CS405 Introduction to Empirical Modelling
Lab 1 – Getting started with tkeden

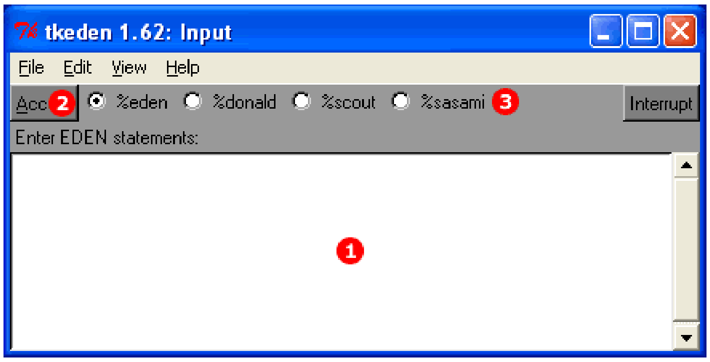
This lab session is designed to give you a brief introduction to the tools used in the Empirical Modelling course.

1 Getting started with EDEN, DoNaLD and SCOUT

EDEN is the Engine for DEfinitive Notations. It is the primary software tool of the Empirical Modelling research group. **DoNaLD** (for line drawing) and **SCOUT** (for window manipulation) are the principal definitive notations used in Empirical Modelling research in this department.

- Eden (files usually with .e extensions)
- Donald (with .d extensions) - a definitive notation for 2D line drawing.
- SCOUT (with .s extensions) - a notation for describing screen layout.
- tkeden - an interpreter for EDEN, DoNaLD and SCOUT.

We will primarily using the tkeden tool in the lab sessions. On the DCS Linux machines you can start the tool by typing "`tkeden`" in a shell. When you start the tkeden tool it will look like this:



The key elements of the interface are: the text entry box (1), the accept button (2) and the notation switches (3). Statements are entered into the text entry box (1) and then executed by clicking on the accept button (2). The environment enables the modeller to switch between notations on-the-fly by selecting the notation switches (3).

Other useful features in the environment include the ability to view all the observables and definitions currently in the model ('View', 'View Eden Definitions'). Basic guides to each of the main notations are available from the Help menu.

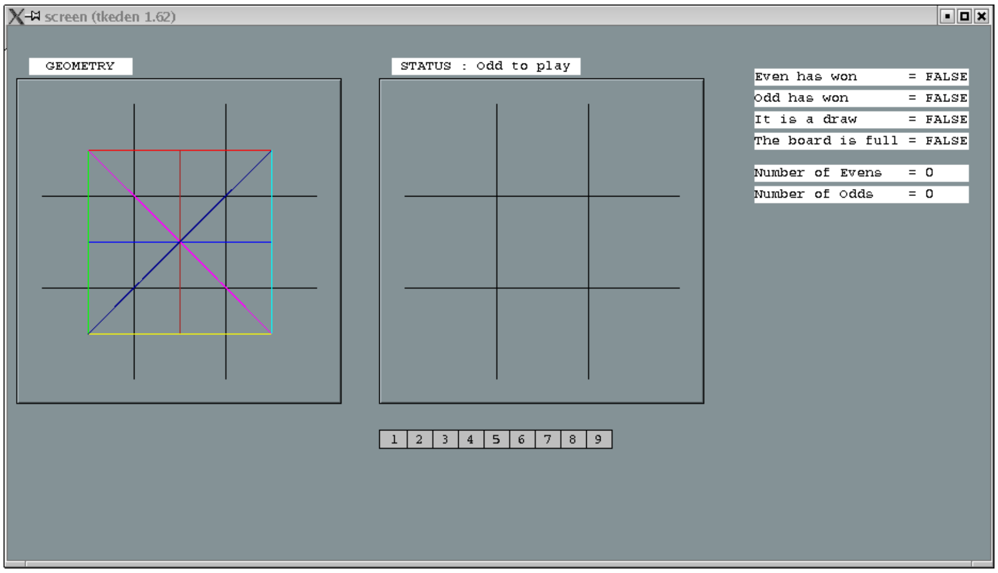


Figure 1: Tic-tac-toe model

2 Dependencies

You should have read the additional handout "Introduction to definitive notations and tkeden" but here's a quick reminder about writing dependencies in Eden. As an example, if we enter the Eden definition:

```
a is b + c;
```

we mean that observable `a` *is always* the sum of `b` and `c`. Whenever the value of `b` or `c` changes, the value of `a` is automatically recalculated. This is called a **dependency**. We can apply this to more complicated formulas that involve functions and conditionals.

