Codeproject Mastergrab

**descriptive title**

MasterGrab: PC strategy game, easy to learn, fun to win. Also write your own C# game robot !

**1-line description**.

Do you need a 10 minute break or want to do some mental warmup before programming ? This free game is for you.

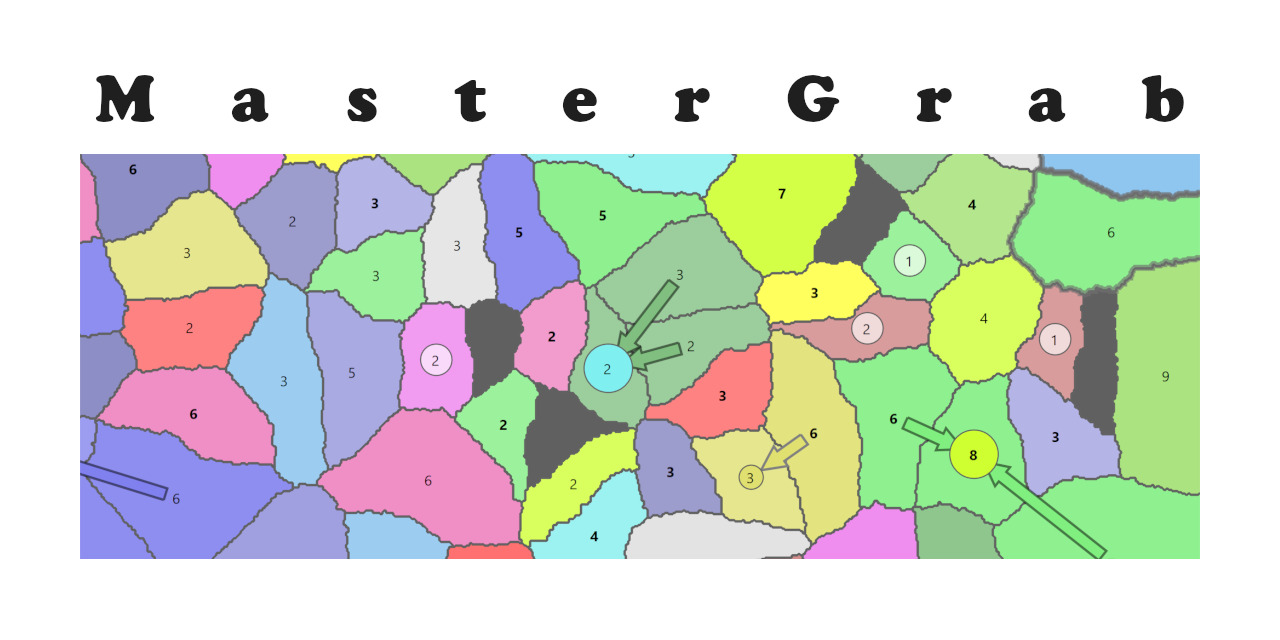
**Tags**

WPF, C#, game,

**1-paragraph abstract**

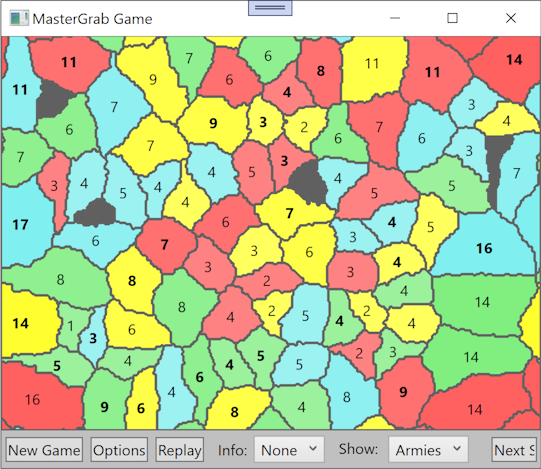
MasterGrab is an open source, turn-based Windows strategy game where a human player plays against several computer players (=Robots), trying to grab all countries on a random map. The game is quite fun, because the random map poses different problems every time and the Robots, who also fight each other, make MasterGrab a dynamic play. For programmers, it might be even more fun to write your own Robot, which can be done with only a few lines of C#.

