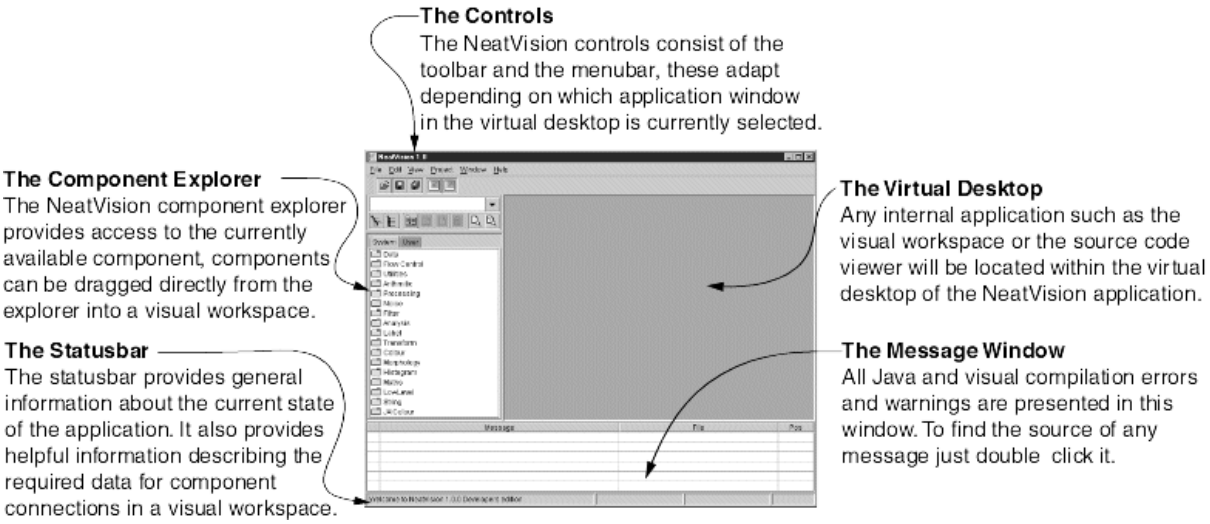
NeatVision: Tutorial Notes



Data files:

NeatVision data files are given the *.dat* suffix in the examples below. Netscape users can download *.dat* files by right clicking on the desired link and selecting 'Save Link As' from the resulting menu.

Colour Codes:

Each data connection has two properties data type and connection status. There are currently 8 supported data types, these are listed below.

Undefined
Image
Integer / Array data
Double precision Floating point data
Boolean data
String data
Fourier data
Coordinate

The other connection property is it's status. There are two main states for a connection, connected and disconnected. There is also an addition sub state disconnected but using default value, these states and associated colours are listed below.

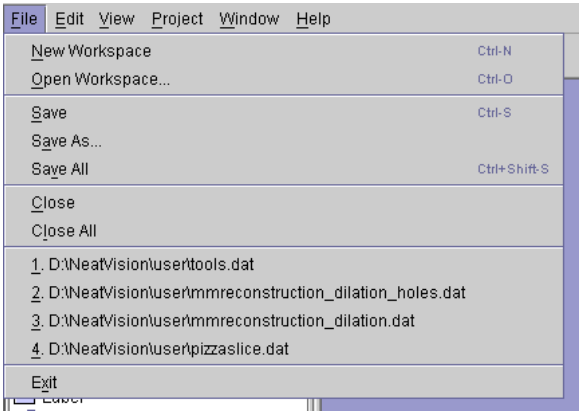
Node connected
Node disconnected
Node disconnected, default available

Basic Concepts

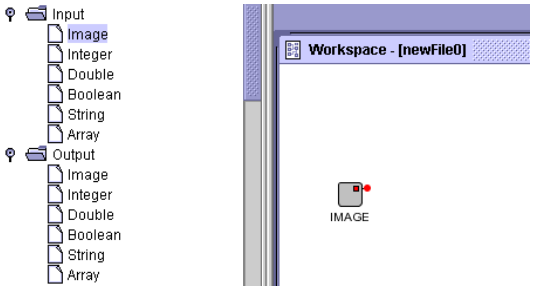
The visual programming concept is quite simple:

- Instantiate data input components
- Process inputs using required data processing components
- Assign system outputs to relevant output components

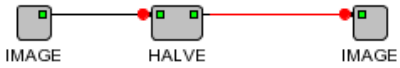
In order to create a visual program with NeatVision, you will need to create a new workspace. This option is available in the FILE menu.



