Charlie Laymon  
[crl50@pitt.edu](mailto:crl50@pitt.edu)  
CS1571  
Homework 5

Usage:

>python flappybird.py

Note that the “Alternate versions” file contains two alternate versions of flappybird.py. “constantbird.py” contains a version where the pipes always spawn in the same position. “repeatbird.py” contains a version that uses a repeating set of pipes that spawn in the same position. To use these, move them to the main directory along with “constantqFile” for constantbird.py and “repeatqfile” for repeatbird.py. Rename whichever q file you are using to “qfile”

State representation choices: The state I chose involved the bird’s horizontal and vertical distance from the back corner of the lower pipe, if the bird was rising or falling, and the bird’s distance from the ground. I initially started with the bird’s and pipe’s x & y coordinates, however, this proved to cause identical states that would have different solutions. This seems to be the state representation that provides the best compromise between state representations that provide too little information and too much information (which causes the state space to become huge).

My exploration methods are intentionally simplistic. Overcomplicating things by attempting to punish jumping in a position that would hit the ceiling or doing nothing resulted in the bird getting stuck in bad paths.

Training the bird took about 20 minutes if the pipes spawn in the same place, 40-50 minutes if given a repeating set of 10 pipes. As far as a totally random situation, I haven’t had time to fully train a bird, but it does seem to continuously improve. I think improving my R-value calculation could cause the training to speed up.

The trained bird will be loaded automatically when the program is run. To “un-train” the bird, simply delete the “qFile” file present in the program’s directory.

Resources used: Various python tutorials, this q-learning example:

http://mnemstudio.org/path-finding-q-learning-tutorial.htm