Tecnológico de Monterrey

Sistemas Inteligentes

Reactive Agent

Martín Noé Márquez García - A01351211

Abstract

The present paper is focused in the development and implementation of a Reactive Agent using Starcraft 2 computer game as a tool. The main objective was to win in medium difficulty, but with the following improvements presented, it was possible to win in hard and very rarely in harder.

Theoretical Framework

Reactive Agents

Reactive agents are the ones that are capable of maintaining an ongoing interaction with the environment, and responding in a timely to changes that occur in it, so we can consider that a Reactive Agent will always respond and interact in a timely fashion to external stimulus.

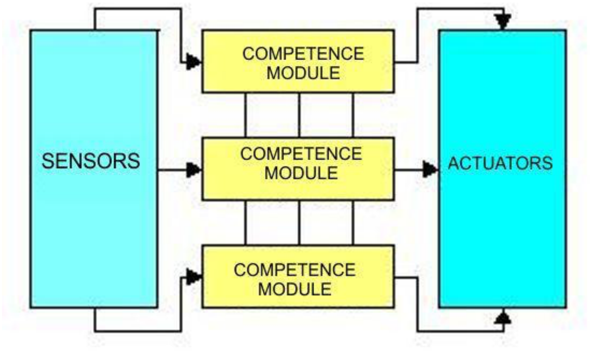


Figure 1: *“Example of a Reactive Agent diagram”*

The diagram of this project can be seen at the end.

Before Start Programming

For the method that will be presented in this paper we need to consider to install some stuff and libraries in python:

Library	Python Enviroment	Conda Enviroment
from pysc2.agents import base_agent	pip install pysc2	conda install pysc2
from pysc2.env import sc2_env	pip install pysc2	conda install pysc2
from pysc2.lib import actions, features, units	pip install pysc2	conda install pysc2

Installing Starcraft 2

Install Starcraft 2 (video guide at: https://youtu.be/KYCxZBghVZ8)

Go to:
https://www.blizzard.com/

Create an account, login and password

Go to:
https://www.battle.net/download/getInstallerForGame?gameProgram=STARCRAFT_2
to download the installer.

Run the installer.
This must be done Outside of the campus network (ports are blocked) or
use a proxy/vpn.

Download and install the game (25 GB).

Installing the maps:

Go to:
http://blzdistsc2-a.akamaihd.net/MapPacks/Melee.zip

download the zip (The files are password protected with the password iagreetotheeula)
create a folder in the installation of starcraft2 called Maps and extract the folder there.

In cmd run:
python -m pysc2.bin.agent --map Simple64

Development/Code:

In the first lines of code you will find that Im choosing Terran as a race to play so the strategy used below, buildings, improvements units, are all of the Terran race.

```
class
TerranAgent(base_agent.BaseAgent):
```

It's important to remember that there are 2 possibilities of location when you start a new game, above or below. So I decided to create a flag and work on the 2 possible scenarios, both to know where we have to attack and to know where our base is.

```
self.attack_coordinates = None
self.pos1 = False
```

Having this we can start choosing the strategy, I decided to build the supply depots first to have more space for food to create more attack units.
Knowing that the reactive agent build the things in a random way I decided to use an array to choose the the exact location to build them faster and that our agent does not take so much time looking for a free space to build them.

I created the following variables to add 7 units in x and 7 in y to the location of the supply depot and get them sorted.

```
self.sepx = 7
self.sepy = 7
```

Then create my two supply depots options depending on where we appear. In my first array I created the reference supply depots with the coordinates above and one below the map.

```
self.supply_depots = [[50, 57], [10,
10]]
```

Here you can see the option when we left at the top of the map.

```
self.sup = [(self.supply_depots[0]),
 (self.supply_depots[0]
[0],self.sepy+self.supply_depots[0]
[1]), (self.supply_depots[0]
[0]+self.sepx, self.supply_depots[0]
[1]), (self.supply_depots[0]
[0]+self.sepx, self.supply_depots[0]
[1] + self.sepy)]
```

So we can say that the coordinates of my supply depots are:

X	Y
50	57
50	64
57	57
57	64

Here you can see the option when we left at the bottom of the map.

```
self.sdown =
[(self.supply_depots[1]),
(self.supply_depots[1][0], self.sepy
+ self.supply_depots[1][1]),
(self.supply_depots[1][0] +
self.sepx, self.supply_depots[1]
[1]),(self.supply_depots[1][0] +
self.sepx, self.supply_depots[1][1]
+ self.sepy)]
```

