A Java Framework for Developing Interactive Image Processing Tutorials

J. L. McLaughlin, D. G. Bailey & W. H. Page
Institute of Information Science and Technology, Massey Univeristy, Palmerston North
E-mail: jesse@clear.net.nz; {D.G.Bailey | W.H.Page} @massey.ac.nz

Abstract

The dataflow model of image processing is utitilised as the basis of a Java framework for facilitating the development of interactive image processing tutorials. Using the framework, Java applets that interactively demonstrate concepts in the field of image processing can be easily constructed. The design of the framework is discussed, both with respect to those developing tutorials and to those wishing to expand the range of image processing operations available to tutorial designers. The framework is initially intended to provide a series of tutorial applets for use in a Web-based instructional system on image processing. A simple example of such a tutorial is described. Future prospects for the framework are then considered.

Keywords: *Java, applets, image processing, dataflow, Web-based instruction.*

1 Introduction

Designers of Web-based instructional systems [1] can use the Java [3] programming language to add interactive elements to their designs. This is achieved by incorporating Java applets into the system's content. Applets are (often small) Java programs that can be embedded into HTML [2] pages, and that provide a graphical interface with which users can interact. Applets offer a number of benefits to designers: 1) they are platform independent, so they can be distributed to a wider range of users; 2) they run in the context of a Java-enabled Web-browser, meaning they integrate seamlessly into a hypertext environment; and 3) they are written in a general purpose language, so there are few restrictions on what can be implemented.

The general purpose nature of Java also contributes to one of the potential drawbacks of the use of applets, which is that not everyone has the time nor inclination to gain the programming skills necessary to develop them. Even those with the skills may feel their time is better spent developing the eductional material for their systems, rather than writing and testing software.

A framework can remove much of the development burden associated with writing software, allowing developers to concentrate on the creative aspects of their systems rather than on unwanted implementation detail. Essentially, a framework is a software system targeted at aiding the development of programs in a specific field of interest. This paper reports on a framework for creating tutorial applets that demonstrate concepts in the field of image processing.

1.1 Dataflow

Image processing algorithms can be expressed as either data- or process-oriented. The process-oriented view leads to the style of algorithm representation familiar to programmers, i.e. a list of commands executed in the order in which they appear (see fig. 1a). The data-oriented view results in a graph-like structure, with image processing operations appearing as nodes and the data being processed appearing as the links between them (see fig. 1b).

1. Load Image into **A**
2. Let **B** equal **A**
3. Threshold **A** at **16**
4. Add **A** to **B**
5. Display **B**

Fig. 1a: Process-oriented view of an algorithm

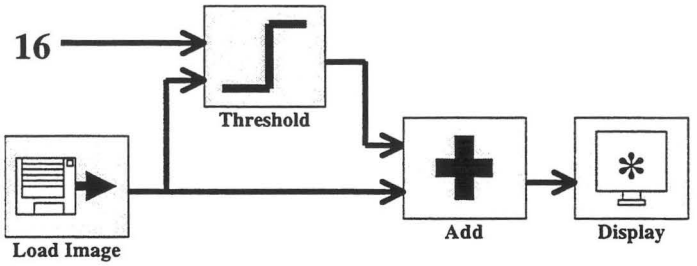

Fig. 1b: Data-oriented view of an algorithm

This data-oriented, or dataflow model of image processing algorithms forms the basis for the design of the framework. Designers specify the construction of tutorials based on connections (links) between framework components (nodes). The facilities provided to users of the framework are now discussed in more depth.

2 Framework End-Users

The framework caters to two main classes of user. Those developing tutorials and those wishing to enhance the range of image processing operations available to tutorial designers. Each group have distinct requirements, although it is possible that one user may fall into both groups simultaneously.

2.1 Tutorial Developers

Ideally, developers would not need to write any code. However, what we have is a framework, not a tool. A tool could be created, using the framework as a basis, however this is another task altogether (one that will be mentioned briefly at the end of this paper). The framework's strength is that it abstracts away a lot of implementation detail associated with a particular task and leaves only an interface that can be quickly learnt and put to use. The framework makes a number of abstractions in order to provide this interface. These are implemented as Java classes hierarchies (see fig. 2).

