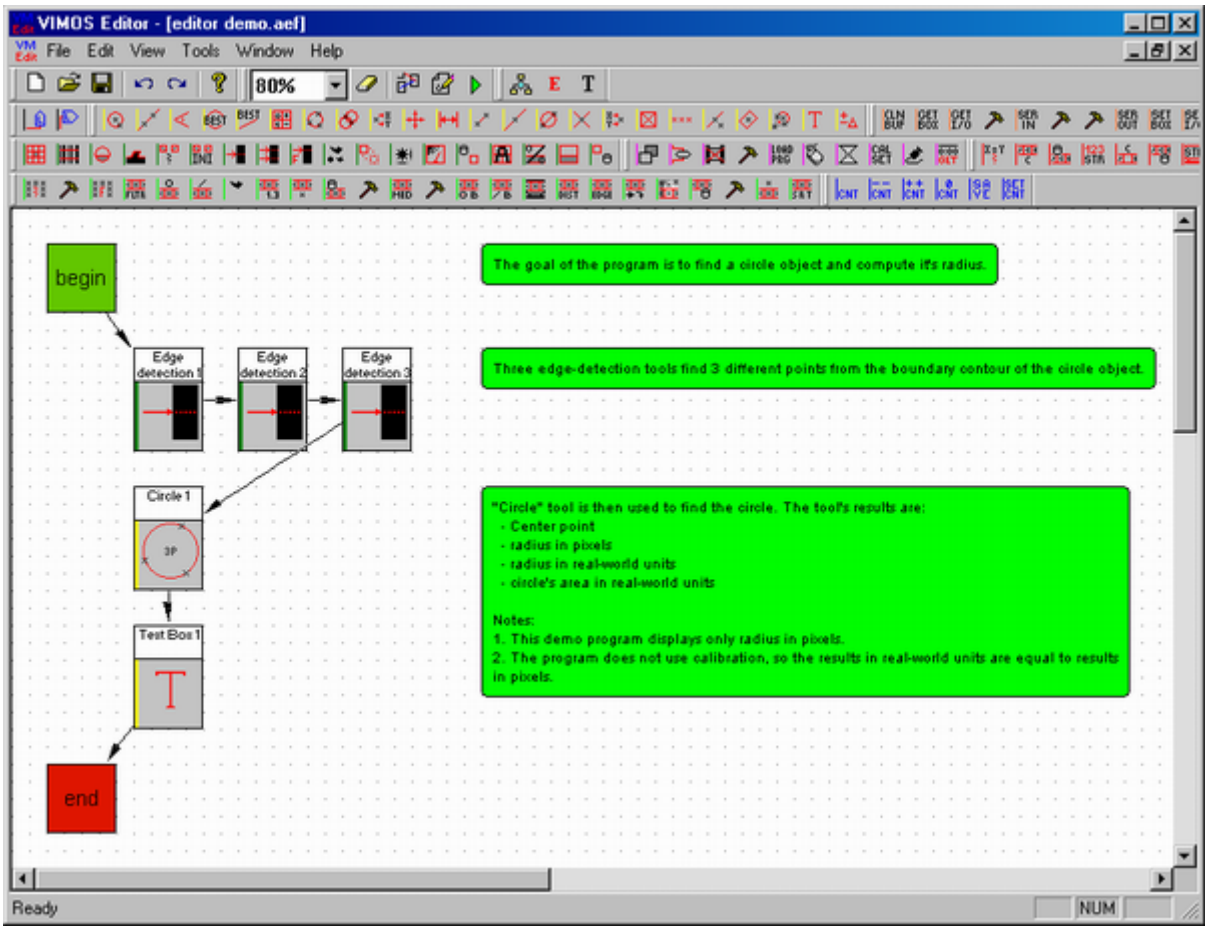




An arrow will appear that must be connected to the next tool in the sequence. Move the arrowhead to the next tool and left click to connect it.



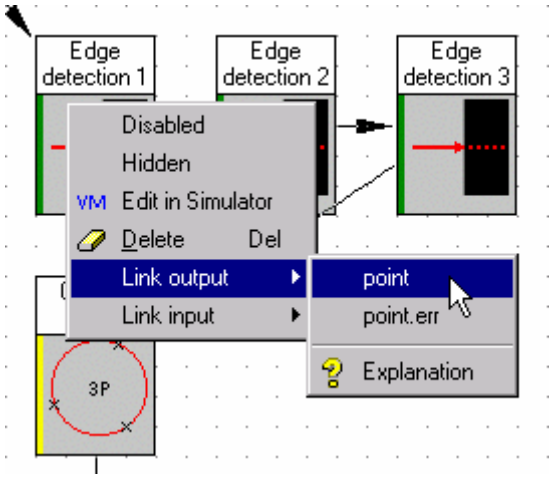
Repeat the above actions to define the whole program sequence. Please, be careful not to select the **Disabled** or **Hidden** options since this will alter the behavior of your program. Finally you must have the following result:



