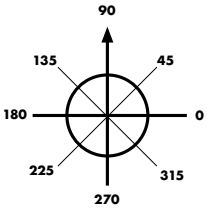
heading



This is a global variable which to which you can assign the desired heading (direction) for your reindeer. It is in degrees from 0 to 360; 0 is due East, 45 is Northeast, and so on. (You can assign values outside that range, but they will be remapped to the standard range on the next frame.) Robo-reindeer can't turn instantly, and can't turn at all when they're out of energy, but they will turn (whichever way is shortest) to this heading as soon as they can.

speed

This global variable sets your reindeer's target speed, as a percentage of maximum speed. Negative values shift your robo-reindeer into reverse. The valid range for this is -50 to 100. Just as with **heading**, the reindeer may will achieve this speed as quickly as possible, given a limited acceleration rate and energy. Note that movement consumes energy in proportion to your actual speed, and when the reindeer's energy level is below 5, it will start to slow down (until at an energy level of 0 or less, it can't move at all).

(**heading** and **speed** are the only special variables you should assign to, though of course you can make up your own variables and assign to those as you please.)

actual

This is a global variable you should only read, not set. It returns a map containing two elements: **heading** and **speed**. These are the actual current heading and speed of your robo-reindeer (as opposed to the global variables above, which are your *target* heading and speed). So, **actual.heading** is which way your reindeer is facing, and **actual.speed** is how fast your reindeer is moving.

deerCount

You can tell how many robo-reindeer are still in the arena by this global variable. This count includes yourself.

energy

This global variable tells you your robo-reindeer's energy level. Energy is used for almost everything a robo-reindeer does: movement, turning, throwing snowballs, even thinking. Energy slowly increases (robo-reindeer are solar powered) up to a maximum of 100. Energy may go negative (dipping into emergency reserve power), but the reindeer will be frozen in place until its energy level increases past zero.

health

Use this global variable to read your reindeer's current hit points. These start at 100, and when they reach zero, the robo-reindeer is scrapped.

position

This global variable is a map containing the current **x** and **y** position of your reindeer. **position.x** values go from -50 (left side of the arena) to +50 (right). **position.y** ranges from -50 (bottom of the arena) to +50 (top).

look

This is a function that returns information on the closest reindeer within 5° of your current (actual) heading. If there is no reindeer in that direction, it will return null. Otherwise, the result is a map containing:

- **distance**: distance to the reindeer seen (in the same units as **position**)
- **heading**: which way the other reindeer is facing
- **speed**: how fast the other reindeer is moving
- **energy**: how much energy the other reindeer has
- **health**: how many hit points the other reindeer has left

throw(energy=20)

This function is used to throw a snowball. You can choose how much energy to put into the snowball; this amount is subtracted from your own energy, and is how many points of health are subtracted from any reindeer hit. If you say **throw** with no parameters, it throws a snowball with 20 energy. The snowball begins just in front of your reindeer, and travels in the direction of your current (actual) heading.

drop(energy=20, delay=5)

This function drops a meadow mine. The mine lays on the field, inert for the specified *delay* number of seconds; after that it is armed and dangerous (including to the reindeer who dropped it!). When hit it explodes, delivering **2** points of damage for every point of energy that was put into it.