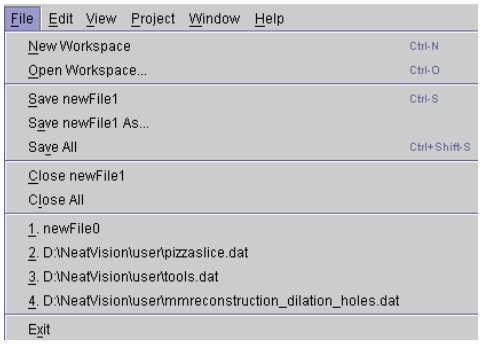
Once the visual workspace is created, you can drag and drop components from the components explorer as required.



Interconnections can be made by dragging the output of one component to the input of another.



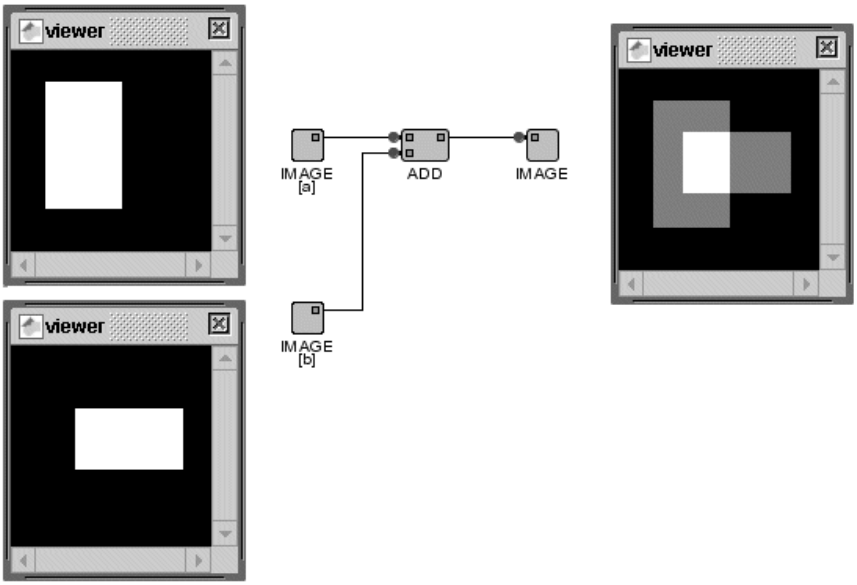
To save your visual program with NeatVision, you will need to save your new workspace. This option is available in the FILE menu. We generally give such files the *.dat* suffix.



Tutorial

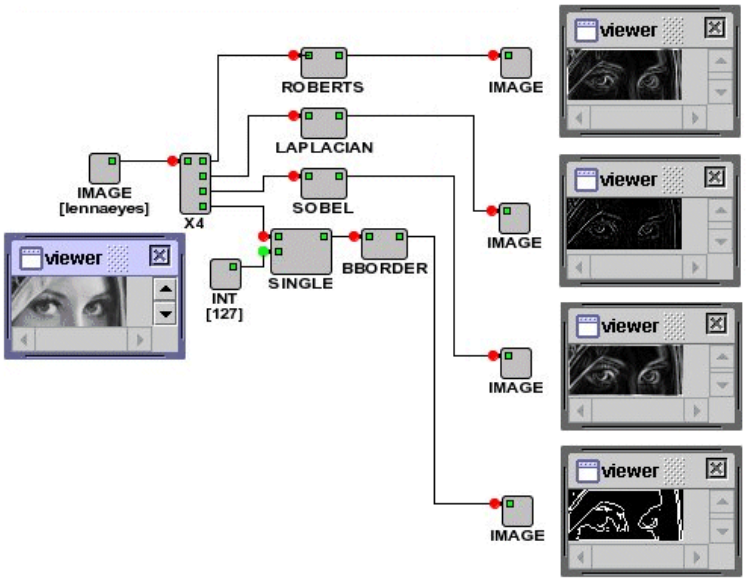
1. Take any two images and add them together (the workspace run tools will appear when you select the workspace). Save your output image as a JPEG file (Select image and go to FILE – SAVE AS and select the appropriate file type)?

Workspace run tools

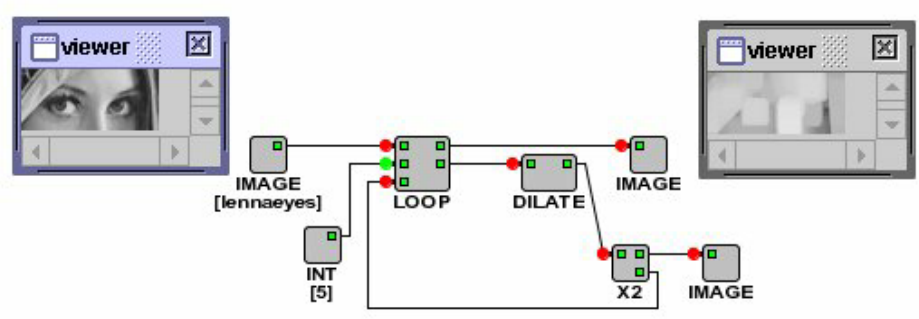


---

2. A selection of the edge detectors available to NeatVision are applied to a sample image (e.g. lenna.gif). A user-defined threshold (integer input) is applied prior to applying the binary border block. Alternatively, this can be chosen interactively using the blocks slide bar (activate by double clicking the threshold block).

