An EM construal of the game of Nim

Nim is a game in which two players take turns to remove stones from piles. At each turn, a player can take one or more stones from just one of the piles. The player who removes the last stone wins.

To load the Nim construal, you should run tkeden and use the Open option on the file menu to navigate to the presnimBurapha2013 subfolder of the UsingEMPE subfolder. You should then open and Accept the file run.eden. This should (after some time!) load the construal together with an EMPE presentation that includes the exercises for this session. It will also open a 'screen' and 'obsdetail' windows that you should *minimise* rather than close. At no stage should you close the EMPE window or the tkeden Input Window.

Slides 2-6 are an introduction to the Nim construal, and precede Exercise 1. Some helpful resources are discussed on Slide 7.

The primary function of an EM script is to provide a context for agent interaction. The file **humanNim.e** supports human-human play for instance. This file can be included via Slide 8.

Exercise 1: Your first exercise is to familiarise yourselves with the game of Nim. For this purpose, you can use the variant of the construal that is derived by including the humanNim.e file. This creates an environment in which there are three piles of stones, each containing up to 15 stones. You can start a game by generating three piles of random size, and use the button interface to remove stones from any one of the three piles.

Exercise 2 involves inspecting the file **humanNim.e** (in the presnimBurapha2013 folder) using a text editor such as Notepad++. The script is quite simple, but the use of several notations potentially makes it more confusing. You should take careful note of the comments on Slide 9.

Exercise 2: Inspect the script that defines the first pile of stones within the **humanNim.e** file. There are three notations and four constituent parts - **%donald** code to define the stones as circles and locate them on the screen and **%eden** code to determine the attributes of these circles, **%scout** code which frames the Donald line drawing and determines how and where it is displayed on the screen and auxiliary **%eden** code to implement the button actions. ...

To explore the Nim construal in more detail, we restore the original form of the construal (via Slide 10) and examine its construction in more detail. For exploration purposes, it is useful to refer to the "symbolviewer" by means of which the principal observables in the construal can be inspected (see the **obsdetail** window). Note that observables may be defined in one of three different notations (**%eden**, **%donald**, **%scout**), but it is their underlying definitions as translated into the EDEN notation that are recorded in the symbol viewer.

The next exercise is intended to help you understand how the screen is specified in %scout. In tkeden, you can redefine the screen directly in %scout, as in the idealised form of Exercise 3 below.

Exercise 3 [as it ought to be!]: Study the way in which the %scout notation is used to specify the layout of the screen in the file Nim.s. Note the implications of making the following redefinitions of the observable 'screen' in %scout:

- screen = <winPileOne / winPileTwo / winPileThree >;

Unfortunately, this kind of redefinition cannot be done via the EMPE interface. Fortunately, the %eden definitions of the screen that are derived by translation are very straightforward, and it is these that are implemented in Slides 11 and 12. Exercise 3 gives the background for understanding how the two variants of the Nim construal introduced previously can be blended. For this purpose, you will need to inspect the file **blendWinPiles.e** using a text editor.

Exercise 4: Study the file **blendWinPiles.e** to see how the windows **winPileOne**, **winPileTwo** and **winPileThree** in which the piles of stones are displayed are transformed and relocated so that they can be integrated into the Nim construal. By applying the same principles to the Scout window **backBox**, as it is used in the last of the screen definitions specified in Exercise 3, introduce an observable **displayBinaryRep** that can be set to to **true** or **false** to reveal and hide the information about the binary representations of the pile sizes.

Displaying the binary representations allows us to explore the winning strategy for Nim. The interested reader who is mathematically minded can study this strategy by referring to Slides 15 through 19 and tackling Exercise 5. The point of Exercise 5 is to raise some of the questions that arise in figuring out the significance of the Nim-Sum strategy. The construal is intended to be helpful in exploring these questions. Slide 19 offers you the chance to hide or reveal the answers - by default they are revealed on the following slide.

As an alternative, or adjunct, to Exercise 5, you may wish to refer to the last two slides of the EMPE presentation and try to develop solutions for the following technical problems:

Exercise 6:

1. Add a textual annotation of the form "This situation is won/lost with best play", linked by dependency to the current Nim-Sum.
2. Change the background colour of clickable buttons on the interface to distinguish them from mere display panels.
3. Adapt the interface so that when play is in progress, the text showing the active pile is highlighted in green and the text for the other two piles is red.
4. Adapt the interface to the windows displaying piles of stones so that you can click on individual stones to remove them.
5. Dispose the stones in a 8-4-2-1 rather than a 5-4-3-2-1 formation in the winPileOne, winPileTwo and winPileThree windows, and display the stones in a way that reflects the binary representation of the pile size (so that for instance, when there are five stones in a pile, this is displayed as 4+1 stones).
6. When the Nim-Sum is non-zero highlight the piles from which stones can be taken to achieve Nim-Sum zero, and display the number of stones to be taken.
7. Automate a Nim player that implements best play.

Getting complete solutions for these problems is beyond the scope of this session - you will do well to understand the principles behind their solution. A solution to Exercise 6.5 is cognitively interesting and can be implemented via the final slide. You may find it instructive to revisit Exercise 1 - understanding how to play Nim - when you have read about the Nim-Sum strategy and when you have seen the construction described in Exercise 6.5.
