Captain Sonoz - Project 2020
LINGI1131
Group078

Benjamin De Cnuydt

December 20, 2020

Chapter 1

Players strategies

This chapter explains different strategies I developed for each player. The first one represent a random player with some other specificities and the second one is an evolution of it with no random.

1.1 Random captain Iglo

Random captain Iglo is a random AI capable of choosing a random prey and follow it until it dies.

1.1.1 Seek and kill

At the beginning of the game, a prey is chosen and the player will try to reach it at distance allowing to throw a missile. If the prey is already too close, a random move is made. Once the prey is dead, another one is chosen until one player remains or captain Iglo is dead.

1.1.2 Charge and fire items

At the beginning of the game, we choose randomly an item to start charging, once an item is charged, it randomly chose another one that is not already charged. If a sonar or a drone is fully charged, it directly fire it, the drone is fired at a random column or row. A mine is landed once charged and captain Iglo waits to have a correct range to fire a charged missile.

1.1.3 Update positions

Captain Iglo keeps track of each player during the game. Once it fires a sonar, each players positions are update regardless the previous ones kept (it updates position even if some correct positions are lost).

1.1.4 Explode mines and missiles

Each turn, during the fire item and explode mine, it first checks if a player is at range of our missile/mine and if captain Iglo is not going to commit suicide.

Because it doesn't update players positions in a correct way, we can consider this approach more or less random.

1.1.5 Analyse what ennemies say

- saymove: update the position of your players record even if some positions may be wrong.
- sayanswersonar: update the players record with the new position of the answering player.
- saypassingsonar: give a random false coordinate among the position of captain Iglo.
- saypassingdrone: nothing to do.
- saydeath: choose another prey and can include itself (captain Iglo suspect a mutiny to steal his recipe)

1.2 Eat my fish stick

This AI is more accurate in locating enemies by creating a trust list for each player.

1.2.1 Trust list

The AI respects three distinct steps played in this order:

Step 1 : localize

The AI starts the game by initialize an empty record that will contain position of each players. A record with the same structure is created but the meaning is different, it's the "trust record". The first item to charge fully and to fire directly is the sonar. Once it gets and answer, it adds the position of the concerned player into the position record (for instance : players(1:pt(x:1 y:5) 2:pt(x:7 y:6))) . For the trust record, some aspects are take into account. The values inside this record have different meaning, instead of the position for the x or y, it puts a trust value :

- 0 : Trust this coordinate (x or y), it's the correct one.
- 1 : The coordinate is not accurate but close from 1 square.
- 2 : Don't trust this coordinate

For instance, a trust record looks like this : info(1:pt(x:0 y:2) 2:pt(x:1 y:0)).

After the first sonar, it decides to choose one coordinate to trust (0), the other one will remain not trust. Even if the one trusted is not correct because of the first turn, it decides to trust it. The first coordinate to trust is not random and depend of the number of row or column, we decide to trust x if there is more row than column and vice-versa.

A prey is chosen at random, but this time the crew is trustful and it never takes itself as a prey.

Step 2 : confirm trust list

Once a sonar is fired, it goes through the step 2 consisting in charging a drone and fire it at the row or column where the AI decided to trust first. If the answer is true, it means that we chose a good coordinate to trust and now we can be sure of this coordinate concerning the player. If the answer is false, it changes the trust list corresponding of the player because the other one is correct and becomes trust (0). If another player that is not the current prey is also concerned by the column or the row where the drone passed through, the AI changes the trust list as well.

Each time a drone is launched, the AI considers to change the prey if there's a player with a better trust status. Indeed, a player with pt(x:0 y:0) will be easier to kill then a prey with pt(x:2 y:0).

Step 3 : Let's the hunt begins

Once step 1 and step 2 finished, the step 3 consists in charging a missile and until the AI fires it, it will land mines all over its path. If the missile is fired, it charges a missile again and wait the good moment to fire it.

Explode mines and missiles remain the same as the previous AI, but it decides to shoot on a enemy on range only if the concerned player has a trust record with no trust value of 2 in it.

Go back to step 1

Every X turns, the AI will go back to step 1 and try to make more accurate the trust and player record.

1.2.2 Analyse what ennemies say

- saymove: Update players positions record.
- sayanswersonar: update the players record, a coordinate is updated only if it is not trust (2) in the trust record (step 1).

- saypassingsonar: Always lies on the same coordinate (X if there is more column than row, Y instead). By lying on the same coordinate again and again, it will be more difficult to locate.
- sayanswerddrone: see step 2.
- saydeath: choose another prey and can't include itself.
- saymineexplode - saymissileexplode : It keeps track of the position of each enemy mine or missile that explose in order to use them in the saydamagetaken instruction.
- saydamagetaken: analyze the position of the last mine or missile that explode and try to correlate with the position of the explosion and the position of the players that the AI track. If the position is significantly different, the players position record and trust list is updated with 0 for the position if the damage taken is 2 (the missile exploded at the extact player's position), 1 if the damage taken is 1.

Chapter 2

Implementation design and choices

2.1 Main.oz

I decided to create the same procedure (launchGame) for the turn by turn and simmultaneous to avoir the maximum of redundancy and keep the same logic behind my conception of how the game works, even if some conditional parts can be annoying in the procedure. I divided the broadcast in two parts, one in a procedure to just broadcast non lethal information and another one to return a stream containing answers of exploding missile or mine. I tried to make a maximum of function to have a clear view of the different steps during a turn inside my main procedure.

2.2 Player078.oz

For my players, I didn't take the same decision of refactoring everything into functions because I hate to scroll from one function to another to much time and I prefer to let the main code logic inside each case (move, sayMove,etc...). If some aspects needs to be clear, I consider creating a function with a relevant name. I didn't chose to implement path finding algorithms because this is pretty difficult with a multi agent game and the fact to optimize the surface time.

Chapter 3

Extensions

Images and sound have been added to the project. Mines are represented with the head of M. Van roy and sounds are played during specials events. Sounds are available only on Linux and the program *mpg123* is needed:

- Start of the game: M. Van Roy motivates the submarines before the battle.
- Submarine hit (only in turn by turn): What ? The AI failed ? It's maybe a bug ...
- Submarine destroyed: It's nice, M. Van Roy is proud of the crew. (Play if the submarine destroyed doesn't put an end at the game).
- End of the game: M. Van Roy gives a good grade to the remaining submarine, 5 points to Gryffindor !