Computer Games Laboratory, Summer Term 2019

Project Notebook

Draxel

Phillip Hohenester

Moritz Kohr

Maximilian Schmidt

Sonja Stefani

P.A.U.L

Contents

1 Formal Game Proposal **4**
- 1.1 Game Description . . . 4
  - 1.1.1 Game Loop . . . 4
  - 1.1.2 Environment . . . 4
  - 1.1.3 Movement/Controls . . . 5
  - 1.1.4 Units . . . 5
  - 1.1.5 UI . . . 6
- 1.2 Technical Achievement . . . 7
- 1.3 "Big Idea" Bullseye . . . 7
- 1.4 Assessment . . . 7

2 Prototype **10**
- 2.1 Prototyping Goals . . . 10
- 2.2 Setup . . . 10
- 2.3 Game Loop . . . 11
- 2.4 Game Rules . . . 11
- 2.5 Results . . . 13
  - 2.5.1 *Version 0.2* . . . 13
  - 2.5.2 *Version 0.3* . . . 15
  - 2.5.3 *Version 0.4* . . . 15
  - 2.5.4 *Version 0.5* . . . 15
  - 2.5.5 *Version 0.6* . . . 16

3 Interim Report **16**
- 3.1 Current state . . . 16
- 3.2 Progress . . . 16
- 3.3 Implementation . . . 16
  - 3.3.1 Game Board . . . 16
  - 3.3.2 Game state . . . 17
  - 3.3.3 Observer pattern . . . 17
  - 3.3.4 User Interface . . . 17
- 3.4 Changes . . . 17
  - 3.4.1 AI . . . 18
- 3.5 Challenges . . . 19

4 Alpha Report **20**
- 4.1 Current state . . . 20
- 4.2 Timeline changes . . . 20
- 4.3 Quality of life improvements . . . 20

4.4 Fixes . . . 21
4.4.1 Highlighting bug . . . 21
4.4.2 Map generation . . . 21
4.5 AIs . . . 22
4.5.1 Random AI . . . 22
4.5.2 Neural networked AI . . . 22
4.5.3 Scripted AI . . . 23
4.5.4 Heuristic AI . . . 23
4.6 Artwork . . . 24
4.7 Future challenges . . . 24

5 Playtesting **25**
5.1 Changes before the tests . . . 25
5.1.1 Design status at playtesting . . . 25
5.2 Organization . . . 26
5.2.1 Personal session . . . 26
5.2.2 Remote session . . . 27
5.3 Questions . . . 27
5.3.1 Personal information . . . 27
5.3.2 User Interface . . . 27
5.3.3 Gameplay . . . 28
5.3.4 AI . . . 29
5.3.5 General questions . . . 29
5.4 Results . . . 29
5.4.1 Personal information . . . 29
5.4.2 User interface . . . 30
5.4.3 Gameplay . . . 30
5.4.4 AI . . . 33
5.4.5 General questions . . . 33
5.5 Changes . . . 34
5.5.1 User interface . . . 34
5.5.2 Gameplay . . . 35
5.5.3 Artificial intelligence . . . 35
5.5.4 Design status after playtesting . . . 35
5.5.5 Design goals for the final release . . . 36

6 Release **38**
6.1 Summary . . . 38
6.1.1 Changes . . . 38
6.2 Evaluation . . . 39
6.3 Questions . . . 40

6.3.1 Maximilian 40
6.3.2 Moritz 41
6.3.3 Phillip 41
6.3.4 Sonja 42

1 Formal Game Proposal

1.1 Game Description

Draxel is a round based strategy game with simple rules, similar to games like chess or "Into the Breach"[1]. The player commands a variety of units on small grid-based maps against an AI opponent, training to defend his homeland in the upcoming conflict with invading alien forces.

Initially the gameplay will implemented with virtual dice rolls, to maintain the tabletop feeling of the game.

1.1.1 Game Loop

Draxel is played in independent games. Each game is divided into turns (or rounds) alternating between the player and his opponent. Each turn one player gets to command a single unit out of all of his active units. This includes one movement and one attack. Short term goals revolve around defeating certain enemy units, getting your own units out of danger or maneuver them into advantageous positions. The long term goals are defending a position/ an object, defeating all enemy units, taking an objective, etc.
Draxel explicitly tries to keep the actions and rules simple, to facilitate a comparably intelligent opponent. As mentioned later, the enemy AI needs to be pretrained, and might not be able to perform well with complicated objectives and rules.
The world of Draxel will be designed to justify these decisions if possible.

1.1.2 Environment

The game is played on a grid-based map. An example for such a map can be found in figure 1. Other games like the aformentioned "Into the Breach" or the well known "Heroes of Might and Magic" series show that a grid-based map is sufficient for deep strategic gameplay, and it should be easier to train the AI with these constraints.

The environment will be used as a key component of the gameplay and provide buffs and debuffs like e.g. a height advantage or slower movement, but first and foremost the environment will be able to block the line of sight between units. Effects of a blocked line of sight could be more subtle, but will initially only affect the attacks and movement naturally (e.g. a ranged attack unit cannot attack without line of sight, and a unit cannot move to a square without line of sight...) – again, Draxel tries to keep it simple.

Figure 1: A map from the game Into the Breach

1.1.3 Movement/Controls

Units move on the grid according to their movement speed. Most likely units will not be able to cross *difficult terrain* or are only able to do so with reduced speed. Additionally there will be obstacles like mountains or tall buildings that block movement completely. This allows for more strategic depth, and will make battles more open-ended (compared to e.g. chess).

1.1.4 Units

Initially there will be three types of units: **Artillery**, **Infantry** and **Mobility**.

Apart from artillery, each unit can only attack units on neighboring fields. Each unit has a special ability distinguishing it from the others. Artillery can attack units further away than one field, although this might imply reduced hit chances. In the first draft of the gameplay

- Artillery units deal more damage to Infantry.
- Mobility units can move multiple fields per round and do more damage against Artillery.
- Infantry units deal damage to multiple units in front of them.

Figure 2: Chess game UI

The trained AI should be able to "understand" this simple rock-paper-scissors scheme, and try to use the right units in battle.

1.1.5 UI

The in-game user interface can be relatively minimal, similar to chess interfaces which only highlight the fields a selected unit can move to and the unit it can defeat. Additional information is reduced to a health for each unit.

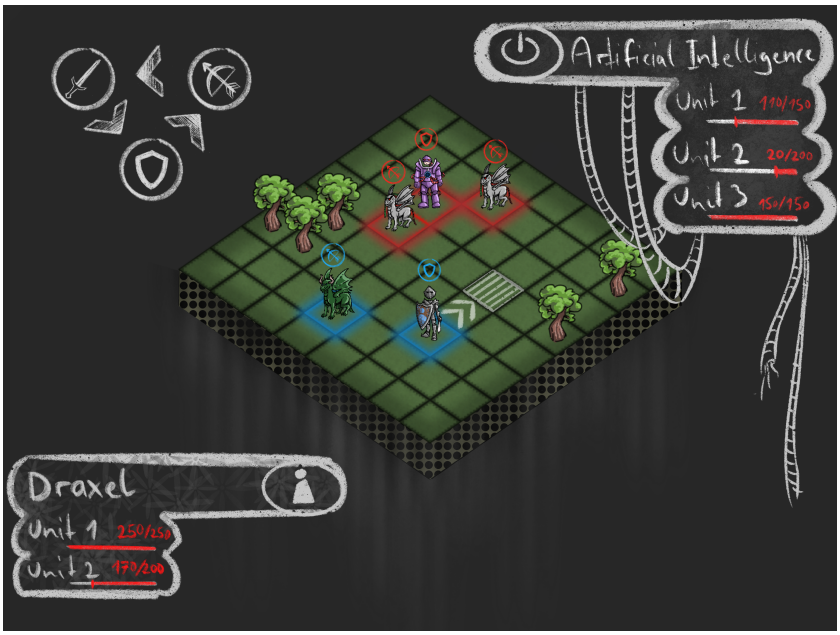


Figure 3: UI mockup of Draxel

1.2 Technical Achievement

The main technical achievement is the usage of "real" AI for the opponents. Draxel uses the Unity `ML-Agents`[3] framework to support AI opponents using pre-trained models to approximate their decision function. The `ML-Agents` framework provides means to implement different learning algorithms and useful abstractions to handle interactions between the agents and their environment. Training can be done via a combination of the Unity specific agent data and external Python notebooks, and the respective pipeline is provided. The final learning algorithm depends on the performance of the several possible approaches.

