**Langton’s Ants with Python**

1. **Baseline: fixed boundary vs. wrapped around boundary**

This section compares two boundary conditions for the baseline Langton’s ants model. For the fixed boundary, the agent turns around by 180 degrees when it is blocked at the borders. The wrapped around boundary connects the North to the South border and East to the West border using the modulo function (e.g. %height or %width of the grid). It can be seen in Fig 1 that the “highway” trail gets stuck at the boundary when the boundary is fixed [left] but wraps around when the borders are connected [right].

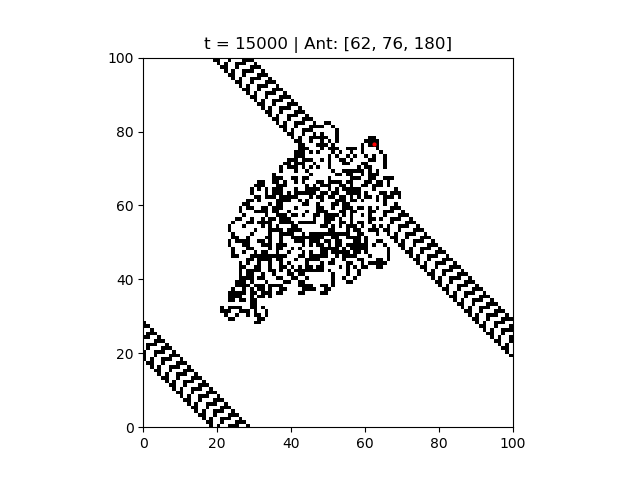
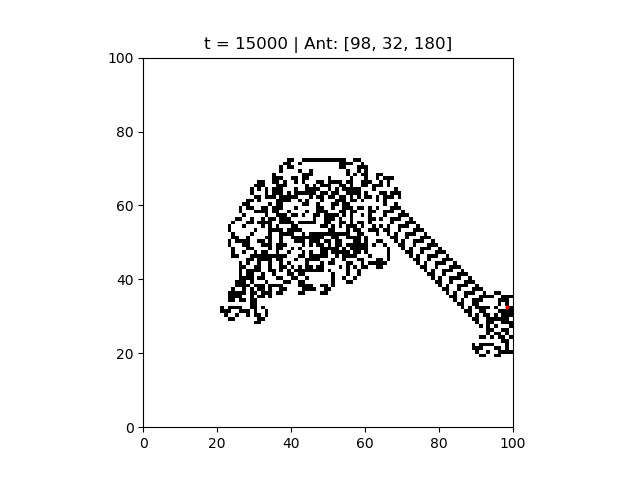
****

Fig. 1: Baseline run with *fixed* boundary [left] vs. *wrapped around* boundary [right] – single ant began at [50, 50], facing true North.

1. **Baseline with different initial orientation (wrapped boundary)**

Using the wrapped boundary condition, this section explores how the baseline run differs under different initial agent positions (i.e. orientation). As shown in Fig. 2, the baseline result depends on how the agent is oriented. The result for an East initial orientation [left] is rotated 90 degrees CW from that for a North initial orientation (Fig. 1 [right]) and 180 degrees from that for a West initial orientation [right].