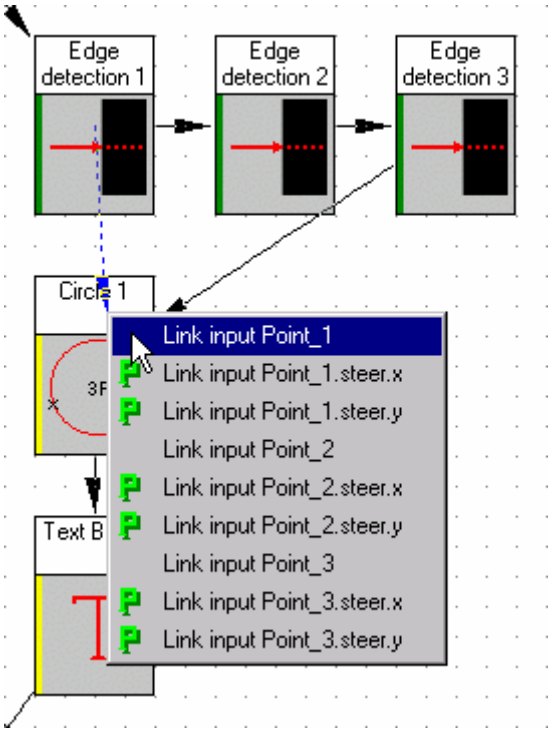
Step 6 – specify the logic links between the tools

In VIMOS programs preceding sets of tools usually generate information for the next tools until a final result is calculated. In this particular case the 3 edge detectors find 3 points that are used by the circle tool to calculate the characteristics of a circle. The circle tool in return supplies the circle radius to the text box tool, which displays the result in the overlay picture. This is a simple example but that is the way the VIMOS System basically works. Before simulating and fine-tuning your program you need to specify the logic links between different tools. You do so by linking a tool's output to another tool's input. You can alternatively link tool's input to another tool's output.

To link the result point of the first edge detection tool to the first input point of the circle tool, position the mouse cursor over the first edge detection tool and right click to open the context sensitive menu. Choose the **Link output > point** option.



A blue arrow will appear. You have to link it to the tool, which will accept the value as an input. The blue color shows that you have selected an output to link to an input. In case you select an input to link to an output the arrow color will be red. Position the arrow over the circle tool and left click. A context sensitive menu will appear that shows all the inputs of the circle tool. Note that you must link inputs and outputs of the same type only (point in this case). Choose the **Link input Point_1** option. Now the result point of the first edge detector will be accepted as the first input point for the circle calculation.



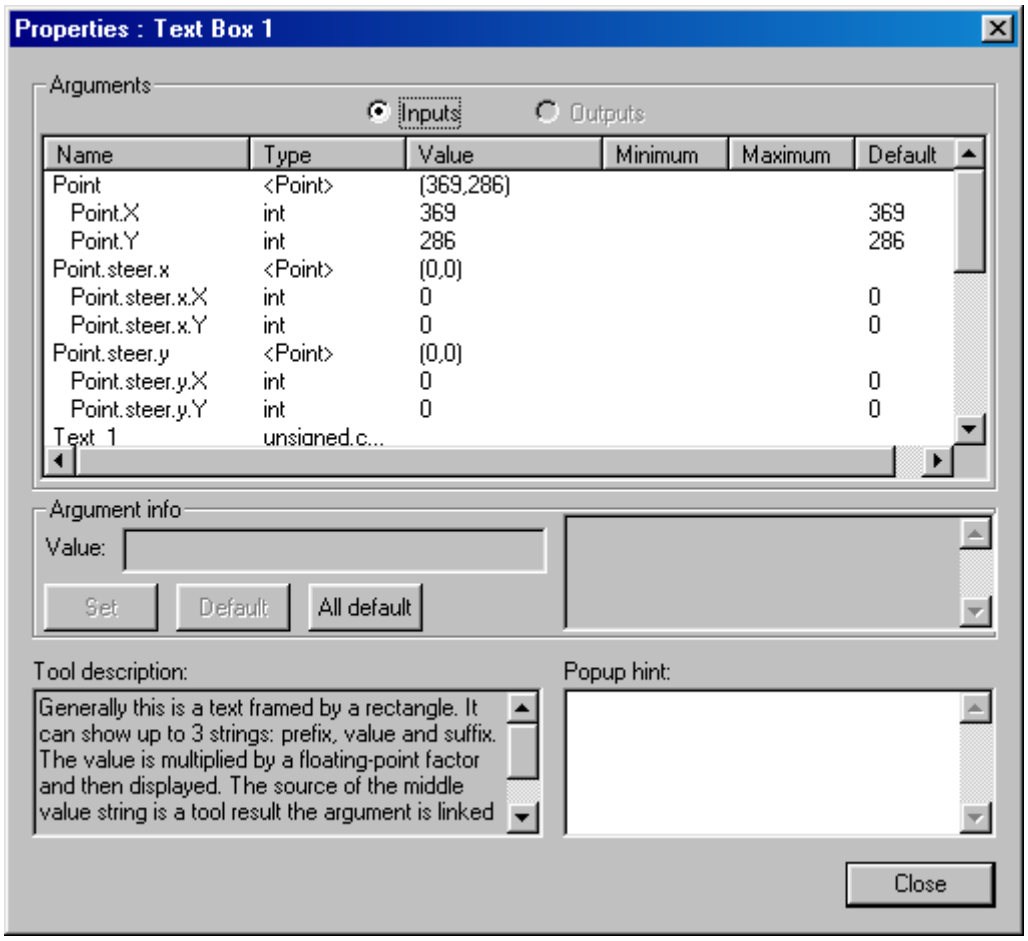
Repeat the above actions for the other two edge detectors selecting **input Point_2** and **input Point_3** in the *Circle* tool respectively. Now you have to define the logic link between the circle tool and the text box tool. Right click on the circle tool and select **Link output > radius1**. Link the radius with the only available option **Link input Text_2.float** of the text box tool. Now the circle radius in pixels (radius 1) will be displayed in the text box.