1.3 "Big Idea" Bullseye

Round based strategy game – akin to typical tabletop war games – but with real AI.

1.4 Assessment

The big selling point are strategy battles against a real AI. Even if the game ends up as a "proof of concept" a lot of insight into the possibilities of the `ML-Agents` framework can be gained, which will be of interest for future projects. In

the best case the enemy AI ends up on a level competitive with human players, and Draxel can deliver basically unlimited different scenarios to test your strategic understanding, and tactical skill. Even if it lacks the adventuring and role playing aspects of e.g. "Heroes of Might and Magic'' it will be a fun short game for anybody enjoying round based strategy games.

						1. Milestone Game Idea pitch		2. Milestone Prototype		3. Milestone Interim Demo			4. Milestone Alpha release			5. Milestone Playtesting		6. Milestone Final Release	
Area	Task	Name	est. Time	rel. Time	Status	15.04	22.04	29.04	6.05	13.05	20.05	27.05	3.06	10.06	17.06	24.06	1.07	8.07	15.07
Functional minimum																			
Organisation	Unity Setup	Maximilian			In Progress		Unity												
Design	Gameplay Desig	ALL			In Progress		Gameplay												
Design	Proto-Tile	Sonja			In Progress			Tile											
Design	Art Design (Terra	Sonja	12				Art												
Implementation	Grid mechanics	Phillip	8		Blocking			Grid											
Design	Build initial map	Phillip	8		Blocking				Map										
Implementation	Unit command sy	Max			Blocking				Command										
Implementation	Unit Movement	Moritz	4		Blocking				Movement										
AI	heuristic (at least	Max	10							AI									
Low target																			
Design	UI	Max										UI							
Implementation	funtional menus	Phillip	4									Menus							
AI	Machine learning	Max / Moritz	20								AI								
Design	Improved Unit ab	ALL			Blocking					Abilities									
Implementation	Unit abilites	Phillip	12								Abilities								
AI	AI for Improved U	Max / Moritz	15									AI							
Desirable traget																			
Implementation	Extended UI (pol	Phillip												UI					
AI	Multiple difficultie	Moritz	20										AI						
Map design	Procedurally gen	Moritz / Phillip	25								Maps								
Design	Attack animation	Sonja	20										Animations						
Implementation	Animations	Moritz	15											Animations					
High traget																			
Design	Tutorial-Screen	Sonja	10												Tutorial				
Sound	Sound effects	Max												Sound					
Implementation	Gamepad contro	Moritz	0.5																
Design/Implemer	More interesting	Phillip											More terrain						
AI	Multiple "tastes"	Max												even more AI					
Implementation	Multiple Agents (	Moritz	8												Multiplayer				
Extras																			
Implementation	Interface to Train and use own AI																		
Design/Implemer	Complex Units	Phillip																	
Design	Story	Sonja	8																
Implementation	Storymode																		

Figure 4: Development Schedule

2 Prototype

This part will discuss the iterations of the gameplay resulting from our experiments with a physical prototype, and the resulting design decisions.

2.1 Prototyping Goals

We refined the goals of the prototype while trying out different gameplay mechanics. Each goal is based on the implementation of the previous goals, starting with the most basic goals:

Version 0.1	Establish initial unit types Establish initial player actions Establish initial turn structure
Version 0.2	Establish initial environment
Version 0.3	Better interaction with the environment
Version 0.4	Faster pace in early gameplay Less “snowballing’’
Version 0.5	Enable more strategic positioning
Version 0.6	More options in lategame Unit balance

2.2 Setup

The setup for a game of Draxel is very easy. The game takes place on a square grid map, we used flipchart paper and drew the board in 8 by 8 size. Bigger boards will result in slower gameplay, while smaller boards get cluttered quickly. Also the typical chess board size allows us to relate some mechanics to well known chess pieces which makes them very easy to understand and learn for anybody who has played chess before.

Each player gets a set of four units, consisting of one unit of the **mobility** and **artillery** type, and two units of the **infantry** type. Later iterations also feature a new unit type called **general** which resembles the "King'' piece in chess.
Obstacles get placed randomly in one half of the board, and then "mirrored'' on the other part, to generate a fair board. You can either represent the obstacles with markers, or draw them directly onto the board, as they are completely static for now.
Each unit gets placed randomly on the first row of two opposing sides of the grid (or on row 1 and 8 if you want to describe it in chess like notation). For this we used an 8-sided die, rerolling if a square is already occupied by another unit or an obstacle, and placing units in **M - I - A** order: the mobility unit first, then the two infantry units, and the artillery unit last.
If the AI will be able to also use this to her advantage, we will allow each player to switch the place of two units. If it turns out that this can be leveraged by a human player much more efficiently, this option will not be available in the final game.

2.3 Game Loop

Goal of the game is to destroy all enemy units, or – if available – the enemy general. To do this each player takes turns based on a *fair* turn sequence (also known as the Thue-Morse sequence[4]). This sequence can be generated by as a binary string by starting with a random number $\in \{0, 1\}$ as a binary string of length 1 and then in each iteration concatenating the existing string with it's boolean complement:

$$0 \to 0 \vdots 1 \to 01 \vdots 10 \to 0110 \vdots 1001 \to ...$$

This may seem a little bit unintuitive, but it leads to more interesting gameplay as the players have to plan around their – and their enemy's – double turns.
When playing the prototype we generated a sufficiently long sequence and allowed every player to look up the turn order. We decided to also display part of this information to the player in the UI, but it will be sufficient to only show the next few turns continuously.

2.4 Game Rules

Draxel has rules for moving a unit, attacking an enemy unit, handling of terrain and finishing the game but the most important part to understand are the unit types (see table 1).
Each unit has 10 hit points and deals three different amounts of damage depending on the enemy unit it attacks. Each unit is especially effective against one other

Unit Type	Movement Speed	Special	Effectiveness
Artillery	1	Ranged	⚔ Infantry ☠ Mobility
Infantry	2	–	⚔ Mobility ☠ Artillery
Mobility	3	Double Move	⚔ Artillery ☠ Infantry

Table 1: Overview of unit types

unit type (denoted by ⚔ in table 1) and deals the most damage when attacking a unit of this kind. Against units of the same type it deals medium damage, and the minimum amount of damage against the type that is denoted with ☠ in table 1. This results in the tournament graph in fig. 5

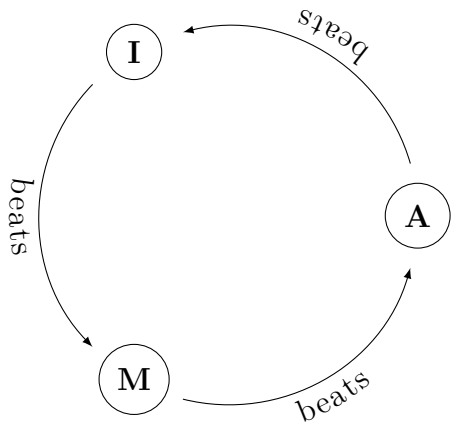


Figure 5: Unit interaction

In version *0.4* we rebalanced the respective damage scale to **2**, **4**, **6** which gave us the best results in the playthroughs.
Each turn the active player can move one unit of his choice according to it's movement speed (in *squares per round*) in any direction that is not blocked by environment or other units, and also attack one adjacent unit with it.