**article text**

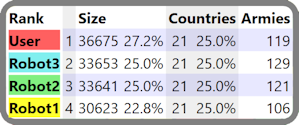


## Game Overview

The MasterGrab game shows by default a map with countries for 4 players. 3 players are controlled by the computer, the countries with the red colour are owned by the human player (you). The goal is to grab all countries.

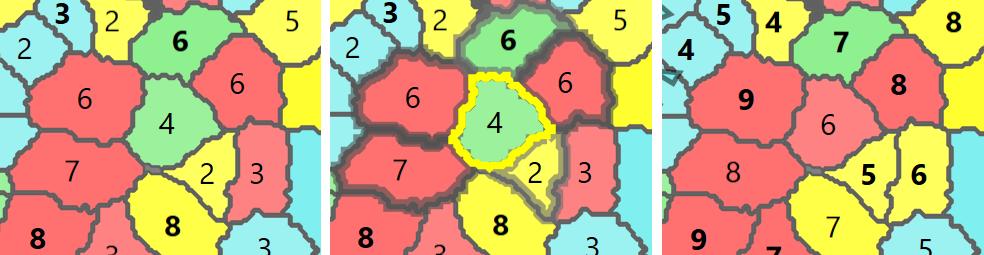


You own the countries in red. In the Info Dropdown you can click on Ranking to see the statistics about the different players:



The total size (number of pixels used on the screen) of all your (User) countries is usually slightly greater than the other players (Robots), which are all played by MasterGrab. The bigger the country, the bigger the army (number in the center of a country) that can be hosted by a country. This number gets randomly created at the beginning, but then grows a little after each move, until it reaches the country's maximum capacity, which is marked by bold numbers.

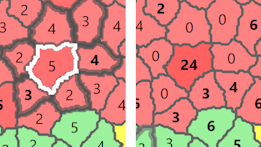
You can attack any enemy country which is next to a country you already own:



In the right picture, your red countries with the numbers 7, 6 and 6 can attack the green country with the number 4, because they are all neighbours.

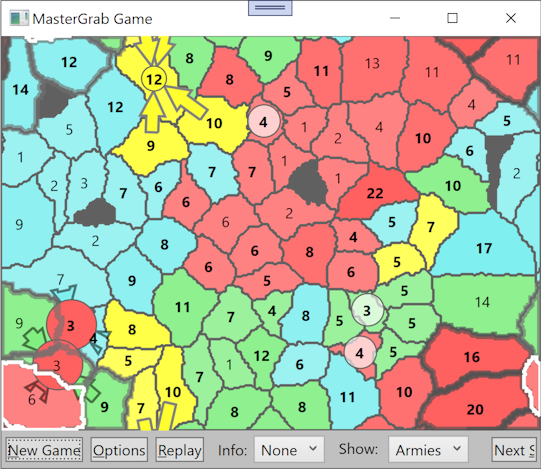
You can easily find out which are the neighbours of a country by hovering the mouse over it. Like in the middle image, the green country with the mouse over it gets a bright border, while its neighbours get a dark border.

You can attack the green country which has 4 armies. You have 3 countries bordering the attacked country with a total of 19 armies. In order to win an attack, the attacker's armies need to be a little more than twice as numerous as the attacked country. Here the attacking countries have 19 armies, which wins easily over the 4 defending armies. On the left picture you can see that the green country is now red, because you won it. When winning a country, you get some additional armies, which get distributed over the attacking and won country according to their size. Note that the numbers in all countries have changed, they grow slightly after every move until they have reached the countries capacity. Also note that some other countries have changed their color, because the robots were fighting each other too.