Step 7 – edit tools' properties

The tools' properties must be correctly adjusted in order to function properly. You can do this either in the Editor or in the Simulator environment. Some properties are more easily edited in the Simulator while others are better to be changed in the Editor. The Editor offers more convenient way to type in numbers and texts. Therefore tool's properties that may be adjusted by simply typing in numerical values or text must be changed in the Editor. Conversely in the Simulator one will find difficulties typing values and text but the Simulator has a very good interactive mechanism for moving, modifying angles, resizing and repositioning the tools on the background image. In the Editor you cannot see the image.

In this demo you must adjust the positions of the edge detection tools, the position and some text properties of the text box tool. Having in mind the above it will be better to adjust the text parameters in the Editor and the positions of all tools in the Simulator.

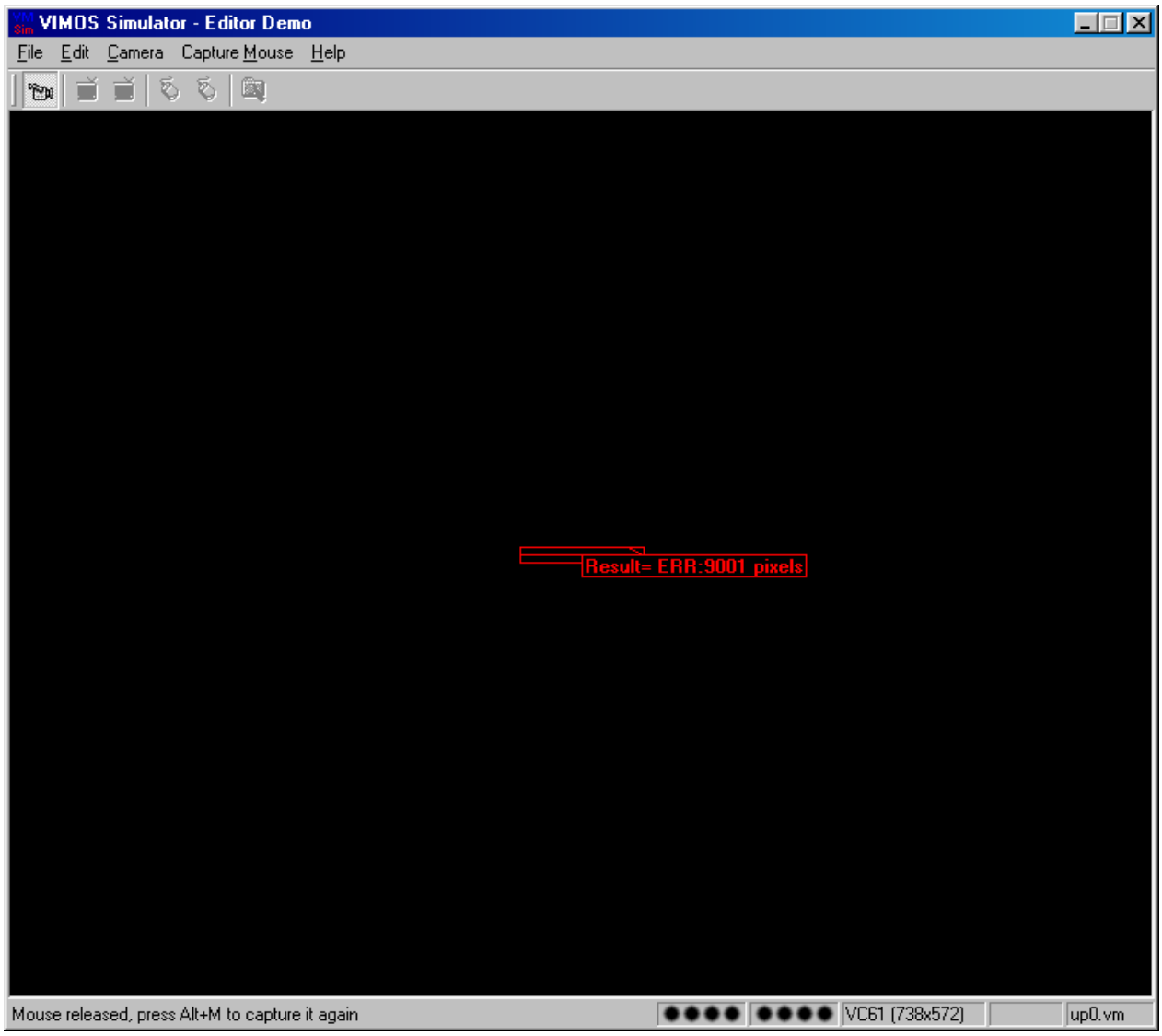
Double click on the text box tool. This will open the tool's **Properties** dialog.



