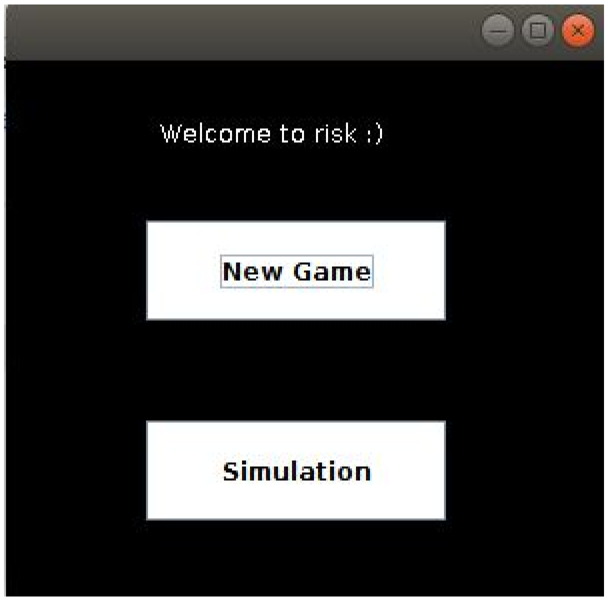
Risk Game

Asmaa Salama	ID: 4443
Menna Ghanem	ID:3798
Esraa El-Hawash	ID:3959
Guehad Mohamed	ID:3861

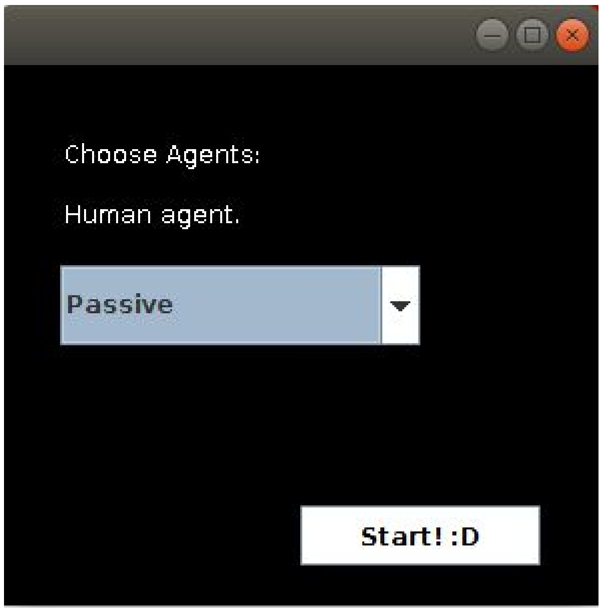
Risk is a popular game for major strategy type players. In Risk, the objective is to conquer the world by attacking to acquire territory and defending your own territory from your opponents.
We were asked in this assignment to handle only reinforcement and attack part in the game .
The game will be played in two modes :

a) Play mode : the case where a human plays against one of the 7 CPU agents (4 of which are AI agent)
b) Simulation mode: the case where no human is involved in playing but if chosen this mode is to show two CPU agents playing a game together

First thing when we run the program the GUI starts by showing us this menu:

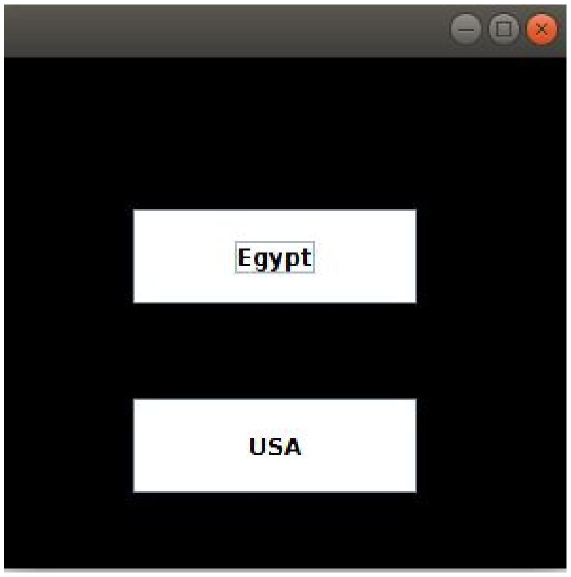


After choosing the mode the user wishes to play in
In case of a New Game
This shows up:



