LINFO1131 : Paradigmes de programmation concurrente

UCLouvain

École polytechnique de Louvain

Groupe 17

Auteurs :
Andrei Nesterov
Jérôme Lechat

NOMA :
4166-02-00
5035-18-00

1 Introduction

For this project, we were asked to make a customized version of a popular game called Capture The Flag (or CTF). The implementation must be made to be played simultaneously (not turn by turn).

For that, we had at hand a Game Controller and a Player Controller.

2 Player Strategies

As we were asked to design new players with behaviour, here are our different strategies :

2.1 Player017Random

This Player got a simple kind of strategies : even if his methodologies are a bit 'dumb', it was our first Player created and thanks to that, we were able to understand the code and to make him more and more robust with time.

Once the game is initialized, players of each teams will spawn in their base.

For the movements, he will only move itself in random maneer : north, south, west or east, depending once again on where he can really move. He will try all the possibilities he has until one is correct. Once he finds the correct one, he will execute it. This is thus a random player (hence the name "Player001Random").

Concerning the weapons, he will choose (also randomly) which weapon he will reload : once the choose is made, if the weapon is already full loaded, the other weapons is going to be reloaded instead. If all the weapons are full loaded, then nothing will happen. Afterwards, the player will be asked if he wants to use one of his weapons : if the weapons he decide to use is full loaded, then the player will instantly answer that he wants to fire with this weapon.

Concerning the flags, player will be asked if he wants to take the opposite team's flag, once he takes it, he won't drop it until he dies or until he arrives at his base. If the player dies while carrying the opposite team's flag, the flag will be dropped exactly where the carrier died. If the opposite team's flag arrive at the base, then the game is finished and we have our winner !

Concerning the life and the potential explosion near the Player, he spawn at the beginning at the game with StartHP (which is equal to 2). If a mine explodes near the Player, the damage taken will depends the Manatthan Distance between him and the mine (the damage can be up to -2). If the Player got no more lifes, he will be ereased from the GUI for a short period of time (SpawnDelay depending the input) and will respawn to his base.

2.2 Player017Smarter

Due to a lot of problems and a lack of time, this kind of Player hasn't reach the level we hoped he would have : we were only able to optimize the move system : he will first try to find the shortest way to the ennemy flag. He can also focus on a ennemy. For the rest, it does the same thing as its 'little brother' Player017Random.

3 Our implementation

3.1 The Game Controller

At first, we thought about stocking a maximum of informations by creating a DB on the GameController... but we had too much problems and complications so we decided to do something

else.

Basically we needed a kind of sink that would gather all the information about flags and mines, from all of the threads in one place in order to guarantee that a flag be not picked by two players simultaneously. So we used a port/stream construction in such a way that all of player threads in the game controller have a pointer to it so that they can send updates about the state of affairs.

More precisely during the initialization of the controller we create a port connected to a stream and each player thread has a way to access it through the `commonPort` field in the State record of the thread. Then there is a dedicated loop called `ProcessCommonStream` that processes the incoming through the port elements of the stream.

This way we can have a shared read/write state without the need of using cells. Implemented functionality includes flag management, as a way to track what happens to flags, who carries it where, or where it is located. It works similarly with mines. If someone drops a mine, it is registered in the state of the `ProcessCommonStream` loop.

3.2 Player017Random

For this player, there is no really implementations done here.

This player gives a little importance to the informations received by the game telling other players are doing something.

4 Our extension

Due to the lack of time and the amount of projects to make for other course, we were unable to make some extensions for the game as we barely finished the main tasks of the project. We still wanted to make the Map Random generator but like we said previously : not enough time...

5 Interoperability

To test the interoperability, we decided to train our players using strategies made by other groups (such as Player018FirstLine / Player018Tactical, Player059Tactical, Player006DefendAndTrack, ...). Thanks to this method, we were able to know the code more and how it works, we were also able to correct some issues we had with our own Player017Random.

Even tho our Player017Random was not fully operational, we decided to share it in the SINF Discord Server, we unfortunately don't know who decided to use it as we didn't have any feedbacks.

6 Conclusion

Due to a lack of time and due to the amount of other projects to be made, we were unable to satisfy all the requests to make this project complete. However, we really enjoyed doing this project and are thinking about making it as a hobby after the exams period.