Address the following questions in your proposal:

* What are you trying to do?
  + We want to do a problem similar to the Malmo Collaborative AI challenge([1](https://www.microsoft.com/en-us/research/academic-program/collaborative-ai-challenge/)). We would have 3 agents in this scenario: A player, an adversary, and a goal. The objective for the player is to catch the goal, which is moving randomly, before the adversary catches the player.
* Motivate the problem: why important/interesting?
  + This problem expands upon what we have done for our projects by introducing a moving goal state. So far in class, we have known exactly where all the goal states are in a problem. However, in the real world because not every scenario has a stationary goal. AI need to be able to find the fastest and safest route to a goal in a not yet known destination.
* What are the challenges? How do you plan to solve them?
* Action plan
  + Write an algorithm that implements the minimax with alpha-beta pruning method of adversarial search, to compare its effectiveness to other search algorithms that we have learned in class. Our plan is to use Malmo to create our own in-depth maps on which to run our minimax algorithm, as well as run other search algorithms that we have already created to log its effectiveness and efficiency.

Brandon’s Ideas/Changes

* Make this a four agent problem with two adversaries. It is trivial to only avoid one adversary as you just move away from them. It is almost an identical problem to HW2 with only one adversary
* If we wanted to add any additional complexity, we could make the goal move in some probabilistic manner so that the agent would have to take this into consideration in the algorithm and reward function

AI Project Proposal

Scott Miller, Brandon Rinderle, Julian Ronda, Nate Olsen, Zach Campo

fsm5yd, br4av, jr2nh, nso4wg, zpc8ff

We are looking to do a project similar to the Malmo Collaborative AI challenge([1](https://www.microsoft.com/en-us/research/academic-program/collaborative-ai-challenge/)). We would have four agents in this scenario: A player, two adversaries, and a goal. The objective for the player is to catch the goal, which is moving randomly, before either of the adversaries catch the player. This project would be implemented in Malmo in a similar fashion to the multi-agent homework we completed.

This project idea was chosen for multiple reasons. First of all, it expands upon what we have done for our projects by introducing a moving goal state. So far in class, we have known exactly where all the goal states are in a problem. However, because not every scenario has a stationary goal in the real world, AI needs to be able to find the fastest and safest route to a goal in a not yet known destination. Additionally, the problem gives us the opportunity to try implementing some of the more complex AI algorithms which we haven’t tried in our homeworks. Furthermore, it is an interesting problem because it is similar to Pac-Man, but has the additional feature of a moving goal state.

To accomplish the goal of this project, we will write an algorithm that implements the minimax with alpha-beta pruning method of adversarial search, to compare its effectiveness to other search algorithms that we have learned in class. Our plan is to use Malmo to create our own in-depth maps on which to run our minimax algorithm, as well as run other search algorithms that we have already created to log its effectiveness and efficiency.

Our main challenges will include implementing the minimax algorithm to avoid the two adversaries, setting up the environment correctly, designing maps give us the best understanding of algorithm efficiency, and finding a good way to measure the efficiency of the algorithms since movements may change from trial to trial. We can create scoring system that measures how many food blocks our player has eaten and balance it out with the time it takes us to get there. For designing efficient and effective test maps, a number of different layouts can be used such as those with walls, dead-ends, holes in the ground, etc.. to make sure that our player can adapt and overcome a number of different world environments. Having multiple world types would allow us to make sure that our algorithm could perform well in many different maze types. Using an environment similar to project 2, but adapted to our needs, could also help in ensuring that our testing environment will be set up correctly.