CS405 Introduction to Empirical Modelling 2008

Labsheet 2: Script mechanics - `eden`, `donald` and `scout`

Mechanics of definitive scripts

We have seen that a simple piece of script can have many different interpretations, and can be refined in many ways. For this reason, it is often useful to be able to extract subsets of definitions from existing scripts and incorporate them in your own models. In order to do this effectively, you need to understand the relationship between the `eden`, `donald` and `scout` notations and know how to tackle the problem of making sense of what may be a relatively complex script. In this laboratory, you will get some experience of this activity by looking at `roomviewerYung1991`. Understanding a script in this way is clearly analogous to what is described as "program comprehension", an activity that doesn't necessarily have much to do with understanding how a program is interpreted in its application context. In the case of a well-written definitive script, we can expect a closer connection between understanding the mechanics of a script and appreciating its relationship to an external referent.

Each exercise illustrates useful techniques that can be exploited in many models and comprises a set of relatively small questions. In tackling them, you may find it helpful to first address one or two questions from each.

Exercise 1: Understanding how `donald` is represented in `eden`

By examining the observables in the `roomviewer.s` model, answer the following questions:

- What is the current definition of `table/SW`?
- How would you print out the current value of `desk/drawer/k` every time that it is changed?
- Why is the line that represents the cable dashed?
- The observable `cableIsShort` is set to `true` when the distance between the socket and the lamp exceeds the length of the cable. Determine what observables influence the value of `cableIsShort` and write down some representative examples of redefinitions that resolve the problem of the short cable, together with their interpretations in the referent.
- How would you adapt the model so that:
  - the cable is coloured red when it is not long enough?
  - the door is coloured green when open and red when shut?
  - in the improved version of the desk those parts of the drawer that are invisible are displayed as dashed lines?
- Place a small circle at the centre of the lamp to indicate a bulb, and set its attributes so that it is coloured solid yellow or black according to whether the lamp is on or off.
- Attach to the centre of the table a `donald` label that displays its current coordinates.
- Explain why nothing happens if you modify the `roomviewerYung1991` model by introducing the definition of the rotated table in the Appendix to this labsheet. How do you resolve this problem? [Hint: you may need to make use of `eden` to define some `donald` observables.]

Exercise 2: Understanding how `scout` describes screen layout

- Trace the set (or a representative subset) of the dependencies that determines the windows that make up the screen display.
- Identify which observables determine the size and location of the lefthand window containing the `donald` floorplan of the room.
- Find the definition that determines the text displayed on the door button (as defined by the `doorButton text` window in `scout`).
- How would you change the background colour and the colour of the text displayed on the door button?
- How would you place a single rectangular window behind the windows displaying the floorplan and the enlarged floorplan in such a way as to provide a blue background?

Exercise 3: Understanding how `donald` drawings are displayed in `scout` windows

- Inspect the definitions of the `scout` windows that display the floorplan and enlarged floorplan. Explain why the `pict` field is the same in both windows.
- How would you interchange the drawings displayed in the two `scout` windows of type `DONALD`?
- How can you determine the `donald` coordinates of a point in the floorplan and enlarged floorplan displays solely by using mouse clicks?
- By assigning new values to the observables `zoomPos` and `zoomSize` and observing the impact on the `scout` window containing the enlarged floorplan, determine the significance of the `xmin`, `xmax`, `ymin` and `ymax` fields within the definition of the window.
- Suppose that a `scout` window of type `DONALD` displays a line drawing without any distortion. What relationship must there be between the dimensions of the window and the values of the fields `xmin`, `xmax`, `ymin` and `ymax`?
- Make use of `donald` variable of type `shape` to specify an image of the table rotated through an angle that can be freely specified via the input window. Revise the definition of the `scout` window that presently shows the enlarged floorplan so that it instead displays the rotated table.

Exercise 4: Understanding how GUI agents are specified in `scout` and `eden`.

