

Notes to accompany the "Rethinking Programming" seminar

Operate in the ~wmb/public/projects/misc/HEAPSORT/HEAPSORT2009 directory

[Preface this demonstration with a look at the Run.e file, and the automated version of heapsort - use the dmt model to show how the dependency is modelled - this gives an idea of the richness of the dependency structure involved.]

1. Introduce

stage2.s, stage2.d, stage2.e

```
include("RunStage2.e");
```

Note the qualities of the artefact as an analogue representation rather than a traditional ADT

Note how observables that are represented visually are meaningful in the comprehension of the heap e.g. ixgtch1 (index of greater child), hc1 (heap condition at node 1)

[In order to define the last value to be the square of the first, we should require definitions such as:

```
val is [v1, v2, v3, v4, v5, v6, v7];  
v7 is v1 * v1;  
v1 = 7; v2 = 91; v3 = 19;  
v4 = 90; v5 = 21; v6 = 3;  
]
```

2. Consider its educational role as an animated whiteboard ...

```
first = 3; last = 5;  
etc.
```

See how the index of the greatest child is 'known to the system'

```
?ixgtch3;  
?hc2;
```

No real behaviour for the model - open to free interaction of

```
val[1] = 23;  
etc
```

Capacity for restoring / entering state at will (without 'invoking a procedure')

```
val0 = [2,4,3,6,5,7,1];  
val = val0;
```

3. Demonstrate how you can take the development 'backwards' to a simpler version of the heap model, where first and last are not changeable, and the definitions of ixgtch1, hc1 are

based on the assumption that all the elements in the array are present in the heap (and are thus not framed in terms of the conditions `inhp1, imhp2, ...`).

```
include("change21");
```

4. Definition of `hc1` and `ixgtch1` are now easier to understand - a conceptual stage preparatory to grasping the general notion of a heap as a segment of an array with a first and last index.

5. Can start to explore effect of actions, such as changing / exchanging values has on the heap condition at nodes etc. - introducing `exc.e` to give requisite procedure.

```
include("exc.e");
```

6. Can turn this into a GUI style environment for interaction with the heap by introducing the file `change23` (with Model 1 in place).

```
include("change23");
```

Now the user can experiment by clicking at nodes.

[Seem to encounter a bug in Scout at this point, whereby the `heapwin` window isn't sensitive, despite its specification:

```
%scout
window heapwin = {
  type: DONALD
  box: [{0, 100}, {500, 600}]
  pict: "Heapview"
  bgcolor: white
  border: 0
  sensitive: ON
};
```

Can check whether get sensitivity through opening the "Command History" and looking for mouse button redefinitions. Re-entering this Scout definition seems to fix the problem. WMB 23/11/09]

Learning issues:

```
## is the user simply responding to the colours of nodes?
    c3col is (hc3)? "blue": "red";

## motivates redefinitions:
    c3col is (hc3)? BLUE: RED;
    BLUE = "blue";
    RED is BLUE;
##     etc

## reset via
    RED = "red";

## accuracy of clicking (near = 10)
    near = 4
## ---> test of skill in use of the mouse
```

```
near = 100
near = 1000
```

Script extension needed for recolouring here is:

```
c1col is (hc1) ? BLUE : RED;
c2col is (hc2) ? BLUE : RED;
c3col is (hc3) ? BLUE : RED;
c4col is (hc4) ? BLUE : RED;
c5col is (hc5) ? BLUE : RED;
c6col is (hc6) ? BLUE : RED;
c7col is (hc7) ? BLUE : RED;
A_l12 is "color="//((ord12==0) ? BLACK:((ord12== 1)?BLUE: RED));
A_l13 is "color="//((ord13==0) ? BLACK:((ord13== 1)?BLUE: RED));
A_l24 is "color="//((ord24==0) ? BLACK:((ord24== 1)?BLUE: RED));
A_l25 is "color="//((ord25==0) ? BLACK:((ord25== 1)?BLUE: RED));
A_l36 is "color="//((ord36==0) ? BLACK:((ord36== 1)?BLUE: RED));
A_l37 is "color="//((ord37==0) ? BLACK:((ord37== 1)?BLUE: RED));

A_c4 is "outlinecolor="//c4col;
A_c5 is "outlinecolor="//c5col;
A_c6 is "outlinecolor="//c6col;
A_c7 is "outlinecolor="//c7col;
```

7. Moving to a more constrained behaviour ...

```
include("change12");
include("change13.2");
include("animate.e");
include("add.e");
```

Can now click to get animation of heapsort where in principle have observables such as the phase of heapsort ...

[demonstrate this]

The *Is this heapsort?* issue ...

The pattern of exchanges is precisely as prescribed in heapsort, but there is no oblivious behaviour as specified by a procedure (consider how the next index at which to make an exchange if necessary is determined in trad procedural heapsort). Proof of this ...

```
first = 4;
last = 7;
next = 0;
```

Start heapsorting, then intervene to put

```
val[7] = 8;
say - then continue the 'heapsort' procedure.
```

Finally: display Rungrattanaubol's extension with the WP precondition spec alongside.