A dialogue box drops showing the options for all 7 CPU agents
(Passive-Pacifist-Aggressive-Greedy-Astar-RealTimeAStar-MinMax)
And the player gets to choose which of these agents he wants to play against.
After choosing the agent
Another menu pops up asking the user to choose the map on which he wants to play

We were given two options : map of Egypt - map of USA



In case of Simulation , the exact same thing happens only 2 dialogue boxes appear for the user to choose 2 agents to play against each other.

RiskMap

Risk map is a class that is considered the base to all our work
In this class we initialize two hashmaps
One called armies and the other called owners
These hashmaps have an object of country as a key and an in indicating number of armies in one case and a string indicating the owner of said country [player1-player2] in the other case.

Playing Agents:

For this part the first thing we did is an interface called “Player” that has 2 methods initialized in it : Reinforce() and Attack()
This class is implemented by each and every player class
With those methods overriden in the respective playing classes
Next thing was a class called “Player Factory” in which an object is created of each class according to the type of agent chosen by the user in the dialogue boxes using if statements.
Passive Player class is sent the playerName string which is the name of the player playing as passive , which is fixed as player2 in human mode and could be either player1 or player2 in simulation mode
The passive player reinforces only and doesn’t attack
So inside the reinforce method of the passive player class what happens is we loop on the hashmaps and check the countries belonging to this player that have the least armies and distribute our bonus armies on those territories and that’s all the passive player does.

The bonus armies are calculated the same way in all classes:

The number of territories of a player is counted and the number is divided by 3 if the result is less than 3 then 3 troops are given as bonus , if the result is more than 3 then the result count of troops if given.

In the case of Pacifist Player class

The Reinforce function stays the same

But this time an Attack() method will be called

In the attack method what happens is:

A loop in the hashmap for all my countries looking for neighbors that aren't mine that the pacifist agent can attack meaning the agent's armies are more than the enemies , and it puts in the min value .

The algorithm keeps looping till it finds the minimum value it can attack and once it finds it , an attack happens and the result of the attack are adjusted in the hashmaps again

The new owner of the attacked country is changed and so is the distribution of the troops according to each attack

We distribute the troops of the attacking and attacked country with a ratio of ⅔ of troops for the newly received country and a ⅓ for the one the agent already owned.

For the Aggressive agent:

The same concept applies with a few changes

For the reinforce it looks for the country with the most troops and keeps adding the bonus armies to it

For the attack it uses the same attack as the pacifist agent but it's repeated multiple times because the aggressive has the ability to attack more than once

Now for the AI Agents:

For the AI agents a heuristic was needed

We thought of the heuristic as 'BSR' which stands for Border Security Ratio

Which is a ratio calculated by BST(border security threat)/number of units
Meaning it calculates how a certain country is threatened by countries around it
BST is the sum of troops of opposing neighbors of the certain country.
Number of units is the number of troops of the country we're calculating BSR for.
This whole calculation is implemented in class "Heuristic" which is used by all 3 AI agents.

Greedy player
For reinforcement the greedy algorithm looks for the countries it owns that are most threatened by the enemy (also calculating the BSR) and supplies that country with troops , not only that but it also looks for the next most threatened country and puts some armies there too. So it can protect it self from the enemy attacks
The greedy player follows the heuristic by calculating the maximum BSR in finds in each turn and attacks the country with the maximum BSR that it can attack(it's armies are larger than that enemy country)
And greedily attacks the highest threatened country it finds till it reaches the goal state and wins all countries on map.

