Meurig Beynon

I hadn't really expected to give a talk here today, but Julie invited me along and I am very pleased to be here. I haven't been talking to this community all gathered in this way before as I wasn't here for the April meeting, but I brought along a few visualisations that I have been involved in constructing. All these are built with the same tool and principles, and I think that the best contribution I can make is to say a little bit about the principles I have in mind when building these visualisations.

I am going to begin by showing you a demonstration, probably the simplest visualisation anybody has seen here today. This I think is helpful by way of background.

<display the tkeden input window ready to load the digit-cabinet presentation model at this point>

We have built a tool which is for much more than visualisation; it is a tool associated with our attempt to rethink the way that we program - in a way that I believe brings the whole thinking about computing much closer to visualisation than to describing processes.

I will demonstrate that by giving you a very simple visual image and putting it in the context of a rather bigger environment. It will take a moment to load that environment.

Essentially you could say that the principles we are using here are somewhat similar to what you see in spreadsheets. In a spreadsheet you record the state of things by setting up a grid in which each cell represents something visible in the world, so that is the link with visualisation if you like. In our context, we use systems - or files - of definitions, to capture the state of something we are experiencing, so that for instance you can view the whole of this presentation environment, if you like, as a state of some kind. It is described by a set of definitions which are hidden from you here, but a little interaction will enable you to get the general flavour of what I am doing. If, for example, I introduce a certain set of definitions, I produce this little visualisation on the left, which as I promised is perhaps the simplest visualisation you have seen today.

You will see that the visualisation comprises two identical shapes in the form of line drawings. My first point is that how we observe these two shapes determines a great deal about what their semantics are. I am thinking of two different intentions associated with these two simple shapes, one is connected with thinking of this as a figure eight, which it may look like on an LED display, the other is thinking of it as if it were the profile of a piece of furniture in a room - say a filing cabinet or something like that.

The files of definitions that have been used to specify the same shape in two different ways are given in an Appendix to this transcript.

I describe one of these shapes as if it were a drawer which is open to a certain degree; it is part of a cabinet, and there are various lines and points that represent the lines and corners and so on, whereas the other shape is described simply as a display associated with an abstract digit, and precisely which lines are illuminated here depends upon which digit I specify by making a definition or redefinition.

In my presentation interface, I am able to introduce definitions to modify the current file of definitions being interpreted. I make use of a window imitating my input box down here on the left. I can take this redefinition, stick it in here (by exercising the 'copy to input biox' option), interpret it, and that will change the current set of definitions. It also has the effect of changing the shape on the right so that the cabinet is now half-open. And if I do the second option below, carrying out the 'execute' definition here, it changes my definition of the current digit being displayed.

So what I have here are shapes associated with observing a digit, and observing something like the profile of a filing cabinet, and I can experiment with these things, to see if you like whether they represent something close to what I intend these lines to represent, and the expected semantics in the world is conveyed.

There are many advantages of modelling in this way, such as being able for example to restore all the definitions I had originally, and doing other kinds of experiments that you might not normally be doing with your cabinet, as in thinking "what if this cabinet is not so wide".

One of the very important aspects of this modelling is that compared with traditional programming - where you have a repertoire of things that you do and you are advised when you write a program to prevent people doing things that go outside that repertoire - here you are actively encouraged to explore, and attempt to do things that probe the border of what is meaningful. For example I could try putting digit equal to 10 and see what happens ... when I do that you see that the visualisation

doesn't really hold up very well - it displays a 9 for some reason, but I can modify my semantics so that something more sensible happens, by making it so that if I change some number num(x) then this will always display that number modulo 10 and so on.

I hope that has given you an impression. My point is that the entire environment here is stored in some very extensive list of definitions and there are many virtues from the point of view of modelling - especially in modelling states as we experience them - that come from this.

I had to introduce this approach to visualisation to some extent to give you some impression of what is going on in the power point presentation.

This is my compendium slide along the lines that Julie suggested we prepare for today. What we have found is that our kind of modelling is amenable to dealing with the semantics of all sorts of things, not just things in science, because the meaning of our state is very much determined by the family of interactions we think are plausible with it.