Mobility units can move double their speed (6 squares) if they don't choose to attack in the same round.

Artillery units can attack non-adjacent units on the same row or column. If

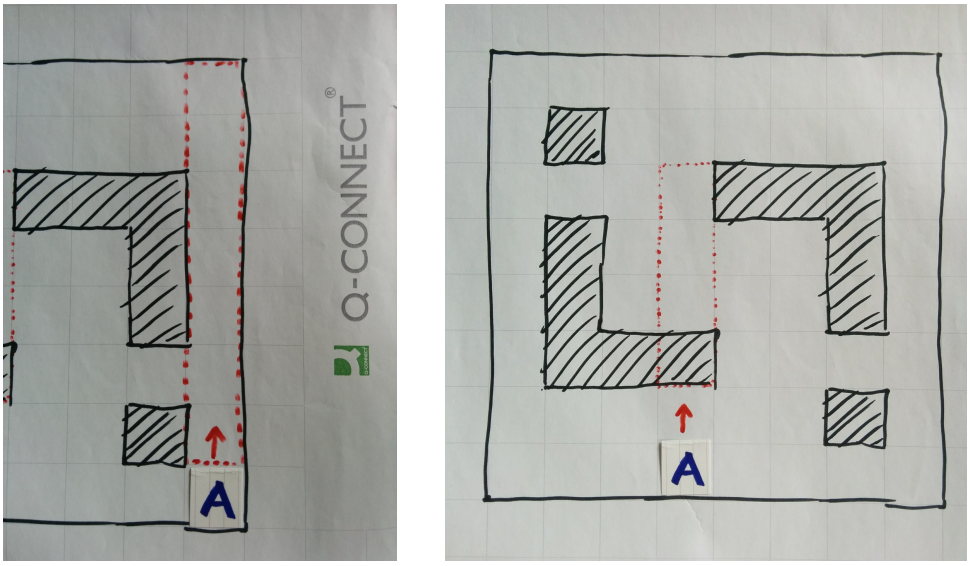


Figure 6: Change of attack range of artillery unit when line of sight is blocked

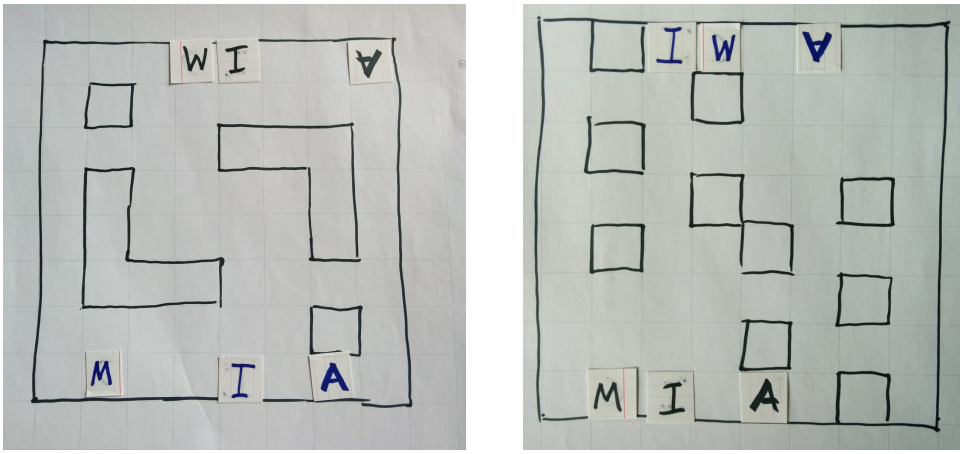
the line of sight to the target is blocked by environment, only the subsequent **4** squares (including the first blocking environment square) can still be attacked. Units further away are out of reach, as seen in fig. 6

2.5 Results

This section will describe the different iterations of the prototype and explain which design decisions were made based on which observations during the playthrough. Every design change was tested independently, but several are grouped together in each *Version* shown below.
To give a better understanding of the gameplay, a short overview over the state of the rules is shown in table 2.

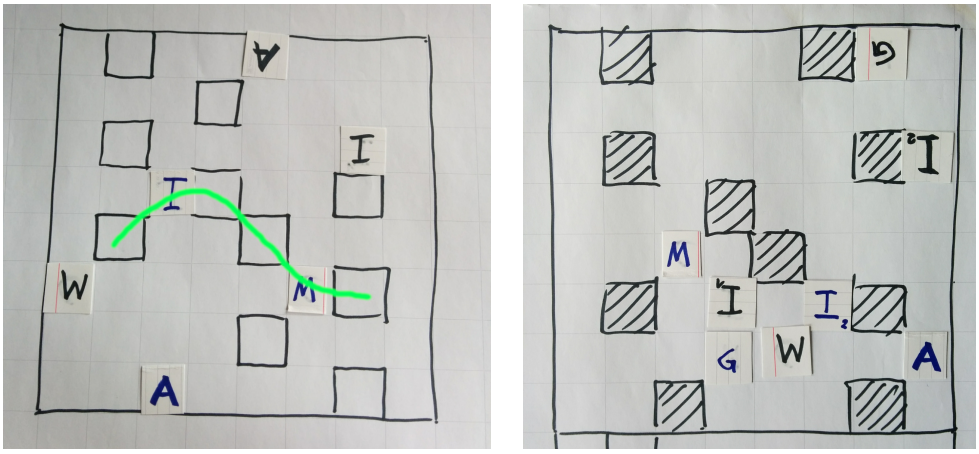
2.5.1 *Version 0.2*

In the initial version Infantry Units and Mobility Units had different special abilities. Infantry Units could attack all 3 squares adjacent in one direction simultaneously, while Mobility Units did not have a double move action but could always move 6 squares (as seen in the column "Movement" in table 2). The board is shown in fig. 7a.



(a) Initial setup (b) More smaller obstacles

Figure 7: Examples from the gameplay tests



(a) Units can better block choke points and affect the movement economy

(b) The General introduces an alternative goal, and rewards quick attacks

Figure 8: More examples

	Units	Movement	Environment	Miscellaneous
Version 0.2	One **Infantry** unit	Speed: 1, 3, 6	two large connected obstacles	Damage scaled 4, 6, 8
Version 0.3			more smaller obstacles	
Version 0.4	rescaled damage to 2, 4, 6			introduced fair turn order
Version 0.5	Two **Infantry** units	Speed: 1, 2, 3 Double move action for **Mobility** unit		Removed **Infantry** special ability
Version 0.6				Introduction of **General** unit

Table 2: Rules

2.5.2 *Version 0.3*

We changed the map layout to avoid stalemate situations where units circle around obstacles, hiding from other units constantly.

2.5.3 *Version 0.4*

The fair turn order allows for "bursts'' of actions, and leads to faster paced gameplay. To avoid an early game where each player just tries to not move his units in range of enemy units as long as possible, we scaled down the damage to make units more survivable and incentivize more aggressive gameplay.

2.5.4 *Version 0.5*

The special ability of the **Infantry** units turned out to be difficult to balance, and also the units did not fulfill their purpose of enabling slower strategies where a player tries to secure space for his units instead of just jumping the opponent. Introducing and additional **Infantry** unit solved this problem, as they could now be used more effectively to block passages in the environment and hurt the other players movement (as seen in fig. 8a). The blue **Infantry** and **Mobility** unit use the environment to create a blockade and cut off the black **Mobility** unit from the other black units.
The introduction of the double move action for the **Mobility** unit as well as the rebalancing of unit movement speed was used to support slower strategies while

still using the **Mobility** units as a deep striking unit to quickly infiltrate enemy lines.