Locate and left click on the **Text_1** parameter to select it. Type the string “Result=” (without the quotes) in the **Value** field. Press the **Set** button to actually change the **Text_1** value. Use the same actions to change the value of the **Text_3** parameter to “pixels”. **Text_1** is a prefix string that will be displayed before the result value while the **Text_3** string is a suffix that will be displayed after the result value. Close the dialog by the **Close** button.

Step 8 – fine-tune the program in the VIMOS Simulator

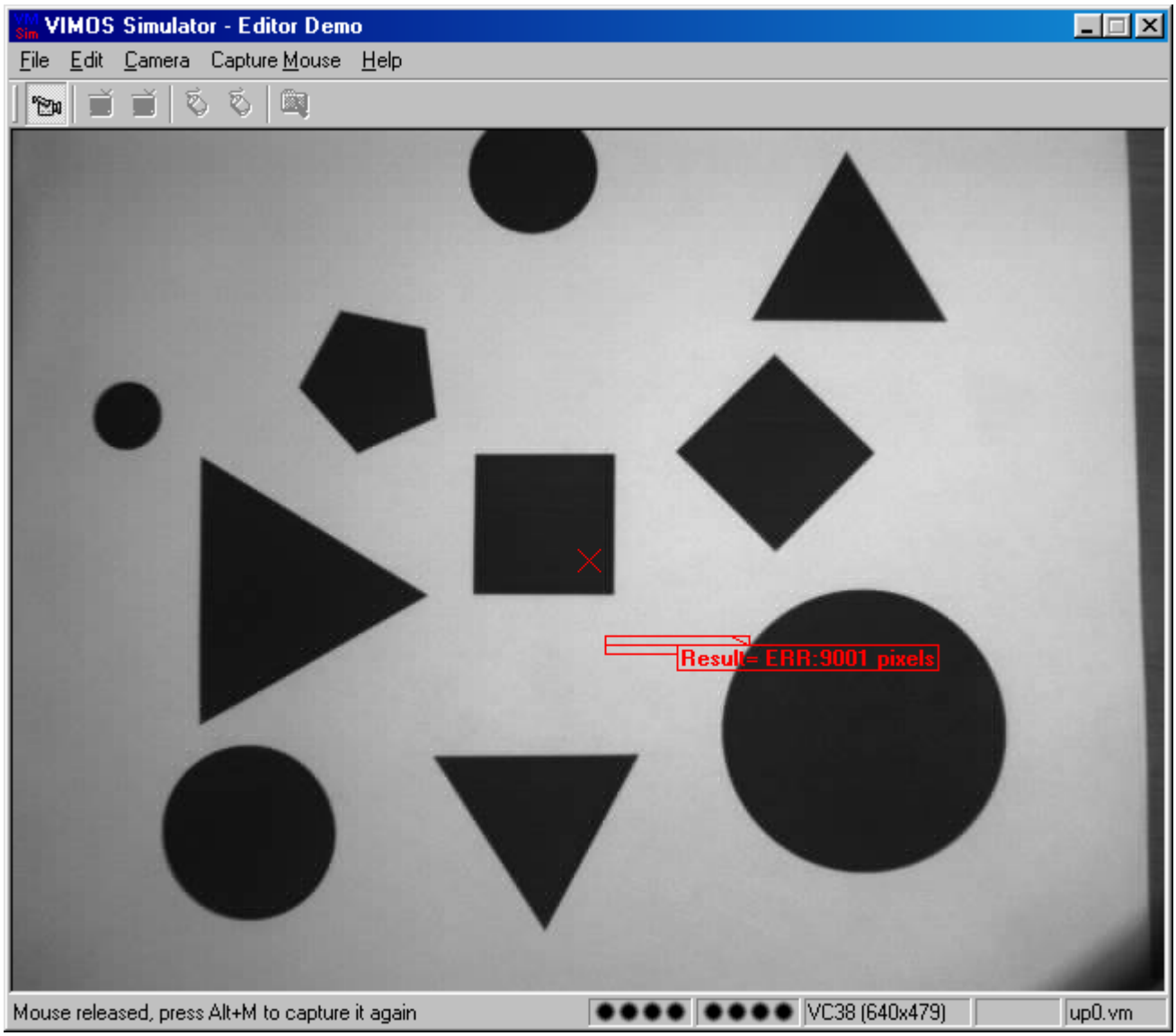
In order to run correctly your program still needs to adjust the positions of the tools on the screen. This is better to be done in the Simulator. Press the Run button  on the Editor’s toolbar. This should start the Simulator, load the user-program in it and start VIMOS in run mode. This is what you must see now:



As you can see there are still no image files defined. All tools have their default positions and are near the center of the screen. So what you must do first is to define image files for your program to process. Press **`Alt+M`** to release the mouse. The mouse cursor will appear on the screen. Select the **Camera > Image source** menu option. This will open the **Image source** dialog. Click the **Setup** button next to **Image file(s)** then press the **Add** button to open the file selection dialog. Go to the `Images` directory in your workspace and select all 5 image files.