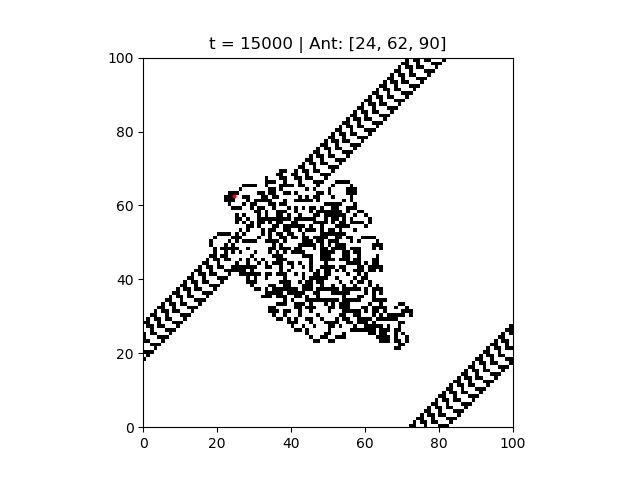
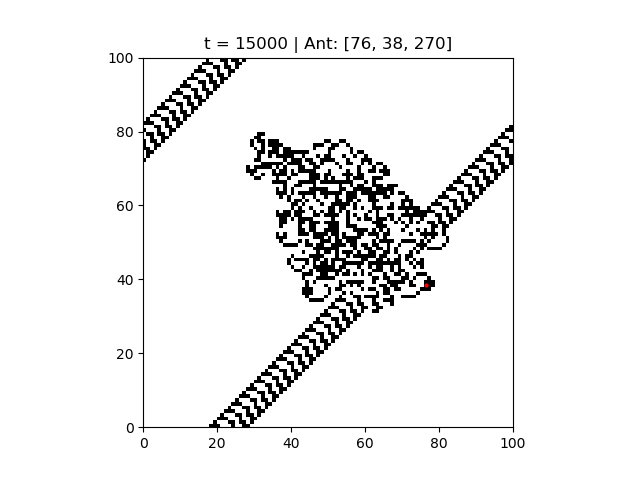


Fig. 2: Baseline run with wrapped around boundary – single ant began at [50, 50], facing true *East* [left] vs. facing true *West* [right].

1. **Multicolor paths: single ant (wrapped boundary)**

This section explores what happens when more than two possible colors are allowed for the patches to create multicolor paths. This is done by increasing the number of possible patch states from 2 to 4 := {0,1,2,3}. The pcolor cmap is updated from binary to YlGn (a yellow-green spectrum) along with vmin and vmax to reflect the new patch state set. The agent-patch interactions are modified as follows:

* Agent turns right on patches with states of odd numbers and left otherwise.
* For state < 3, state += 1.
* State 3 patches return to state 0 the next time an agent lands on them.

As shown by Fig. 3 [left] and Fig. 1 [right], the modified agent-patch interactions in the multicolor run did not result in substantial changes and the overall shape of the random walk and the “highway” trail remained. Supplemental to Fig. 2, Fig. 3 also shows that the results depend on the initial orientation of the agent.

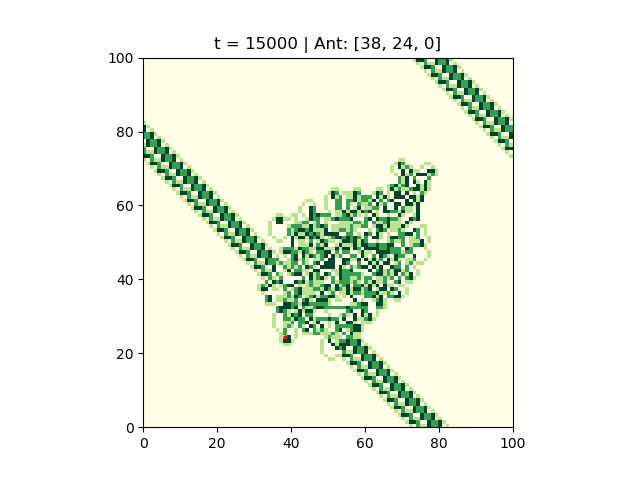
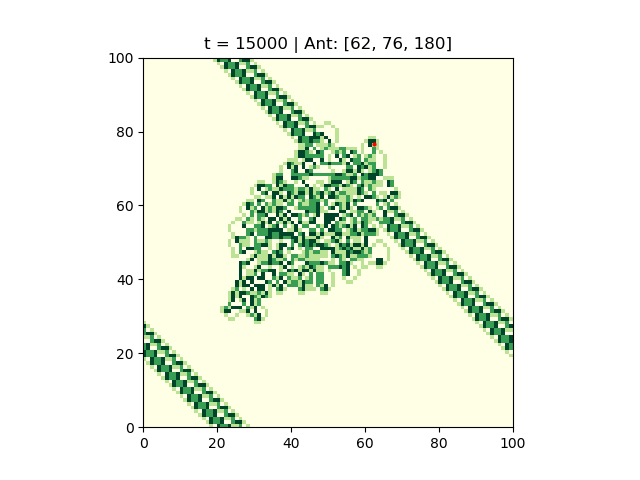