2.5.5 *Version 0.6*

The introduction of a **General** unit with 10 hit points, and medium effectiveness against all units, with a movement speed of 1 and the special ability to lose the game when killed, finally solved the problem of "snowballing'' effects, where a player who gained a lead by killing an enemy unit first, was often able to win the game. Now a player could sacrifice some units to allow a final hit on the enemy general, which lead to more dynamic endgames with more interesting results and deeper strategic decisions. In fig. 8b you can see that even though the blue player has the advantage in terms of units, his General is surrounded and will most likely be killed before blue can leverage his strength in numbers to turn the game.

3 Interim Report

3.1 Current state

Currently we are somewhere in-between the low target and the functional minimum. The basic game play is done and a first iteration of the neural networked AI is working.

3.2 Progress

For this release we managed to create a fully playable version of our game. It includes all originally planed units with all their abilities. There is already a neural networked AI (although at the current training state it is not very smart).The game can be played alone against AI or by two players on the same machine.

3.3 Implementation

3.3.1 Game Board

The game board is build around unity's tile map and grid system. It is constructed out of several layers (one for the terrain, one for units, and one for effects). Everything from units to terrain is a tile which is then placed in its respective layer. Right now we only have simple 2D tiles. In the next release we hope to switch to isometric tiles.

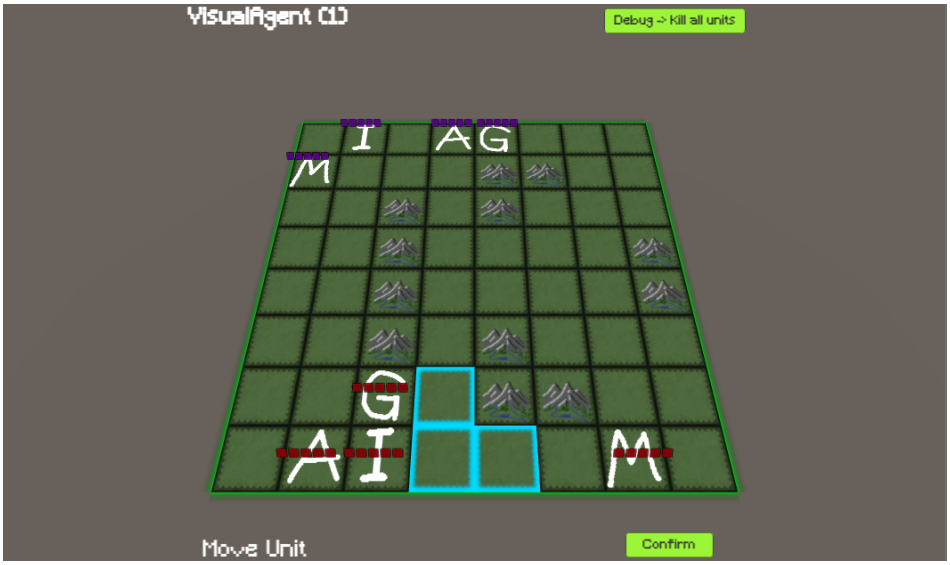


Figure 9: Current sate of the game

3.3.2 Game state

The game is build around a game state system. We wrote the entire system ourselves instead of using premade code for it, since this enabled us to properly encapsulate it.

3.3.3 Observer pattern

The game state is tracked inside a single class that knows where everything on the board is. All other class that have to change something in relation to this board are then registered as observers to it. Every time a change occurs they are then notified and can update their own behavior respectively.

3.3.4 User Interface

The current version has a main menu from which you begin a match. Additionally it supports a simple in game overlay for the actual game play. This overlay contains the unit lives, the active player, and the current turn order.

3.4 Changes

The original game idea already had clear phases for every player. First you moved a unit, then you made an attack action. To improve the learning rate of our AI we divided these phases into more precise steps. Now each player first selects a

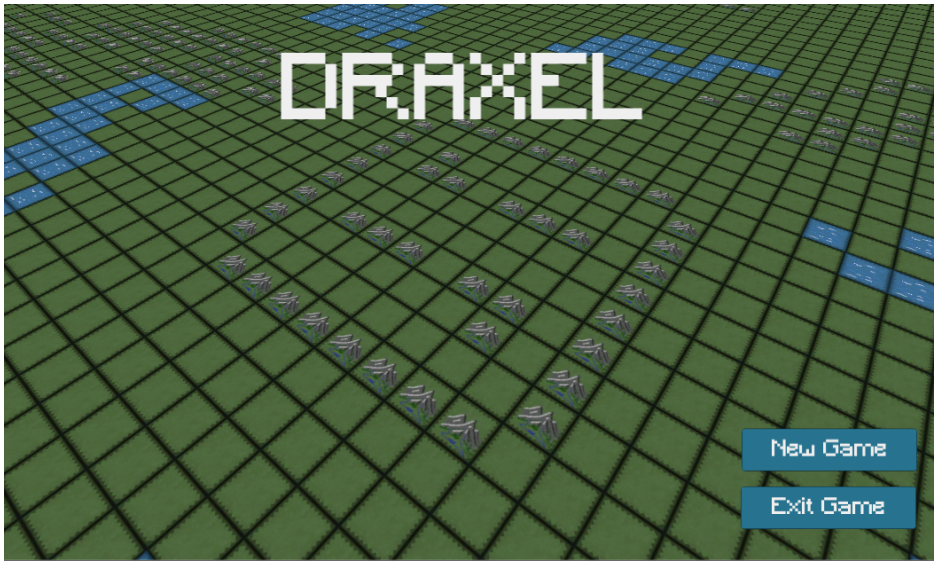


Figure 10: Current sate of the game

unit. In the next phase he can then move this unit up to its movement limit. Next he can perform an attack action. In the end there is another movement phase for our mobility unit if it has not performed an attack (technically all units have this phase, but with zero movement left).

3.4.1 AI

The first tests of a neural network based AI worked quite well. The AI learned the basic rules and is able to play the game although, at this point it is still selecting units at random.
The current AI uses a visual approach. We use a compute shader to render a portion of our game state into a texture. This texture and some additional vector observations are the input to the AI.
Even a completely random behaviour gives a somewhat pleasurable game experience because our AIs can only perform valid actions within their phase.
We found a nice side effect of training an AI in our game: We get dozens of hours of human playtime in minutes and can spot bugs very quickly.
In addition to this we are working on a normal maximin based AI.

3.5 Challenges

Our biggest challenge so far was a lack of initial planning. Initially we created a list of tasks and an accompanying time line. Additionally we distributed these tasks evenly between all members to ensure no one had long down times or was overworked.
The main problem with this was and still is, that these tasks are not really well thought out. It happened allot that new task could not be started because older ones were not yet finished or we completely forgot certain aspects which had to be implemented first.

4 Alpha Report

4.1 Current state

We managed to get everything from our desirable target done and are currently working on tasks from the high target. In general the game is in a state that we are comfortable with handing over to testers.

4.2 Timeline changes

Do to poor planning on our part that already manifested in the last report and dragged over to this one, we had to make some adjustments to our timeline. These changes range from switching tasks around to outright canceling them.
Animations and effects were moved from the desirable target to the high target. This was mainly done because our artist did not have the time to get animations done. Additionally we had planed to add multiple flavors of neural networked AIs in the desirable target. This proved harder than originally planned and was therefore moved to the high target as well.

To counter balance this we added multiple different AIs in the desirable target. We also improved the UI and added other quality of life improvements to help our testers enjoy the game.