Astar Player:
The reinforce for the Astar is done the same way as for the greedy.
The attack however needs a G(n) [Cost] to find the optimal path to take
Uses the same heuristic but the only difference is a cost is added. The cost in our case is the number of moves made by the class to win the game in other words the more moves i make the more countries i attack the more territories i own , which takes me closer to my goal of world domination and owning all countries

The optimal solution would be to make the most amount of moves and to have the most possible BSR meaning to be be as much of a threat on the enemy as possible .
This class creates a tree and handles all operations using states, there’s an initial state, a goal state, a parent node and leaf nodes .
Real-Time Astar:
For the real time Astar almost the exact same implementation for the Astar class with the only difference of a limitation in algorithm
The path we look for doesn’t make it all the way till the goal node , it only looks for a path at a certain depth or limit that we specified in the algorithm

Lastly and Evaluation Class:
In this class we calculate the evaluation of all 3 AI agents using the function mentioned in the PDF
P=f*(L+T)
L= number of turns to win the game
T= number of search expansions
f= weight (1,100,10000)
In this class 3 functions are called to calculate the Performance using the 3 weights for each called agent.
The function is called from within player classes after all their attacks and reinforces are done , the number of turns is a counter as to how many times the reinforce, attack methods were called
And the search expansion is calculated depending on the agent playing
The player class sends both T and L to the evaluation class where the performance is calculated with the 3 weights and printed in the console.

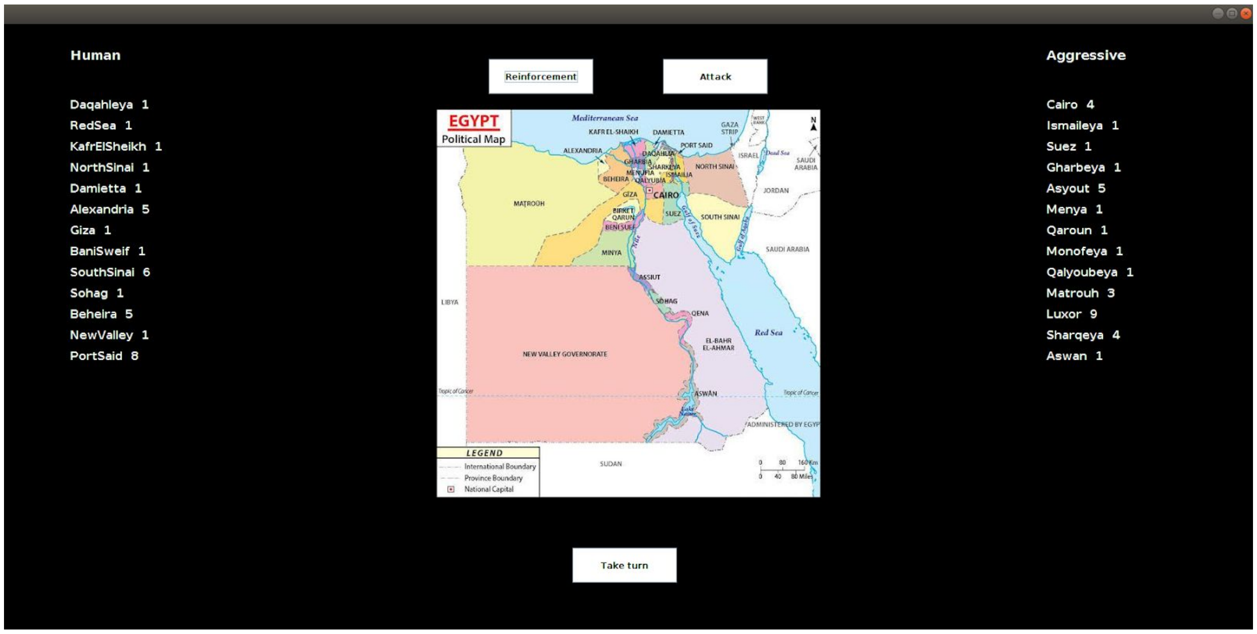
Here are some sample runs of how to play:

At the start of the game after choosing the map you want to play on

All the countries in the map are split equally and randomized between the two playing ends of the game .

