|  |  |  |  |
| --- | --- | --- | --- |
| Indexes  **267007**  **?????**  **?????** | Names  **Bartłomiej Cerek**  **Kseniia Sych**  **Vlad Leonov** | Field of study  **Computer Science SMaDA** | Department  **WEAIiIB** |
| Subject  **Agent-Based Modelling** | Topic  **Agent-Based Model of Battle** | | |

### 

### Introduction

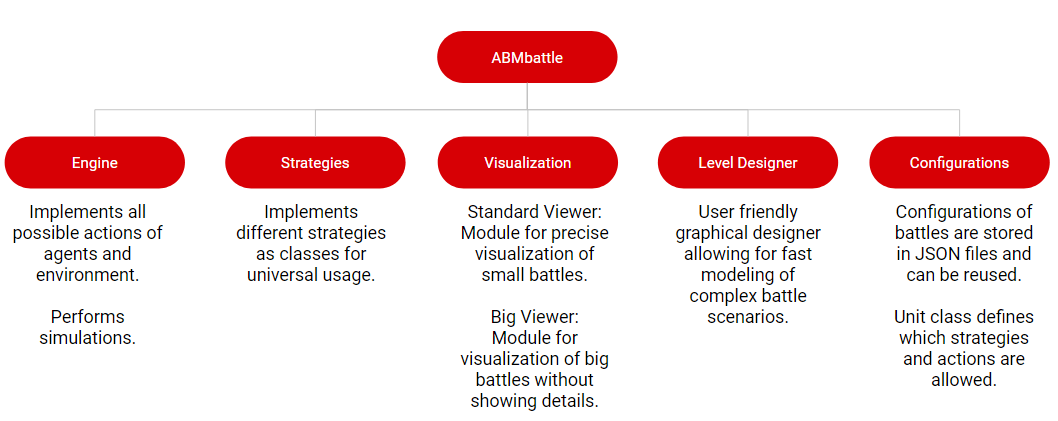
The goal of this project is to create an Agent-Based model of battle, without using any specific modeling framework. The main requirement is scalability, developed software should allow the user to model any kind of simple battle scenario, giving flexibility when it comes to the number of armies, types of units and properties of the environment. As a benchmark for accuracy and testing purposes Battle of Thermopylae from 480 B.C. was selected. It was simulated with different configurations and resolutions. The project was realized using Python programming language with an object-oriented approach. Code can be found in the GitLab repository under the link: <https://gitlab.com/bartlomiejcerek/abmbattle>.



*Image 1. The geographical setting of The Battle of Thermopylae.*

### Structure of Project

The project consists of the simulation core and many utilities and tools that allow for configuration and building of models. The structure is depicted in *Image 2*.



*Image 2. Structure of the project.*

**Engine** consist of Engine, BattleField and Unit classes. Instances of those last two represent independent agents and their environment. Engine class modifies their states during simulation by defined rules and used **strategies**.

**Strategies** are classes that implement the *make\_move* method. Whenever the Agent wants to perform the move, the engine passes the current state of the simulation and agent ID to the strategy that this agent implements. Based on this data decision which move, from possible, to choose.

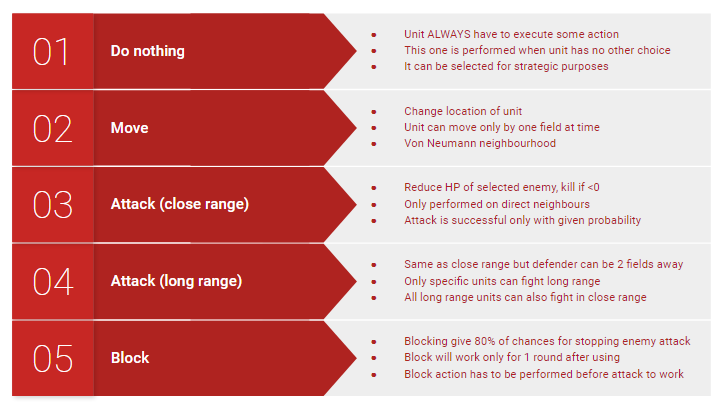
**Visualization** modules allow for creating graphical representations of simulated battles and latter saving them as GIFs.

