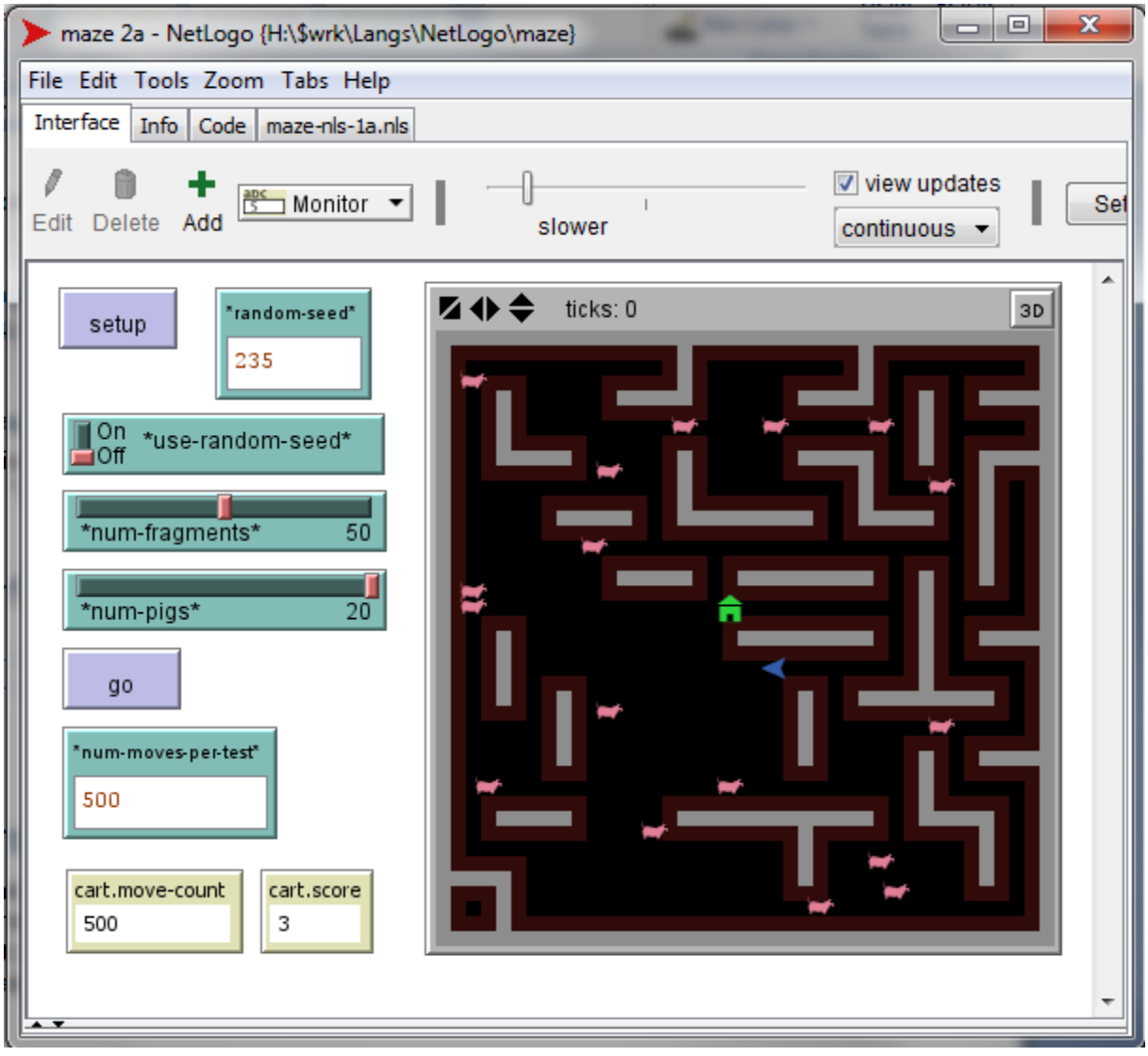
The tPIG competition 2012

brief

This year's competition from tPIG is a (poorly disguised) maze traversal problem.

Your objective is to write the controller for a golf cart that trundles round a maze rescuing pigs that have got lost & brings them back to a pig-sty.

The competition will be played out in a Netlogo world something like the one below.



The pink things are pigs; the green house is the sty & the blue dart is the cart.