The original plan included adding more units and terrain. We decided to completely drop these plans, since the game board is already quite full with the current amount of units and terrain features. Also our AIs could probably not handle them to well and would need a lot of work to function properly.

4.3 Quality of life improvements

The menus were streamlined to work better, look similar across all of them, and provide additional functionality.
The main menu now allows the selection of the opponent AI and the map seed. It also contains a plain text tutorial.
The in-game menu allows to customize the delays between phases and restart the game.

In the last release players had to click ”confirm” to end a phase. This was really annoying when you had selected a unit and wanted to move it but had to click the button first. To combat this players can now double-click a field to move the

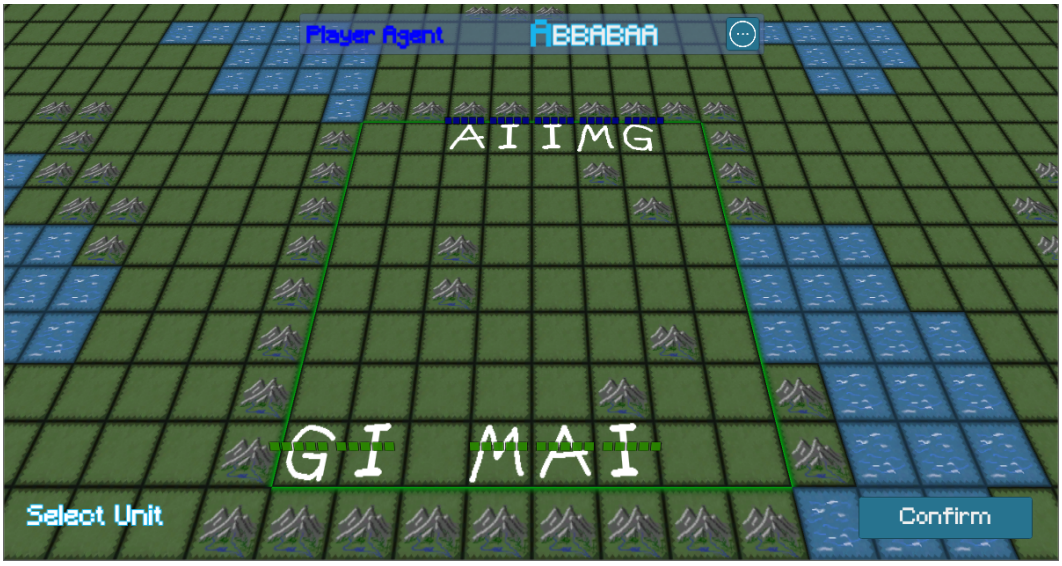


Figure 11: The current state of the in-game UI.

selected unit there without clicking confirm.
Also there is idle music in all scenes now.

4.4 Fixes

4.4.1 Highlighting bug

It could happen that the movement highlights from previous turns for a mobility unit were not removed and stayed visible although the original unit was no longer selected or could no longer reach the highlighted fields. This not only looked weird but was also confusing. This was caused by the mobility unit being the only unit which could do two movement actions. When the highlights were removed at the end of a round this second movement was ignored and its highlights were not cleared.

4.4.2 Map generation

Since the mountains on our map are procedurally generated it could happen in rare cases that they formed a complete barrier in the middle of the map, effectively seperating both players. This is avoided in the current version by simply selecting one x-coordinate where no mountains can be placed.

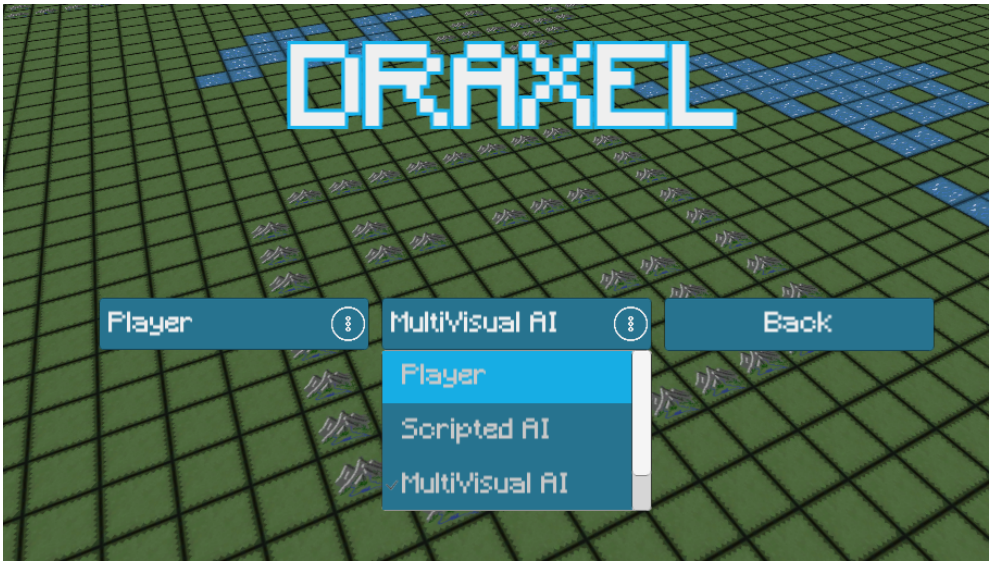


Figure 12: The AI selection menu allows to choose the Agent for both players. It is possible to choose the same AI-type for both players. This also allows to play against an other player.

4.5 AIs

The artificial enemies were one of the main focuses for this milestone. Their skill should be sufficient for play-testing.

4.5.1 Random AI

The completely random acting AI can now attack and is basically used as baseline for benchmarking our other AIs. The random AI shares a lot of code with our Player Agent.

4.5.2 Neural networked AI

We optimized a lot in regards to GameState rendering to Textures. It is now way more flexible and lead to a new NN-AI that uses multiple render textures to represent the GameState. It uses one Texture to represent terrain information (including Obstacles, movement cost and a marker for the currently selected Unit). The second render texture represents information about all units (PlayerID, UnitType and Health). Only the movement points of the selected unit are provided additionally as an vector observation. This new version can also select a unit on its own, even though it is not aware of the current GameLoop sate.
The MultiVisualAgent was trained only against itself in around 8 hours. It achives

a winrate of 92% against the unseen random acting agent, proving the potential of our approach.

4.5.3 Scripted AI

As an substitute for a game-solver, which precomputes the best turn, we implemented a scripted AI. It computes a score (based on weighted sums of damage, kills and proximity to enemies) for each unit and its actions and performs the best. This AI needs some tweaking but achives already 95% winrate against the random agent and 68% against our trained agent.

4.5.4 Heuristic AI

Initially we planned to use a heuristic AI (comparable to the way a chess engine works) instead of a simple scripted one. Many of the best engines around use very simple symmetric heuristics, for example one of the first formulated by Claude Shannon in 1949[5] evaluates the state of a chess board by the simple function

```
f(p) = 200(K-K')
       + 9(Q-Q')
       + 5(R-R')
       + 3(B-B' + N-N')
       + 1(P-P')
       - 0.5(D-D' + S-S' + I-I')
       + 0.1(M-M') + ...

KQRBNP = number of kings, queens, rooks, bishops, knights and pawns
D,S,I = doubled, blocked and isolated pawns
M = Mobility (the number of legal moves)
```

With such a function, you are then able compute the optimized MinMax Score[2] from a series of alternating turns (one player trying to minimize the score, the other to maximize) and use this to search for an optimal turn sequence of a certain depth.
Of course the notion of doubled, blocked and isolated pawns does not carry over to Draxel, but it is just an example for more "knowledge'' that could be added. All other ideas from this simple heuristic could be trivially adapted to Draxel.
Nonetheless it turns out that we are still not able to implement a heuristic AI, due to problematic design decisions we made in the beginning. The most important part of the heuristic evaluation isn't the function itself, it's an efficient and preferably deep search for the best score. In particular, one has to be able to

efficiently generate new possible board states from the current board state representation. While our approach is able to convert the given board state to a certain representation (that we use as input for the neural network), we did not consider a way to generate new board states efficiently from this representation in our current design. Instead of making the representation the core of the implementation, every interaction with the game is implemented in a high level object oriented way, so generating possible moves and resulting board states – although possible in principle – would either be intolerable slow and error prone, or require a major refactoring of the codebase, which in turn would be very prone to errors at this late stage of the project.
Unfortunately we did not find this problem at an earlier stage, and spent too much time on it, but eventually we will now reallocate the work force, and just stick with the scripted AI (using some of the ideas from the heuristic function).

