Intro to Generative Art
Problem Set 4

Harvard University
Taught by Sudhan Chitgopkar

This problem set is optional, but *highly encouraged*. You may work in groups and/or solicit outside help including but not limited to Generative AI tools. All outside sources/code used *must* be properly credited in the code documentation submitted.

1 Getting Antsy?

While we focused on Conway's Game of Life during lecture, there are no shortage of cellular automata with intricate and complex emergent behavior. Though not technically cellular automata, Langston's ant is a two-dimensional Turing machine that is played on a grid with binary states (much like Game of Life!). In its most basic form, the ant's rules are as follows on a grid of shaded/unshaded cells:

- At an unshaded square, turn 90° clockwise, flip the color of the square, move forward one unit
- At a shaded square, turn 90° counter-clockwise, flip the color of the square, move forward one unit

Modify our code to Conway's Game of Life to instead visualize Langston's Ant and its trail. For reference, the code we developed during lecture can be found here.

Too easy? Let's bump it up a notch. Instead of continuing to work with ants on a square lattice, we'll instead work with worms on a triangular one. Paterson's worms. Though there are $\approx 1,300$ different rules that these worms can take on, find a set of rules that you find interesting and use them to visualize Paterson's worms. Though there are lots of interesting rules, rule 1,0,4,2,0,1,5 is of particular interest, as it's currently unknown whether the Paterson's worm using this rule ever terminates.

2 Let There Be (F)light!

In lecture, we developed a quick toy model for Boids. Here, we'll work on refining that model to make it more interesting. For reference, our original Boid code can be found here. The first thing we'll address is our Boid's shape. We used a `circle` to avoid the calculations associated with having a Boid "point" to the direction it's going. Begin by modifying our existing Boid model so that each Boid is now an isoscles triangle which points towards the direction it's heading.

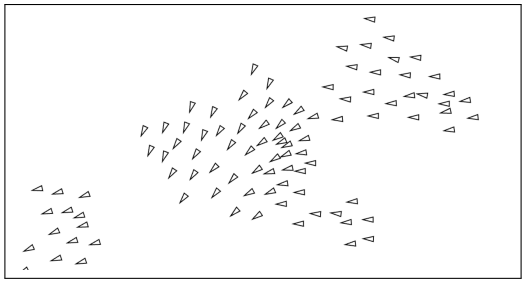

Figure 1: Boids rendered as isosceles triangles

Now that you've gotten a sense of how those calculations work, let's make our Boids more realistic.

For some Boid, b, rather than letting *any* Boid within `flockDist` affect the velocity of b, modify our existing Boid model such that only Boids in front of b affect b's velocity.

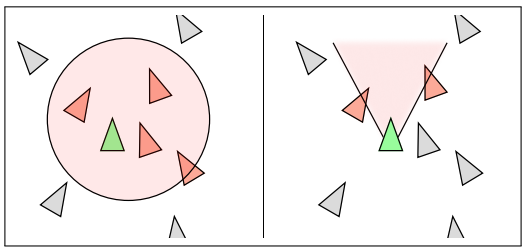

Figure 2: Boid with 360° vision (left) and Boid with 45° vision (right)