2.1.1 Operator

Operator is the base class for all tutorial elements, including image processing operations and user-interface components. It provides named inputs and outputs through which data can flow,

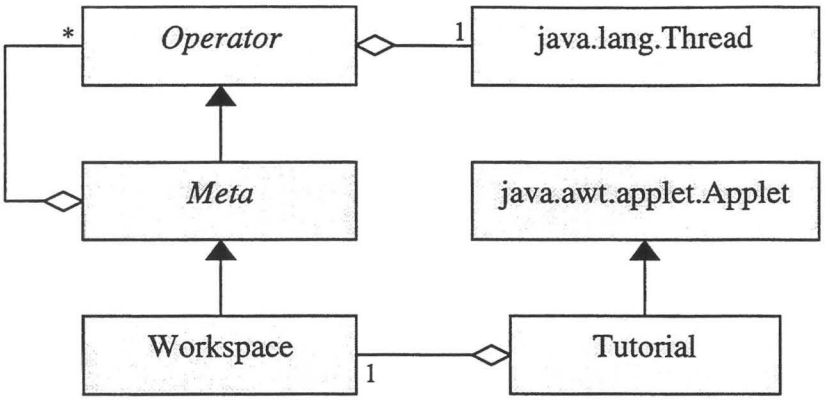

Fig. 2: Class hierachy visible to tutorial developer

and the functionality to connect these together. The following demonstrates how this is represented in Java:

```
abstract class Operator {
  void connect(String argIn, Operator op, String argOut) { ... }
  void disconnect(String argIn) { ... }
}

class Threshold extends Operator { ... }
```

To use an Operator, an instance of the required sub-class is instantiated and then connected to other Operator instances by making the necessary calls. In Java, this would look like:

```
Operator loadImage = new LoadImage();
Operator threshold = new Threshold();
threshold.connect("IMAGE_IN", loadImage, "IMAGE_OUT");
```

Each Operator also maintains its own Thread object, which allows it to carry out its processing without affecting the responsiveness of other Operators or the user-interface. This is an important consideration since to be interactive, the tutorials need to react to user input.

2.1.2 Meta

Meta is a container class that encapsulates a group of related Operators. This allows aggregation of functionality into a single object that can be treated just like any other Operator. The relationship between Operators and Metas is recursive so that Metas can contain Metas, etc.

2.2 End-user Requirements

The requirements of the tutorial developer are:

- *support for an extensible range of image processing operations;* this ensures a wide scope for tutorial development. This also means the provision of a range of data-types used in

image processing; not only for different types of images but also for histograms, scalar types, chain-codes, etc., since many concepts depend on these types of data.

- *provision of flexible input/output capabilities;* that is, a range of facilities for graphically accepting user input and displaying output. For example sliders, dials and buttons for user interaction; image, histogram and graph plotting displays for output.
- *integration with Java's Applet support;* so that the use of the framework is naturally aligned with the steps required to actually generate working applets, thus minimising the effort required to use the framework productively. This requirement also implies the framework should take into account the nature of the browser-applet environment, so that the applets generated work efficiently, without hogging resources.

3 Example of a Simple Tutorial

Figure 3 shows how a simple tutorial could be constructed. Once the image is loaded, it is displayed permanently so that the original is always visible. Then, when the user adjusts the slider, the thresholded image is recalculated dependingon the user input. This thresholded image is then (re-)displayed. This helps the user visualise the effect of changing the threshold level of a threshold operation.

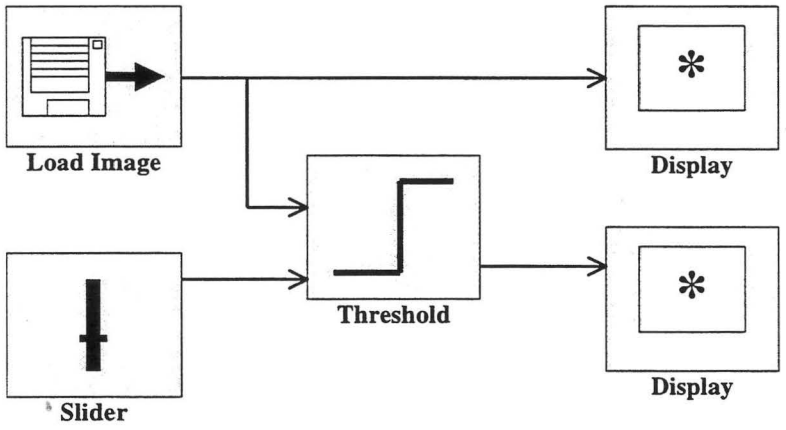

Fig. 3: Tutorial structure to demonstrate interactive image thresholding

4 References

[1] B. H. Khan (Ed.), *Web-Based Instruction*, New Jersey: Educational Technology Publications, 1997.

[2] D. Raggett, A. La Hors & I. Jacobs (eds.), *HTML 4.0 Specification*, http://www.w3.org/TR/REC-html40/, 1998.

[3] J. Gosling, B. Joy & G. Steele, *The Java Language Specification*, Addison-Wesley, 1996.