4.6 Artwork

Not yet done, due to time limitations.

4.7 Future challenges

As mentioned above we had planned to create multiple different neural-networked AIs. If we have time in the next few weeks we might try to train additional AIs for this, especially an AI which utilises a rNN. We also want to further train existing agents and cross-train different agents.
We also worked on a heuristic AI that tries to predict the entire game and chooses its action based on the best options. We currently have no good solution for the problems we are facing by the implementation.
One of our goals for testing is to find weaknesses in our current AIs and steer our future AI development into the right direction.

5 Playtesting

5.1 Changes before the tests

We changed the perspective to isometric like originally planed and improved positioning of units.

5.1.1 Design status at playtesting

At the time of playtesting, the used build includes all necessary assets to work and provide the player an experience that is close to the final product. We distanced ourselves from using pixel art as it has proven too difficult for the artist to learn to do properly, so the used sprites for the units and tiles are all high-resolution drawings that try to capture a cartoony style. At this point, the player characters are in the game, the grass tiles have been put in as well as the mountain obstacle tile and some water tiles to add variety.

Figure 13: The look of the game in the playtesting stage.

In 13 it can be seen how water tiles are used outside of the designated play area. The mountain obstacle is used to define the play area as well as add additional strategic gameplay elements on the battlefield. In this build, the player characters and the enemy characters use the same sprites as the enemy sprites were close to being finished, but not yet done.

Figure 14: The player character sprites.

In 14 the full, finished player character sprites can be seen as used in the playtesting build and upcoming builds. The design follows the originally formulated idea of having a medieval setting clash with a setting in which aliens invade earth. The characters in 13 reflect upon the medieval side of this clash, with the player's general being a king with a stereotypical crown. The cavalry unit is a chevalier, the infantry unit a simple knight and the artillery unit a dragon. The designs were chosen to fit into the "knights & dragon" fantasy of medieval times.

5.2 Organization

Initially we spend the first week bug fixing and writing the questionnaire. For ease of use we used a Google Form and also uploaded a final build to a Google drive folder. This way the access to our test was centralized for our testers. Our questionnaire was set up in such a way that users would only have to answer the multiple choice questions, the text answer ones were optional.
We got twelve tests done. Nine of those were friends and family and three were volunteers from the FMI magistrale.

5.2.1 Personal session

The personal play testing sessions were done one on one, or very small groups. We thanked them for their participation, sat them down with the game, and observed their progress. We only answered questions or helped them, if they were truly stuck. The average session lasted about half an hour.
Overall we observed nine tests.

5.2.2 Remote session

Additionally we posted the link to the Google drive folder into various Discord groups. This way we got further three testers who were not under direct supervision and had to text us for questions.

5.3 Questions

5.3.1 Personal information

We collected some personal information about or testers to verify the meaningfulness and to give context to our results.

How experienced are you with strategic tabletop games (eg Chess)? Since our game is inspired by chess, it felt necessary to ascertain how good players already were with these kinds of games.

How experienced are you with computer games? If players are more familiar with the concept of point and click controls, they might have less problems with our game.

Are you a student in natural sciences? / How old are you? Asked to gather information about our test demographics.

5.3.2 User Interface

A lot of the information necessary to play our game is conveyed over the UI. We asked several questions regarding separate parts of our UI, to spot potential problems or room for improvement.

Did you like the Style of the UI? Asked to get a general feel of how players perceived the user interface.

How intuitive was the main menu? The main menu is the first point of contact with our game. If players already struggle here it might hint at severe problems with our style of UI or structure of the game start.

How intuitive was the Unit selection/ movement? Unit selection and movement is part of the core aspect of the game. Ideally it should be as clean and easy to use as possible. Unfortunately we had to make some compromises

between ease of use and a clear turn structure so that our AIs were able to play the game as well.

How clear was the health display? Unlike chess our units have multiple hit points and not only one. Additionally units have varying effectiveness and deal different amounts of damage. So understanding how much live each unit has left is vital to winning.

How clear was which players turn it was? Unlike most games players don't switch after each turn.

How easy was it to see what unit is which type? The units are only distinguishable by their sprite (and it is not mentioned in the tutorial which unit is which), so being able to immediately recognize them is very important.

5.3.3 Gameplay

This section contains questions regarding the gameplay and the fun players had during play.

How engaging did you find the effectiveness system? As explained before units deal different amounts of damage depending on the attacked unit. This mechanic is vital to winning the game without sheer luck.

Did you feel like being the starting player is an advantage? In many turn based games the starting player has an unfair advantage over his opponent. We tried to mitigate this with our uneven turn order. We asked this question to verify if this system works.

Was the target of the game clear at all time? Asked to see if the tutorial explained the game well enough that players did not feel lost during game play.

How important did your decisions feel? In games we have to make decisions to achieve our goal. It is part of the fun of a game, being responsible for your decisions. If it feels like all you do has no effect, it is not fun.

Were you able to get better during playing? Asked to see if the learning curve is not to steep or existent at all.

What was your favourite unit? / What was your least favourite unit? If to many players favour the same unit, it might be overpowered. Consequently if a unit is disliked by the majority it might be unbalanced.

5.3.4 AI

The artificial intelligence is the core aspect of our game. This section dealt with its general performance against the players and how the testers liked them.

What AIs did you play against? This question was mainly to see how representative our survey is. Together with individual responses we can try to extract further information regarding specific opponents.

What AI was most fun to play against? We want to develop some of the AIs further. This question helps us to decide which AI is worth making a new version of, or needs necessary changes.

What was your impression of enemy behavior? / Did you have a winning strategy, if yes, which? These questions target undesirable behaviour of players and AI we might have to counteract.

If you Played against the scripted AI, was it hard to win? We have no way of measuring the skill of our scripted AI against humans. The answers might show if we have to keep working on the logic of this AI.

5.3.5 General questions

This section contains questions about general topics like how well the testers liked our game, or what they liked best overall/ what they disliked the most.

Please describe where you see the game on this graphic. As stated in the project structure handout, players had to specify where on the graphic they see our game. We wanted to double check if they perceive the game as we do and as we intended.