Another example:

```
d is (e == 1) ? f : g;
```

In this case, if `e` equals 1, then the value of `d` is the same as `f`, otherwise the value of `d` is the same as `g`.

3 Tic-tac-toe

This week we'll be getting started with tkeden and the Eden notation by playing a game, tic-tac-toe. However, it will be necessary for you to complete the model before you can play.
Start the tool 'tkeden' and load the model with the following commands:

```
cd /dcs/emp/empublic/projects/tictactoeRoe2002/
tkeden Run.e
```

If you cannot see the entire screen (see figure 1) then either resize the window with the mouse, or type "`windowSize = 200;`" into the input window and click on 'Accept'.

3.1 Rules of Tic-Tac-Toe

This game is similar to noughts and crosses in that the object is to get three counters in a line. A player wins if they place the last counter in a line and the sum of the values in that line is equal to the target (which is 15).

Players take it in turns to place numbers in the grid. One player (Odd) can only play odd numbers and they always start. The other player (Even) can only play even numbers. They always go second. No number can be used more than once.

3.2 The model of Tic-Tac-Toe

Play with the model and have a go at the game. You should notice that the information windows on the right of the screen do not change. You can also cheat! There is no restriction on playing only odd or even numbers. There is no restriction on playing the same number twice.

The Task

The aim of this task is to add some dependencies to the model to complete it. The variables that need to be defined have been set up in the model already. By defining them correctly you can get the model to work properly. These are the variables that need to be defined:

1. **n1valid, n2valid, n3valid, n4valid, n5valid, n6valid, n7valid, n8valid, n9valid** – defines whether the first, second, etc. number can be selected.
2. **nofeven** – defines the number of even numbers on the board.
3. **nofodd** – defines the number of odd numbers on the board.
4. **evenwon** – defines whether Even has won.
5. **oddwon** – defines whether Odd has won.
6. **full** – defines whether the board is full.
7. **draw** – defines whether the game is a draw.

To redefine one of these you need to type in a command (e.g. `n1valid = TRUE;`) into the tkeden input window and click on the 'Accept' button.

Resources for the task

The existing model already has some definitions and functions present. In this exercise you may need to use some of the variables in this list:

1. `nofsquares` – is the number of squares in the grid (i.e. in this example 9).
2. `lastselection` – is the number that was last entered into the grid.
3. `target` – is the number that each player is trying to get in a line.
4. `player` – the person whose turn it is to put a number on the board.
5. `evenplayer` – a constant.
6. `oddplayer` – a constant.

In addition to the variables above, you may need to use some of the following functions:

1. `not_used(n);` n is a number. This function will give you back a value of TRUE if the number n does not appear in the grid. This function will give you back a value of FALSE if the number n appears in the grid.

2. `countevens();` will return the number of squares in the grid that contain even numbers.

3. `countodds();` will return the number of squares in the grid that contain odd numbers.

4. `isodd(n);` n is a number. This function will give you back a value of TRUE if the number n is odd. The function will give you back a value of FALSE if the number n is even.

5. `iseven(n);` n is a number. This function will give you back a value of TRUE if the number n is even. The function will give you back a value of FALSE if the number n is odd.

6. `checklines ( target );` target is the number which each player is trying to get in a line. This function will give you back a value of TRUE if there is a line containing three numbers which adds up to the value target. If there is not a line then this function will return a value of FALSE.

4 Summary

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

- know how to start tkeden, and execute definitive scripts;
- have an understanding of Eden, and be able to modify and query Eden variables;
- know how to write a dependency in Eden to link two or more observables; and,
- have an understanding of definitive principles and the general use of tkeden from the "Introduction to definitive notations and tkeden" guide.

5 Revision history

This worksheet was adapted from Chris Roe's Tic-Tac-Toe worksheet from the EM 2002 course, and the tic-tac-toe model preparation should be credited to him.