Monks Knights Barbarians:

**Path following:**

Originally, the monks used path following. They moved along a set path going from on monastery to the next in a never ending loop. However, we took it out of the game because it was not helping the game in any way and replaced it with path finding.

**Path finding:**

The monks, knights, and the barbarians all use path finding. We use a navigation mesh for the movement, with the positions of the monasteries baked into the mesh.

**Decision Tree:**

The knights use a decision tree to determine their behavior.

**State Machine:**

The barbarians use a state machine to determine their behavior

5. Am all alone

1. See unguarded Monk

3. See more friends than knights

5. Am all alone

2. See more knights than friends

5. Am all alone

2. See more knights than friends

1. See unguarded Monk

**Behavior Tree:**

We did not implement behavior trees into the game. We created a prolog behavior tree for the monks.

**Genetic Algorithm:**

All three use genetic algorithms to change their attributes every round. These attributes were all changed by a single fitness value that was evolved through a single threshold.

The traits evolved by each unit is as followed:

Monks – Sight range and move speed

Knights – Sight range, speed, and attack delay

Barbarians – Hit chance, sight range, speed, and attack delay

We used a population of 30 units, 10 of each class. Each chromosome had 10 bits and the phenotype range was 0 to 200. We calculated fitness based on the time that the unit survived and, for the knights and barbarians, how many units they killed. For the monks, since they could not kill, the time survived counted more heavily towards their fitness value. As a result, fitness values ended up in the range of about 10 to 180.

Only one threshold was evolved during all testing. During this testing, it took about 12 rounds for preferable behavior to evolve, which was noted by the increased variance in the win condition that was occurring each round. The values of the traits that we ended up with varied about 30% either higher or lower than our initial hard coded values.

I believe this feature is viable in the game as it adds a nice variance to each team’s performance. Given our old scenario, it was extremely common for the Knights to win every round that was played, but when varied attributes were introduced, their win rate began to drop if they received lower fitness values. As a result, it can be said that genetic algorithms enhanced our game by making the teams more balanced. Balance is also an important factor in making a game more fun, as it reduces predictability and provides more unique occurrences as it plays out. A scenario involving this would likely revolve the changes between each situation a character is in, where sometimes your fitness is allowing you full control of the situation, and other times it makes you nearly helpless.

**Bayes Classifier:**

Need to Complete