5.4 Results

5.4.1 Personal information

Overall our testers were reasonably familiar with strategic tabletop games and very familiar with computer games. Interestingly we had exactly an 50/50 split

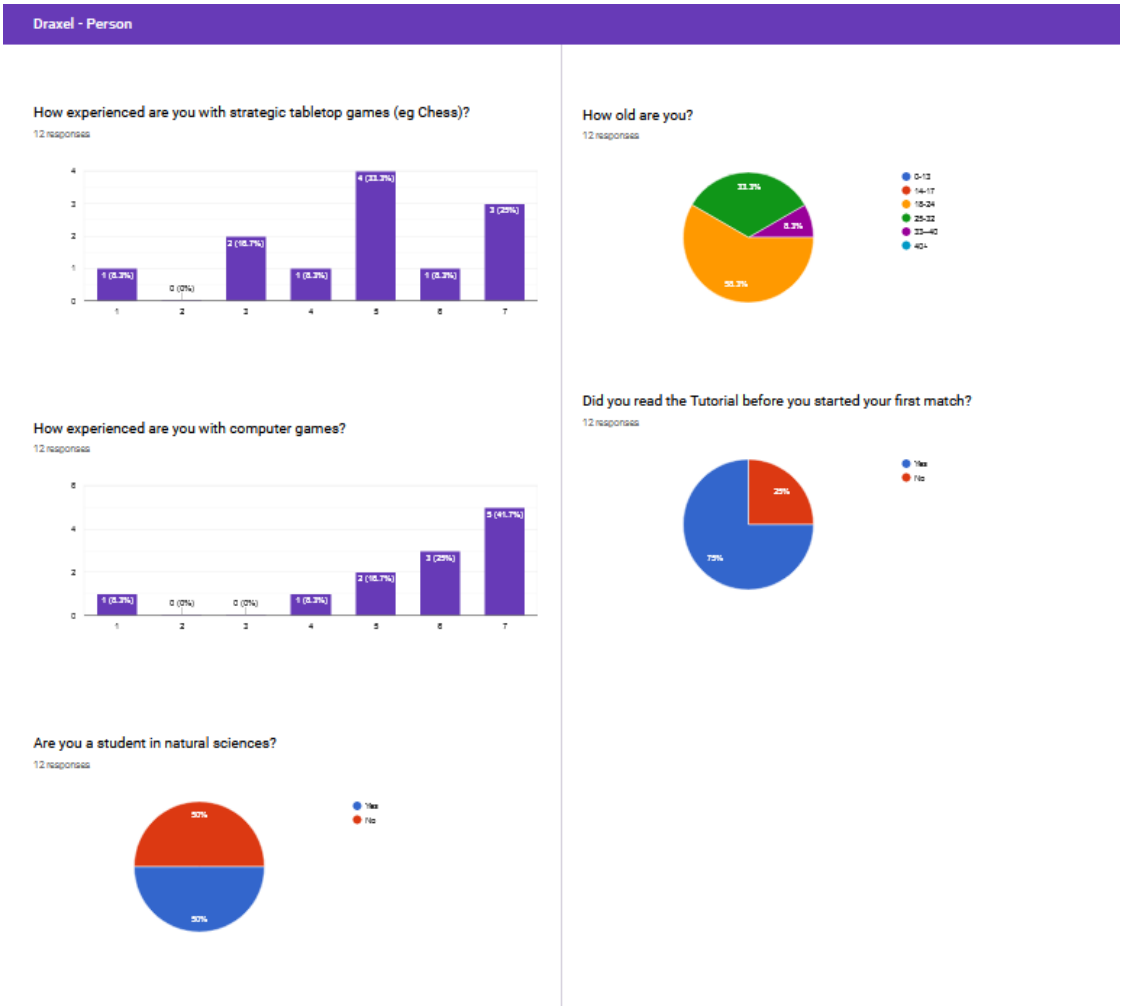


Figure 15

between students of natural sciences and others. Additionally most were in the age range of 18-24.

5.4.2 User interface

In general the user interface was quite well received. Some areas that nee improvement are the tutorial and the unit selection/ movement process.

5.4.3 Gameplay

In general the game seems to be fun, engaging and there is no aspect the players really hated. But it also became clear that there are several areas that still need

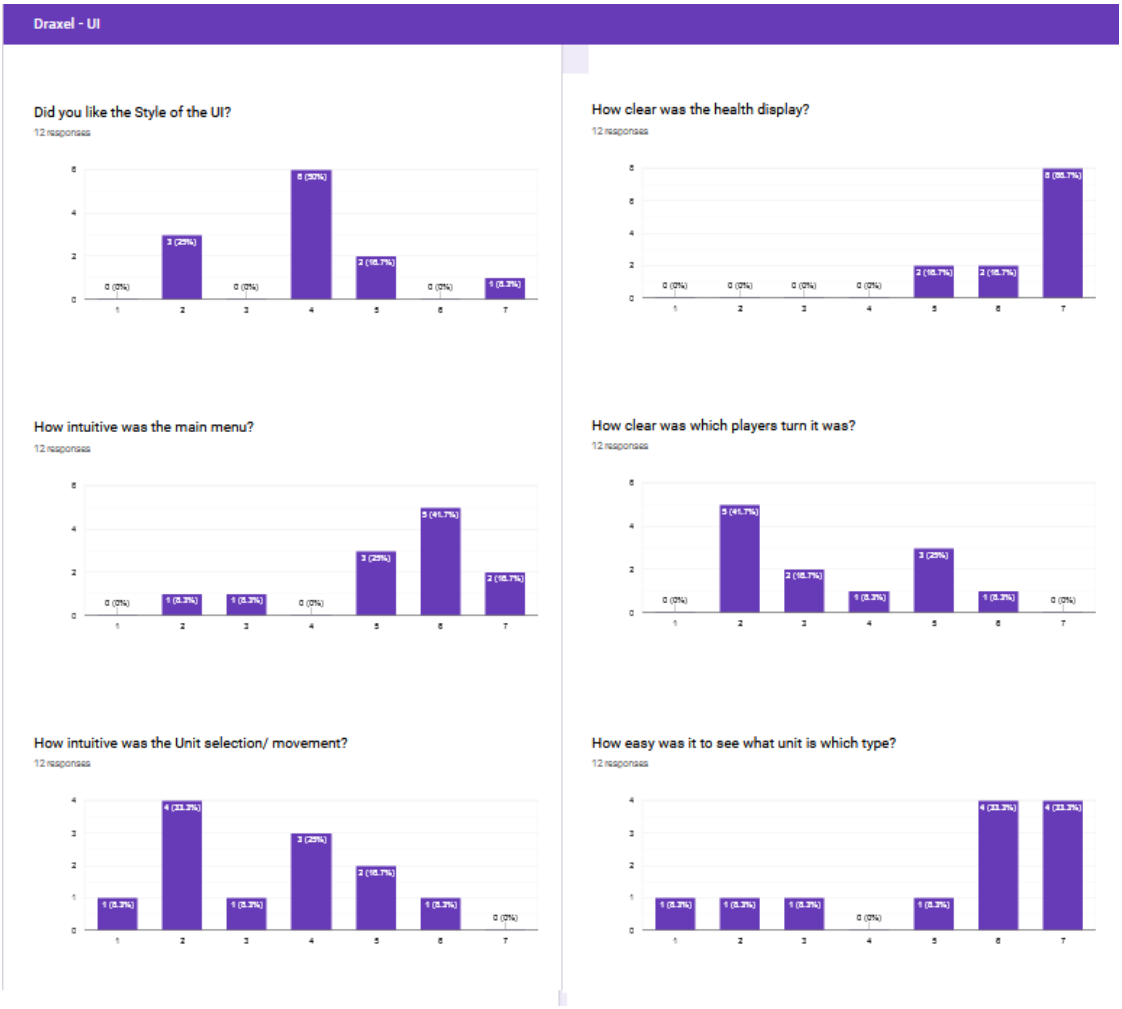

Draxel - UI
Did you like the Style of the UI?
12 responses
How clear was the health display?
12 responses
How intuitive was the main menu?
12 responses
How clear was which players turn it was?
12 responses
How intuitive was the Unit selection/ movement?
12 responses
How easy was it to see what unit is which type?
12 responses

