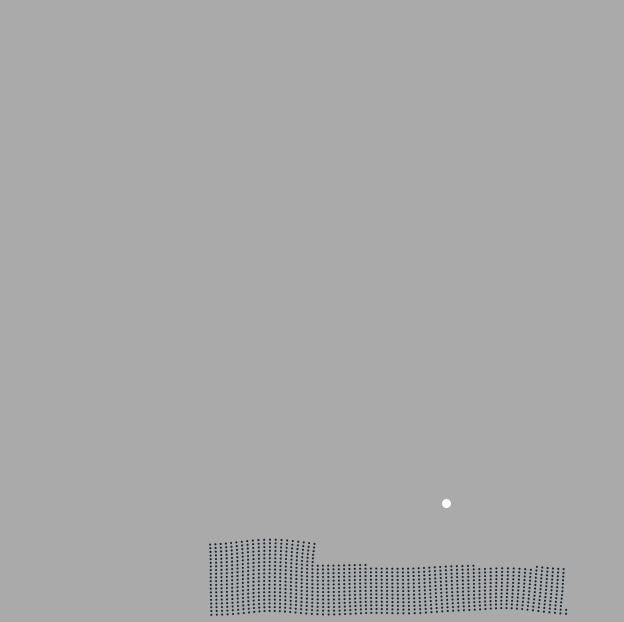
Artificial Life – Lab 3

Charlie Seifert

I modified diffmpm\_simple to generate a snake with a number of body segments, each with a certain height. The height of each of them is optimized according to a cost function that rewards smoothness, causing the snake to gradually develop segment heights that are similar in height. The other part of the cost function is original to the example code, rewarding creatures that use their initial velocity to reach towards the target. I have zeroed initial horizontal velocity so that the snake's vertical velocity as well as its segment heights are the only things that may propel it sideways towards the target. Additionally, because the target was a moderate distance above the snake, the segment heights not only tended towards being the same height, but also being as tall as possible, to bring the center of mass of the creature closer to the target; this was an inadvertent consequence of how I designed the cost function. An example snake midway through its evolution from random initial morphology to tall and smooth body segment heights is shown below.



More specifically, the mechanical property being changed is the height of each segment of the snake’s body. This means that the snake’s topology is fixed, and certain properties of segments within that topology are changed and optimized according to a cost function, using a gradient-based approach to do so.

The biggest challenges I faced were in organizing the data of the snake in a way that made difftaichi happy; it was very particular about needing the segment heights (stored in array seg\_heights) to be held in a taichi field. I learned this was necessary for it to effectively do the gradient calculation with respect to the segment heights.

The cost function is shown below. The first part adds to the loss variable the distance from the creature’s center of mass to the target. The second part, which I added, adds to the loss variable the difference in height between any two adjacent body segments.

|  |
| --- |
| @ti.kernel  def compute\_loss():      dist = (x\_avg[None] - ti.Vector(target))\*\*2      loss[None] = 0.5 \* (dist[0] + dist[1])      # reward smoothness      loss\_ht\_wt = 0.12      loss[None] += loss\_ht\_wt \* (seg\_heights[0] - seg\_heights[1])\*\*2      loss[None] += loss\_ht\_wt \* (seg\_heights[2] - seg\_heights[3])\*\*2      loss[None] += loss\_ht\_wt \* (seg\_heights[3] - seg\_heights[4])\*\*2      loss[None] += loss\_ht\_wt \* (seg\_heights[1] - seg\_heights[2])\*\*2 |

A problem I dealt with but did not completely solve was the allocation of particles within the simulation. As the simulation proceeds, you can see that the particles on the far right of the window are gradually removed from the body of the creature to be put elsewhere. I believe the fixed nature of the number of particles in the creature’s body is a limitation presented by difftaichi, making my approach of simply adding or removing particles from each body segment not efficient; I should remember this for my final project and perhaps use it to my advantage and let body mass be a fixed constraint.

The loss function of an example run is shown below. You can see that without huge cost assigned to each body segment height, it levels off relatively quickly.