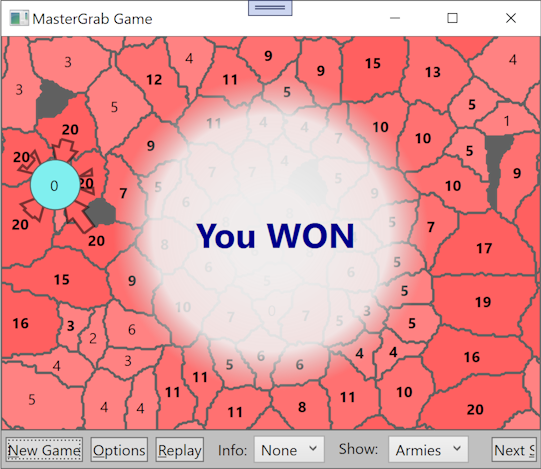
You can also click on your own country, in which case the armies from your neighbouring countries get moved into the clicked country. The countries which do not have an enemy as a neighbour give all their armies, while your countries with enemies keep some soldiers for defence. Note that the clicked country can normally host at most 7 armies through natural growth, but it has now 24. You can move them with your next moves to an enemy for an attack. Or you just leave it at your border, which will prevent your enemy from attacking you and he will attack another robot instead.

Back to the overall game explanation. After you have made a move, all the robots make a move and the armies in every country grow slightly. It is a good strategy trying to win some countries next to each other:



Note that the colored arrows show the moves the robot made after your last move. The big circled numbers show the result of a battle. The small circled numbers mark the smallest countries, which are easiest to attack.

The game is over once the human player has lost or won all countries:



As you can see, the rules are quite easy to learn. The game is fun because:

* It is always different, since the map is randomly created
* It is livelier than games with just one opponent, like in chess, especially since the robots attack also each other
* You need to develop your own strategies how you can make the robots attack each other instead of you
* The robots are not too clever, meaning most random games can be won, although one might need several tries.
* The robots are not too stupid, it can be a challenge to beat them all.
* You can choose how many countries and robots you want, which allows you to choose between a quick or longer game.
* It's a turn based game, you can think as long as you like. The robots make their move nearly instantly, no waiting time for you.
* **Last, but not least, you can develop your own robot in C#, which you can do in a few days.**

## Getting the game

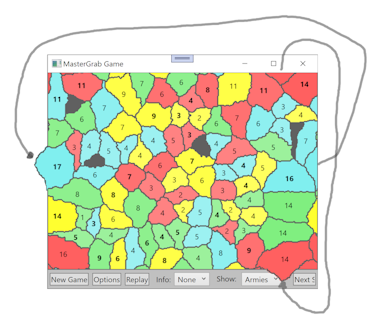
The easiest way is just to get the complete code from Github:

<https://github.com/PeterHuberSg/MasterGrab>

You need the code if you want to develop your own robot or if you just want to look at the source or even better, improve the game.

If you just like to play the game, you can also download a zip file from CodeProject, which contains MasterGrab.exe. Unpack the .zip in a folder and double click on MasterGrab.exe to start the game.

## The Map



The map itself has no borders, like the earth orb, which also has no edge. The countries at the left border of the window wrap around to the right window border and the countries at the top border continue at the bottom border. Interestingly, the 4 map corners are actually touching each other. The benefit of a map without any borders is that no country gets the advantage of being a border country, i.e. it would have fewer other countries attacking it.

## Mastergrab Window

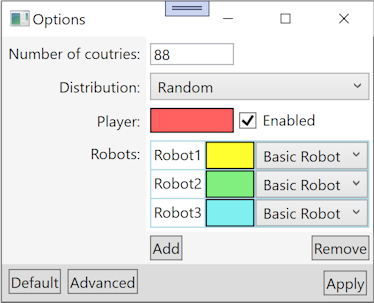
If you resize the MasterGrab window, start a New Game and the map will fill the new window size properly.

On purpose, it is not possible to undo a move, otherwise winning becomes too easy. But if you made too many mistakes, you can click Replay to start the same game again.

If you don't want to make a move (click on any country), you can click on Next Step which asks the robots to make each one move.