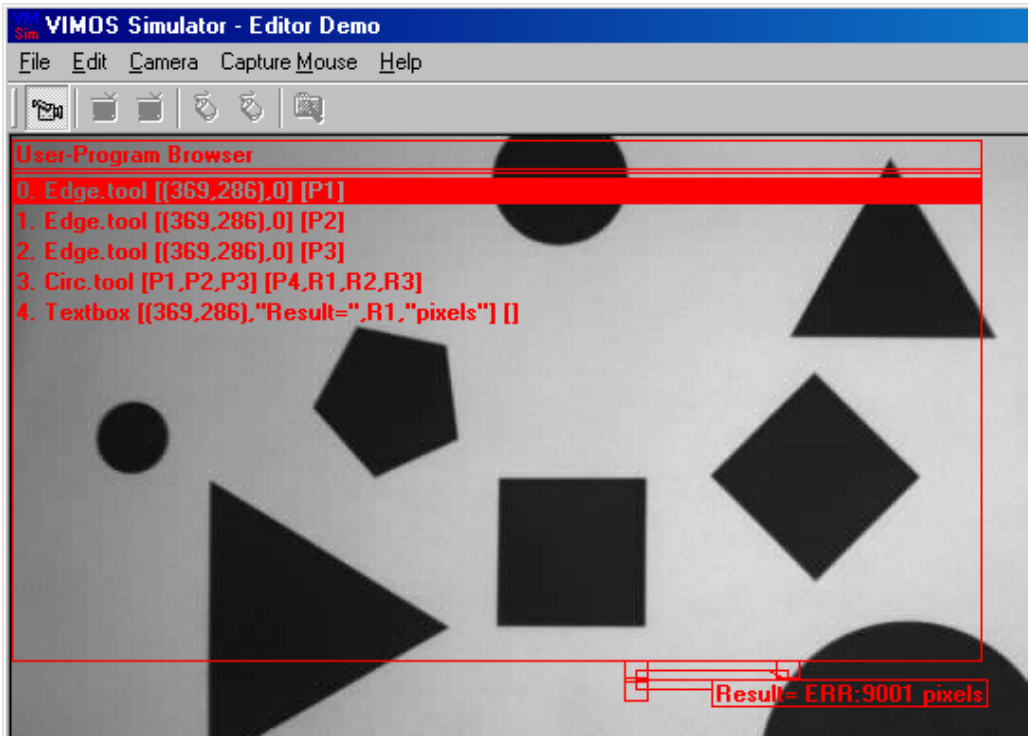
***TIPS & TRICKS**. Hold down **`Shift`** key to select multiple files.*

Confirm the sequence of **OK** buttons. Now you must see the background images change i.e. you are in camera simulation run mode. To be able to adjust tools' properties you must switch to edit mode. Press **`Alt+M`** to capture the mouse again. Right click to open the run mode main menu. Left click to activate the only available option **Stop run-mode**. Now the images cycling must stop and you are in edit mode. Your screen should look like this:

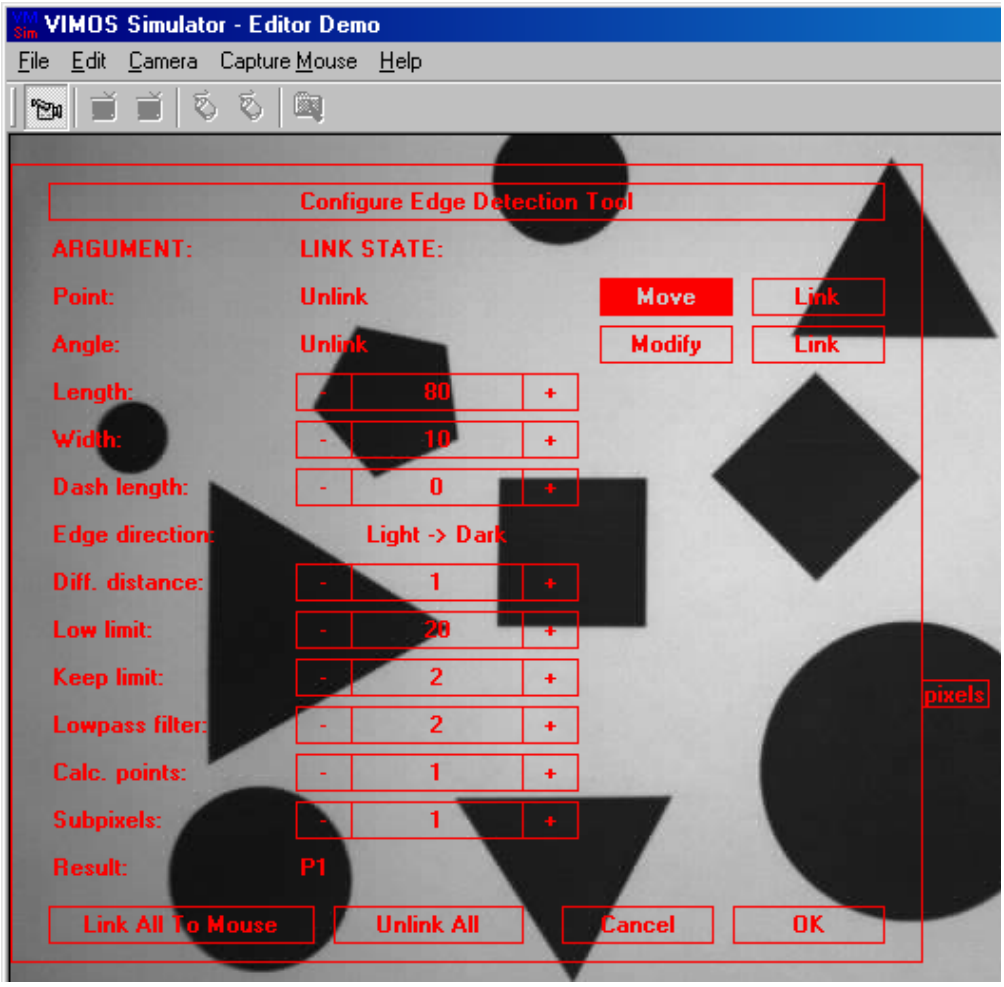


Note that your background image may be some of the other 4 images. It depends when you exactly stopped the run mode. This is not a problem for this exercise since each of the circles will allow you to properly adjust the image processing tools.

