Lab 2: Repairing The Lift

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

Introduction

In lab session 1 you were introduced to the tkeden environment and learnt how to write dependencies in the `%eden` notation. This lab session you will learn more about the `%eden` notation. In particular, by the end of the session you should know how to use functions, procedures, triggered procedures and clocks.

During this lab session you will be using a model of a lift.

1 Loading the model

The lift model for this lab session can be found in:

```
/dcs/emp/empublic/projects/liftHarfield2005
```

Move to this directory and load tkeden. Execute 'Run.e' followed by 'prepare.e':

```
tkeden Run.e prepare.e
```

> **Task 1**
>
> Investigate the model. What problems can you find in the model? How is the model different from the DCS lift? Discuss in your groups or with a partner.

2 Writing functions and procedures

Recall from the notes that a function in eden has the basic form:

```
func max {
   para m;                      /* m is alias of first argument */
   auto i;                      /* local variable */
   for (i = 2; i <= $#; i++)    /* for each argument */
      if ($[i] > m) m = $[i];   /* keep max */
   return m;                    /* return max of all arguments */
};
```

A procedure is similar to a function except that it does not return any value:

```
proc printhelloworld {
   writeln("hello world");
}
```

The first issue to be addressed in the model is the American convention used for floor numbers. The function that displays the floor number in the lift is called ‘formatfloor’.

> **Task 2**
>
> Query the function ‘formatfloor’ to find out what it currently returns. Rewrite the function so that it returns British-style floor numbers (i.e. G, 1, 2, 3, etc).

3 Triggered procedures

A triggered procedure is a special type of procedure that gets called automatically when a specified observable changes. In the following, the procedure is triggered by the variables `a`, `b` or `c`. If any of these variables changes then the procedure is called, and the values of the three variables are printed out.

```
proc print : a, b, c {
   writeln(a, " ", b, " ", c);
};
```

Currently the buttons in the lift do not work. Instead, a message is sent to the terminal when a button is pressed. This message will help you locate the action that is responding to button presses.

> **Task 3**
>
> Change the lift button action so that when a button is pressed it is illuminated. Hint: lift button 1 can be illuminated by setting the value of ‘_car1’ to true.

4 Clocks

A clock is a special agent that updates an observable at regular intervals. When a clock is started it can be used to trigger some change regularly, and therefore is useful for animation and automation. This is a new feature since tkeden-1.64 and is only documented in the Eden Quick Reference (from the Help menu).

> **Task 4**
>
> Within the Help menu, look up the information on creating a clock. Start a clock which updates the observable ‘clocktick’ every 100 milliseconds. Check that the clock is ticking by querying it several times to see that it has incremented.

5 Making an animation

In this section you will be modifying the behaviour of the lift so that it no longer randomly jumps between floors. Instead, it should act like a regular lift in that it will move in one direction servicing requests before changing direction. The second part, which animates the movement of the lift, will show the behaviour of the lift more clearly.

> **Task 5.1**
>
> Look at the procedure ‘startmoving’. Currently it randomly selects a floor. Rewrite the procedure as specified in pseudo-code below. Check that the lift no longer randomly selects a floor.

```
## choose a direction
IF ‘direction’ is 1 (i.e. up) AND
   current floor (‘atfloor’) is greater or equal to ‘highestbuttonselected’ THEN
   change the ‘direction’ to -1 (i.e. down)
ELSE IF the ‘direction’ is -1 AND
   ‘atfloor’ is less than or equal to ‘lowestbuttonselected’ THEN
   change the ‘direction’ to 1
END IF

## move one floor in the direction
IF ‘direction’ is 1 THEN
   increment ‘liftposition’ by 10
ELSE
   decrement ‘liftposition’ by 10
END IF
```

This will cause the lift to select a direction and move one floor in that direction. The next task is to animate the lift, so that instead of jumping from floor to floor (i.e. adding 10 to the lift position at once), the lift moving can be observed moving in between floors. This is done by using the clock created earlier to increment the ‘liftposition’ every 100 milliseconds. We also need to introduce a new observable (called ‘moving’) to ensure that the clock only updates the ‘liftposition’ when required.

Task 5.2

Change the procedure ‘startmoving’ again by removing the part that increments the ‘liftposition’ and replacing it with ‘moving = TRUE;’. Change the procedure ‘visitfloor’ by adding a line at the top which stops the lift moving (i.e. ‘moving = FALSE;’). Create a procedure which is triggered by ‘clocktick’. The procedure must increment/decrement the ‘liftposition’ when the lift is ’moving’.

Once you have completed this task you should have a working lift model. This completes the lab session. If you have time then you can investigate the lift model that was demonstrated in the lectures (found in `/dcs/emp/empublic/projects/liftBeynon2003`) and try to run through the scenario.

Summary

In this lab session you should have learnt how to:

- write a function and a procedure.
- create triggered actions which are triggered by an observable and execute a procedure.
- start a clock which updates an observable at regular intervals.
- animate a model.

Revision history

This worksheet was written by Antony Harfield for CS405 Introduction to EM.