MicroWar

Physical Prototype

Team 4
David Niggli, Marcel Lüdi, Gaetano Paganini, Jonas Krucher

Prototype description

We used Lego for our prototype.

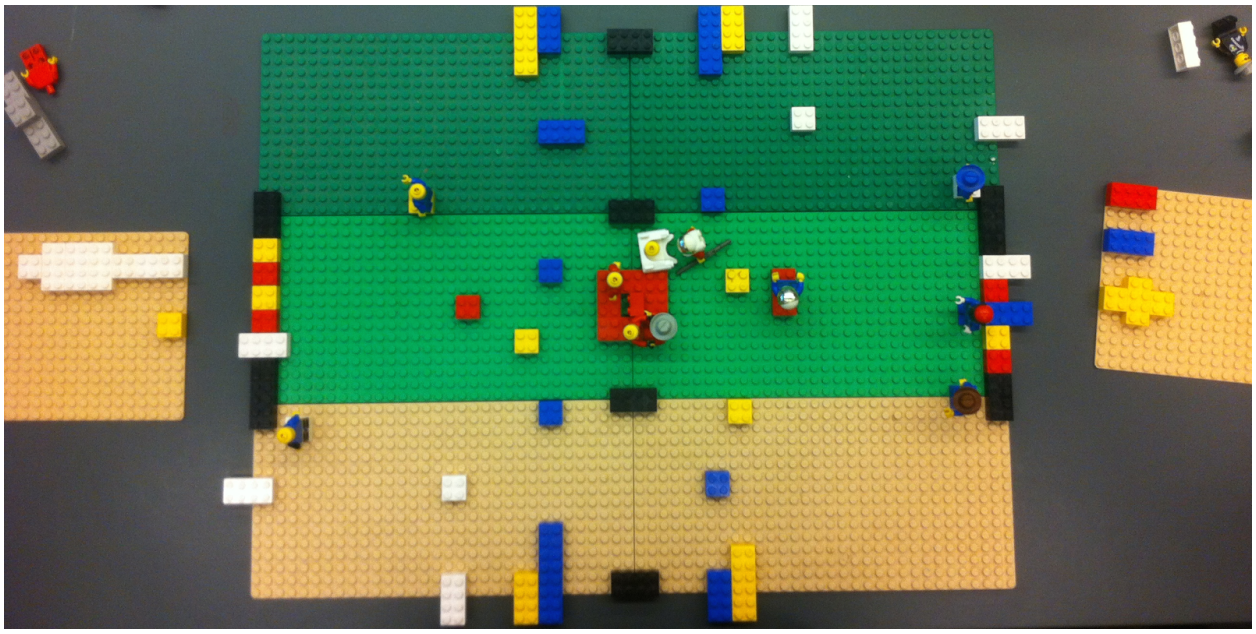

Figure 1: In the middle is the game board with the game elements. On both sides are the resources of the two players.

The game is discretized in rounds.

One round consists of the following actions (in this order):

- Units move with a given speed
- Units act (gather, attack…)
- Resources move
- Resources spawn every second round (altering blue-white/red-yellow)
- Players input (expand/mutate sequence)
- Units spawn (every 5 rounds)

The board consists of the two DNA bases for the players and five spawning pools for the four different types of resources: red, white, blue and yellow blocks.

Resources spawn at certain locations and flow along certain point symmetric paths: blue and yellow resources flow from one side to the other, whereas the white resources flow to the near spot on the same Lego-board.
Blue, white and yellow resources disappear when reaching the other side. The red blocks spawn in the middle and do not move.
Players can trade Resources three to one.