Each competitor will get a chance to run their solution in the same 5 randomly produced mazes. There will be a time limit in each maze (measured in ticks). The winner will be the solution which rescues the greatest number of pigs in the shortest amount of time.

notes

1. the cart can only move one whole patch at a time – it cannot move part patches or multiple patches;
2. the cart only turns by 90 degrees – so only ever faces in one of 4 directions;
3. the pigs do not move (they are happily eating);
4. the golf cart controller cannot drive over maze walls;
5. the cart controller cannot look around, the primitives supplied in the model allows your controller to see if there is a wall directly in front of it (or directly to the left or right) but not look further than the next patch. It also knows when it is on top of a pig or the sty.

the rules

What follows are some of the competition rules (what you allowed to do and what you are not allowed to do). If we have left something out (ie: forgotten to specify a rule) which is obviously in the spirit of the competition & your solution breaks that then it will be disqualified from that maze. If you are disqualified from a maze you will gain no pig rescues for that maze & be scored the maximum number of ticks – you will still be allowed to run your solutions through the other mazes.

Here's a list of things you get disqualified for...

- moving onto blocked or wall patches
- moving out of the maze
- changing position without using the *cart* methods (ie: using any of the normal Netlogo primitives for moving turtles like *forward*, *jump*, etc)
- finding out where pigs (or other features) are other than by landing on the patches they occupy (ie: you should only use *cart.pig-here?* to find them not *ask* them for their positions)

As a rule-of-thumb we recommend that you only use the cart methods supplied (see section below "supplied procedures/methods") to move the cart and interrogate its environment.

the supplied code

The NetLogo code for the maze model is in a ".nls" include file. Check the release notes for its name. Also provided is an example maze model. To write your own solution you should build your own maze model in the following way.

importing the include file

to import the include file put the following at the start of your model code...

```
__includes ["maze-include-1a.nls"]
```

global & breed variables

you may declare/use global variables, you may not declare/use *cart* breed variables.

initialising carts

your model should contain the definition for a procedure/method called cart.setup, this will run automatically, leave the body of cart.setup empty if you do not want any code to run.

```
to cart.setup
  ;; put set up code here
end
```

running carts

put the code to run your cart in a procedure/method called *cart.make-move*.

```
to cart.make-move
  ;; put code to run your cart here
end
```

other procedures/methods

you may include other procedures/methods in your model (but make sure you follow the restrictions/rules).

supplied procedures / methods

The include file we supply contains a collection of procedures to move your cart and interrogate its environment.

cart.fwd

- move the cart forward one patch (it will not move if the patch ahead is a wall)

cart.right

- turn the cart right 90 deg

cart.left

- turn the cart right 90 deg

cart.turn180

- turn the cart around 180 deg

cart.sty-here?

- reports true/false if the cart is at a sty

cart.pig-here?

- reports true/false if the cart has reached a pig

cart.blocked-ahead?

- reports true/false if the patch ahead is blocked (ie: an internal or boundary wall)

cart.blocked-left?

- reports true/false if the patch ahead is blocked (ie: a wall)

to-report cart.blocked-right?

- reports true/false if the patch ahead is blocked (ie: a wall)

cart.grab-pig

- pick up a pig (NB: the cart must have reached a pig to pick it up). When the cart grabs a pig, the cart changes color & the pig disappears. A cart can only carry one pig at a time and can only drop them off at a sty.

cart.drop-pig

- drop a pig & score some points (NB: the cart must be carrying a pig and have reached a sty)

cart.set-patch-tag [tag]

- allows a cart to mark the patch it is on (you should not use any other approach to do this kind of thing)