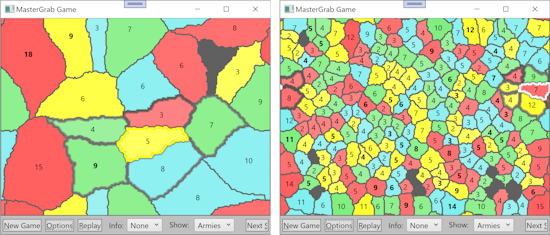
## Options

When starting MasterGrab, it automatically generates a new game using default options. You can change them by clicking on the Options button:



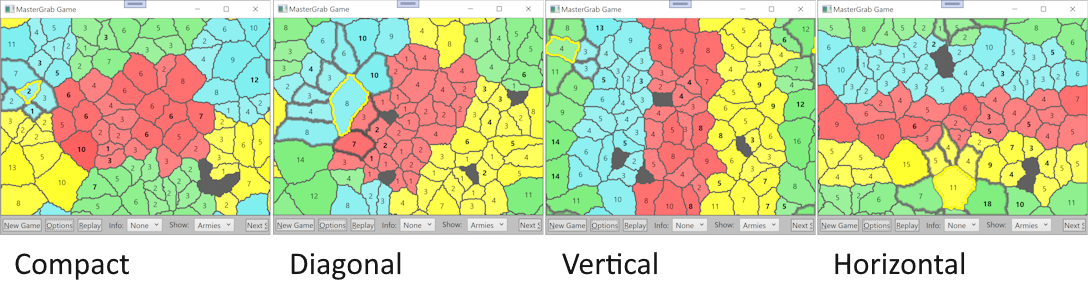
### Number of countries

Here you tell MasterGrab how many countries should be created in the map. It can be few dozens or hundreds:

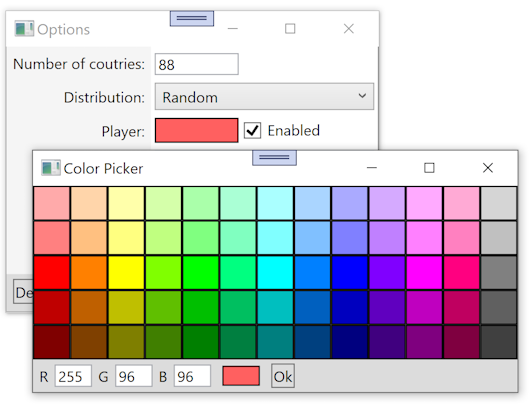


### Distribution

Per default the owners are assigned randomly to the countries all over the map. This is the easiest mode to play, since the robots play better when the countries of one robot are next to each other, like in the following 4 distributions:



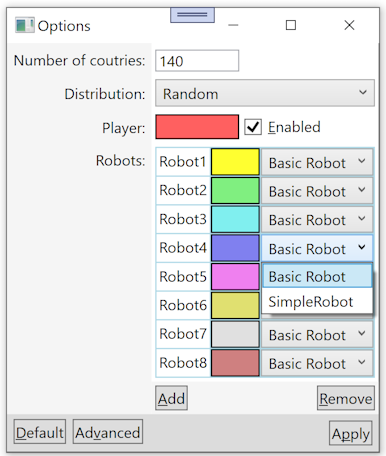
### Player



You can change the colors of your countries by clicking on the colored rectangle, which opens a color picker.

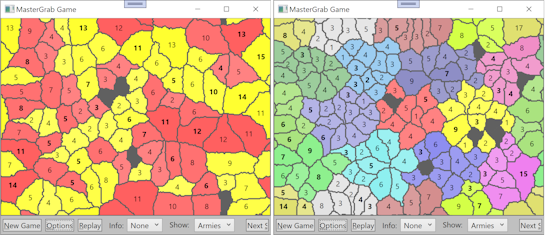
If the Player is not enabled, only robots are playing. This can be fun to study the robots' strategy, or, if you write your own robot, to see how it behaves. The robots only make a move if you click on the Next Step button or if you choose in the DropDown Autoplay how often the robots should make a move.

### Robots



Clicking on the Add button allows creating up to 15 robots, Remove removes them. With the dropdowns on the right you can choose which type of robot you want to use. Presently there are only two. Basic Robot is ok, Simple Robot is just an example how the code of a very simple robot can look like, which is a good starting point for writing your own robot, which you then can also add to this dropdown. My hope is that over time other people will share their robots on Github.

