The Fox & Hounds Challenge

the game

Based on the concept of an old board game called fox & hounds, the world contains 10 hounds and a fox (but you can change the numbers with a slider). The hounds try to catch the fox by landing on it & the fox tries not to be caught (by running away from the hounds).

the board

The set-up used in the challenge is a 40 x 40 grid of squares (called "patches" in NetLogo) with no obstacles or blocked off squares. The board looks like a square but is actually a toroid (ie: wraps around both horizontally and vertically).

setting up

The fox & hounds are placed at random locations in the world & then the fox is allowed to make 10 moves (this gives the fox a chance if it is placed in a really unfortunate position).

playing

Every turn, the fox is allowed to move, then each of the hounds. If any of the hounds land on the fox the game ends. Two foxes can be played off against each other in two ways...

1. they are both put into the same pack of hounds at the same time & the one that lasts the longest is the winner;
2. they take it in turns to face the same pack of hounds & the one that lasts the longest is the winner.

the challenge

Each team builds a fox in NetLogo (see comments about hounds & details about NetLogo implementations provided later). Teams enter their foxes into a tournament which runs at the end of the competition.

Teams will (almost certainly) need to build hounds to try out their foxes, teams may enter their hounds into the tournament as well as their foxes. Part of the tournament will be judging the best hounds.

NetLogo coding details

See skeleton NetLogo code provided – this gives a file structure (which must be followed by competing teams) and specifies basic code for setting up and running the *Fox & Hounds* NetLogo model. It also contains the code for very simple foxes and hounds.

Comments in NetLogo start with a semi-colon and end at the end of the line they are on. There are three types of comment you need to notice in the file...

1. comments which divide the model into sections, eg:

```
;----------------------------------
; setup hounds
;----------------------------------
```

2. comments which start ;; <*>
   These comments are used to highlight key points in the model where competitors need to add their own code. These key points are described as below. Note that other sections of the code may **not** be modified (if in doubt ask one of the tournament organisers).

;; <*> extensions
declare any NetLogo extensions used in your model (table, array, etc) at this point.

;; <*> globals
declare any global variables used

;; <*> fox-variables
declare variables owned by foxes

;; <*> hound-variables
declare variables owned by hounds

;; <*> setup-globals
provide any model code needed to initialise your global variables

;; <*> setup-smart-foxes
provide the model code needed to setup your foxes
you may **not** include any code to position or move your foxes

;; <*> setup-smart-hounds
provide the model code needed to setup your hounds
you may **not** include any code to position or move your hounds

;; <*> smart-hound-move
provide the model code needed for each hound to get it to point in the direction you would like it to move. Hounds may not move location during this process. After running this code, existing model code will move each hound forward 1 patch.
Note: if there are circumstances when you want your hounds to stay still you may wrap the *forward* statement in an *if* block.

;; <*> smart-fox-move
provide the model code needed for each fox to get it to point in the direction you would like it to move. Foxes may not move location during this process. After running this code, existing model code will move each fox forward 1.5 patches. Note: if there are circumstances when you want your foxes to stay still you may wrap the *forward* statement in an *if* block.

other code notes

- please write code in the *spirit* of the game, ie: do not attempt to subvert the principals of play (it would not be acceptable, for example, to ask a fox to turn or move in the move-hounds procedure);
- do not change the interface (by adding buttons, etc);
- do not change the "shape" of turtles (the default shape allows users to see the directions of turtles.