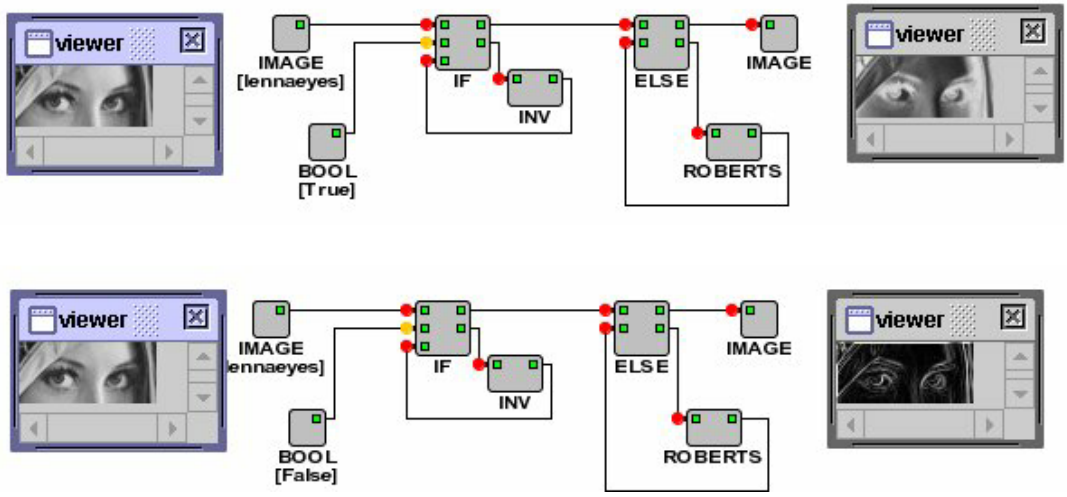


3. **Feedback implementation**. This example will allow the input image to be processed a number of times (as determined by the user defined integer input to the feedback block). The second (lower) output window allows the user to see the effect on the input image for each pass through the feedback cycle.



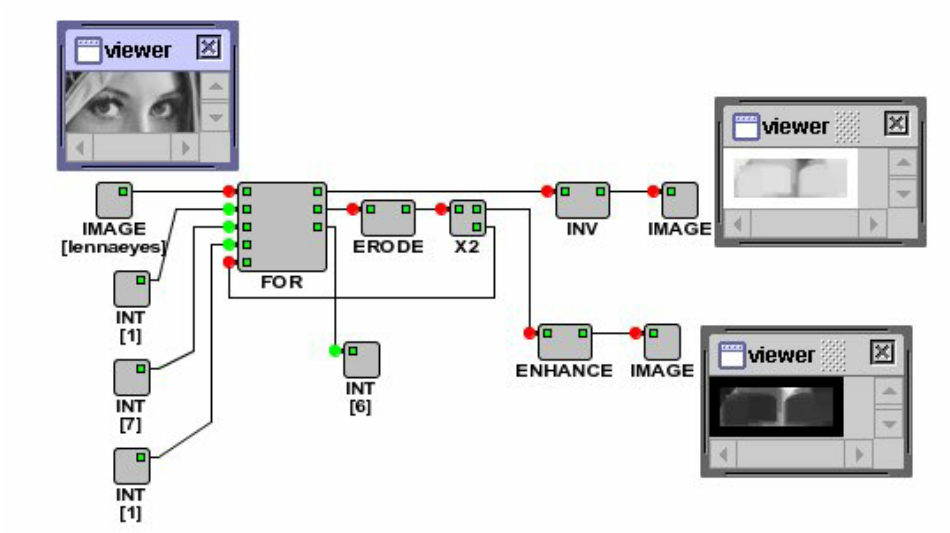
---

4. ***If .. else*** **implementation.** This example will allow conditional processing of the input image (as determined by Boolean (orange) input to the if block). The processing path can be redirected based on the conditional Boolean input.



---

5. ***For loop*** **implementation.** This is a standard implementation of a *for loop* structure. As per the example given below, the feedback actions will be implemented as the loop variable increments from 1 to 7 in steps of 1 (as determined by integer (green) inputs to the *for* block). The second (upper) output window allows the user to see the effect on the input image for each pass through the feedback cycle. The loop variable is indicated by the third (lower) output from the *for* block.



---

6. **Image Probing.** Using the graphic tools (these will appear when you select an image) generate 2D and 3D image profiles.

Graphic Tools