Maps with 2 and 15 robots:



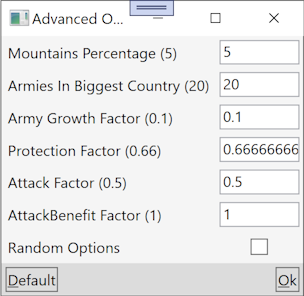
The more robots you use, the more countries you should produce. Playing against just 1 robot is challenging because his whole focus is on attacking you, in which he quite excels. Playing against many robots is fun because you have to beat first your neighbour robots, while the robots further away battle each other and some of them can get quite big. To win the game, you have to beat also the big robots.

As a beginner, it might be helpful to start with only 1 robot and 10 countries. It's a good way to explore what effect your attack on a country has. The consequences might not be that clear with more countries and robots.

### Option Window Buttons

**Default:** Fills the Options Window with default values: 140 countries, random distribution and 3 robots.

**Advanced:** There is actually a second window with advanced options. Check it out once you have some experience with MasterGrab:



Hovering your mouse over the option TextBoxes shows you a ToolTip explaining what the options do.

It might be fun to enable Random Options, which will set random values for all options. Once it is enabled, click on Ok, which will close this window and show the Options window. There you click on Apply and MasterGrab shows a readonly Advanced Options window with the new random values, which you should know when playing that game.

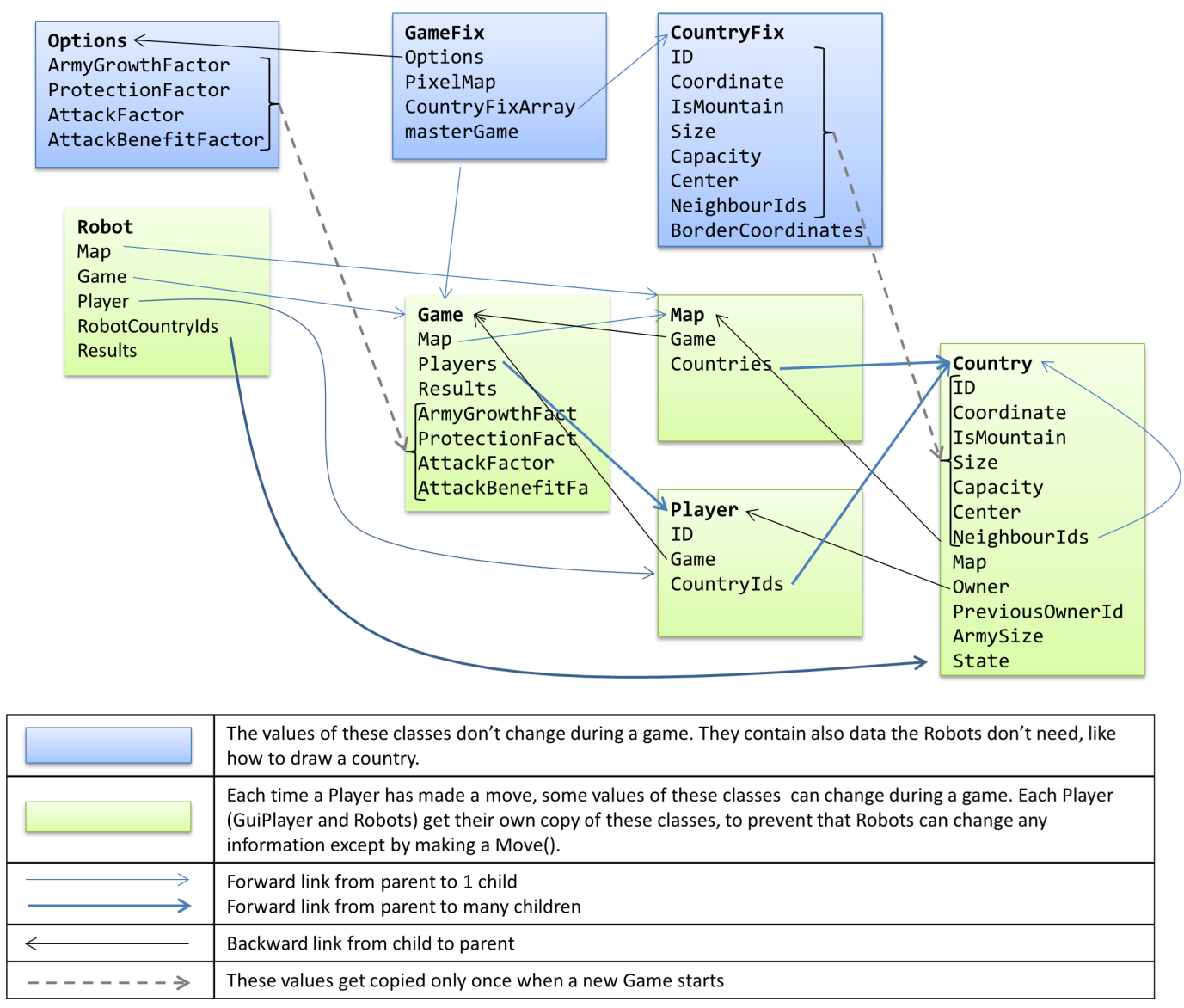
**Apply:** Clicking on Apply closes the Options Window and starts a new game with the options.