So we can say that the coordinates of my supply depots are:

X	Y
10	10
10	17
17	10
17	17

To build my barracks I also use arrays with specific locations so that at the time of building the reactors they had enough space to be built.

```
self.barracksup = [[38, 62], [70,
60], [75, 45]]
self.barracksdown = [[15, 25], [17,
40], [35, 15]]
```

Building Supply depot

In the next function we can see that I'm using here my flag of position so that my agent can know if it is up or down and thus use the special coordinates of each case. Im also using a for function to call all my array options.
So my complete function to create my supply depots is the next one:

```
def supply_depot(self, obs):
    supply_depot =
self.get_units_by_type(obs,
units.Terran.SupplyDepot)
    food_cap =
obs.observation.player.food_cap
```

```
    if len(supply_depot) < 4 or
food_cap < 20:
        if
self.unit_type_is_selected(obs,
units.Terran.SCV):
            if self.can_do(obs,
actions.FUNCTIONS.Build_SupplyDepot_
screen.id):
                if self.pos1:
                    self.i += 1
                    if self.i == 5:
                        self.i = 1
                    return
actions.FUNCTIONS.Build_SupplyDepot_
screen("now", ((self.sup[self.i-1]
[0]),(self.sup[self.i-1][1])))
                else:
                    self.i += 1
                    if self.i == 5:
                        self.i = 1
                    return
actions.FUNCTIONS.Build_SupplyDepot_
screen("now", ((self.sdown[self.i-1]
[0]),(self.sdown[self.i-1][1])))
        else:
            SCV =
self.get_units_by_type(obs,
units.Terran.SCV)
            if len(SCV) > 0:
                SCV =
random.choice(SCV)
```

```
                return
actions.FUNCTIONS.select_point("sele
ct_all_type", (abs(SCV.x),
abs(SCV.y)))
```

Building Barracks

In the next function we can see that I'm using here my flag of position so that my agent can know if it is up or down and thus use the special coordinates of each case. Im also using a for function to call all my array options.
So my complete function to create my barracks is the next one:

```
def barracks(self, obs):
    barracks =
self.get_units_by_type(obs,
units.Terran.Barracks)
    if len(barracks) < 3:
        if
self.unit_type_is_selected(obs,
units.Terran.SCV):
            if self.can_do(obs,
actions.FUNCTIONS.Build_Barracks_scr
een.id):
                if self.pos1:
                        self.i+=1
                        if self.i >
3:
                            self.i=0
```

```
print(self.i)
                        return
actions.FUNCTIONS.Build_Barracks_scr
een("now",
(self.barracksup[self.i]))
                else:
                        self.i+=1
                        if self.i >
3:
                            self.i=0
```

```
print(self.i)
                        return
actions.FUNCTIONS.Build_Barracks_scr
een("now",
(self.barracksdown[self.i]))
        else:
            SCV =
self.get_units_by_type(obs,
units.Terran.SCV)
            if len(SCV) > 0:
                SCV =
random.choice(SCV)
```

```
                return
actions.FUNCTIONS.select_point("sele
ct_all_type", (abs(SCV.x),
abs(SCV.y)))
```

Building Reactors

For my strategy I decided to implement the construction of reactors, this to be able to create units 2 times faster than normal.

