Peter Schaldenbrand

Artificial Intelligence, Dr. Hwa

Project 3

November 10, 2016

I began trying to model the flappy bird game in many different ways. My first attempt had the state as a vector of the y coordinate of the bird, the x coordinate of the pipe, and the height of the opening of the pipe gap. This created a humungous state space. Since the space was so large, it would be too difficult for the bird to learn. So, I tried to quantize the values of this vector. I divided everything by 10 to reduce this state space. I figured that 10 was a small enough number that it wouldn’t affect the decision making and a large enough number to reduce the state space and allow the bird to actually enter a state more than once.

My next attempt at making the state was even simpler than the first. I wanted to make the state space even smaller so the bird could learn faster. I figured that the only things that mattered position wise were how far the bird is from the pipe and the difference in y coordinates between the bird and the opening of the pipe. So, these two differences now made up my state space. I added the quantization factor of 8 in to consolidate the states even more. Now the bird learned a lot faster than before. The biggest issue with this approach is that the bird doesn’t know how far it is from the top or bottom of the screen. So, you’ll see that the bird usually loses by hitting the top or bottom rather than hitting the pipe. I had tried to fix this by adding the y coordinate of the bird to the state space vector, but this made the learning take too much time, so I scrapped that idea.

In the end, it took about 600 games to get the bird to learn to go an average of 5 pipes per game. I think the learning could have been better if I tried a few different parameters on the reward function. Another thing that would have affected the learning rate would be the beginning of the algorithm where the bird takes many random paths. These random choices help the bird take a route that it might not usually take, this then helps fill out the Q values. Finding the rate of random choices is difficult, but picking the optimal frequency could help the bird learn quickly.

To run the program just do the normal “python flappybird.py” using python 2.7. This will automatically load the Q values and other data necessary to run the trained bird. I didn’t mess with any of the other code such as changing frame rate, or the height of the pipe gap. Everything about the game should be as it originally was.