An amount of 20 armies is randomly distributed on all territories as seen in the picture below

This is the initial state of any game.



Human		Aggressive	
Daqahleya	1	Cairo	4
RedSea	1	Ismaileya	1
KafrElSheikh	1	Suez	1
NorthSinai	1	Gharbeya	1
Damietta	1	Asyout	5
Alexandria	5	Menya	1
Giza	1	Qaroun	1
BaniSweif	1	Monofeya	1
SouthSinai	6	Qalyoubeya	1
Sohag	1	Matrouh	3
Beheira	5	Luxor	9
NewValley	1	Sharqeya	4
PortSaid	8	Aswan	1

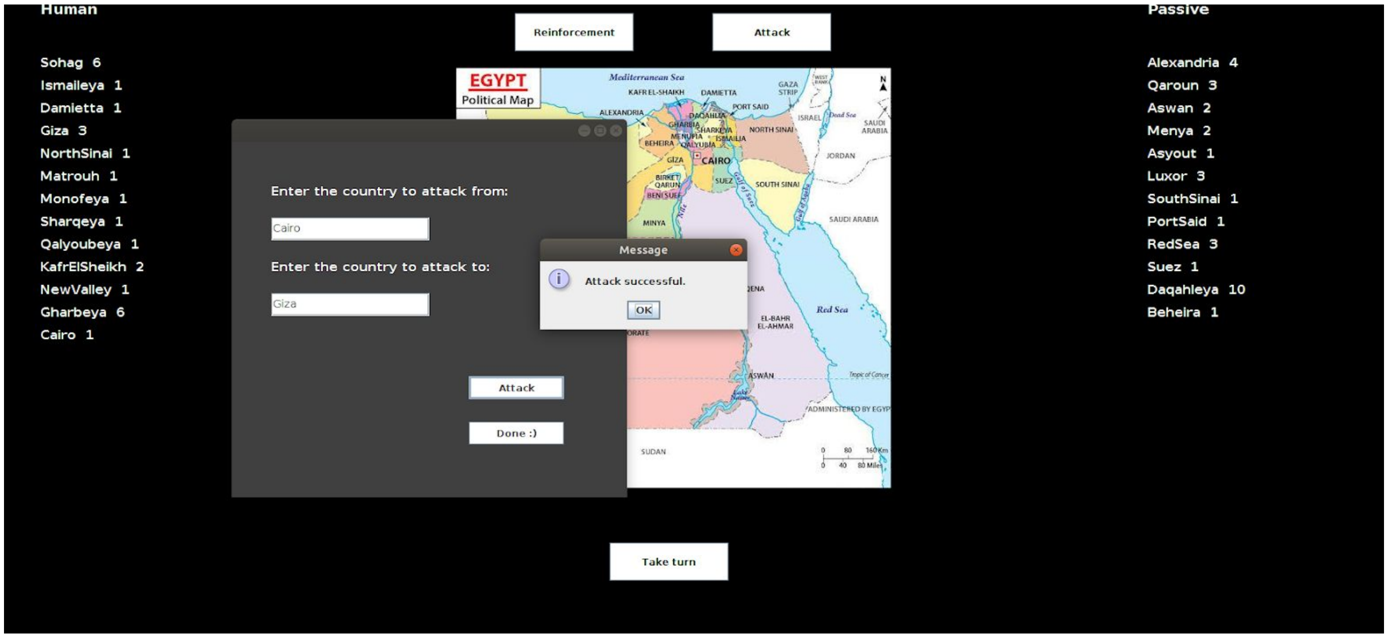
Here is a sample run of a human playing against the Aggressive agent on the Egypt map

How a human plays :

You click on the Reinforce button and a window will show up with your bonus troops for this turn , the user should then choose the number of armies they wish to add , and choose the country on which they will add chosen armies .

