COMP222 A2: Robocode

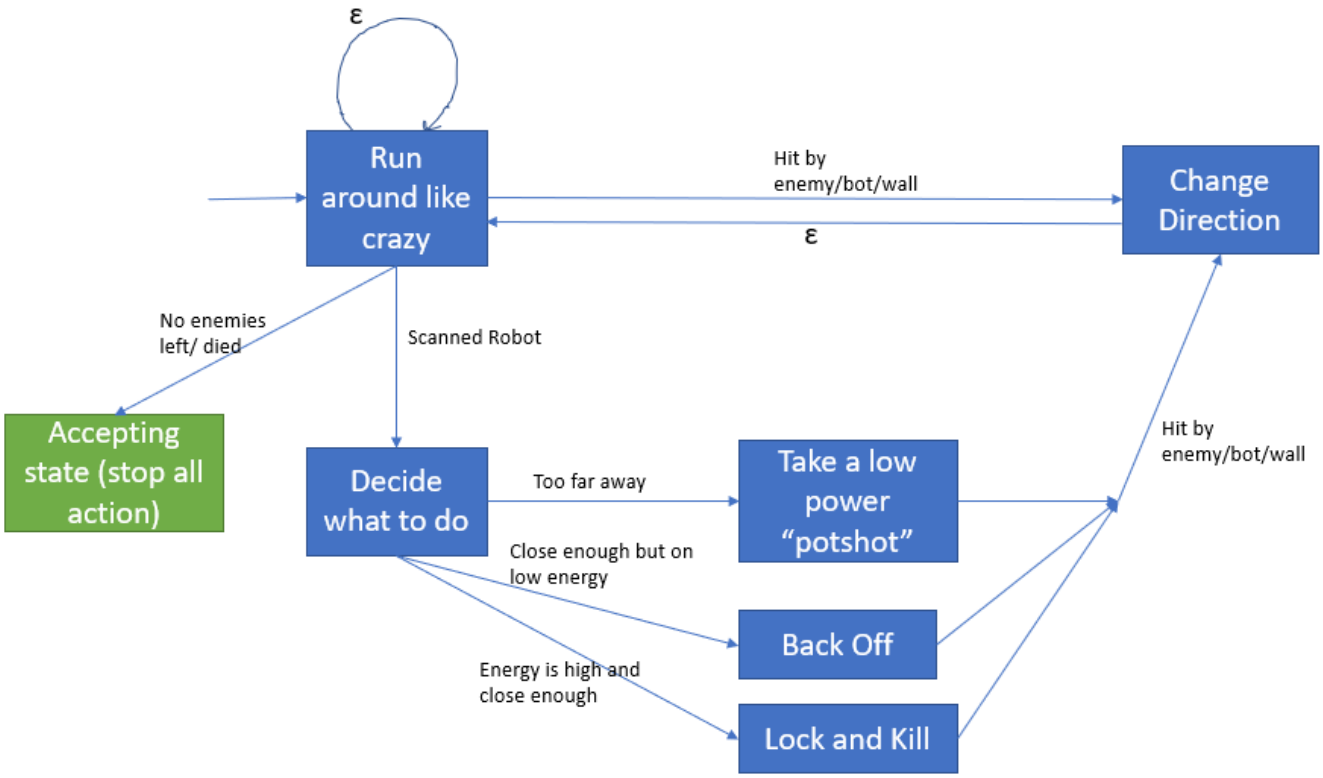
Behaviour Control Model

I decided to base my robot's behaviour on a finite state machine because I thought this would suit the event-based nature of the game environment. The robot could loop in the start state with some default behaviour like avoiding trouble. Events could then trigger the transition into different states so the robot could respond appropriately. If there are no further events it could go back to its default behaviour in the start state. I guess the accepting state would only be accessible if the robot has died or there are no other robots left as it would no longer need to loop on its default behaviour.

I thought on this basis I could continually improve the robot by adding more behaviour states within the initial response states. This would allow it to deal with more situations. I also thought that it would be important to have some states accessible from most other states so it could deal with emergency situations. For example, my robot could be in the middle of attacking an enemy but if it hits a wall it needs to immediately go to another state to respond appropriately.

Robocode Bot Design

So, here is what I meant in practise:



The idea is that my robot will constantly run around to avoid being targeted by other bots. If it scans an enemy, it will consider its options before acting to avoid recklessness. If the enemy is too far away, then it will not bother pursuing but will take a low power shot. If the enemy is close and my bot has plenty of energy, then it will head towards the enemy whilst firing increasingly powerful shots as the distance between bots is decreased. (This is beginning to sound like the government's current health advice). If the enemy is close but my bot is on low energy, then it will still attack but will move backwards whilst doing so.

In practise this has worked quite well because my bot is very aggressive but knows when to back off and when to preserve energy. I think these are characteristics that will help it do well in a tournament. However, in battles with many enemies, my bot is sometimes unlucky and gets caught in the middle. In these situations, it is normally pesky bots like "sample.Walls" and "sample.Spinbot" that somehow come out on top. Damn those frustratingly simple but effective bots!

Implementation

Okay so the “Run around like crazy” state is implemented in the robot’s main loop. I was inspired by the bot “sample.Crazy” for its manic dodging and weaving abilities so my loop looks similar.

“onHitByBullet”, “onHitWall” and “onHitRobot” are all event handling methods that simply call the “changeHeading” method. This means that at any time from any state my bot knows what to do if hit by a bullet, wall, or enemy. The “changeHeading” method just changes whatever direction was headed in previously to the opposite direction. This means that if my bot goes into a wall it will move away from it etc.

I added a method to change the colour of my bot so it would be easy to tell what its current behaviour is. When it is “running around like crazy” my bot should be black (its intended default colour).

“But what about when the bot scans an enemy”? As detailed in the FSM diagram my bot checks its current situation and decides what to do accordingly. If the enemy is of a distance greater than 400 (away from my bot) then my bot responds with a “fire(1)”. The direction of the gun should be the same as my bots heading at this point. The other responses on enemy scans are more complicated and involve some if statements.

“lockAndDestroy” is a method that will be called every tick an enemy is scanned, and the distance is less than 400 and energy is more than 50. It “locks” on to the enemy by moving the scanner and gun towards the target so that the method is called in the next tick when the target is scanned again. It then checks which shot to use while moving closer to the target. If the target is far away, then my bot will fire the weakest shot. If the target is between 400 and 300 distance then the shot is upgraded to “fire(2)”. If the target is between 300 and 100 distance then the shot is upgraded to “fire(3)”. If the target is closer than 100 distance, then my bot fires the most powerful shot and stops moving towards the target. Often by this point my bot has already collided with the enemy so it moves back in response. This means that the bot sort of goes back and forth while shooting appropriate shots at the target. When in the “lock and kill” state my bot should be coloured red.

“backOff” is the same as “lockAndDestroy” but instead of moving towards the enemy whilst shooting it, it moves back. It should be coloured white when my bot is in this state.