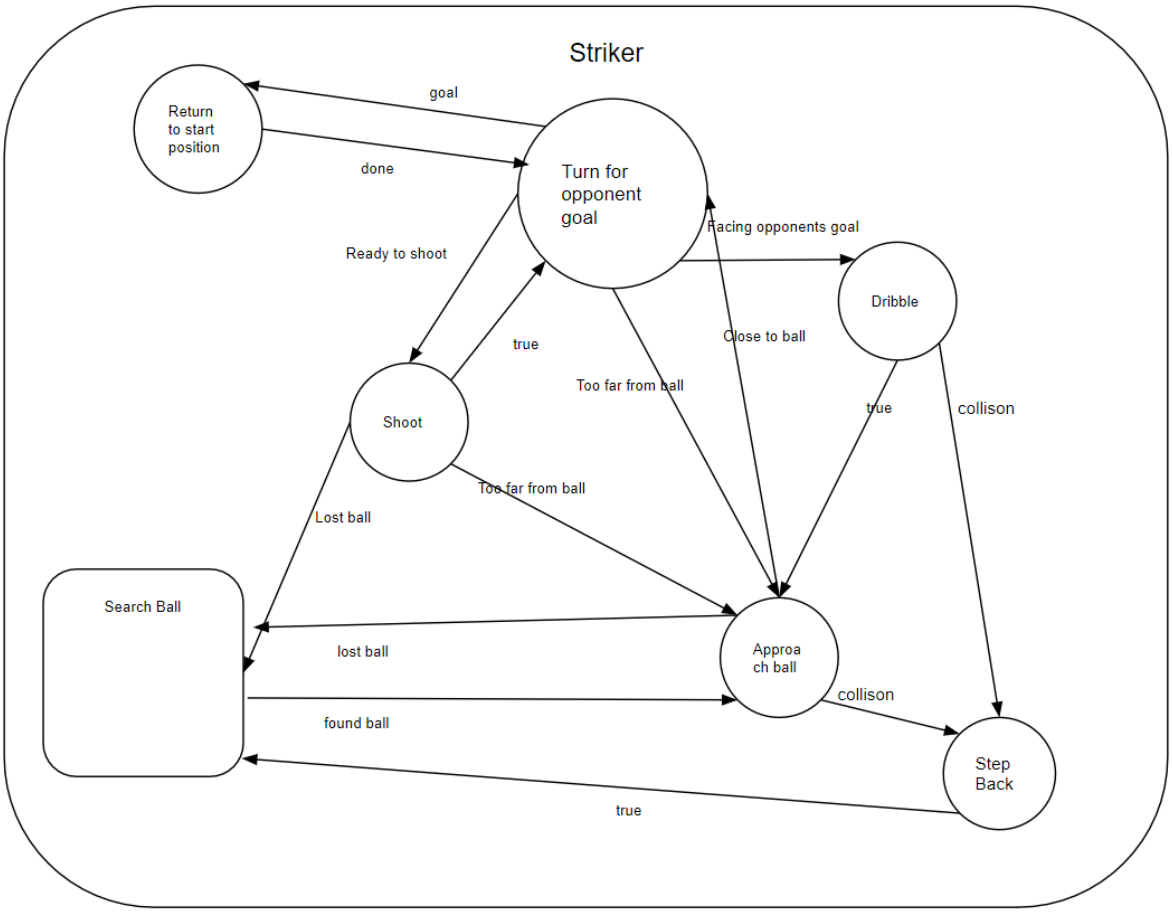
Robocup Behavior

Project by Simon Thomes and Dominic Schialer

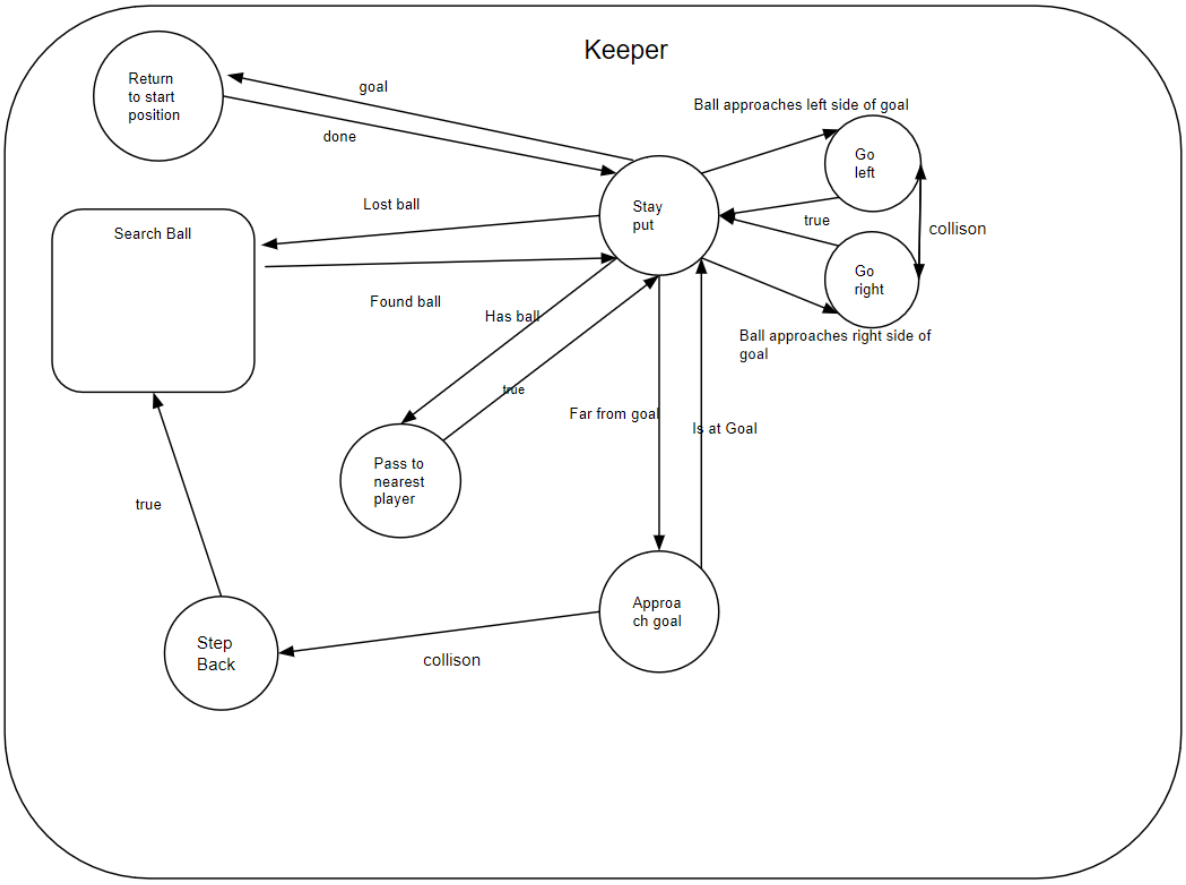
We wanted to create behavior for the robots by using state machines. We designed three different state machines to define three roles for the robot. The *Striker*, the *Keeper* and the *Defender*. The state machine for every role is simple and can be easily implemented. We use a TeamViewer, which simulates the robots and the field. We had to create our own and we couldn't use simpspark, because we would have had to create vision, communication, localization and motion. So we created a TeamViewer that simulates these things.

The Striker:



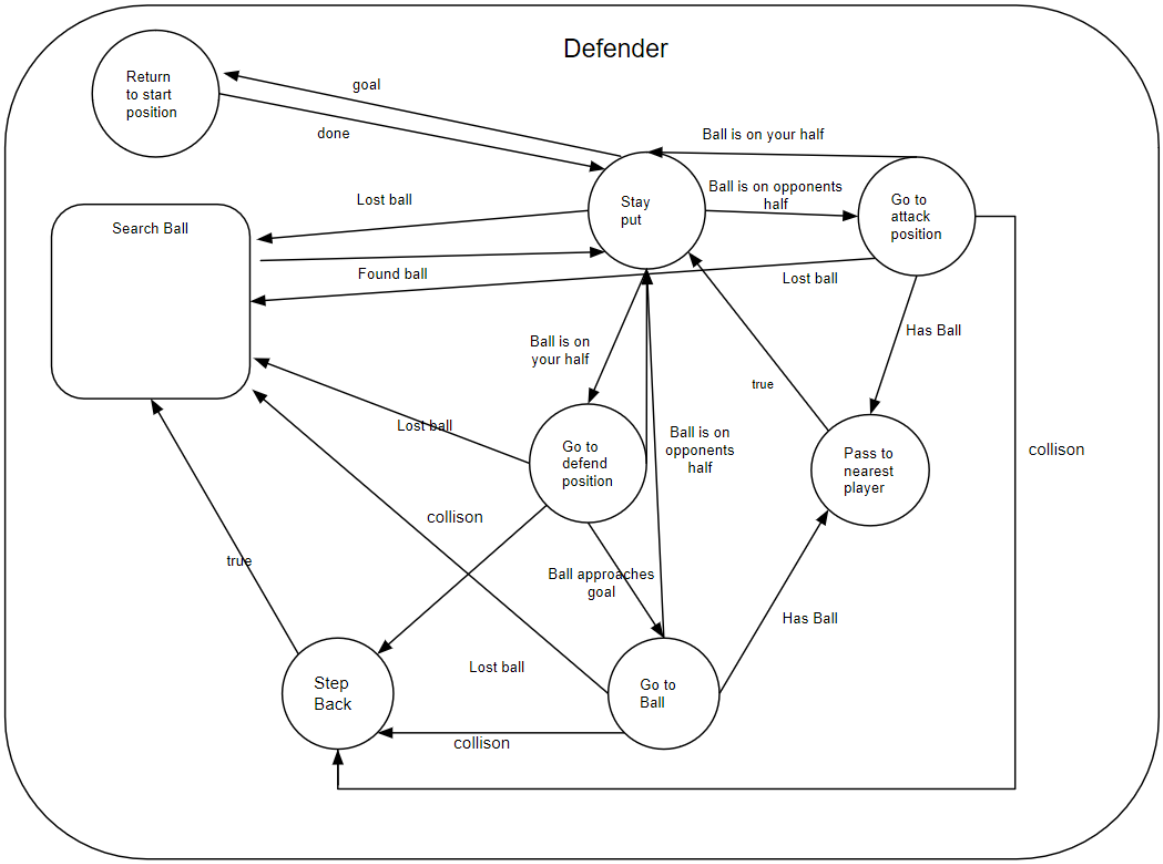
The idea behind the *Striker* is to create an aggressive player who always follows the ball. When he has the ball he will try to score a goal for his team. When he loses the ball he tries to find it by looking and when required turning. The main purpose of our *Striker* is to get the ball in the opponents goal. Our *Striker* could be improved by making him more complex for example passing to other players and trying to steal the ball from other players more effectively. Sadly we did not have enough time to realize all our ideas, because we wanted to create more classes than one.

The Keeper:



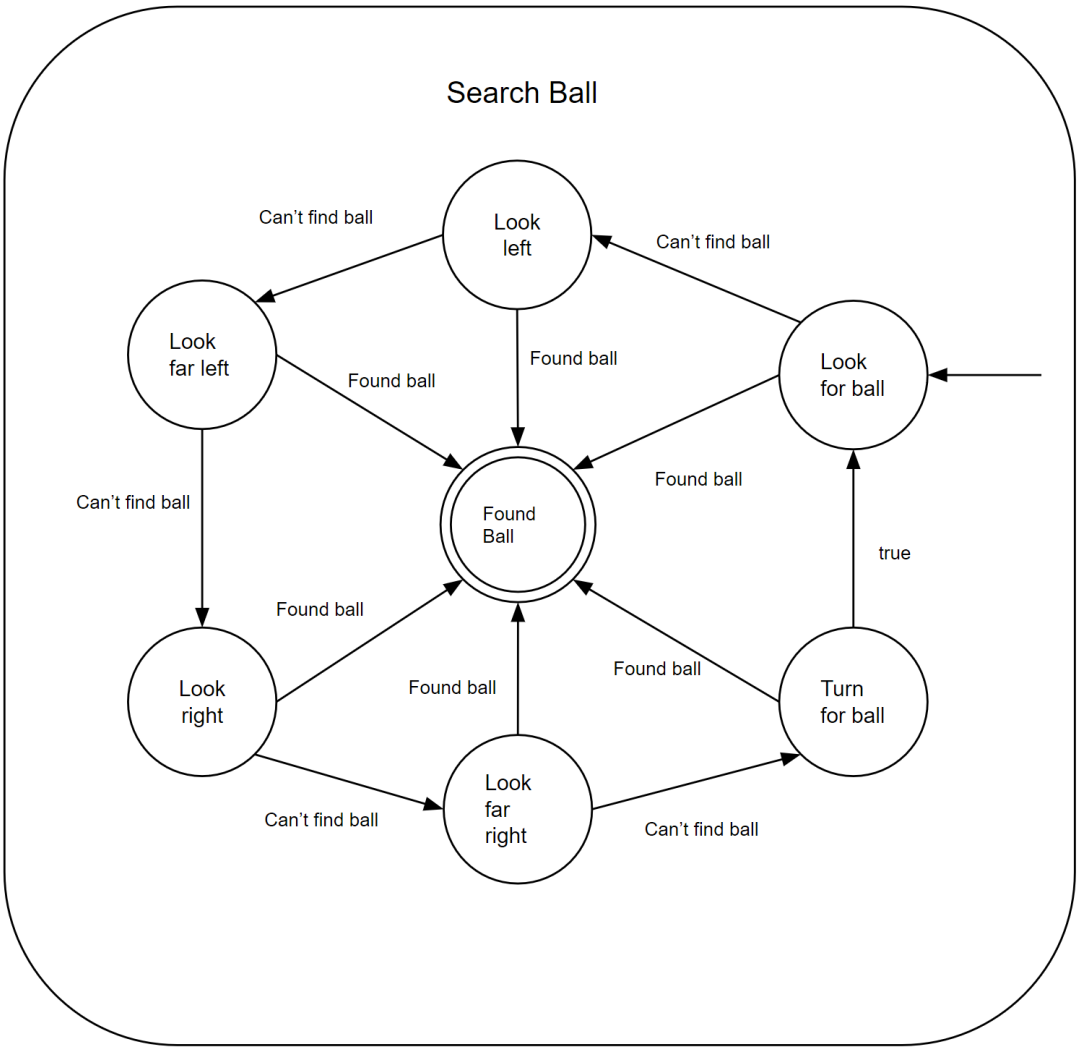
The idea behind the *Keeper* is to create a very defensive player, whose only goal is to prevent goals. At the start of a match he goes to his goal and stays there to defend it. He can look for the ball and can step to the left or the right depending on whether the ball is approaching that side. When he has the ball he will pass it to the nearest *Striker*.

The Defender:



The *Defender* is a kind of mix between the *Striker* and *Keeper*. Every *Defender* has two main positions: an attack position and a defend position. When the ball is on his half he goes to the defend position and tries to block balls which are coming toward the goal. In the attack position (the ball is on the opponents half) he tries to be a support player for the *Strikers* by being in a good position to pass the ball to other players and getting the ball passed to.

Search Ball:



Search Ball is its own state machine. It starts with *look for ball*. First he looks to the left and then to the right, if he didn't find it in the current position he will turn and look again, until the ball is found. All state machines have collision detection and if they have a collision the will step back a bit.

The state machines are coded in python and can be found in the StateMachine folder. To use the state machine you just have to use .run(action) on it and with the right action it will change to a different state. The current state can be found with .currentSate. It is very easy to use our state machines with different programs.

TeamViewer

Our TeamViewer is implemented using python and uses the pygame (www.pygame.org) to create the visuals. We encountered problems running pygame on Linux, your mileage may vary.
The TeamViewer simulates the robots and the ball. Movement of robots is dictated by the capability of NAO robots. For example the robots in our TeamViewer can only rotate their head between 119.5° and -119.5° and have the same FOV (67.4°DFOV). We do this that way, so our robot and the NAO robot share the same limitation. The TeamViewer uses the implementation of the state machines and also decides what to do in which state exactly. Some states are not implemented like the state machine describes them but the main functionality is there. For example, *Pass* does not shoot to the nearest friendly player. We have a bug with searching the ball. Sometimes a robot can't find the ball even though he looks at it. Sadly we did not have enough time to correct these things.
We created test scenarios which are defined in game.py.

Scenario 1: 1 Goalkeeper (blue) and one Striker (red). (best scenario)
Szenario 2: 1 Striker trying to find the ball.
Scenario 3: Defender (red and blue) 1 Keeper (blue) 1 Striker (red) (you can see defenders are not very effective)

You can add players by using game.add_player() in main of game.py. To start the TeamViewer execute game.py (with number for scenario) and a window should show a field and robots. It is possible to speed up the simulation by pressing the number keys (1,2,3,4 for 1x, 2x, 3x, 10x normal speed). You can reset the field by pressing “r”.