And after the user is done reinforcing

Click the button Attack , another window will show looking like this :



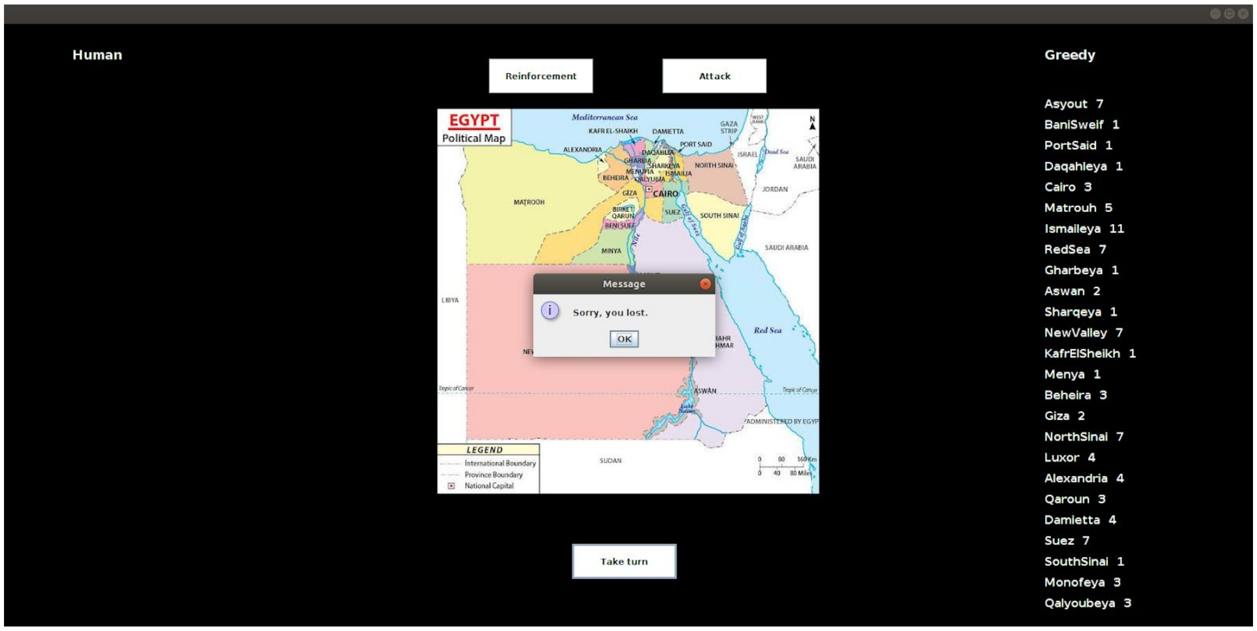
```
Human                    Passive

Sohag 6                  Alexandria 4
Ismaileya 1              Qaroun 3
Damietta 1               Aswan 2
Giza 3                   Menya 2
NorthSinai 1             Asyout 1
Matrouh 1                Luxor 3
Monofeya 1               SouthSinai 1
Sharqeya 1               PortSaid 1
Qalyoubeya 1             RedSea 3
KafrElSheikh 2           Suez 1
NewValley 1              Daqahleya 10
Gharbeya 6               Beheira 1
Cairo 1
```

```
Enter the country to attack from:
Cairo
Enter the country to attack to:
Giza
```

```
Message
Attack successful.
```

you choose one of your countries on the map and write it down , and then chose which country you wish to attack on, if the attack is valid , a message will show indication the attack was successful and the attacked country will be added on you side of the window.

