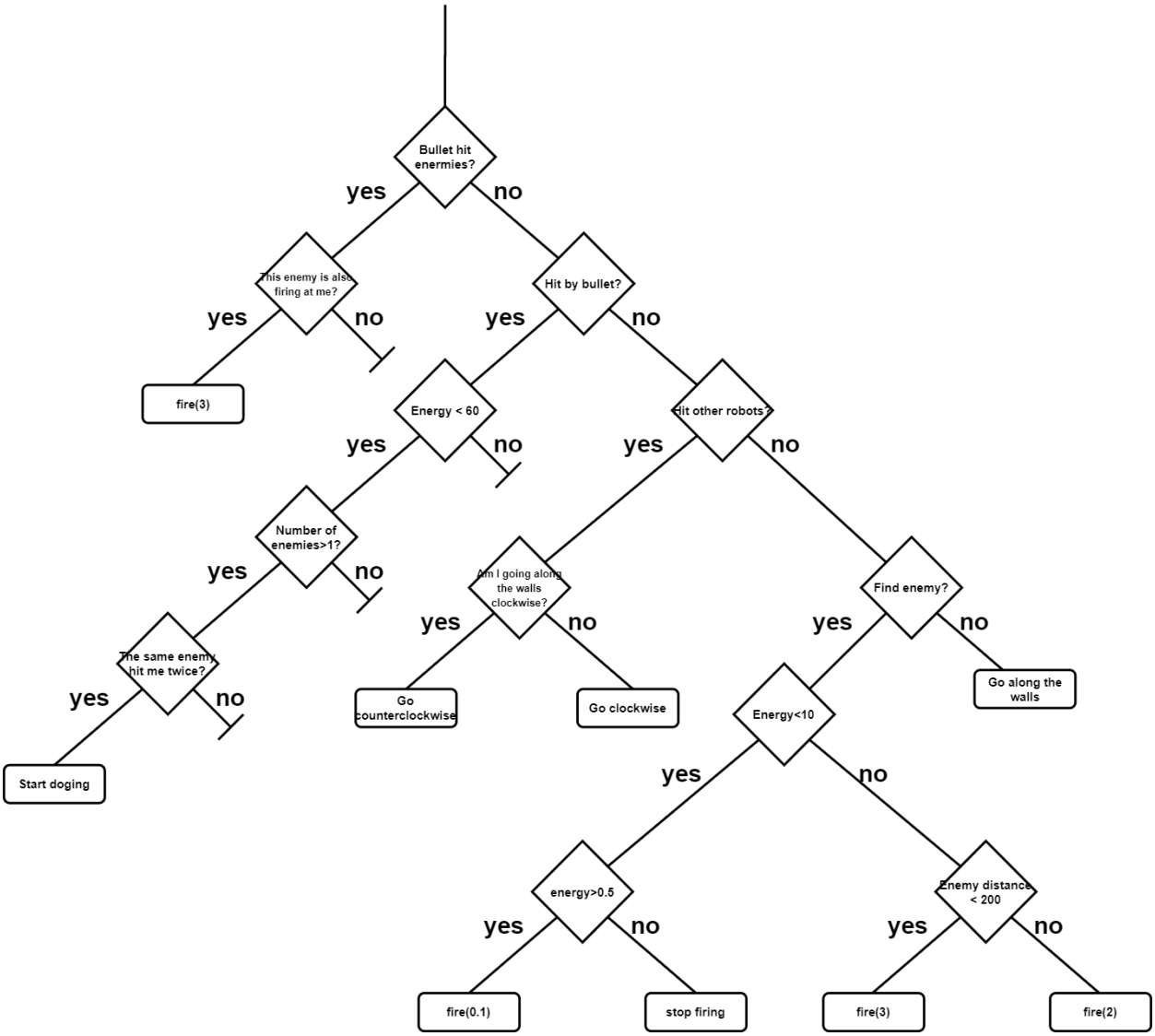
COMP222 - 2020 - Second CA Assignment
Individual coursework
Robocode tank Documentation

1. **Behaviour control model**
   Decision tree is chosen as my Robocode robot's behaviour control model. It is a flowchart structure which has tests from root node to leaves. If all the tests along the way from root to leaves hold, it will trigger the robot to do the action on the leaf node.

2. **Robocode robot design description**
   2.1 graphical representation of the tree



2.2 Explanation for the tests and actions

2.2.1 Priority of tests

According to Robocode API [1], all battle events have a set of default priority value as figure shows. Larger value means higher priority. Tests in decision tree also obeys the priority for events.

WinEvent:	100 (reserved)
SkippedTurnEvent:	100 (reserved)
StatusEvent:	99
CustomEvent:	80
MessageEvent:	75
RobotDeathEvent:	70
BulletMissedEvent:	60
BulletHitBulletEvent:	55
BulletHitEvent:	50
HitByBulletEvent:	40
HitWallEvent:	30
HitRobotEvent:	20
ScannedRobotEvent:	10
PaintEvent:	5
DeathEvent:	-1 (reserved)

Figure: priority of different events in Robocode

2.2.2 OnBulletHit

- BulletHitEvent: this event occurs when one of my robot's bullets has hit another robot.
- Decision steps:
  1. If this event occurs, then next test will be 'is the robot which my bullet hits also firing at me'.
  2. If it is the case that two robots are firing at each other, then my robot will set bullet power to the maximum which is 3.
- Comment and Explanation:
  Assume two robots *A* and *B* are sitting and firing at each other and their initial energy is the same, the one with higher bullet damage will finally win the game because it will deal more damage and get more power back from its opponent. In this case, if my robot raises the bullet power to the maximum, then it is undefeated.

