CS405 Introduction to Empirical Modelling
Lab 3: The SCOUT windowing notation

Empirical Modelling Research Group
http://www.dcs.warwick.ac.uk/modelling/

Introduction

This week we are introducing the SCOUT (SCreen layOUT) notation. This provides a notation for describing the graphical aspects of models. Both the *tic tac toe* and lift models used in previous weeks make use of this notation. Points to note are that:

- SCOUT is a *definitive notation*, so there are no procedural elements to the language.
- The symbol `=` represents dependency and so has the same semantics as the Eden keyword `is`. There is no equivalent in SCOUT of procedural assignment (Eden's `=`);

We are providing you with an Eden model of two jugs, the aim for this lab is for you to construct an interface to this model using the SCOUT notation.
The model presents two jugs of known capacity and we must use a combination of filling, pouring and emptying to achieve a specified target content. There are no identifying marks on the side of the jugs.

1 The jugs model

The jugs model is available in `/dcs/emp/empublic/teaching/cs405/lab3/model`. To run the model include the file `Run.e`. An ASCII representation of two jugs will appear on the terminal.
The model includes five proceedures that can be used to control the model. The five proceedures are:

1. `FillA();` – fill jug A until it is full.
2. `FillB();` – fill jug B until it is full.
3. `EmptyA();` – completely empty jug A.
4. `EmptyB();` – completely empty jug B.
5. `Pour();` – pours liquid from the jug

Task 1

Use the above procedures, try filling and emptying the jugs.

There are a number of key eden observables in the model which be used in later tasks.

- `capA` – Capacity of Jug A.
- `capB` – Capacity of Jug B.
- `contentA` – Content (liquid level) of Jug A.
- `contentB` – Content (liquid level) of Jug B.
- `widthA` – Width of Jug A.
- `widthB` – Width of Jug B.

- `valid1` – That button 1 (Fill Jug A) is enabled.
- `valid2` – That button 2 (Fill Jug B) is enabled.
- `valid3` – That button 3 (Empty Jug A) is enabled.
- `valid4` – That button 4 (Empty Jug B) is enabled.
- `valid5` – That button 5 (Pour) is enabled.

2 Adding a SCOUT window to the screen

A graphical layout in SCOUT consists of one or more *displays*, each containing one or more *windows*. The display used by most SCOUT scripts is `screen` and is created automatically by tkeden (or dtkeden). Other displays have to be created using a built-in Eden command.

Note: The SCOUT display `screen` is implemented as a window manager toplevel window, SCOUT windows are boxes within a SCOUT display.

There are various types of SCOUT window and you will begin by using the TEXT windows. A simple text window can be declared using the following SCOUT:

```
%scout
integer myScoutWindow1_X1 = 10;
integer myScoutWindow1_Y1 = 10;
integer myScoutWindow1_X2 = 100;
integer myScoutWindow1_Y2 = 100;

window myScoutWindow1 = {
        type: TEXT
        string: "This is myScoutWindow1"
        frame: ([{myScoutWindow1_X1, myScoutWindow1_Y1}, {myScoutWindow1_X2,
myScoutWindow1_Y2}])
};
```

and is placed on the screen using the definition:

```
%scout
screen = <myScoutWindow1>;
```

If you wish to place more than one window on the `screen` then the window names are separated with `/`. Note that any SCOUT object needs to be declared to the SCOUT symbol table before it is used.

```
%scout
window myScoutWindow2; ## windows need to be declared before use
window myScoutWindow3; ##  but can be left undefined.
screen = <myScoutWindow1/myScoutWindow2/myScoutWindow3>;
## screen will be updated to represent all three windows once the
## two new windows are defined.
```

Task 2

Add a SCOUT window called `containerA` to the screen, check that you can move and resize the window by redefining the position properties.

3 Window attributes

It is possible to adjust various attributes of a SCOUT window including background colour, border colour etc. For a full list consult section 3 of the *SCOUT Quick Reference* available from the help menu in tkeden. Here is an example TEXT window containing some optional attributes:

```
%scout
window myScoutWindow2 = {
        type: TEXT
        string: "This is myScoutWindow2"
        frame: ([{100, 10}, {200, 100}])
        font: "{times 12 {bold italic}}"
        border: 2
        bgcolour: "green"  ## green background
        fgcolour: "white"  ## white text
        bdcolour: "black"  ## black border
};
screen = < myScoutWindow1 / myScoutWindow2 >;
```

Task 3

Define `containerA` to have a light grey background and a black border.
Hint: Various colours are predefined, see the Colour Names *in the help menu.*

4 Using dependency in SCOUT

The window that you created in Task 1 (`containerA`) will represent one of the two jugs in the underlying model.

Task 4.1

Using dependency, define the height of your jug container so that it is dependent on the capacity of jug A (Eden observable `capA`). Also the jug's width should be dependent on the Eden observable `widthA`.

Represent the liquid in the jug with another SCOUT window called `liquidA`. Make the height of this window represent the amount of liquid in the jug (Eden observable `contentA`). Choose a colour which represents the liquid (e.g. blue) for the background colour of `liquidA`. Make sure that the liquid window is overlaid on the container window.

Hint: See section 2 in "SCOUT - a definitive notation for SCreen layOUT".

If you wish to make a dependency to an eden observable `x`, you need to introduce `x` to SCOUT first.

Task 4.2

Check that you can still move the jug, does the liquid move with it? If not, correct your definitions so that it does.

5 Adding another jug

To complete the visualisation, you will need to add a second jug and link it into the model.

Task 5

Add a second jug (Jug B) that uses the observables `capB`, `contentB` and `widthB`.
Define the distance between the two jugs to be a third of the display's width (`screen_width`).

6 Buttons to control the model

Creating buttons will require use of the sensitive attribute to register mouse messages on a window. Remember that mouse events will cause the redefinition of an observable named by the window name concatenated with `_mouse` for image windows and `_mouse_` and the box number for text windows.

Task 6

Using the five procedures outlined earlier, add buttons connected to the five proceedures, `FillA`, `FillB`, `EmptyA`, `EmptyB` and `Pour`. These buttons should only be enabled when the respective procedure is available.

Summary

You should by working through this lab sheet **and by additional study**:

- be able to build a SCOUT interface to an existing model;
- know how to create SCOUT windows and display them on the screen;
- know to alter the colour, border, font, layering and other properites of a SCOUT window;
- be able to dependency with SCOUT; and,
- know how to create buttons in SCOUT and connect them to other parts of the model.