Fig. 3: Multicolor path run with wrapped around boundary – single ant began at [50, 50], facing true *North* [left] vs. facing true *South* [right]

1. **Multicolor paths: two ants (wrapped boundary)**

This section explores the combined effect of multicolor paths and multi-agents. A second ant with a different starting position (same orientation) than the first ant is added to the Multicolor path model. Fig. 4 [left] shows that although the two ants crossed patches due to proximity, they both produce “highway” trails eventually. However, when the second ant started at a different initial orientation, the inter-agent effect of the ants was large enough that it prevented the formation of orderly trails at 15,000 time steps. This illustrates the possibility of synergistic effects among model parameters, thus underscoring the importance of parameter sweeps in data analysis.

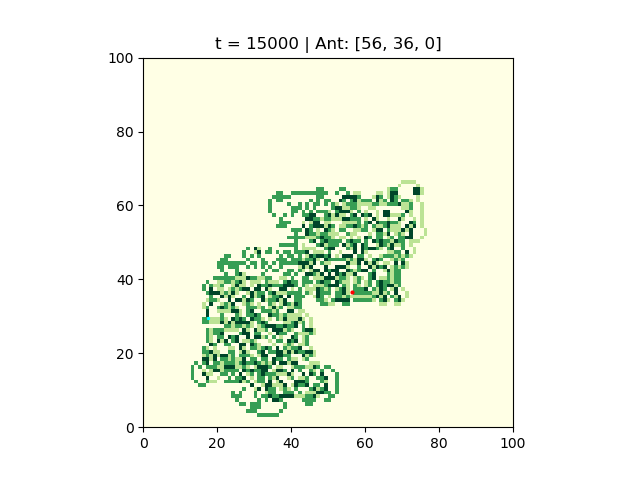
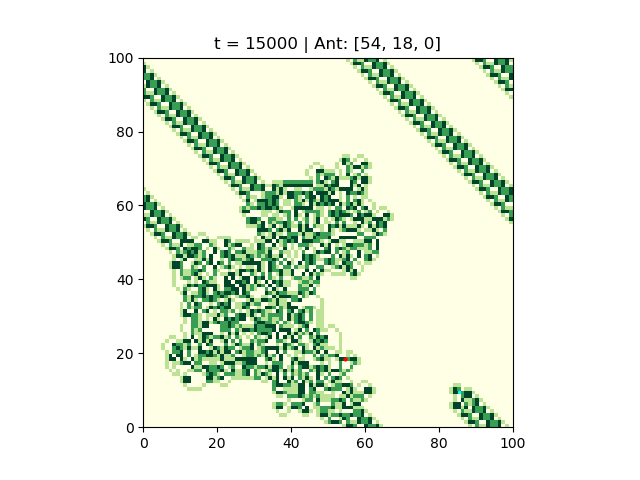


Fig. 4. Multicolor path run of two ants with wrapped around boundary – Ant 1 began at [50, 50], facing true North & Ant 2 began at [33,33], facing true *North* [left] vs. Ant 1 began at [50, 50], facing true North & Ant 2 began at [33,33], facing true *East* [right]

**Reflection**

This assignment is challenge but a fun way to learn about cellular automata while practicing my beginner python skills. The pycx file we studied in class provided a good starting point for me and I could google-search most of the functions I did not understand or would like to include in my models. I constructed my models in an agent-patch environment from the start, knowing that agent-patch interactions are required for simulating more than one ant. This proved somewhat challenging and the trickiest part was formulating the triple for-loops to update the data structures for the agents and patches correctly. Updating the orientation of the agents took a bit work but was overall straightforward once the proper math operations were incorporated.

I find that the quickest way to simulate the models is by running a while loop over a maximum time step (specified as 15,000 in my models) in the code. It will take only 1 Step Once in the GUI interface to get to that max time step, although the step may take a while to finish. This wait time depends on the number of operations in the code and almost doubled when I increased the number of ants in my model.