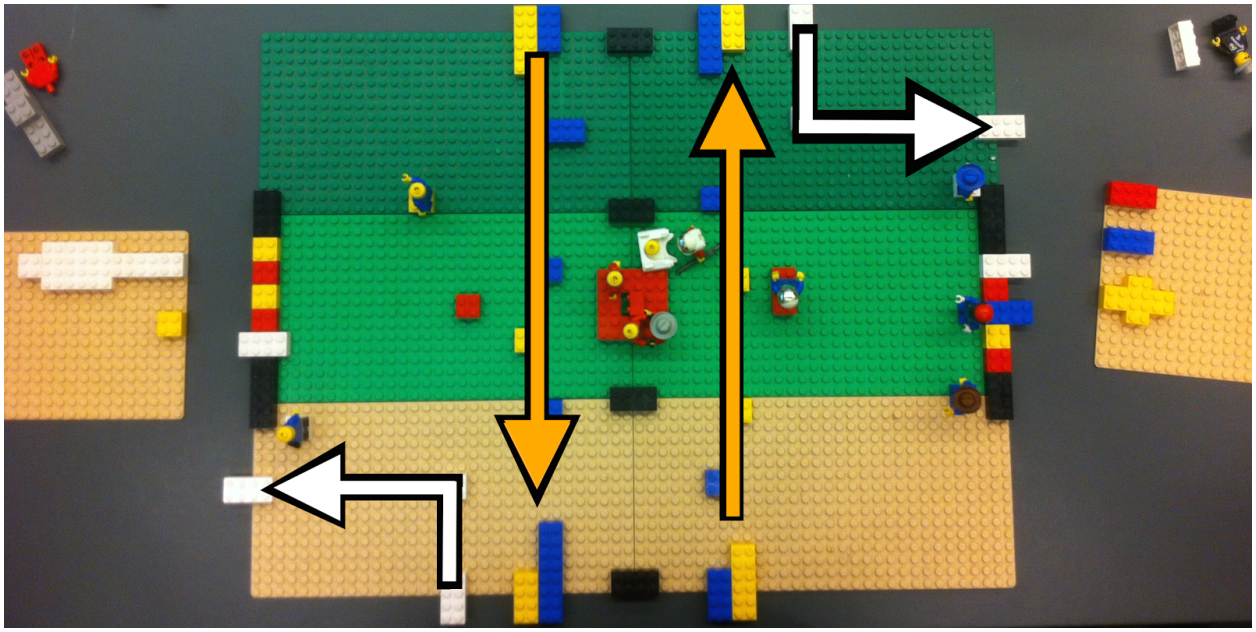

Figure 2: The flows of the white, yellow and blue resources (white arrow: white resources, orange arrow: yellow and blue resources).

With their DNA sequences, the players can code for four types of units: blue gatherers (one types for each resource type, which was binary coded with the characters hands), red attackers, black DNA attackers and white healers. The resources for the units are in figure 3 encoded.
Units can move 10 fields (one field equals one lego “nop”) per round.

There are two types of enhancement for units (achieved through coding suffixes): jetpacks for all units (speed boost) and guns for attackers (more damage).
Units with the faster movement can move 15 fields instead of just 10 and units with stronger attack takes of two parts of a unit it attacks instead of one.

A gatherer moves toward the closest resource of its type (or if not present: to the closest spawning location) until it reaches it, then it brings it back to its base.
An attacker moves toward the closest adversary unit until it reaches it, then it follows it until he or the attacked unit dies. Each round, an attack will shorten the attacked adversary unit (first it will lose legs, then torso, finally die). After an attacker kills a unit he has to return to his base (this can be avoided in the implementation by adjusting hitpoints and such, but we needed it for balancing). When a gatherer dies he drops the carried resource, which stays there.