- Determine what `eden` redefinition is executed when the door button is clicked. Confirm that this redefinition can be entered directly through the input window and explain why it has the effect of opening or shutting the door.
- Modify the `eden` action associated with the door button so that it toggles the colour of the door from red to green. Explain why this is a less satisfactory way of specifying the colour of the door than using a definition as proposed in Exercise 1 above.
- Adapt the `eden` action associated with the door button so that it counts the number of times the door is opened. Modify the text on the door button so that it displays this count.
- Identify the `eden` action that is triggered by clicking the mouse in the `scout` window displaying the floorplan. Explain how this action moves the table.
- Explain why the buttons that relocate the table no longer work correctly when the rotating table mentioned in Exercise 1 is introduced. Fix this problem.

Questions

1. The syntax and structure of the definitive notations associated with `eden`, `donald` and `scout` are very different. This can be frustrating, especially for novice users. Can you think of reasons why this diversity in styles of definition might be appropriate? Can you suggest a design for a single unifying definitive notation that could express all the various types of dependency relationship that can be expressed by using `eden`, `donald` and `scout` in EDEN?

2. *Making a magic room.*

   Outline the kinds of EDEN definitions, functions and actions you would need to introduce in order to adapt the `roomviewerYung1991` model so that it could support the following "magic" scenario:

   - when the lamp is positioned so that cable is not long enough, the table obstructs the door and the desk drawer is pulled out completely, the drawer is transformed into a magic drawer.
   - as the magic drawer is put back in the desk, the dimensions of the drawer and the desk converge to the point where - when it is completely closed - both are square with a side of length equal to the original width of the desk.
   - when the magic drawer is completely closed, the square representing it becomes itself another image of the room such as is displayed in the original floorplan window.

   This is the kind of mental exercise that is useful when you are conceiving possible developments of a model. It is important to be able to think in terms of observables, dependencies and agency you could in principle model in EDEN without necessarily actually creating the model.

3. *This is not the kind of topic that would be included in an examination question, but may give you some insight into why using EDEN should be significantly different from using Visual Basic. It may also raise issues of interest to those who are curious about the design and implementation of EDEN.*

   a. The way in which `donald` line drawings are associated with `scout` windows using "viewports" is not very well-aligned to the principles of modelling with definitive scripts. It can also cause practical problems in model development. Why is this? Can you suggest a better way of associating line drawings with windows?

