`wumpus-core` Guide

Stephen Tetley

May 5, 2010

1 About `wumpus-core`

`wumpus-core` is a Haskell library for generating 2D vector pictures. It was written with portability as a priority, so it has no dependencies on foreign (i.e. C) libraries. It supports output to PostScript and SVG (Scalable Vector Graphics).

`wumpus-core` is rather primitive, the basic drawing objects are paths and text labels. A secondary library `wumpus-extra` has been prototyped containing some higher level objects (arrowheads, etc.), but hasn't been officially released - the code needs more thought before being put into the wild. Previews are available from the `copperbox` project repository hosted by Googlecode.

Although `wumpus-core` is heavily inspired by PostScript it avoids PostScript's notion of an (implicit) current point and the movements `lineto`, `moveto` etc., instead `wumpus-core` aims for a more *coordinate free* style.

2 Exposed modules

`wumpus-core` exports the following modules:

`Wumpus.Core.` Top-level import module, re-exports the exposed modules. Exports as opaque some of the internal data types, where the export is necessary for writing type signatures to user functions but access to the objects themselves is hidden by *smart* constructors.

`Wumpus.Core.AffineTrans.` The standard affine transformations (scaling, rotation, translation) implemented as type classes, with a of derived operations - reflections about the X or Y axes, rotations through common angles.

`Wumpus.Core.BoundingBox.` Data type representing bounding boxes and operations on them. This module is potentially important for defining higher-level graphics objects (arrowheads and the like).

`Wumpus.Core.Colour.` Colour types (RGB, grayscale and HSB) and conversion between them. Some named colours, which should be hidden or import qualified if a more extensive package of colours (e.g. the named SVG

colours) is used. RGB is the default format, where black is `(0.0, 0.0, 0.0)`, and white is `(1.0, 1.0, 1.0)`.

`Wumpus.Core.FontSize.` Various calculations for font size metrics. Generally not useful to a user but exposed so that variations of the standard Label type are possible.

`Wumpus.Core.Geometry.` Usual types an operations from affine geometry - points, vectors and frames. The `Pointwise` type class which is essential for defining transformable drawable types.

`Wumpus.Core.GraphicsState.` Data types modelling the attributes of PostScript's graphics state (stroke style, dash pattern, etc.). Note `wumpus-core` annotates primitives - paths, text labels - with their rendering style. PostScript has a mutable graphics state, changing via inheritance how the curent object is drawn.

`Wumpus.Core.OutputPostScript.` Functions to write PostScript or encapsulated PostScript files.

`Wumpus.Core.OutputSVG.` Functions to write SVG files.

`Wumpus.Core.Picture.` Operations to build *pictures* - paths and labels within an affine frame. Type classes overloading convenience constructors for building paths, labels, ellipses... The constructors are convenient in that attributes - colour, line width, etc. - may be specified or not. The technique is due to Iavor S. Diatchki's XML-Light.

`Wumpus.Core.PictureLanguage.` Composition operators for pictures. The operators are somewhat analogue to the usual operators or pretty-printing libraries, but work in 2D rather than largely horizontally with some vertical concatenation.

`Wumpus.Core.TextEncoder.` Types for handling non-ASCII character codes. This module is perhaps under-cooked thou it appears adequate for Latin 1...

`Wumpus.Core.TextLatin1.` A instance of the TextEncoder type for mapping Latin 1 characters to the PostScript and SVG escape characters.

3 Drawing model

`wumpus-core` has two main drawable primitives *paths* and text *labels*, ellipses are also a primitive although this is a concession to efficiency when drawing dots (which would otherwise require 4 to 8 Bezier arcs to describe). Paths are made from straight sections or Bezier curves, they can be open and *stroked* to produce a line; or closed and *stroked*, *filled* or *clipped*. Labels represent a single horizontal line of text - multiple lines must be composed from multiple labels.

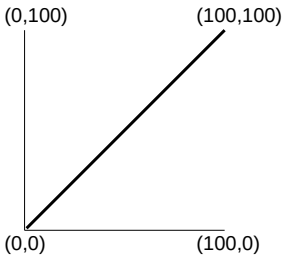

Figure 1: The world frame, with origin at the bottom left.

Primitives are attributed with drawing styles - font name and size for labels; line width, colour, etc. for paths - and place within a picture. The function `frame` lifts a primitive to a picture within the standard affine frame (the standard frame has origin at (0,0) and unit bases for the x and y axes). The function `frameMulti` places one or more primitives in a frame - this will produce more efficient PostScript and should be preferred for creating scatter-plots and the like.

`wumpus-core` uses the same picture frame as PostScript with the origin at the bottom left, see Figure 1. This contrasts to SVG where the origin at the top-left. When `wumpus-core` generates SVG, the whole picture is produced within a matrix transformation [1.0, 0.0, 0.0, -1.0, 0.0, 0.0] that changes the picture to use PostScript coordinates. This has the side-effect that text is otherwise drawn upside down, so `wumpus-core` adds a rectifying transform to each text element.

Once labels and paths are assembled as a *Picture* they are transformable with the usual affine transformations (scaling, rotation, translation) and multiple pictures can be composed with the operations provided by the `PictureLanguage` module. The operations should be largely familiar from pretty-printing libraries although here they are extended to 2 dimensions.

Once assembled into pictures graphics properties (e.g. colour) are opaque - it is not possible to write a transformation function that turns a picture blue. In some ways this is a limitation - for instance, the `Diagrams` library appears to support some notion of attribute overriding; however it is conceptually simple. If one wanted to make blue arrows or red arrows with `wumpus-core` one would make colour a parameter of the arrow creating function.

4 Font handling

Font handling is quite primitive in `wumpus-core` . Particularly the bounding box of text label is only estimated rather than calculated from the font's metrics. For the its intended use (producing diagrams, pictures rather than high quality text) this is not such a draw back - implementing a font loader to read TrueType fonts would be a significant effort, probably larger than the effort put into `wumpus-core` itself.

attributes -
matrix alignment -

5 References

PostScript is a registered trademark of Adobe Systems Inc.