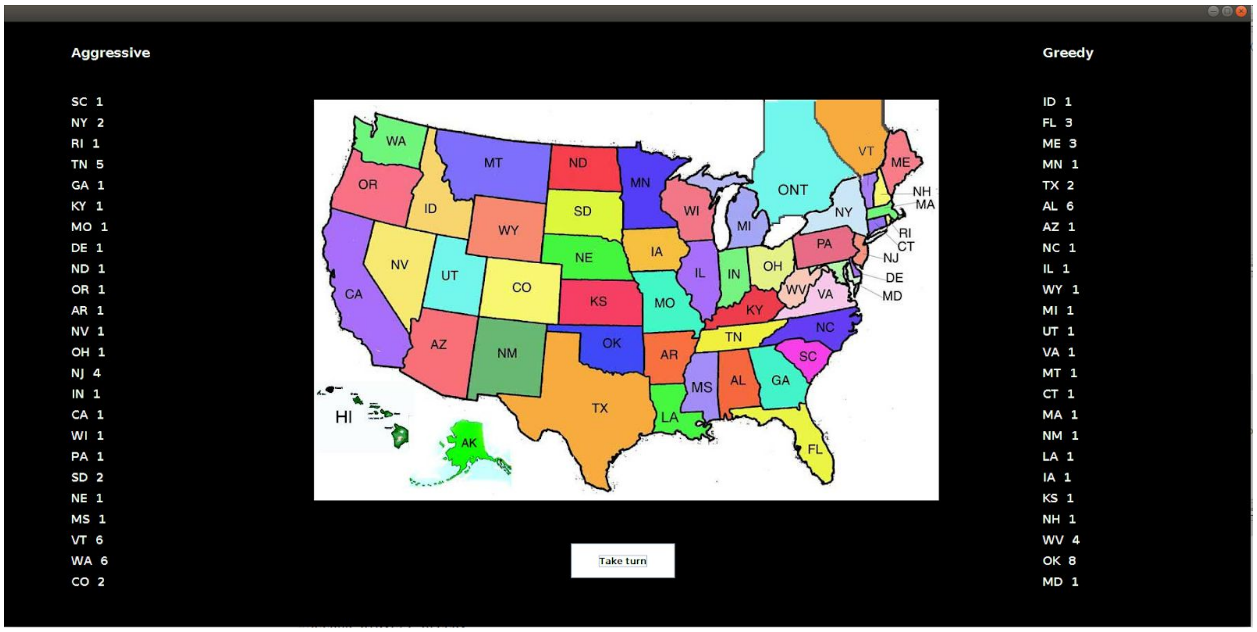
Here's a sample run of the greedy algorithm winning against a human :



```
Human                    Greedy

                         Asyout 7
                         BaniSweif 1
                         PortSaid 1
                         Daqahleya 1
                         Cairo 3
                         Matrouh 5
                         Ismaileya 11
                         RedSea 7
                         Gharbeya 1
                         Aswan 2
                         Sharqeya 1
                         NewValley 7
                         KafrElSheikh 1
                         Menya 1
                         Beheira 3
                         Giza 2
                         NorthSinai 7
                         Luxor 4
                         Alexandria 4
                         Qaroun 3
                         Damietta 4
                         Suez 7
                         SouthSinai 1
                         Monofeya 3
                         Qalyoubeya 3
```

```
Message
Sorry, you lost.
```

This is a sample run of a simulation , Aggressive and Greedy playing against each other , the user controls when each algorithm gets to play by clicking the button “Take Turn” , every time the button is pressed the turn is alternated between the two CPU agents, until one of them wins.



```
Aggressive                                   Greedy

SC 1                                         ID 1
NY 2                                         FL 3
RI 1                                         ME 3
TN 5                                         MN 1
GA 1                                         TX 2
KY 1                                         AL 6
MO 1                                         AZ 1
DE 1                                         NC 1
ND 1                                         IL 1
OR 1                                         WY 1
AR 1                                         MI 1
NV 1                                         UT 1
OH 1                                         VA 1
NJ 4                                         MT 1
IN 1                                         CT 1
CA 1                                         MA 1
WI 1                                         NM 1
PA 1                                         LA 1
SD 2                                         IA 1
NE 1                                         KS 1
MS 1                                         NH 1
VT 6                                         WV 4
WA 6                                         OK 8
CO 2                                         MD 1
```