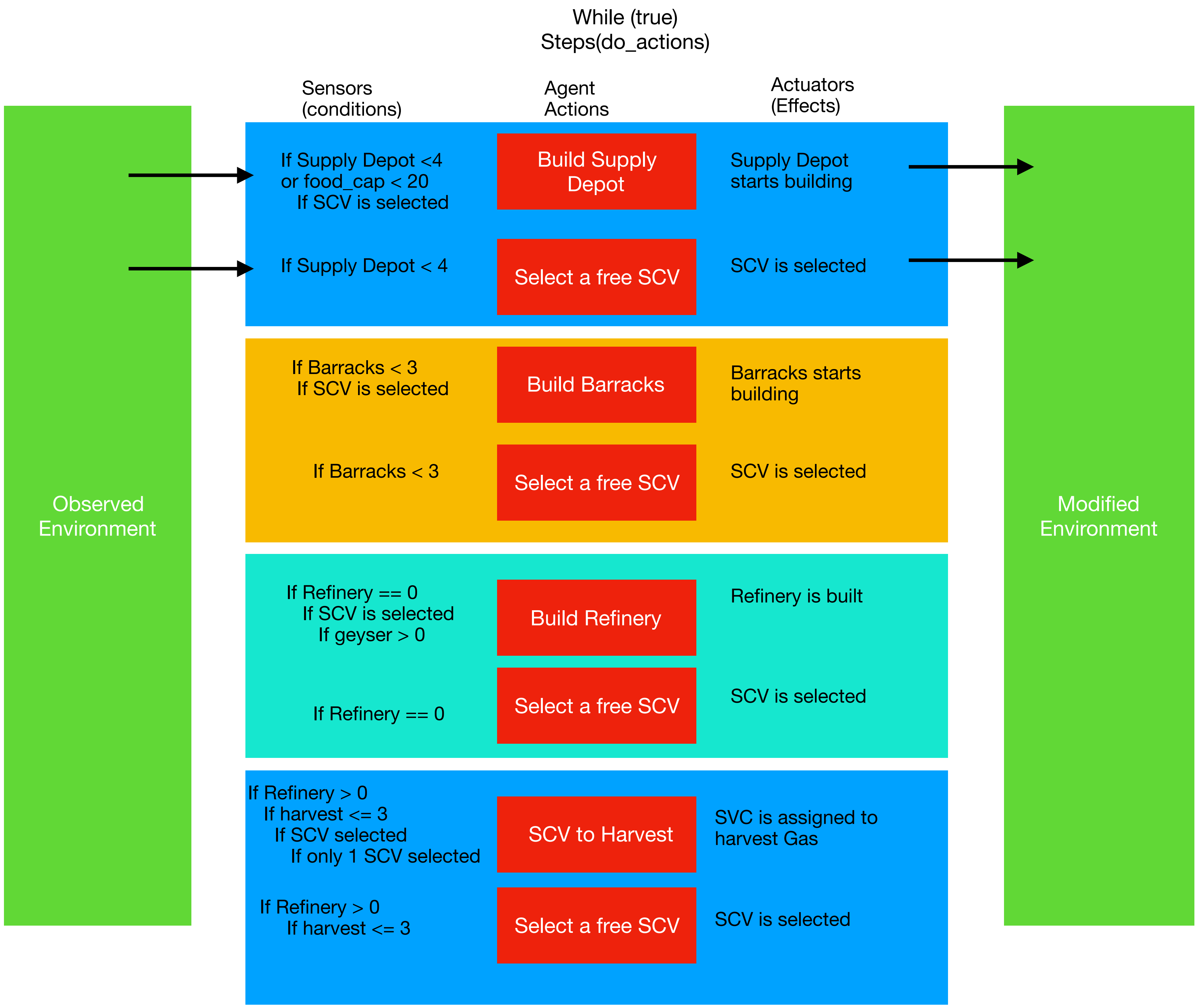
```
def reactor(self, obs):
    reactor =
self.get_units_by_type(obs,
units.Terran.BarracksReactor)
    if len(reactor) < 3:
        if
self.unit_type_is_selected(obs,
units.Terran.Barracks):
            if self.can_do(obs,
actions.FUNCTIONS.Build_Reactor_quic
k.id):
                return
actions.FUNCTIONS.Build_Reactor_quic
k("now")
        else:
            barracks =
self.get_units_by_type(obs,
units.Terran.Barracks)
            if len(barracks) > 0:
                barracks =
random.choice(barracks)
                return
actions.FUNCTIONS.select_point("sele
ct_all_type", (abs(barracks.x),
abs(barracks.y)))
```