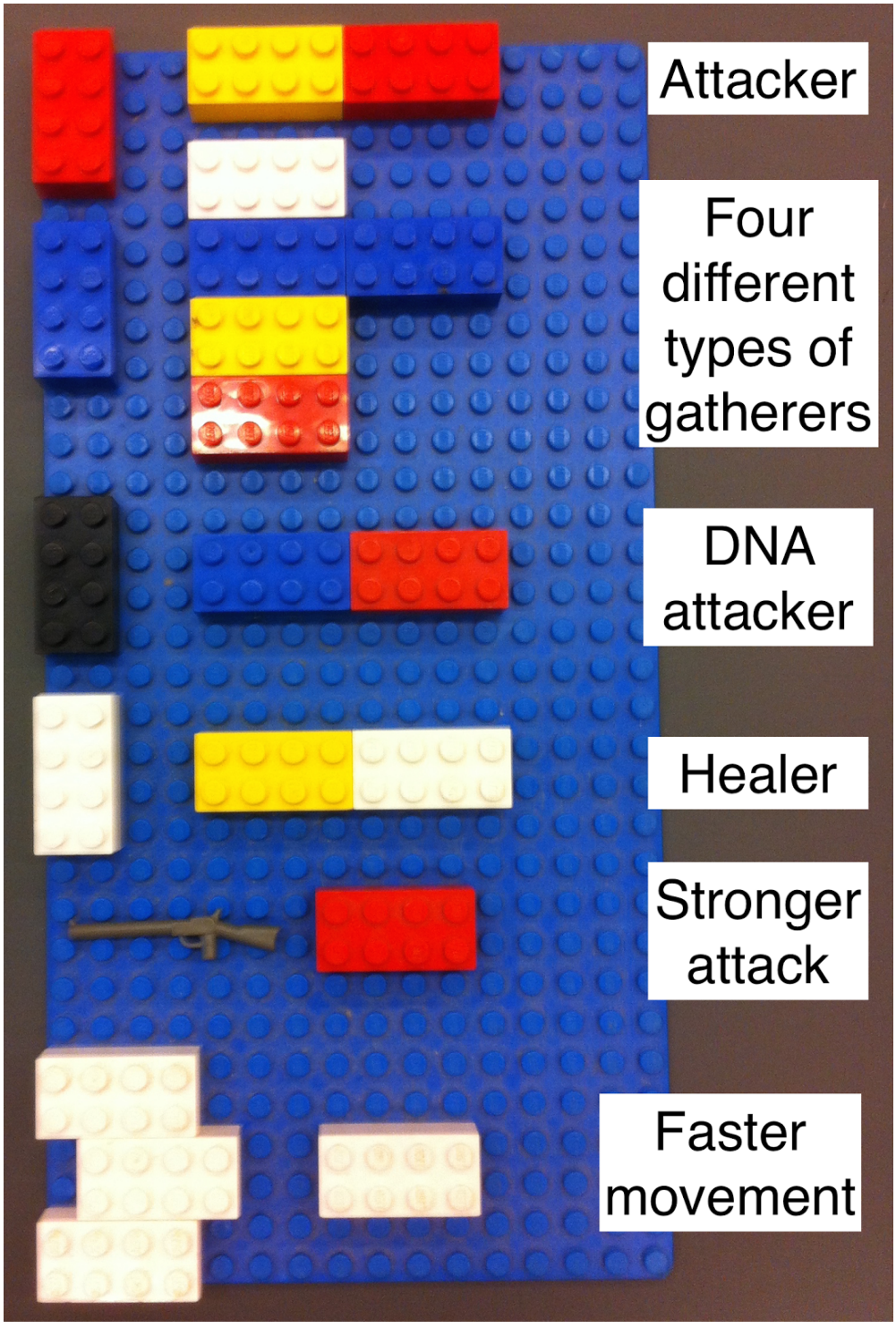

Figure 3: Overview of the necessary resources (lego bricks) for each protein type, gun and jetpack (on the right side). The colors of the lego brick on the left side show the color of the lego units.

Experience

It was undefined before, how gatherers react to different resources.
Solution: have different gatherers per resource
A further Problem was, that you could get stuck when creating a bad sequence in the beginning, for example making a wrong type of gatherer.
Solution: Adjust starting conditions, make it impossible perform a bad start decision (for example spending the first resources on an attacker and then not be able to do anything else).
There was the question of the starting conditions: should the players get resources in the beginning or rather gatherers. Should they get one gatherer of each type?
Solution: One gatherer of each type to make it interesting from the beginning.
For the round based prototype the question arose, how many fields the resources and the units should move per round.

Two positive points were noticed:

- As the game progresses, the players have more and more resources and DNA slots
- Not too many units flood the board, as they get killed

Conclusions

- After implementation, we need to balance the following attributes: Resource spawn frequency, attack and movement speeds (depending on type)
- Unit speeds, lifepoints and hitpoints depending on type
- The frequency of DNA translation (unit spawning)
- Unambiguous exact algorithm for the DNA translation (unit generation)