Here on the top left we have got something which illustrates that. It is an exercise I did in modelling a Schubert song - Schubert's Erlkönig. These images are all stills taken from a dynamic presentation; you can play this visualisation in conjunction with Erlkönig, if you wish, but also all these visualisations were dynamically put together by modifying states and they are still open to being changed. There is no time to develop on that point again but the images in this corner illustrate that.

Overall the objective can be seen as tryinbg to give visual expression to affect.

Over here I have got visualisations that are more to do with trying to understand certain mechanisms. For example, this object here is a planimeter, a kind of obsolete instrument of measuring areas, and this is an animation in which you can trace a curve on paper and its area as it is being registered through the rotation of this red dial. The way in which all the connections between these elements of the state are maintained (as you would expect) is by making use of files of definition that record state.

Over here on the right we are introspecting about what cognitive processes are involved in playing a game of noughts and crosses. So in this image we are registering the fact that you recognise lines, you can interpret a current position in a game s of noughts and crosses etc. We find it interesting that this kind of visualisation supports all the many different levels of understanding, so I that can mimic a child who cheats or who doesn't know the rules, change the rules on the fly, change the lines on the fly, those sort of things.

The key original element here is the way we are using visualisation: we are trying to use it as really the fundamental semantic mediator (though of course there could also be other forms of perceptualisation) and build computing from the bottom up, from a very concrete, grounded semantic origin.

That doesn't prevent us from doing things we can do in classical computer science. This is a UML diagram (based on a state chart devised by David Harel) recording the state of a digital watch. Harel contends that statecharts, as a generalisation of finite state machines, give a powerful model of state, but I feel that this kind of claim is probably is even more appropriate in relation to our files of definitions. In this model, for instance, a few hundred definitions will keep all the interface state of the watch in the statechart - whilst at the same time - it will keep an analogue display and a digital watch. In addition to that the model reflects the entire watch functionality (not merely the interface functionality), so it gives us a very rich power to model states. This is what one of my research students did to adapt the original digital watch model, by way of modelling his own watch, and how he conceived its state in a less formal manner than the statechart reflects.

Down on the bottom right is a case study that we have looked at in some detail, and is perhaps of more interest in relation to research in the humanities. It is a visualisation we made as a kind of reconstruction of railway operation in the vicinity of an accident that occurred at the Clayton Tunnel in 1861. Our first effort was just this 2D depiction of the state, where you have a tunnel and you have the fairly simple depictions of the elements on which the attention of the human agents in the accident is focused you see here. We have a signalman, who is responsible for maintaining traffic through the tunnel, and who has to respond to whether a train has emerged when the train has emerged from this end - this indicates when it is time to let the next train into the tunnel, and so on. We have distributed this model so that we have 5 or 6 schoolchildren at different workstations playing the parts of the various agents in the

scenario. Of course, the 2D version is not a physically very realistic animation or visualisation of what is going on - by adapting the 2D model, we have since developed the 3D visualisation to the right of this, which is a VRML visualisation in some sense more realistic. It is interesting though that (for example) a signalman waiting for trains to approach the tunnel has some conception of when the next train is due, even though they can't see it, and in the 2D kind of model you might get that impression - you may even have a little blob up here to register that there is a train approaching - and that could mentally correspond quite well to what the station master "sees" in his mind's eye, though there will be no record of it on the VRML display, as you won't actually visibly see it. This is bearing out what the previous speaker was saying about different perspectives on what is real. In that respect, both of those animations have their distinctive and different qualities.

Finally I have one or two images in the bottom left corner here. As a mathematician by background myself, I am enormously impressed with Julie's work in mathematics, and this is the first reason for our common interest in this topic. There are some visualisations here which are based on enhanced versions of our basic modelling tool in some sense, building in different kinds of visual notation to support essentially different metaphors to guide the way we interpret diagrams. These are two visualisations from my own research. Below that is a visualisation that I have been using recently for teaching graphics, in a sense using the very presentation environment I showed you before. All three of these models are - in effect - concerned with visualising transformations and so on. Exploring these models allows us to try out different ways of visualising things.