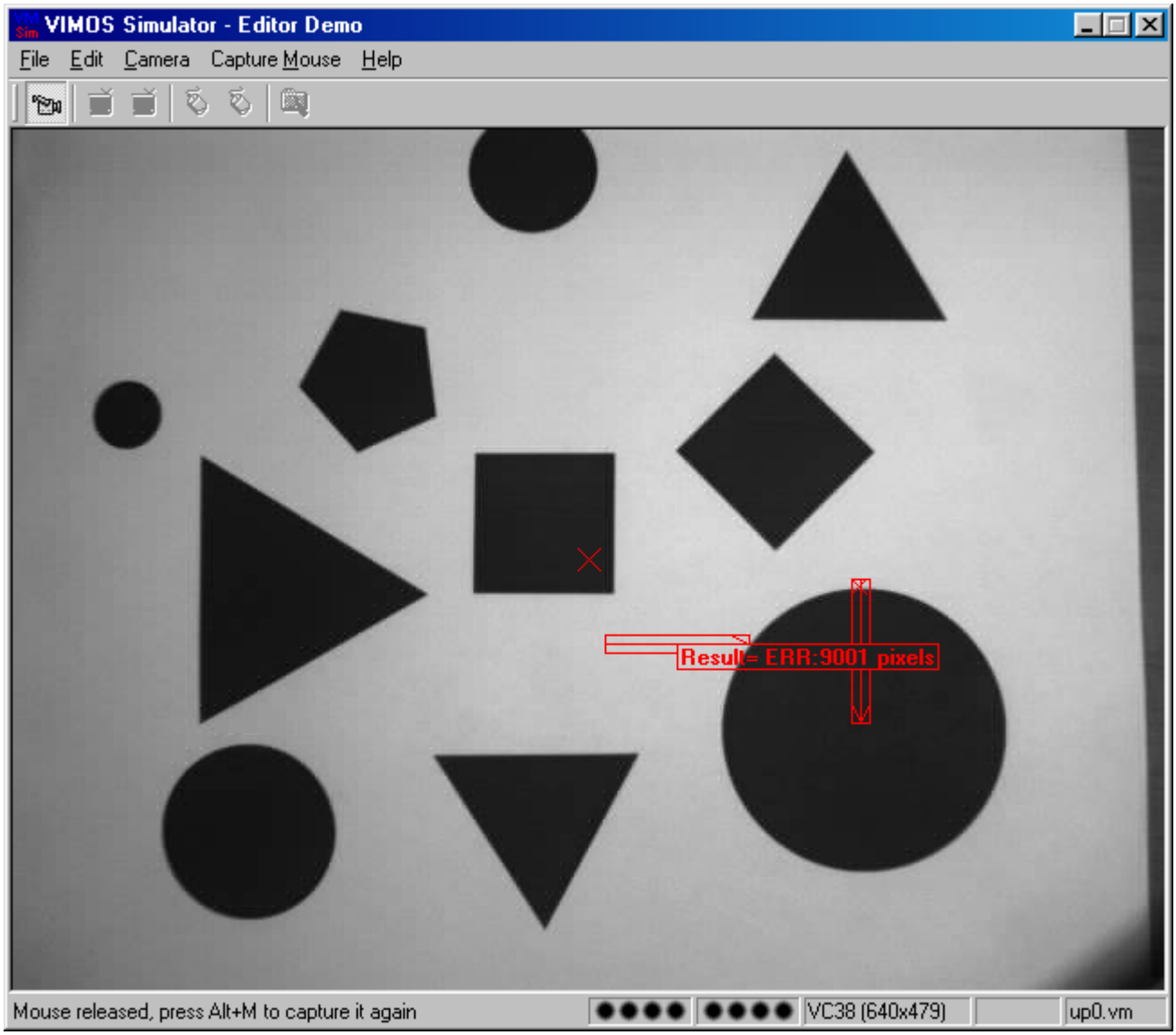
Now you will change the positions of the 3 edge detector tools. Right click to open the edit mode main menu. Successively select and left click the **Edit user program > Configure program element** menu options. The user-program browser window will open.