## Programming your own robot

Basically, you have just to write a new class inheriting from Robot and place it into the Robots project. When MasterGrab starts, it searches for these classes and displays them in the Options window, where the user can select them for the next game.

### Classes involved in a Game

The game is managed by the GameController. The user can select in the Options Window which robot types and how many of them should play, how many countries should be on the map and other settings. When the user presses New Game, the options get sent to the GameController, which starts a new Game. It waits first for the user to make a move, then it asks each Robot to make a move (DoPlanMove()). The robot gets a copy of the latest game data, like which Country is owned by which Player (human or robot) and what is the size of the army in that Country. The Robot cannot change any of this data, that would open the door to cheating. The Robot can only tell the GameController which is the next move it would like to make.



Some data changes during a game, like who owns which country and other data does not change during one game, like which countries are neighbours (Options, GameFix, CountryFix).

Options contains all the information needed to create a new game,

Whenever a robot can make a move, it gets the Robot instance which represents him, holding all the information a robot needs, like RobotCountryIds which is a collection of all the countries (actually the countries' IDs) he owns.

Player has a Game property, which is actually the root of the class tree. Map is a special collection holding all the countries.

Game properties

Map is a collection of countries

Players allows the robot to investigate the data of all other players

Results shows the move of every player in the last round and if it was a success or failure.

AttackFactor is important for a robot to decide if he can attack an enemy country. He first adds up the army sizes of his own countries neighbouring that enemy country. That total gets multiplied by AttackFactor, which is smaller than 1. If the result is greater than the defending army's size, the attack wins.

Player properties

The robot's Player instance also holds CountryIds with a list of all countries that player (robot) holds. Note that these are only unique country ID numbers, which can be used to get the country data like this:

var country = Map[countryId];

Country properties

Some Country properties are not interesting for robots, like its coordinates on the screen. Country properties used by robots to calculate their next move:

NeighbourIds list the neighbour countries' IDs.

ArmySize number of attackers in own countries or defenders in enemy countries.

Code of a very simple Robot

One can use SimpleRobot.cs as starting point for your own robot:

using System.Collections.Generic;

namespace MasterGrab {

/// <summary>

/// Simple Robot player using only 1 country to attack

/// </summary>

[Robot(name: "SimpleRobot", isUsedForDefault: false)]

public class SimpleRobot: Robot {

Any namespace can be used. Important is that SimpleRobot inherits from Robot. Using the RobotAttribute is optional. It allows changing the class name to a nicer name seen by the user and to indicate if that robot should be used as default options.

Robot provides SimpleRobot easy access to Player (the robot itself), Game, Map, RobotCountryIds (owned countries) and Results.

readonly List<Country> attackers;

Note that a robot gets for each move completely new instances of Game, countries, etc. For example, it is not possible to reuse a country instance from a previous move. However, Country.ID and Player.ID are constant. So if a robot wants to store information about another player over several moves, he must combine that player's ID with the rest of the data he wants to store.

