Base Scripts

--------------------

These are the same player movement scripts used from Assignment 1. The only modifications were made on PlayerScript.cs. We add an Ai Controlled field that would recieve its movements from the PathfindingNetwork. Otherwise, controls such as movement and rotation are not any different than from Assignment 1.

Sensor Scripts

---------------------

These are the basic sensors used from Assignment 1. The only modifications made were on how they can send information to either a trainer or a neural network. Additionally, we normalized the data. These scripts include the WallSensors.cs and AdjacentAgentSensor.cs files.

Training

--------------------

TrainingScript.cs - This class handles training output of the sensor data to a text file. Each of the sensors finds this component in the game and calls setters to update the training data. Then upon normal frame updating, the script takes its current values and saves them to the current output.

Neural Network

---------------------

BackPropNeuralNet.cs - This is a basic neural network that learns via back propogation. A good deal of the code was retrieved from the visualstudiomagazine article provided to the team by the professor. It was modified to be slightly more OO.

Helpers.cs - This was another file that came with the BackPropNeuralNet.cs project from the same article. We only update console logging for Unity mainly.

TestScripts.cs - These were just proofs of concept and include the XOR problem.

PathfindingNetwork.cs - This file hooks the Player's AI-controlled movements up with the neural network. GetNextMovement() method is called from the PlayerScript with its parameters as sensors, and is returned movement and rotation values. Additionally, this class handles the creation of the specific neural network needed for pathfinding, logs to the console, and handles the loading of training data. This is also where we set final or default weight values for the NN.

How to Train

---------------------

Select the lawn mower in the Hierarchy. Turn on the TrainingScript. Set file options and TrainingTrials. Trials dictate how many times you can run over the bunny. After the number of times is reached, logging to the output file will stop. The Record Each parameter dictates the capture rate (in frames). Each scene ran will successively output to the root directory as sensorDataX.txt, where X is the attempt #. So the first training session will be called sensorData0.txt, then sensorData1.txt, and so on.

After you've trained, you can load the training data.

Deselect the TrainingScript on the lawn Mower. Select the PathFinding Network script. Set NN options such as Learn Rate and Momentum. Select Load Training File to tell the network to load all files in the sensorDataX.txt sequence from the root file (the default values are randomized). Asynchronously, the data will load (even if you stop the scene! so be sure to wait until it's all done). You can see progress in the console if you have Verbose Logging turned on. When it's complete, the final weights will be displayed.

At this point you can copy the weights into weights variable in PathFindingNetwork.cs line 34 (Start method), uncomment line 45 to set the weights to this double array, and restart the scene without the Load Training File option checked on the PathFindingNetwork script (but still keep the script checked on). On the PlayerScript, select the AI Controlled boolean option, and restart the scene. The AI will now control the player's movement.