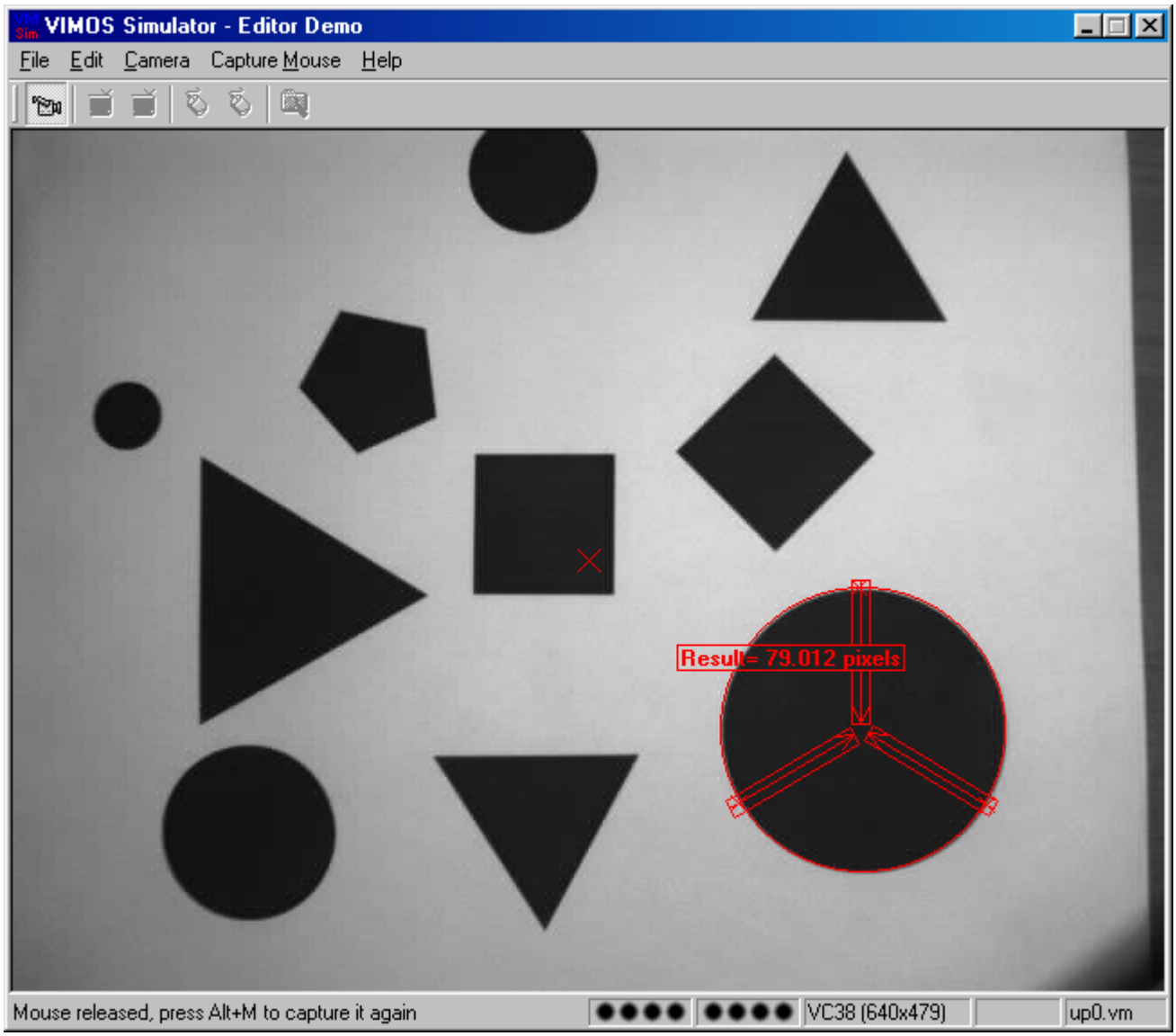
In the user-program browser window choose the first edge detector tool. Then select the **Configure…** option, which will open the **Configure Edge Detection Tool** dialog box.