The variable attackers is not used to store information between 2 moves, but for efficiency reasons it gets created only once and then constantly reused.

public SimpleRobot(): base() {

attackers = new List<Country>();

}

GameController calls DoPlanMove() when it is time for the robot to make the next move. Here is where you place the code for your robot.

protected override Move DoPlanMove() {

Country? attacker = null;

Country? target = null;

double ratio = 0;

Strategy used by SimpleRobot:

loop through all countries this robot owns

find the best neighbour to attack. SimpleRobot tries to optimise:

size of the neighbour country (the bigger the better)

how many defenders (the fewer the better)

loop through every country the robot owns

foreach (int countryId in RobotCountryIds) {

var country = Map[countryId];

calculate how big is the attacking army

double attackingArmies = country.ArmySize \* Game.AttackFactor;

loop through every neighbour of this country. Remember the weakest enemy country compared to your attacking country.

foreach (int neighbourId in country.NeighbourIds) {

var neighbour = Map[neighbourId];

if (

!neighbour.IsMountain && //cannot attack mountain

neighbour.OwnerId!=Player.Id && //don't attack own country

neighbour.ArmySize<attackingArmies) //is robot strong enough ?

{

//calculate how much stronger the attacker is then the defender

double newRatio = attackingArmies / neighbour.ArmySize \* neighbour.Size;

if (ratio<newRatio) {

//found a weaker country to attack

ratio = newRatio;

attacker = country;

target = neighbour;

}

}

}

}

Decide what will be your move after looping through all your countries.

Move move;

if (attacker==null) {

If a robot decides not to attack, he can select one country and move all armies from countries he owns around that country into that country. See BasicRobot.cs for an example of how this can be done.

SimpleRobot decides to skip one move if he cannot attack.

move = Move.NoMove;

} else {

attackers.Clear();

////SimpleRobot is a bit stupid and uses only 1 of his countries to attack

////It would be better if he would use all of his countries which are neighbours of the attacked countries,

////like this:

////foreach (int targetNeighbourId in target.NeighbourIds) {

//// Country targetNeighbour = Map[targetNeighbourId];

//// if (targetNeighbour.OwnerId==Player.Id) {

//// attackers.Add(targetNeighbour);

//// }

////}

attackers.Add(attacker);

move = new Move(MoveTypeEnum.attack, Player, target!, attackers);

}

return move;

}

}

}

SimpleRobot is kept on purpose to the bare minimum. A bit more sophisticated is BasicRobot, which attacks an enemy country with all his own countries being neighbours of the enemy and when there is no good attack to perform, it moves his own armies neighbouring one of his countries into that country. This is rather important in the later phase of the game.

However, BasicRobot is lacking any strategic planning, like:

Detect attacks and defend against them

Once reaching a certain size, also grow armies in different countries instead of attacking as much as possible

Which attack improves the "shape" of his countries:

When his countries bild a "circle", the countries inside the circle have no enemies.

When he manages to get all countries in a column on the screen, less defence is needed.

Taking into account which is the weakest and strongest opponent

Preventing a strong neighbour from attacking by increasing the armies at the border

To tempt 2 enemies to attack each other: If the Robot owns a country surrounded by 2 enemies, that country prevents the enemies from fighting each other. If the Robot's country attacks an enemy country, it will most likely lose and the 2 enemies will start to attack each other.

When you play MasterGrab yourself, you will discover also other strategies, which you can then teach your Robot.

To my surprise, even BasicRobot has no strategy, it is actually difficult to beat. Meaning it should not be that hard for you to write an even better Robot.

## Known problems

There is one bug in the random map creation which I could not solve. Luckily, this happens very seldom, in which case just a new random map gets created. I will write another CodeProject article about the MasterGrab design where I will describe the problem in more detail.

I noticed that 2 robots keep attacking each other with exactly the same countries and stop reacting to what was played. Again, this happens very seldom, usually when I have a country with a really big army. I usually help them by attacking one of the countries in that fight, Luckily, also this problem happens very seldom.

## I need your help

I have played MasterGrab since many years every day, because it is fun and engaging. I would like to share it with other people, but I don't know how to do that. Can you share the game with friends and, even better, advise me which other websites I can use to make the game more popular ?

And, if you do improve the code or add more functionality, can you share it with us on Github ?