NetLogo Hill Climbing Challenge

Basic Description

Your task is to create a breed of turtles that is able – individually or collectively, but with limited communication abilities – to find the highest point in an unknown terrain. That terrain will be smooth, but not necessarily convex or unimodal – i.e. there might be many hills that contain the high spots in their local neighborhoods, but there's no guarantee that the peak of a given hill is the highest point in the entire terrain.

Time Limits

- The challenge will run for an unspecified number of ticks. However, midway through the running time, turtles will be notified of this milestone via a **`mark-halftime-<breeds>`** procedure. If you want your turtles to change their behaviors after half the time has elapsed, it's your responsibility to implement that logic in this procedure.
- Shortly before time expires, a **`mark-warning-<breeds>`** procedure will be invoked for all turtles. The amount of time remaining will not be specified; however, it will be enough time for a turtle to move in a straight line from any point in the world to any other point in the world – but probably not much more than that.
- When time expires, only those turtles within a distance of 1 of the highest point on the terrain will survive; all others will be killed. There is no limit to the number of turtles that can stand on any given patch, including the patch representing the highest point.

Turtle Abilities and Constraints

- You may define a single breed, or multiple breeds; if the latter, the initial combined population size will be the same as if you had defined a single breed, but the proportions of the total for multiple breeds will be random.
- Every tick, each turtle will be asked to perform the **`move-<breeds>`** procedure; you must include this procedure in your code. This is the only procedure in which your turtles are allowed to move.
- Turtles may not move a distance greater than 1 in any single tick. (Please remember that a move from the center of a patch to the center of a diagonally adjacent patch is a distance greater than 1.)
- Turtles may not examine any aspect of the terrain at a distance greater than 1. For example, **`patch-here`**, **`patch-ahead`**, **`patch-right-and-ahead`**, **`patch-left-and-ahead`**, and **`patch-at-heading-and-distance`** may all be used by turtles to refer to nearby patches, as long as the specified distance is less than or equal to 1. **`patch-at`** may also be used, but with more care (since, for example, specifying 1 for *dx* and 1 for *dy* will result in a distance of $\sqrt{2}$).

- Turtles may remember any number of patches they've visited, but if you want turtles to use some attributes of such patches (e.g. **`height`**, **`pxcor`**, **`pycor`**) when the patches are more than 1 step away, the turtles will need to store that information as well. The only thing turtles are allowed to do with distant patches is **`face`** them, use the direction to the patch given by **`towards`**, and get the **`distance`** to them. (The implication is that once a patch is visited, the turtle has a perfect sense of direction to return to the patch, and knows how far away it is. Anything else must be remembered in custom turtle attributes.)
- Turtles may communicate with other turtles from the same breed, or from any of the breeds defined in your code. This communication may consist of reading other turtles' attributes, asking other turtles to do some work, etc.
- For best results, use breed-specific commands and reporters for communication between turtles.
- Turtles may not communicate with turtles defined in another participant's code.
- Turtle communication can only take place between turtles on the same patch – i.e. using some selection from **`turtles-here`**, **`<`*`breeds`*`>-here`**, etc..
- All standard built-in turtle attributes (**`who`**, **`size`**, **`shape`**, **`color`**, **`breed`**, **`heading`**, **`xcor`**, **`ycor`**, etc.) may be read by your code; only **`heading`**, **`xcor`**, and **`ycor`** may be modified (directly or indirectly) by your code, using the movement primitives.
- Every turtle will also have a **`health`** attribute; this may be read – but not modified – by your code. Some hazards (below) decrease the value of this attribute; when a turtles' health falls to 0 or lower, it dies.
- You may define and use additional attributes for your turtles, using a **`<`*`breeds`*`>-own`** statement. Your code may freely modify these, as long as no turtle either reads (via **`of`**) or modifies (using **`ask`**) the attributes of a turtle that isn't on the same patch. If initialization of these attributes is needed, your code should include a **`setup-<`*`breeds`*`>`** procedure, which will be called automatically for each turtle. (Don't forget: All uninitialized attributes have a default value of 0; that may not be appropriate for some attributes, especially if they're intended to store lists, agent references, or agent sets.)
- At the start of play, some number of turtles (the same number for each participant in a heat) will be placed on the terrain randomly; no turtle will be placed on a hazard patch (see below).
- Turtles may read the standard built-in patch attributes (**`color`**, **`pxcor`**, **`pycor`**, etc.), but aren't allowed to modify any of them.
- Each patch will have a **`height`** attribute, which may be read – but not modified – by your code.
- Your code may must not define any custom patch attributes – i.e. you're not allowed to have a **`patches-own`** statement in your code.
- The only predefined global variable (besides built-in constants, such as the symbolic

color names, the **base-colors** list, **pi**, **e**, etc.) that may be read by the turtles is **ticks**. (Of course, turtles aren't allowed to update the clock using **tick** or **tick-advance**.) In particular, turtles aren't allowed to read **world-height**, **world-width**, **min-pxcor**, **min-pycor**, **max-pxcor**, and **max-pycor**.

- Since the reporters **random-xcor**, **random-ycor**, **random-pxcor**, and **random-pycor** give information (particularly when used multiple times) about the world dimensions, these reporters may not be used by turtles in your code.
- Turtles may not ask patches, ties, or other turtles to perform any action which would indirectly violate the above rules.

Terrain and Hazards

- The topology of the terrain will be a box; no horizontal or vertical wrapping will be used.
- The actual size of the terrain will not be published in advance.
- All of the normal terrain (including the hilltops) will be colored some shade of brown. (If needed, use the **shade-of?** primitive to test for shades of a color.) However, there will also be hazards on the terrain:
    - Radioactive pits – These are red patches; a turtle landing on one of these will die instantly. Generally, they will placed as individual patches, but random placement may result in multiple adjacent pits.
    - Tar pits – These are green patches, which slow down travel for turtles passing over them. A turtle landing on one of these will not be able to leave that patch until a randomly determined number of ticks (between 10 and 20) has elapsed. These will also be placed individually and randomly.
    - Barbed-wire fence – Gray patches which inflict damage, reducing the health of turtles that land on them. These will be placed in strings of connected patches, but no more than 10 in a row.
    - Turtle-pult – These violet patches are catapults which will fling a turtle to a random spot on the terrain (but not directly on a hazard patch). This can be good or bad, in terms of a turtle's location, but they definitely damage a turtle's health, and will require some number of ticks (between 10 and 20).

  The quantitative effects of fences and turtle-pults on turtle health will be somewhat random, but will not be so great as to cause death with fewer than 5 incidents.

Winning the Game

- The preliminary round will consist of heats including 2-5 players, and will be won by the player with the most surviving turtles – i.e. those that are at the highest point of the terrain when time expires. If time (and the number of participants) permits, the preliminary round will be done in round-robin style, so that each player has the opportunity to compete multiple times.
- The final round will include the highest 2-5 scorers (in terms of heats won) from the preliminary round; there will be 3-5 heats in the final round, with the scores averaged across the heats to determine the winner.

Clarification and Judging

- Requests for clarification of the rules should be directed to instructors and RAs in advance. Those same people will serve as judges of the event, to resolve any disagreements or unforeseen ambiguities in the rules, and they will have the last say. Attempts at bribery are allowed, but aren't guaranteed to be effective.