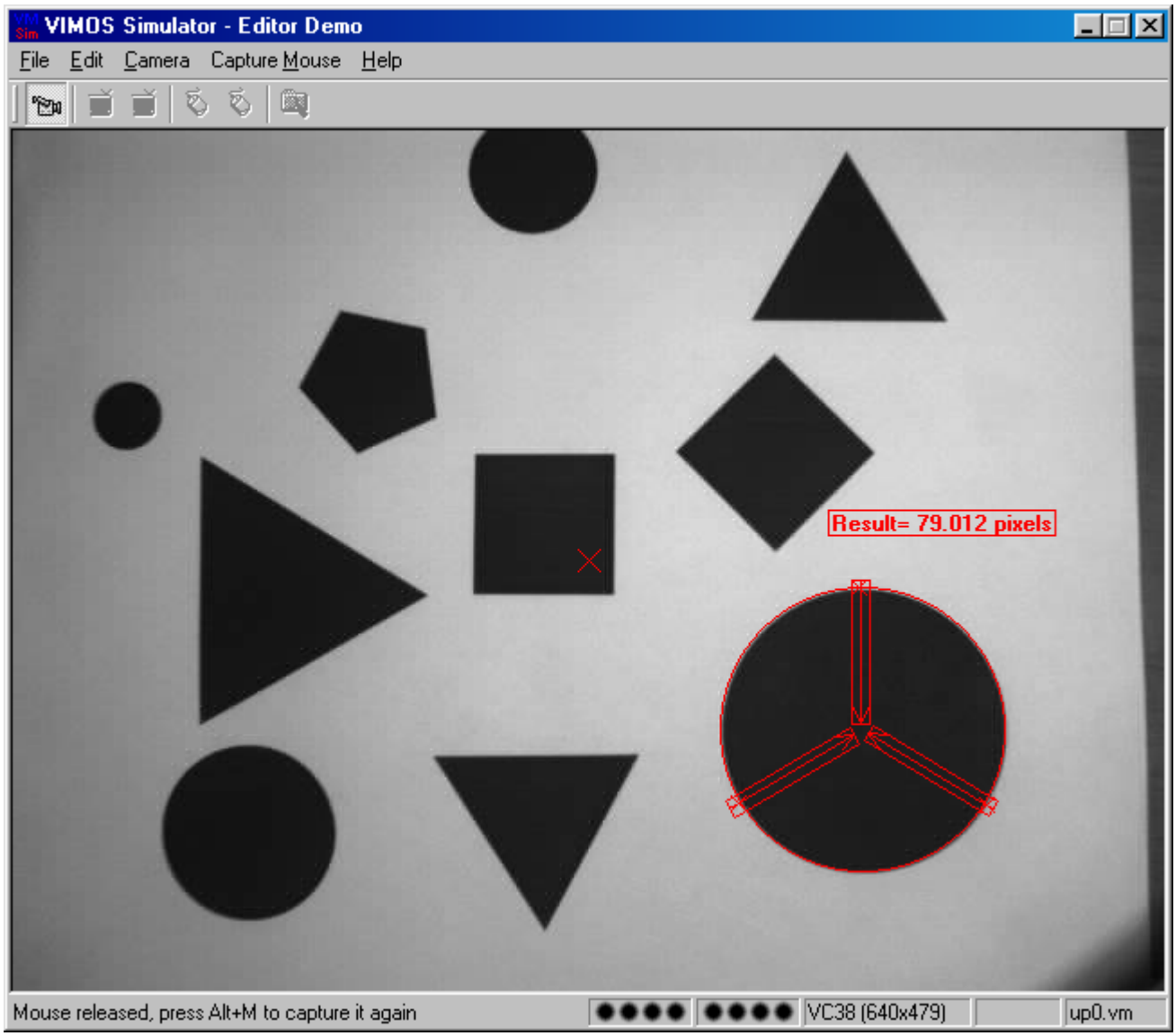
Use **Point > Move** and **Angle > Modify** buttons to make the edge detector vertical and pointing down. Place the edge detector as shown in the picture below.



Use the same actions to reposition the other 2 edge detectors as shown on the picture below.



Use the same actions to move the text box tool and locate it on a more suitable place. Finally your screen must look like the one below:



Step 8 – transfer the changes back to the Editor

It is very important to update the changes you have just made in the Simulator back to the Editor. Press **F6** to synchronize the Editor with the Simulator.

Step 9 – simulate your program

To start simulating your program right click first to open the edit mode main menu. Select and left click the **Start run-mode** menu option. Now you should see the different image files being loaded and processed. The result must also change according to the current radius of the circle. If you do not have a correct result in each case due to the positions of the edge detectors you can stop the simulation and reposition the tools by selecting **Edit user-program > Configure program element > … > Move** menu options.

Step 10 – stop the simulation and exit the Simulator

Right click to open the run mode main menu. Select and left click the **Stop run-mode** menu option. Right click to open the edit mode main menu. Select and left click the **Exit** menu option to stop the camera simulation mode. Now the mouse is released and the mouse cursor appears on the screen. Close the VIMOS Simulator as any other Windows application.

Step 11 – save your work and exit the VIMOS Editor

Do not forget to finally save your work before closing the VIMOS Editor.

3.5. What to do next

We suggest that you now spend some time familiarizing yourself with the tools available in VIMOS. Most useful for this purpose are the *“Tools’ descriptions”* and *“VIMOS programming”* manuals.

4. Documentation

4.1. Structure

VIMOS documentation consists of 13 independent manuals, which cover different aspects of VIMOS usage. Here is a short list of all available manuals:

🕮 *Installation guide*

Includes information about installing VIMOS on a PC system as well as on a camera.

🕮 *Using VIMOS on TI camera*

Describes installation and usage of VIMOS on TI camera.

🕮 *Using VIMOS on sensor camera*

Describes installation and usage of VIMOS on sensor camera.

🕮 *Using VIMOS on ADSP camera*

Describes installation and usage of VIMOS on ADSP camera.

🕮 *Getting started*

The basic information you should know before starting to experiment with VIMOS.

🕮 *Using the VIMOS kernel*

How to work with VIMOS kernel on camera and PC simulator.

🕮 *PC & Camera*

How the camera and the PC interact.

🕮 *Resources*

Description of resources available to the user.

🕮 *Using the Simulator*

How to use the Simulator to develop and test user-programs and to communicate with the camera.

🕮 *Using the Editor*

How to use Editor to develop and test user-programs.

🕮 *VIMOS programming*

Detailed explanation about VIMOS kernel and how to develop user-programs.

🕮 *Tools description*

Descriptions of all the tools available in VIMOS together with some useful tips & tricks.

🕮 *Examples*

Sample programs to help you get familiar with VIMOS.

4.2. Where to find documentation

You should have received all manuals in an easy to read electronic form together with your copy of VIMOS. If some of the manuals are missing you can download them for free from our web-site http://www.vimos.com. Upon request, all manuals are also available in print.

4.3. Web resources

All manuals, as well as the demo version of the system are available on the official VIMOS web-site http://www.vimos.com.