Intro to Intelligent Systems
Project 1
Due: 10/10/18
Oscar Chacon (orc2815@rit.edu)
Leul Berhane-Meskel (lgb9445@rit.edu)

Introduction:

For this project, we had to modify an agent to play the Atari 2600 version of space invaders. The agent that was provided acted as a random agent, which meant that the actions it decided upon were pseudo randomized, following no such strategy to try and get the highest score. Our modified agent tries to take a few steps in its attempt on getting the highest score possible. The agent is able to utilize all the actions it has available but only after running several checks to make the optimal action.

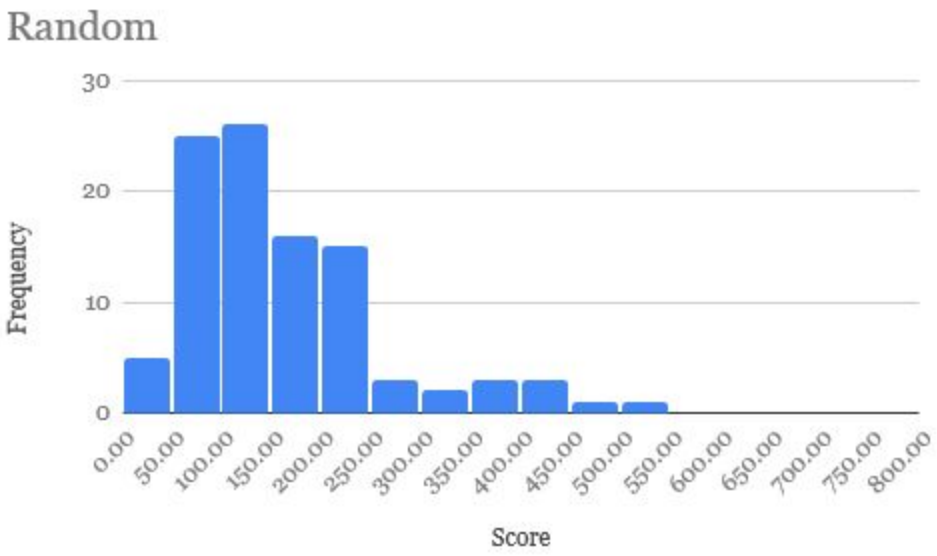
Methods:

The agent we designed relies on all the actions it has available. The idea behind the design is to emulate an aggressive player in a First Person Shooter; dodge incoming threats and never stop shooting, unless behind a barrier. We wanted the agent to be smart enough to not fire while under a barrier because if it hits the barrier with its own bullet, the barrier still breaks apart which is not beneficial to our agent. Each time the act method is called the ship is found by searching the row that the ship spawns on. Once that x value is found we know the ship's hitbox is roughly 3 pixels to each side, so we give a few pixels of buffer and save the ship's left and right edges for our search. Identifying the pixels relies on the value of ob[y][x][0] because that value is the red color value for the given pixel. Since each possible color has a different red value, we created a dictionary where the red value is the key, and the value is the string describing what it is. This is to improve code readability.

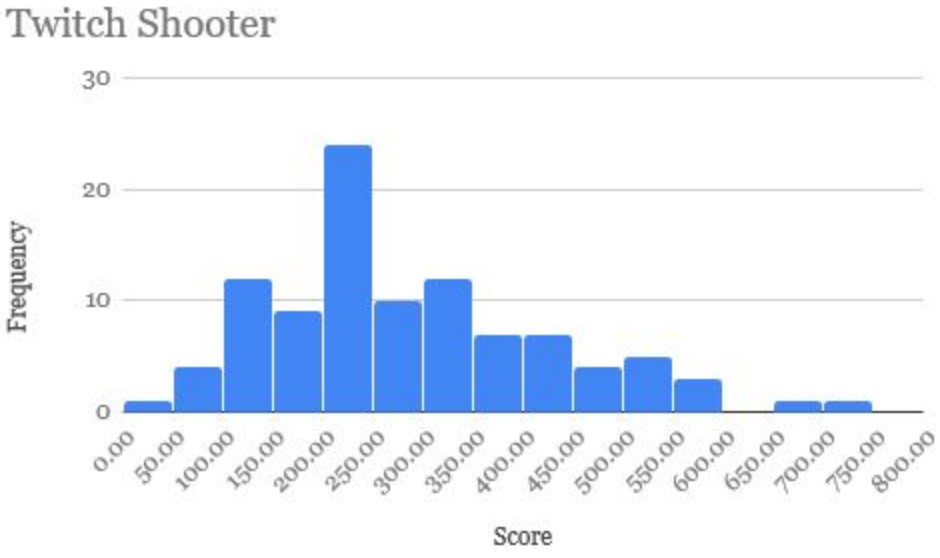
If a barrier is found, then a variable is set to inform the ship to not shoot as it moves this turn, as to prevent damaging the barrier for no reason. The next step is to search the column above the ship for the first bullet or alien it finds. If a bullet is found, then the ship will try to move away from the bullet, shooting if it is not under a barrier. If the bullet is more on one side or another of the ship, it'll move and shoot in the opposite direction. If the bullet is in the middle of the ship, the ship will move and shoot to the right as by and large because the ship spawns to the left, we want it to move right and explore more of the space. If an alien is found, then the ship will move towards the alien while shooting.. If none are found, the search column is expanded 3 units to each side and the search for the aliens is renewed. If one is found, the ship moves over while shooting to the nearest alien. If one side of the search leaves the playable area, the ship moves in the opposite direction.

Results:

When we ran the original random agent 100 times, and compare that to our "twitch shooter" agent who was also run 100 times, we get the following results. Our twitch shooter on average can score 126.15 points more than the random agent.



Average: 157.8
Top Score: 515
Lowest Score: 15



Average: 283.95
Top Score: 725
Lowest Score: 35

Conclusion:

From our results, we can see that our agent outperformed the original random agent. We had over 100 points higher for our average, and both the high score and low score were higher than the random agent. This shows that instead of just randomly picking an action each frame, analyzing the situation at that time and choosing a movement based off of the current environment will lead to better results. Given enough trial and error, we may have been able to identify more specific situations to further cover our cases when looking for the optimal action, however since the search space for any one game of space invaders is insurmountably massive, there was not an efficient way of analyzing enough games frame by frame to deduce any other factors that might have helped in strengthening the agent's decision making. This is also compounded by the increased time the game takes to compute a step as it has to search further and further away from the ship as it clears more aliens away. Had we been able to maintain state between function calls of act(), some more options could have opened up to allow for more efficient searching, such as removing areas that all aliens had been eliminated from so that future calls to act() could avoid searching areas that no targets could exist.