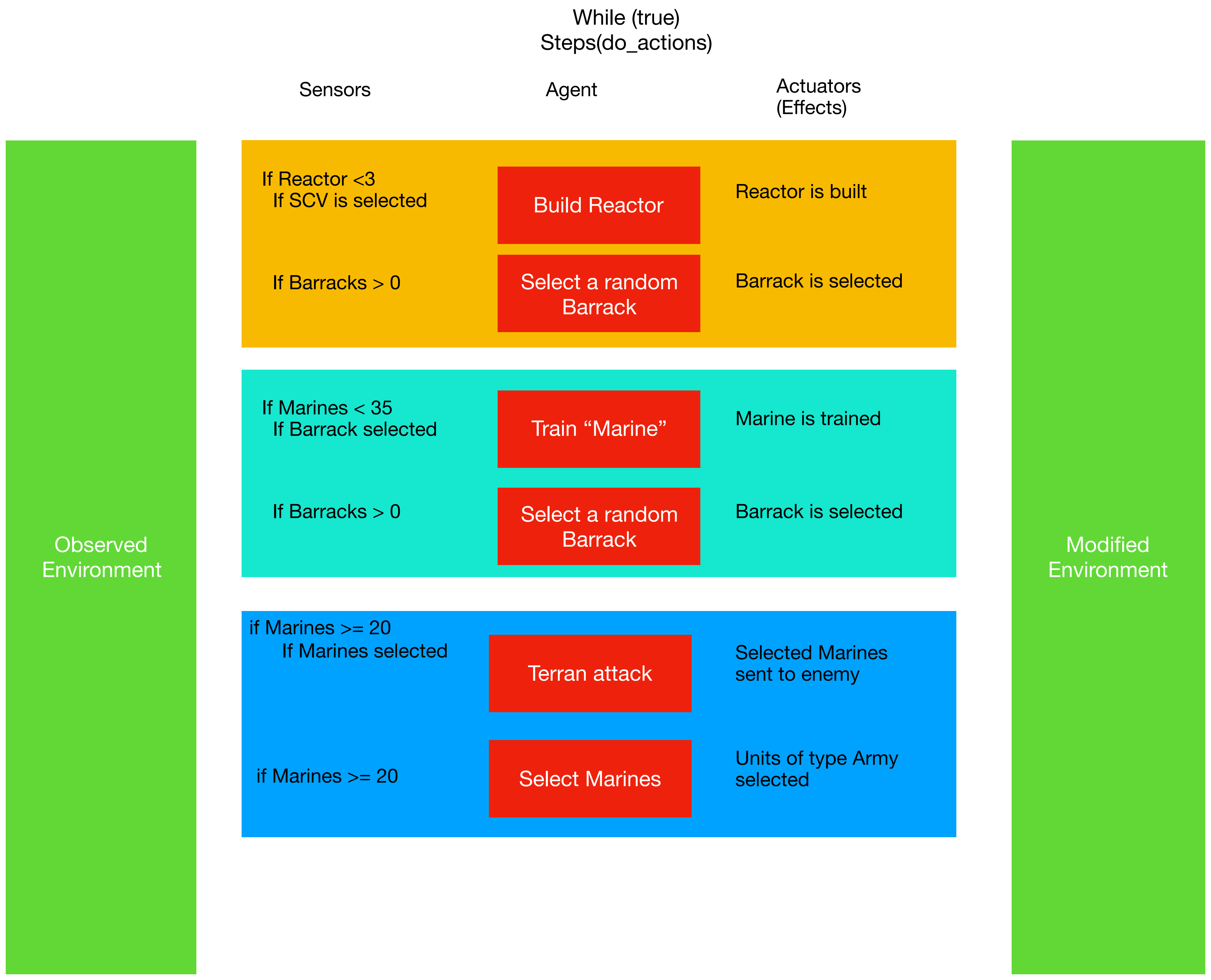
Conclusion/Improvements

The above presented were only some improvements to the documentation that is already on the internet and that were thought during the project for the improvement of the agent.

I think that the project could still have areas for improvement, as it could be fixing a problem when selecting scv to build since sometimes the agent takes the one who is working in the refinery and does not return for what is no longer It produces gas and this brings with it some problems.

I hope this documentation is useful for the following semesters and can improve what has already been done this semester.


While (true)
Steps(do_actions)
Sensors (conditions)
Agent Actions
Actuators (Effects)
Observed Environment
Modified Environment
If Supply Depot <4 or food_cap < 20 If SCV is selected
Build Supply Depot
Supply Depot starts building
If Supply Depot < 4
Select a free SCV
SCV is selected
If Barracks < 3 If SCV is selected
Build Barracks
Barracks starts building
If Barracks < 3
Select a free SCV
SCV is selected
If Refinery == 0 If SCV is selected If geyser > 0
Build Refinery
Refinery is built
If Refinery == 0
Select a free SCV
SCV is selected
If Refinery > 0 If harvest <= 3 If SCV selected If only 1 SCV selected
SCV to Harvest
SVC is assigned to harvest Gas
If Refinery > 0 If harvest <= 3
Select a free SCV
SCV is selected


While (true)
Steps(do_actions)
Sensors
Agent
Actuators (Effects)
Observed Environment
If Reactor <3
If SCV is selected
Build Reactor
Reactor is built
If Barracks > 0
Select a random Barrack
Barrack is selected
If Marines < 35
If Barrack selected
Train “Marine”
Marine is trained
If Barracks > 0
Select a random Barrack
Barrack is selected
if Marines >= 20
If Marines selected
Terran attack
Selected Marines sent to enemy
if Marines >= 20
Select Marines
Units of type Army selected
Modified Environment