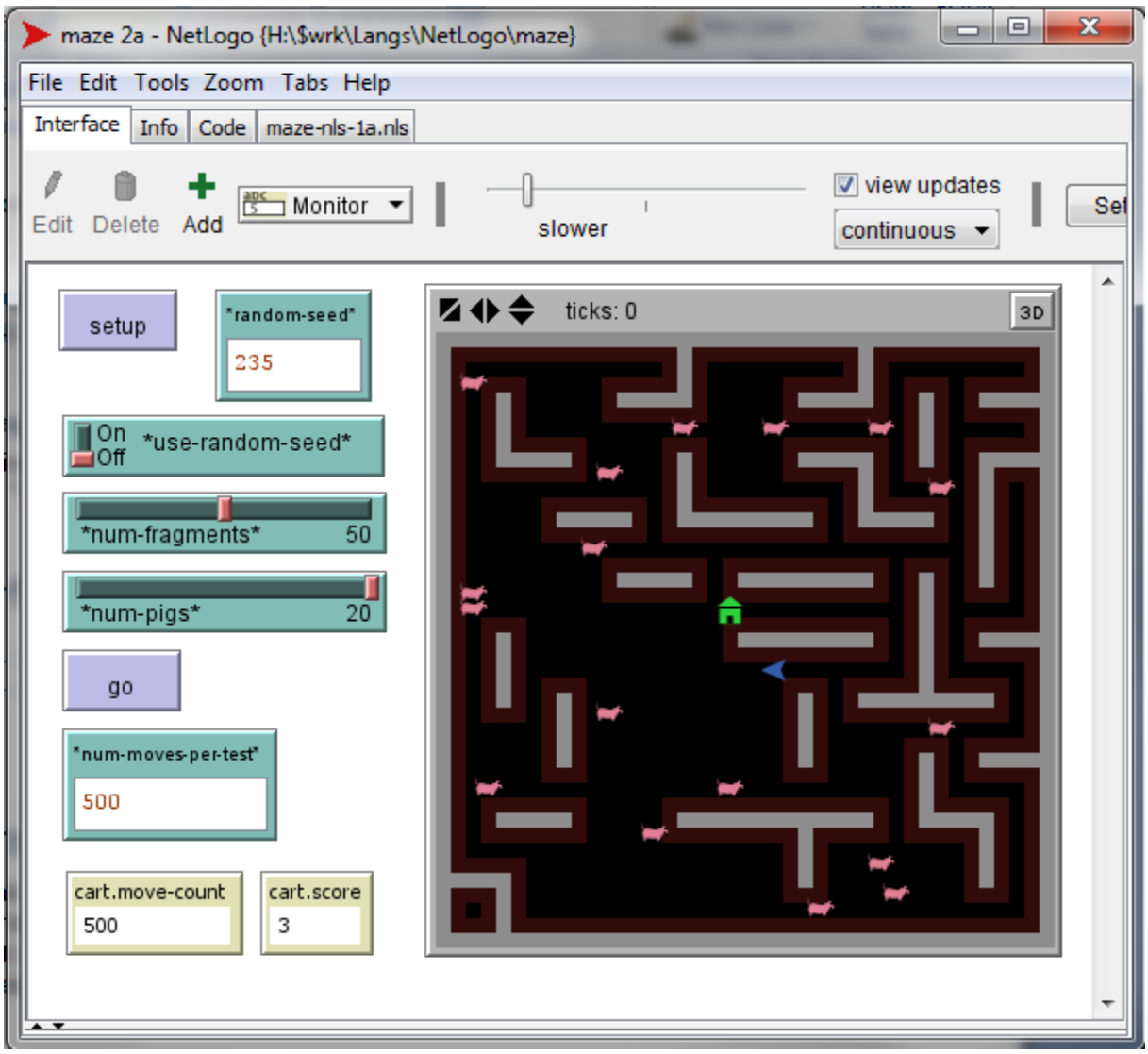
cart.get-patch-tag [tag]

- reports a mark previously made using *cart.set-patch-tag*. If no patch tag has been set the value will be "undef".

cart.stopping?

- this is a variable that will normally stay "false", set it to "true" if you want to stop your model.

the model interface & running the model



setup

- this sets up the model, drawing a new maze, placing pigs, etc. It also calls your cart.setup procedure.

random-seed

- creation of the maze, position of pigs, etc. are randomised. The use of a random-seed allows mazes/tests to be repeated – if the same random seed is used then the maze will be the same.

use-random-seed

- when this is "on" the random seed (above) is used, otherwise no known seed is used (so tests cannot be repeated).

num-fragments

- the number of maze fragments (pieces of maze wall) to be drawn. Each piece of wall is 4 patches long & 1 patch wide.

num-pigs

- the number of pigs to put n the maze.

num-moves-per-test

- the number of moves a cart is allowed before the test is stopped.

cart.move-count

- the number of moves the cart is made so far – this is updated as the model runs.

cart.score

- the score for the current cart – based on the number pigs successfully rescued.

judging the competition

All the competition entries will be run through the same 5 mazes for a set number of moves. The entry that rescues the most pigs (or rescues all pigs in the shortest number of moves) will win.

known problems

- the pigs look like pink sheep;
- it is not possible to hide/disable NetLogo primitives so competitors (you!) are trusted not to use those that break the rules (but since it's a competition expect us to look at your code closely – especially if you look like you might win :o)

contact us

For more information check the "PIG" link at www.agent-domain.org or contact Simon at s.c.lynch@tees.ac.uk