Decision Making
Pinak Jalan
U0965201

Disclaimer/Abstract: This report is a reflection of decision making algorithms implemented using Mit Doshi's repo(I have re implemented a bunch of his code, but he was unable to provide me a clean assignment 2, which is why I have analyzed his code for assignment 3 and modified/reimplemented a lot of it for the purposes of this project, because I was having problems with Assignment 2). This assignment encircles decision trees and behavior trees.

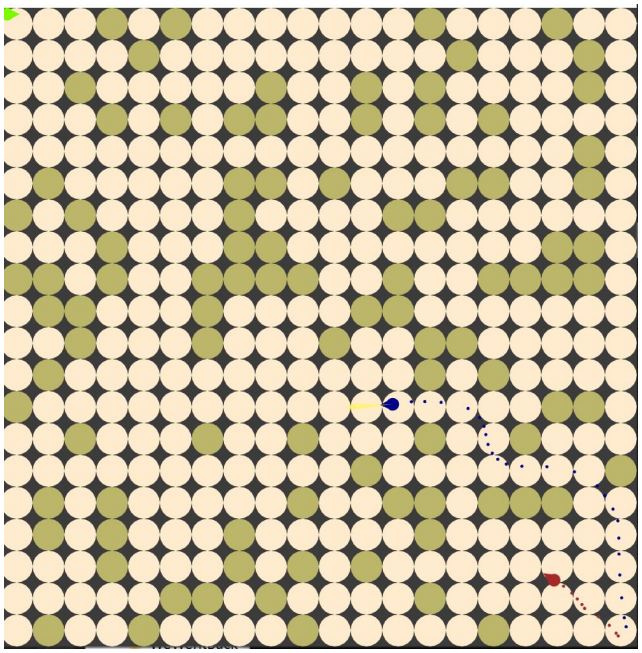
Algorithms:

Decision Trees: Decision trees, usually two parts - Decision and Actions, are usually created using graphs. Actions are usually tasks that the decision tree has to perform, while decisions are the part of the decision tree that are used to execute said actions. If we think of a hierarchy of a graph , decision nodes are parents and action nodes are leaves.
Decision trees consist of managers that keep track of various states in the world and execute actions accordingly. The usual states consist of

- **priority,**
- **time taken(for action to execute and finish),**
- **parallel execution,**
- **aborts and interruptions etc.**

Custom behaviour tree implementations: The implementation of the decision tree used here if a wander+walk+random speed enemy bot with custom priority. The boid will keep walking and wander upto an "expiry time", after which it charges up its velocity to anytime up to a 1.5 multiplier, and then goes normal speed after "expiry time" is over. Since all of this conforms to a decision tree, there will be decision and action nodes. The decision nodes, as we discussed, will tell the action nodes what to execute. The action nodes here will be: wandering around, walking, and randomly sprinting. If we are talking about design, for testing purposes, I used a variety of different scenarios. Walking around intuitively is the default, but if that is changed to random sprinting, in conjunction with the monster created using the behaviour tree, it would die a lot quicker, and the player would zoom around the map instead of taking its time. The random velocity multiplier also made it appear a little "jittery".

This was a very simple implementation of a decision tree implemented for this assignment. Code compile and design problems made it harder to test, but after researching more, and looking into Mits code from what he did, he had a "digger AI", which used energy to wander, and if it had enough energy and found an obstacle, it would destroy it. Similar to the action and decision nodes, having enough energy, obstacle finding and wandering would be three actions with their own priority.



<---- The following shows both the boids moving at random velocity multipliers while wandering.

Decision trees typically also need world information to work correctly and efficiently. We need a way to store this information and pass it off to the boids so that everytime the program is restarted and the world is renewed, the boid makes its decisions unbiased and in the same way. The simplistic functionality i have written for behavior trees for these particular boids were not very hard to implement, because essentially the functionality i am providing these boids encircle moving, wandering, and random “sprinting”.

Behavior trees: Another type of decision making algorithm, every node in this graph is always active. Each node here is its own entity, which means that there is no need for an external manager because each node already has them in-built. Data in a behavior tree is encapsulated into a blackboard, which have states, conditions and more data about the nodes in the tree. Tasks in behavior trees include sequencers, selectors and repeat until fails. These usually execute left to right. Sequencers usually execute true only if their children return true.