**Level Designer** is a tool for creating simulation scenarios and saving them to files. It is crucial as more complex simulations require a lot of time to model.

**Configurations** are files and classes that allow storing, changing and remodeling created simulations. Same as level designer those are necessary for the robust workflow with simulations.

### Agent Actions

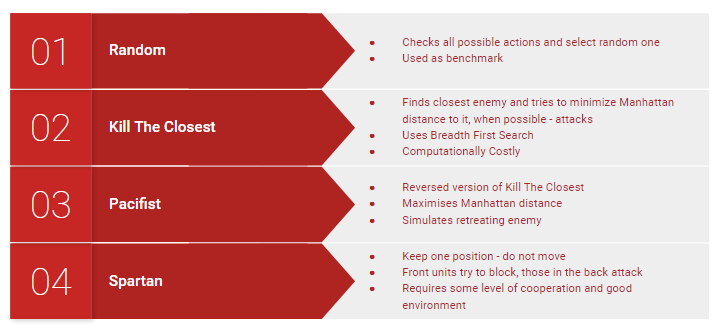
In the agent-based system, one of the crucial elements to define is a list of actions that agents can perform. Actions allow for interacting with the environment and other agents. In battle scenarios, the most important issues for units are moving and fighting. All actions are implemented as methods of class BattleField, as the battlefield is composed of environment and units. Performing those changes the inner state of the battlefield but also units (for example reduces their HP).



*Image 3. List of actions that can be performed by agents.*

### Strategies

The key element of simulations in this project are strategies. Similar units can behave completely differently depending on their strategy. Each unit has one strategy associated with it, however different units in one team can use different strategies.



*Image 4. List of strategies that can be used by agents.*

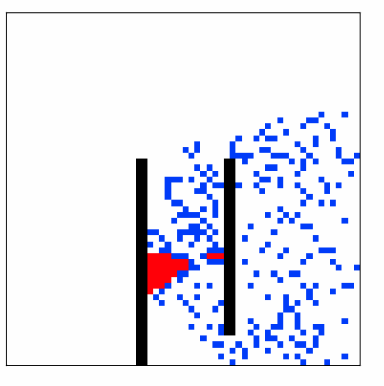
### Level Designer and Visualization

### Level Designer and Visualization modules played a crucial role in analyzing simulation results and creating models. Graphical representation allows not only for faster work but also for better reasoning. Seeing in what patterns units move or which moment of battle was crucial for the outcome allowed for creating better next simulations.

*Image 5. Examples of visualization.*

### Results and conclusions

The final version of the Agent-Based Modelling of Battle allowed us to create versatile simulations and scenarios, most notably Battle Of Thermopylae. The well-tuned model behaves as the most commonly accepted historical description. Without breach in Spartans’ formation, only a great number of Persians had chances for winning. However, in simulation taking into consideration treason, and giving Persians the ability to surround Spartans their loose was inevitable.



*Image 6. A simplified view of Persians breaking Spartan formation.*

Conclusion about Spartan strategy is quite obvious, it is efficient only in a very propitious environment. The less straight forward result is that ‘*Kill The Closest*’ strategy turned out to be way worse than expected. It feels quite natural to treat each unit completely independently and let it focus on the closest enemy. It supports the independent agent paradigm, creates a natural setting of not-strictly-directed battle which was a case in ancient times and even works very well in small simulations. The problem arises in massive battle simulations (+100 units) when many soldiers focus on one enemy and constantly block movements of their fellows. This underlines one more advantage of a simple version of the Spartan strategy, even though it is not centrally controlled it turns out to be quite scalable.

It seems that creating a better strategy for Persians would have to include the creation of a supervisor, general that imposes better distribution of attacking units, this however, would break agents’ independence. Improving the existing version of the algorithm to take into consideration the spacing of units would definitely increased Persians efficiency, but for sure not as much as introducing big-scale strategy.