COMP222 Robocode tank Report

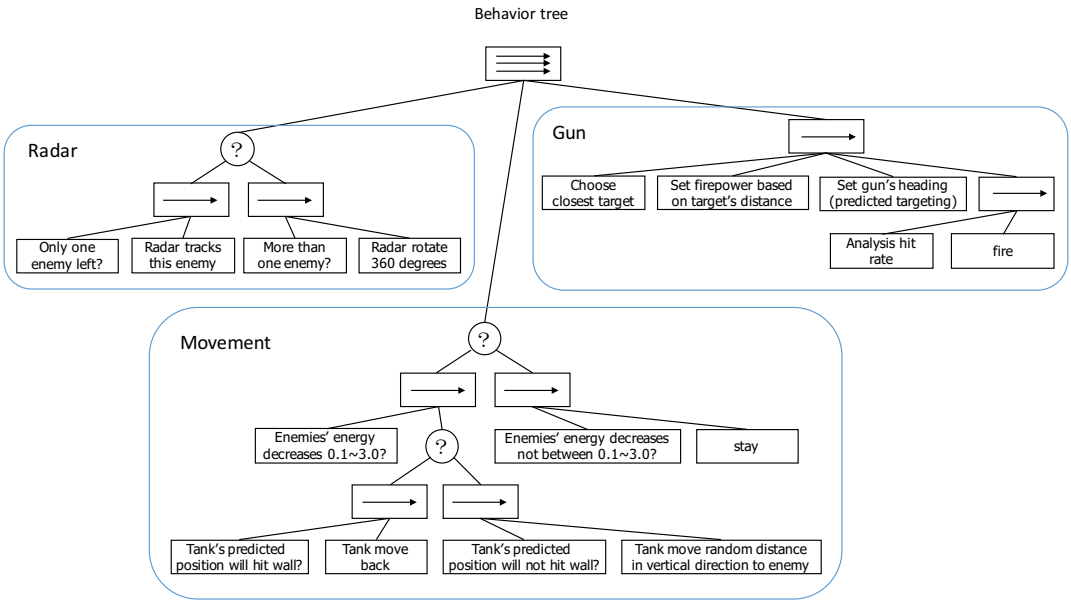
Behavior control model

Behavior trees is a mathematical model and it contains hierarchical nodes to control the flow of decision making of an AI entity. The behavior tree can extend many branches and nodes to control the action of robot tank. Each behavior tree is made up of different types of nodes and return one of three statuses- success, failure and running.

Composite, Decorator, Condition, Action are four behavior tree nodes which provides different functions to define the AI behavior. The composite node can have one or more children. It will process these children in sequence order(Sequences), select the children that first successes(Selector) or process these children together(Parallel). The decorator node contains inverter, ignorer, limit(loop) and until Fail which can modify the behavior of a task. Composite node and Decorator node almost control the decision making. The other two nodes, condition node and action node, are leaf nodes which is the commands to tasks. When the system implements the behavior tree, the system will traverse down from the root of the tree and test each node until it reaches an active node, then constantly traverse the tree until the game complete.

Design description

According to the research about robocode, I decided to use class advancedRobot to build my robot tank. AdvancedRobot has non-blocking calls which can control the radar, gun and vehicle individual. Therefore, I choose behavior tree as the behavior control model to process multitasking. The parallel node can express these three parts (radar, gun, vehicle) working together.



Based on the total score mechanism, I found that longer live in the battlefield and more

damages to others will get high score. Therefore, there are three small parts of my design. For the first part I try to let my tank get less damages so my tank can avoid bullet as far as possible. If enemy's energy decrease from 0.1 to 3.0, it has high probability that enemy has fired a bullet so my tank will move in vertical direction to avoid this bullet. For the second part, my tank should decrease energy waste, such as do not hit wall, compute hit rate to decrease useless fire, set the firepower suitable. In order to hit wall, I set a rectangle small than battlefield. Then, before every movement, my tank should compute the future position. If this future position is in the outside of rectangle, the tank should move back. For the compute of hit rate, my tank will record the times ("bulletHitTime") when it fires to a specific enemy. If the bullet hits the enemy, the value of "bulletGetTime" will be updated. These two value can help my tank compute the hit rate roughly. Finally, the firepower will be set based on the distance of target. The larger distance, the smaller firepower. The third part is try to get more damages, I decided to use circular targeting to predict enemy's position. Using iteration method to adjust and check enemy position can obviously improve the hit rate. Some tank moves in linear line can be considered as circular movement in a larger circle, so this targeting method is useful for hitting linear movement robot and circular movement robot.

Implementation description

The implementation of my robot tank can be divided three parts, radar, movement and gun. The connection between these parts is a enemyList (Vector). The class Enemy is used to store all the information of each enemy, such as enemy's heading, bearing, currentTime(when radar scan the enemy), bulletHitTime and bulletGetTime (for computing hit rate). There is some helper method for rectify the angle of heading and bearing in order to reduce the rotate angle. The method "getAngle" is used to compute the degrees based on two coordinates. The method "isinVector" is check the enemy is in the enemyVector. If yes, enemy's information will be updated and return this enemy, else this method will return null.

For the radar, from the start of battle, radar will rotate infinite in 360 degrees. While the radar finds the enemy (onScannedRobot event), the information of this enemy will updated. If this enemy is not in the enemyList, it will be added. While one enemy dead (onRobotDeath), this enemy will be removed from the enemyList. If there is only one enemy in the battle, radar will turn around in small angle to track this enemy.

For the movement, because the radar can always update the information of enemy. If enemy's energy reduces from 0.1 to 3.0, my tank will turn to vertical direction between enemy and my tank, then turn right or left 30 degrees to close to the enemy. Before the movement, my tank will predict the future position. If the future position is outside of the rectangle which is smaller than battlefield, the tank will move back. If the tank hit wall, robot tank will trigger onHitWall event and move to the center point of battlefield.

The class Gun is used to control the gun of my tank. There are three steps before fire a bullet. Firstly, the method "closestTarget" will search the closest enemy from enemyList and set the

closest enemy as target. Then, according to the distance of enemy, the method "setFirepower" will set the value of firepower. Finally, the method "setGun" is use the circular targeting to predict enemy's position. Assumed all the enemies' moving path as a circle, linear movement's circle is very large. According to the radius of circle and the heading of enemy, we can compute the future position of enemy and set the gun turn to that direction. Before fire a bullet, the tank should consider the hit rate of this enemy. If the hit rate is low, the tank will choose longer interval to fire a bullet.