2.2.3 OnHitByBullet

- HitByBulletEvent: this event occurs when my robot has been hit by a bullet.
- Decision steps:
  1. If this event occurs, then check the robot's current energy.
  2. If the current energy is less than 60, then check how many enemies left in this battle round.
  3. If there are more than one enemy, then check does the same enemy hit my robot twice.
  4. If two bullets come from the same enemy, then my robot should start dodging the bullet.

- Comment and Explanation:
  1. If the energy is greater than 60, my robot will be considered as healthy and able to keep firing.
  2. If there is only one enemy left, dodging might be unnecessary because the goal is to win the battle. Attacking is more important than defending.
  3. If my robot catches two sequential bullets from the same enemy, it will be regarded that this enemy is shooting at my robot intentionally. Thus, it is necessary to dodging bullets currently.

2.2.4 onHitRobot

- HitRobotEvent: this event occurs when my robot collides with another robot.
- Decision steps:
  1. If this event occurs, then check the robot's current moving status.
  2. If it is moving clockwise along the walls, then it should move anticlockwise.
  3. If it is moving anticlockwise along the walls, then it should move clockwise.
- Comment and Explanation:
  My robot has only 2 basic moving patterns: go clockwise or anticlockwise along the walls. Hitting other robots means there some robot is in its way. If it keeps moving forward, it may get stuck at that position.

2.2.5 OnScannedRobot

- ScannedRobotEvent: this event occurs when my robot scans another robot.
- Decision steps:
  1. If this event occurs, then check the robot's current energy.
  2. If current energy is less than 10 and greater than 0.5, the set the bullet power to 0.1.
  3. If current energy is less than 0.5, stop firing.
  4. If current energy is greater than 10, then check the distance to the scanned robot.
  5. If the distance is less than 200, set the bullet power to 2, else set to 3.
  6. If my robot does not discover any other robots, keep moving along the walls.
- Comment and Explanation:
  ScannedRobotEvent means the robot find an enemy. Since the bullet power will be taken from the robot energy and its speed varies based on its power, it should be calculated before firing, considering energy left and the distance to the enemy.

3. Robocode robot implementation description

- Overview

  After doing battle experiments within 11 standard robots in sample package, I found that among all the robots, 'Walls' is the robot with the highest winning rate. Since it is always moving along the walls during the battle, it is hard to be hit by other robots. Therefore, I implemented a robot which imitates the moving pattern of 'Walls' and made some modifications based on it.

- Modifications

  1. Moving to the nearest wall when battle begins

  Code:

```
private int getNearestWall() {
    myLoc=getMyLoc();
    double[] distances=new double[4];

    distances[0]=getBattleFieldWidth()-myLoc.y;
    distances[1]=getBattleFieldHeight()-myLoc.x;
    distances[2]=myLoc.y;
    distances[3]=myLoc.x;

    double shortest_dis=distances[0];
    int shortest_index=0;
    for(int i=1;i<distances.length;i++) {
        if(distances[i]<shortest_dis) {
            shortest_index=i;
            shortest_dis=distances[i];
        }
    }
    return shortest_index;
}
```

  Explanation: At the beginning of a battle, distances to 4 walls will be calculated. Then my robot will go straight to the nearest wall and start moving along the walls.

2. Dodging bullets
Code:

```
public void onHitByBullet(HitByBulletEvent e) {
    if(getEnergy()<60 && getOthers()>1) {
        if(e.getName()==lastHitByBullet) {
            if (e.getBearing() > -90 && e.getBearing() < 90) {
                if(getHeading()%90==0) {
                    if(clockwise==1) {
                        turnRight(90);
                    }
                    else if(clockwise==-1) {
                        turnRight(-90);
                    }
                    ahead(-moveAmount);
                }
            }
            else {
                if(getHeading()%90==0) {
                    if(clockwise==1) {
                        turnRight(90);
                    }
                    else if(clockwise==-1) {
                        turnRight(-90);
                    }
                    ahead(moveAmount);
                }
            }
        }
    }
    lastHitByBullet=e.getName();
}
```

Explanation: If the robot catches a bullet, it will record the name of the robot which fired that bullet. Then it will check whether this bullet and last bullet come from the same enemy. If this true, then the enemy will be regarded as firing at my robot intentionally. In this situation, my robot will change the current move pattern to dodge the bullet.

3. Smart fire

Code:

```
public void onScannedRobot(ScannedRobotEvent e) {
    if(getEnergy()<=10&&getEnergy()>=0.5) {
        fire(0.1);
    }
    else if(getEnergy()>10 || getOthers()>1) {
        smartFire(e.getDistance());
    }

    if (peek) {
        scan();
    }
}

public void smartFire(double robotDistance) {
    if (robotDistance > 150 ) {
        fire(2);
    }
    else{
        fire(3);
    }
}
```

Explanation: If an enemy is discovered, my robot will first check its left energy. If it is still healthy, the distance to the enemy will be obtained. If the distance greater than 150 pixels, then set the bullet power to 2. Else if the distance is less or equal to 150 pixels, then set the bullet power to 3.

Reference:

[1]"AdvancedRobot.SetEventPriority Method", *Robocode.sourceforge.io*, 2020. [Online]. Available: https://robocode.sourceforge.io/docs/robocode.dotnet/html/1a261629-e10b-29a6-8896-e45ba1e267bc.htm. [Accessed: 06- May- 2020]