Homework 4 — Turmites

02-601

1. Set up

Unzip and copy the `turmites.zip` starter code (from the Box drive linked from Diderot) into your `go/src` directory. Ensure you have the packages required to draw to a canvas installed in your `go/src` directory as well.

2. Assignment

2.1 Turmites

A "Turmite" is a simple animal with the following characteristics:

- It lives in the 2D plane on an $n \times n$ grid. At any give time, its position is one of the cells of this grid.
- It can face either North, South, East, or West (but no other direction).
- It can be in one of 26 different "states" called `a`, `b`, ..., `z`.
- It can sense the color of the cell that it is currently in.
- It has a simple behavioral pattern, describe below.

A Turmite "move" takes the following actions:

- Change the color of the square that it is on to some color.
- Set its state to a new value in `a...z`.
- Turn 0, 90 or 180 degrees relative to the direction it is facing (forward, left, right, or backward).
- Walk 1 step in the direction it is now facing.

These actions are completely determined by the Turmite's current state and the color of the square. Which direction it turns to, what color it paints the current cell, and which new state it enters is entirely dependent on its current state and the current color of the cell it is one. This means we can think of the Turmite "brain" as having a collection of **if** statements of the form:

```
if cur_state == 'a' && cur_color == 0 {
    state = new_state
    set_color(current_square, new_color)
    facing_direction = new_direction(facing_direction, turn)
    move_1_step(facing_direction)
} else if ...
```

We can specify the contents of this "brain" with a set of rules of the form:

```
cur_state cur_color -> new_state new_color turn
```

where `turn` is one of `left`, `right`, `backward`, `forward`.

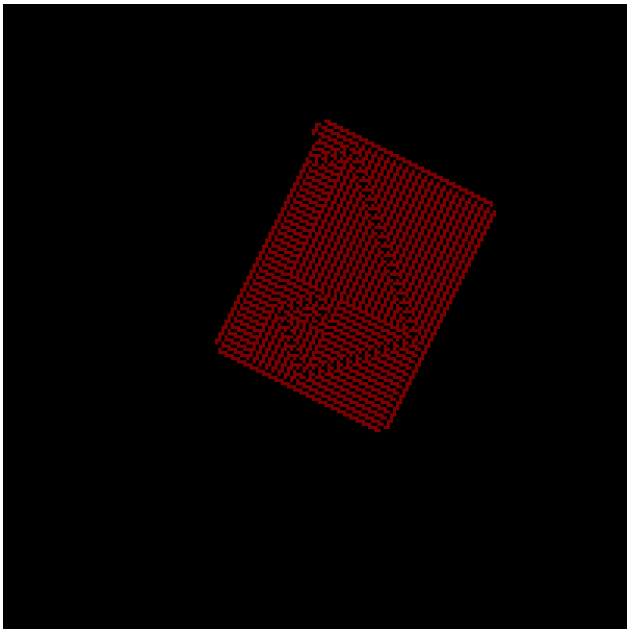
One of these rules "fires" if the Turmite is in state `cur_state` while sitting on a cell that is colored `cur_color`. When it fires, the Turmite paints the current cell `new_color`, changes its internal state to be `new_state`, changes its direction according to `turn` (if `forward` the direction stays the same; if `backward` the Turmite turns 180 degrees; if `left` or `right` the Turmite turns 90 degrees in that direction) and then moves one cell in that direction.

For example, here's a small Turmite brain:

```
a 0 -> a 1 left
a 1 -> b 0 forward
b 0 -> a 1 right
b 1 -> a 1 right
```

Here, there are two states (`a` and `b`) and two colors (0 and 1).

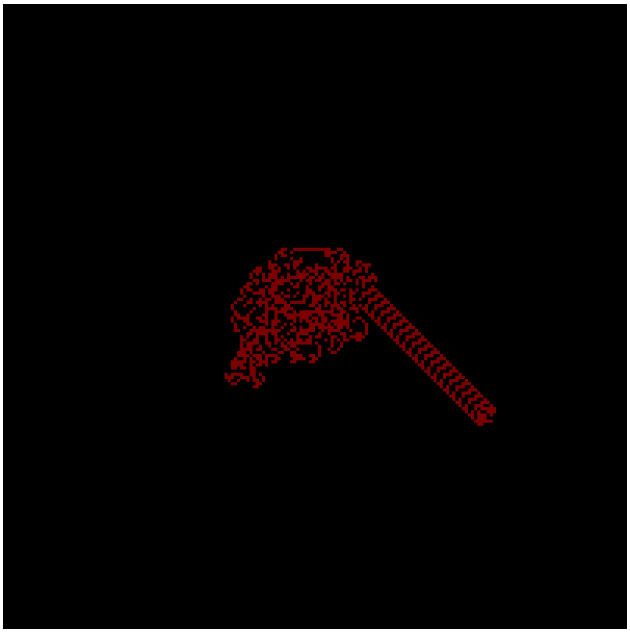
If we put this Turmite onto a 2D plane where every cell has color 0 (black) to start and let it do its thing for a while, we get the following picture:



Example 2. Consider this even simpler Turmite:

```
a 0 -> a 1 left
a 1 -> a 0 right
```

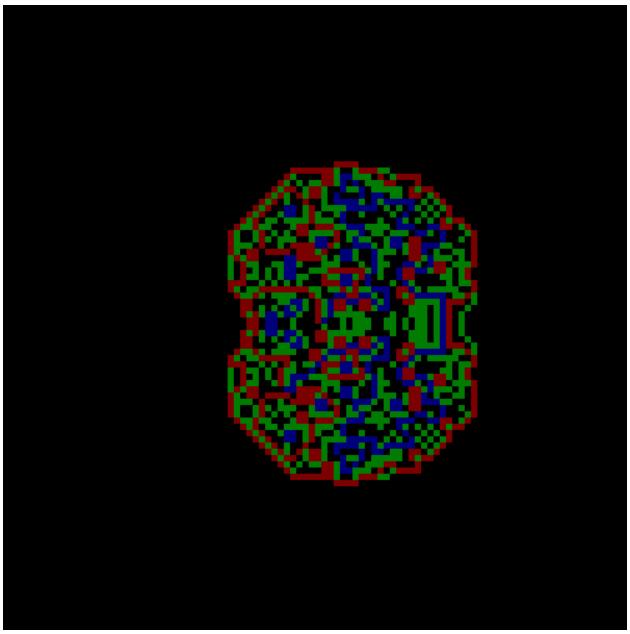
This produces:



Example 3. Here's another 'mite brain:

```
a 0 -> a 1 right
a 1 -> a 2 right
a 2 -> a 3 left
a 3 -> a 0 left
```

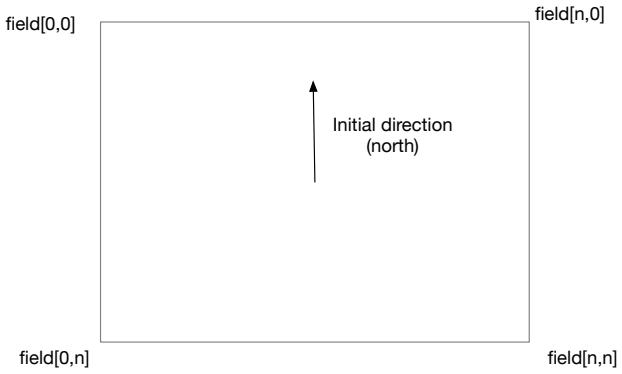
which produces something that actually looks like a brain:



2.2 What you should do

2.2.1 Task 1: Fill in the `Step` function

Write the `Step` function in the code template that computes one step of Turmite behavior according to the above description. Note the following coordinate system:



2.2.2 Task 2: Design a cool Turmite

Create a file named `coolest.mite` that is the coolest Turmite ruleset that you can come up with. How will this task be graded, you ask? According to the following rules: (1) if you ask me (or any course staff) how it will be graded you get a 0; (2) if you produce something non-trivially interesting, you get 100%; (3) if you produce something, but it is trivial or totally uninteresting, you get 50%.

2.3 Autograder

As usual, you should create a gzipped tar file containing the Go files of your program (do not include any folders or subdirectories). Your gzipped tar file should also contain a file called `coolest.mite` that specifies the coolest turmite that you constructed.

The autograder will run your turmite simulation program using commands of the form:

```
turmite -prog PROGRAM -s SIZE -step STEPS
```

The template code supports all these parameters. The autograder will run it against a few example turmite programs and also run it against your `coolest.mite` turmite.

2.4 Learning outcomes

After completing this assignment, you should have:

- Learned about turmites.
- Incidentally learn about Turing machines.
- Seen (again!) that very simple rules can to complex behavior.
- Had some fun creating cool patterns.