Finally - in the bottom left corner - we have recently been playing with a sudoku modelling environment. What you can do is to attach colours to the various digits (1 to 9) - then when - by some naive criteria - you have got a choice about what you can put in a square (such as whether a digit is still available for use in its corresponding region, row and column), you can blend the corresponding colours and display the blend in the square. For example, when you do this, a dark colour corresponds to something with relatively few choices and you can spot what squares are most promising targets for entering the next digit. Using the auxiliary interface below, you can actually play, in real time, with the assignment of colours to digits, and this becomes an interesting environment for solving sudoku. It seems to have particular merits if there are several of you trying to solve a puzzle together, because it seems to help a lot in terms of focusing your attention on where it is plausible that you could guess an answer for example, especially towards the end of the process when there are fewest choices for digits.

This then is a sample of the things we have been doing in trying to apply our visualisation principles. I guess a big problem is making contact with serious visualisation on the scale that engineers want to do it. Obviously it gets very difficult. We typically deal with a few hundred definitions and we have an interpreter which maintains the relationships between those, as in maintaining spreadsheets dependencies, but if you scale that up by a factor of 10, it becomes much more difficult for our particular software to cope. Despite this, we think this has promise as a way of rethinking how we approach programming even in its full generality.

Appendix

The %donald files displayed on the right are the two visualisations of the identical line drawing referred to in the talk.

A little experiment with the visualisation of the filing cabinet shows that it is quite unsatisfactory in some respects. Though the model does do a satisfactory job of displaying what a filing cabinet would look like from a top-down view, the model of the drawer does not reflect its physical characteristics faithfully. This is clear from the manner in which the length of the drawer depends upon the extent to which it is opened - see the definition of drawer/length. This becomes apparent when the drawer is "pulled right out", for instance, by redefining the observable open to a value that exceeds 1. A sequence of interactions that serves to fix this problem is embedded into the EDEN presentation referred to in the talk (though this involves interactions beyond the scope of what is discussed in the talk itself).

```
%donald
openshape cabinet
within cabinet {
   int     width, length       # the dimensions
   point   NW, NE, SW, SE  # the 4 corners
   line    N, S, E, W          # the 4 edges

   N = [NW, NE]
   S = [SW, SE]
   W = [NW, SW]
   E = [NE, SE]

   width, length = 300, 300

   SW = {600, 200}
   SE = SW + {width, 0}
         #  the other 3 corners are . . .
         #  ... relative to SW corner
   NW = SW + {0, length}
   NE = NW + {width, 0}

   ##### cabinet/drawer #####

   openshape drawer
         # the file cabinet has a drawer
   within drawer {
      real    open            # 1 = open; 0 = close
      real    length    # the length of the drawer
      line    N, S, W, E
         # the 4 edges of the drawer

         length = ~/length * open
         open = 1.0
         # the drawer is opened initially

      N = [~/NW+{0,length}, ~/NE+{0, length}]
      S = [~/NW, ~/NE]
      W = [~/NW + {0, length}, ~/NW]
      E = [~/NE + {0, length}, ~/NE]
   }
}
```

```
openshape led
within led {
   int     digit
   point p1, p2, p3, p4, p5, p6                #
6 points
   line l1, l2, l3, l4, l5, l6, l7             #
7 segments
   boolean on1, on2, on3, on4, on5, on6, on7
         # status of the 7 segments

   digit = 8                  # initially display all
segments

   p1 = {100, 800}
   p2 = {100, 500}
   p3 = {100, 200}
   p4 = {400, 800}
   p5 = {400, 500}
   p6 = {400, 200}

   on1 = digit != 1 && digit != 4
   on2 = digit != 0 && digit != 1 && digit != 7
   on3 = digit != 1 && digit != 4 && digit != 7
   on4 = (digit == 0 || digit >= 4) && digit != 7
   on5 = digit == 0 || digit == 2 || digit == 6 ||
digit == 8
   on6 = digit != 5 && digit != 6
   on7 = digit != 2

   l1 = if on1 then [p1, p4] else [p1, p1]
   l2 = if on2 then [p2, p5] else [p2, p2]
   l3 = if on3 then [p3, p6] else [p3, p3]
   l4 = if on4 then [p1, p2] else [p1, p1]
   l5 = if on5 then [p2, p3] else [p2, p2]
   l6 = if on6 then [p4, p5] else [p4, p4]
   l7 = if on7 then [p5, p6] else [p5, p5]
}
```