ROWS - description of design & implementation

September 6, 1999

1 Introduction

ROWS is a widget set, an abstract description of a user interface. As such, it must be portable, flexible and easily extendable. This document details how and where these aims have been achieved.

2 Widgets

2.1 What is a widget?

A widget is a user interface object of some description. Under ROWS, the base widget is just an object, nothing about the code defining any form of physical property - no bounding rectangle, no events, nothing. What it does describe is a set of properties required to define other widgets.

3 Graphics subsystem

3.1 'Abstract'

The ROWS graphical subsystem describes a set of operations that can be applied to any suitably advanced graphical device, bitmapped or vector[1]. All that is really needed to support a ROWS graphical interface is an ability to draw filled and unfilled polygons, but the effort of clipping and rendering a shape increases as the amount of predefined facilities decrease. The system is intended to support devices ranging from graphical displays to printers and plotters.

Graphics are drawn using three widgets: the drawable, the brush and the colour: A drawable represents a physical object that can be drawn on, be it widget, window, printer or file; a brush represents a method of drawing on a drawable; a colour represents a colour on a given drawable. Once a drawable for a given device has been created, it can be treated as the drawable for any other device - this way, drawing code can be used in a variety of contexts with little or no alteration.

[1] Actually, the description is really of a vector device, although some methods are not really suitable for such an environment (points, for instance, make no sense on a vector device)

However, not all devices are created equal, and some features and properties may not be reproducable on all devices. For example: clipping regions are hard to emulate on printers and plotters; some printers require an image to be rendered in strips; colours are different on different devices. For these regions, it is important to bear in mind that there may be differences between the various devices, and a rendering routine that is expected to perform well on a wide range of devices should avoid using graphics routines that may not translate well[2].

3.2 Colours[3]

A colour is a complex object to describe, partly because there is no really absolute definition of its properties. In order to keep things simple, a ROWS colour will be internally represented at least by an RGB value (although I reserve the right to add optional alternative representations). On a device, a colour may have several values allocated to it - for example, where it is used for anti-aliasing. A colour, therefore, has an abstract representation (its RGB values), and a physical representation (a set of device values).

3.3 Fonts

One property of having abstract graphics is that some things may have many different implementations. One example is fonts - there is (at least) the Impulse font system, X11 fonts, Postscript fonts and Truetype fonts to support. Not all of these font systems can be rendered correctly on all devices. Under ROWS, therefore, a font descriptor is as follows: '<Typographic system>/<Font name>[;<alternative>]'. So, for example, 'Impulse/serif/roman/Times New Roman/regular;Truetype/times.ttf;X11/adobe-times-medium-r-*-*-*-*-*-*-*-*-*-*;Postscript/Times New Roman' would be a font name with alternatives. Where the ';' character is part of a font name, it should be escaped with the '\' character, as should the '\' character itself. All font names must be in 7-bit ASCII, and it is up to the graphics system to pick a preferred font and to provide a fallback method for where no suggested typographic system is supported.

4 Events

[2]Basically clipping functions. At some point, a geometry library may be incorporated into libImpUtil to provide support for clipping rectangles and regions, to ease writing complete interfaces for vector devices.

[3]This currently describes an interface that is still to be implemented. The current implementation forces colours to be created specifically for a drawable.