Aryk Anderson

Intelligent Systems Task 1

Part 1:

This part was absolutely the simplest, I just used a java Random to create n+1 (n=10) (x, y, z) points within the bounds of the world. Choosing random points works out fine, though sometimes it creates incredibly sharp angles. The con of this would just be that there is no intelligent behavior in the decisions for the points to travel to, but that was outside the scope of this assignment.

Part 2:

This part actually came at a different point in my algorithm, but to do this, I just grabbed two points, and created a unit vector along those two points and then created points along that line by multiplying some fixed step size along that unit vector until the end was reached. This approach worked out great, I see no downsides to taking this approach.

Part 3:

For this part, I used the unit vector associated with the line connecting the bird's current position and previous position to determine the heading. From the unit vector I was able to calculate yaw and pitch directly from the vector components without any additional math outside of a call to atan2(). I had one downside that took some time to figure out with this approach and that was that it didn't play very well with the way that I was adding noise to the line, and I had also forgotten about the zero vector case, but those problems have been resolved. Another downside to this approach is that sometimes the angles just end up weird and the bird seems to thrash awkwardly.

Part 4:

I added noise to the segmentation points by taking 3 random Gaussians and then adding them component wise to those individual points. I also decided that I wasn't going to jitter the end points of the original lines as this made the behaviour look much more erratic. The downside of the approach I took is that sometimes the points would end up very close to each other by chance and cause an amount of abnormal looking jitter. This could probably be improved by introducing some other form of non-linearity for use in jitter so that it would be a bit more regular.

Part 5:

In order to smooth out the path of the bird, I decided to use Bezier curves, because I found that to be much easier then trying to come up with some criterion of how smooth a path is. I know that you said it was overkill in class, but I actually didn't know of a good way to make this "look about right" without using some form of interpolation. I created the curve points in two passes - the first pass to generate curve points happens after the original lines are created, then segments are added and jittered, then I do another pass to generate more curve points between the jittered positions to smooth out the paths a bit further before calculating yaw and pitch. This technique worked really well to create smooth paths that look somewhat realistic. I'm not sure that the way I approached this really handles turns of <10 degrees, but it seems to do a good job on most other turns and dealing with the jitter. The biggest downside of the way I approached this is that it generates a huge amount of additional points for the bird to follow along. I'm sure this method could be improved, but I'd really need a good reason to try and change the way I'm doing things. The numbers for jitter, number of points to generate, segment length, etc. could all be tweaked further to make it look better. This method also makes it appear as though I'm using far fewer original line segments then I am.