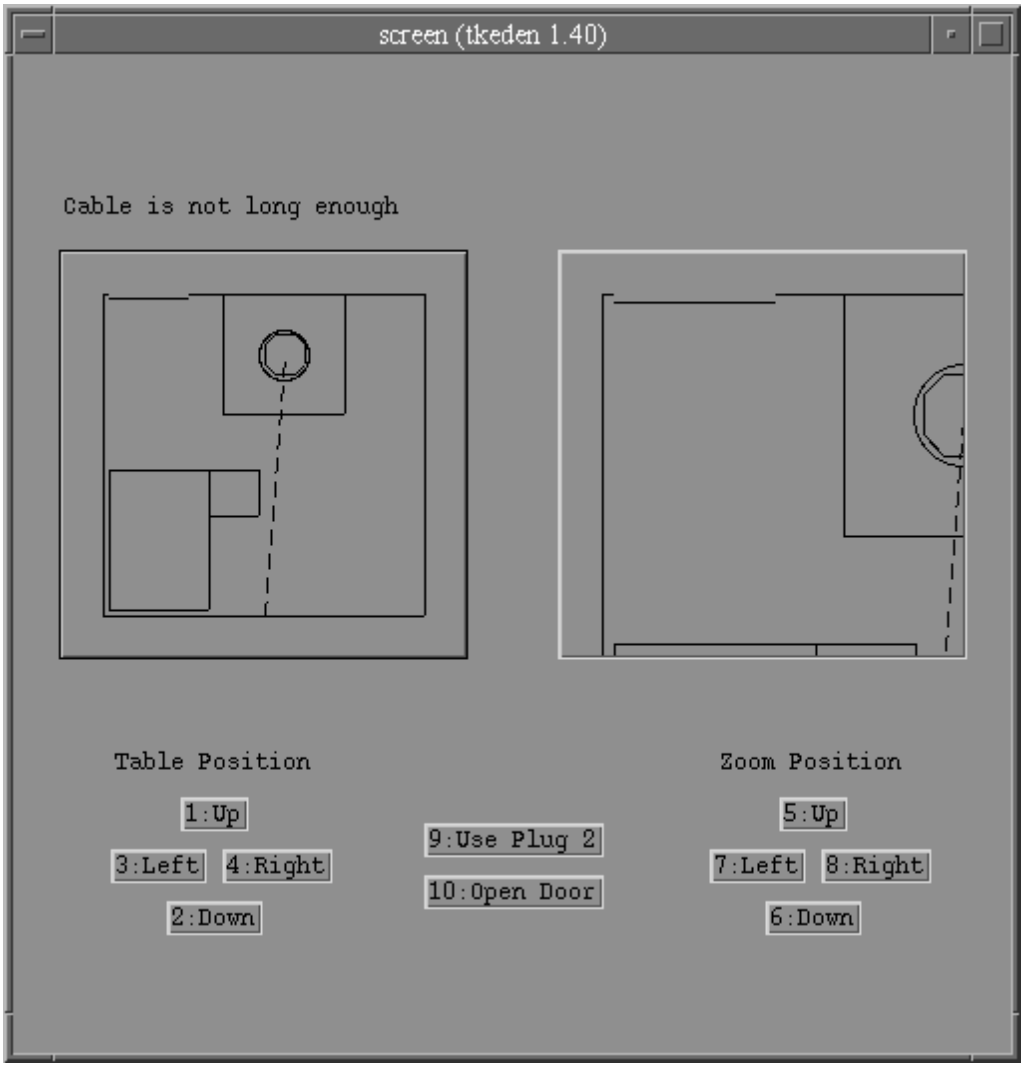
   b. There were significant developments in EDEN after `roomviewerYung1991` was created. In earlier versions of EDEN, the only way to attach a behaviour to a sensitive button was to use the mechanism illustrated in the `roomviewerYung1991` model. That is to say: clicking on a sensitive `scout` window (say `doorButton`) caused the value of an associated mouse-related observable (such as `doorButton_mouse_1`) to be redefined. The modeller then had to write their own action (such as `doorButton_to_input`) to be triggered by a redefinition of this observable. Later versions of EDEN introduced a "VB-like" feature, whereby redefinition of a mouse-related observable directly invokes a procedure (such as `doorButton_mouseButtonPress`). You can expose this mechanism by querying the observable `doorButton_mouse_1` in `roomviewerYung1991`, and observing the actions it triggers:

   ```
   doorButton_mouse_1 ~> [doorButton_mouseButtonPress, doorButton_to_input];
   ```

   By default, VB features are no longer enabled in current versions of EDEN (see the `ChangeLog` of the `Help` menu option for more info). Can you figure out
   - how these VB mechanisms were intended to be used?
   - why their use was to be deprecated?

Appendix to Labsheet 2

```
%donald

## first redefine coordinates of table relative to its centre

within table {
        point centre
        SW = centre - {width div 2, length div 2}
        NW = centre - {width div 2, -length div 2}
        SE = centre + {width div 2, -length div 2}
        NE = centre + {width div 2, length div 2}
}

## now define the rotation

within table {
        point centre
        real tablerotangle
        SW = centre - rot({width div 2, length div 2}, {0,0}, tablerotangle)
        NW = centre - rot({width div 2, -length div 2}, {0,0}, tablerotangle)
        SE = centre + rot({width div 2, -length div 2}, {0,0}, tablerotangle)
        NE = centre + rot({width div 2, length div 2}, {0,0}, tablerotangle)
}

## specify angles in radians - as floating point numbers

table/tablerotangle = pi div 4
```



A screenshot taken from `roomviewerYung1991`