Figure 16

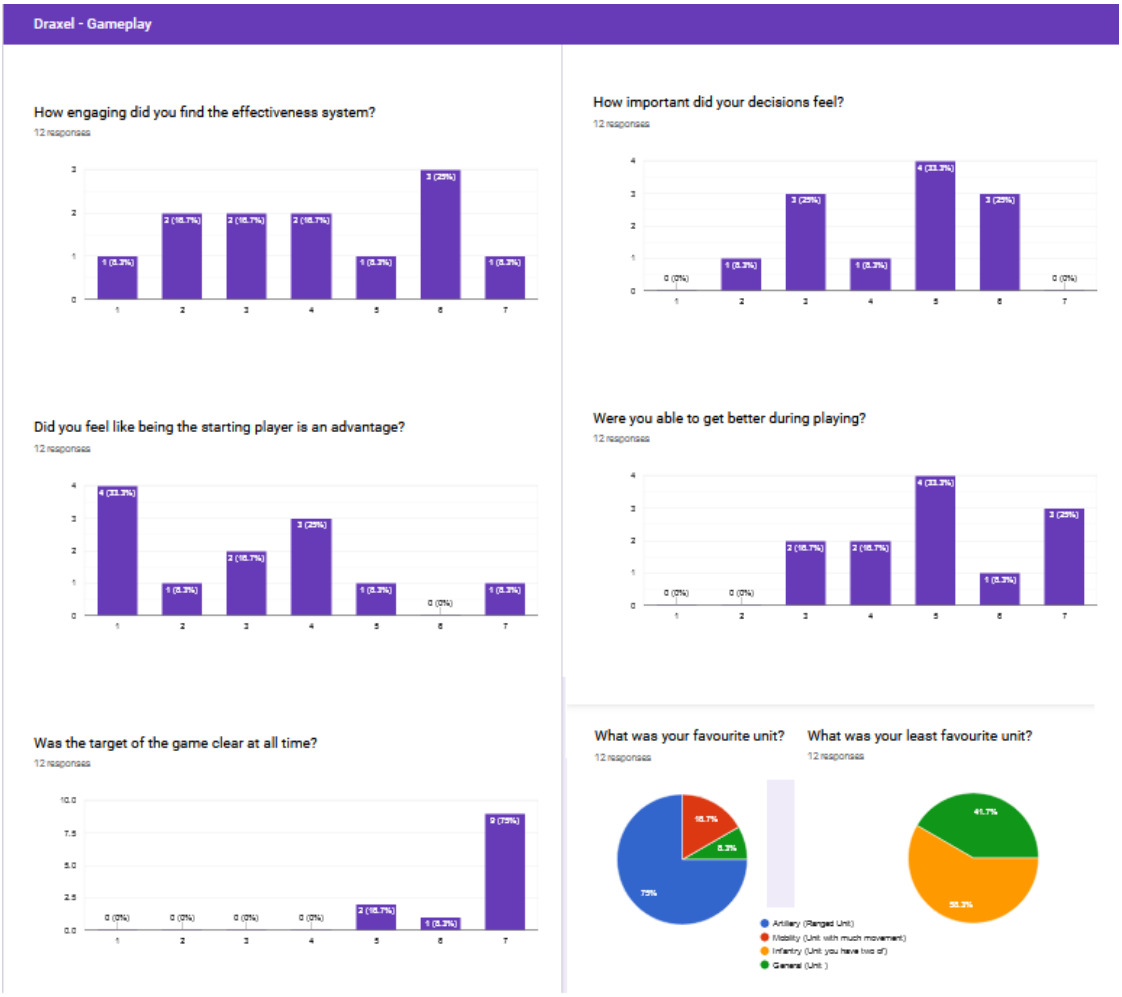


Figure 17

some improvements.
Almost all players heavily preferred the artillery unit to all other units. In fact some players almost exclusively used it during play. This strongly suggests that it is to strong and needs to be rebalanced to make the overall gameplay more diverse.
Interestingly roughly two thirds disliked the infantry most. This was mainly caused because it is the unit the artillery is strongest against and because it can't do anything special.
Many players were somewhat confused by the effectiveness system and took some time to understand the turn order.

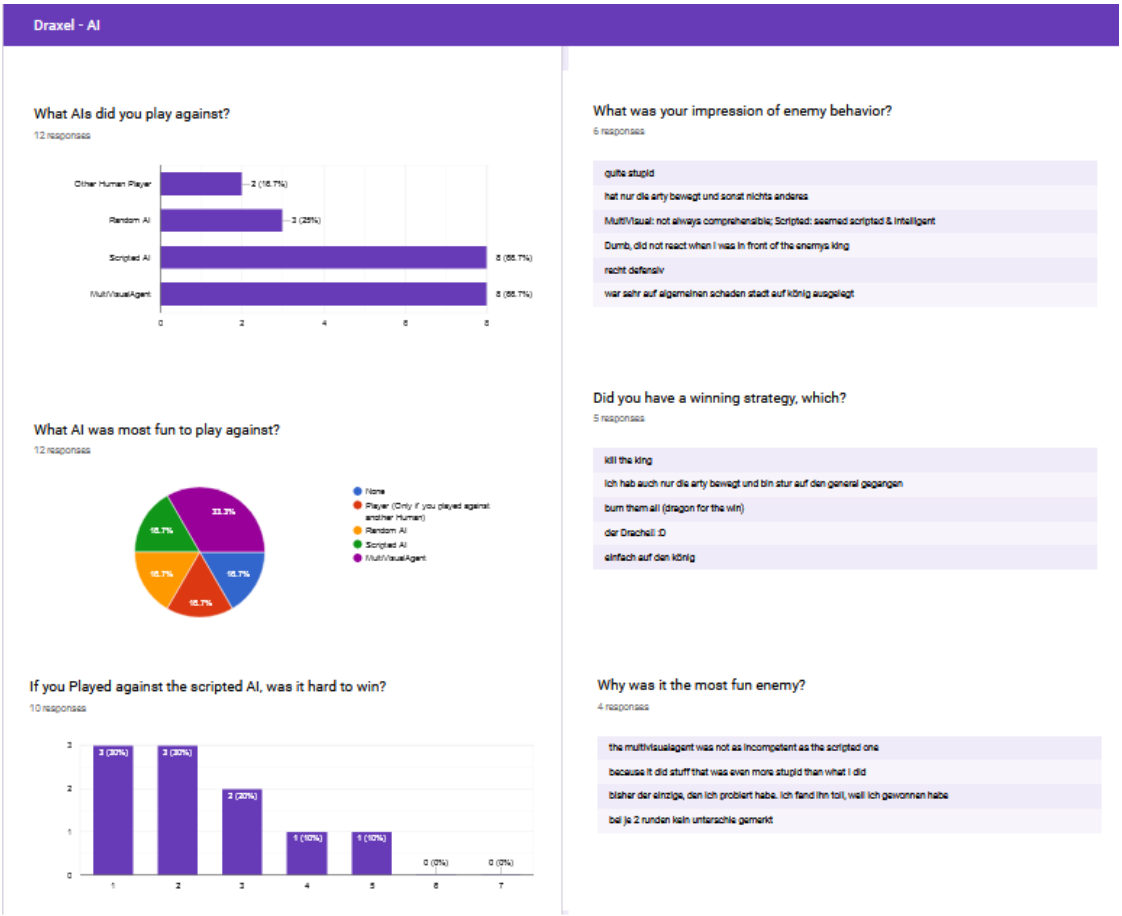


Figure 18

5.4.4 AI

Almost all players played against the multi visual AI and the scripted one. Three testers played against each other. (One pair filled the form only once)
The currently implemented AI seems to weak. Players had way more fun and understood our design decisions better if they played against other humans.

5.4.5 General questions

Our testers liked the style of our game. Especially the dragons were well received. On the contrary all of them complained about the controls, because they felt quite unintuitive.

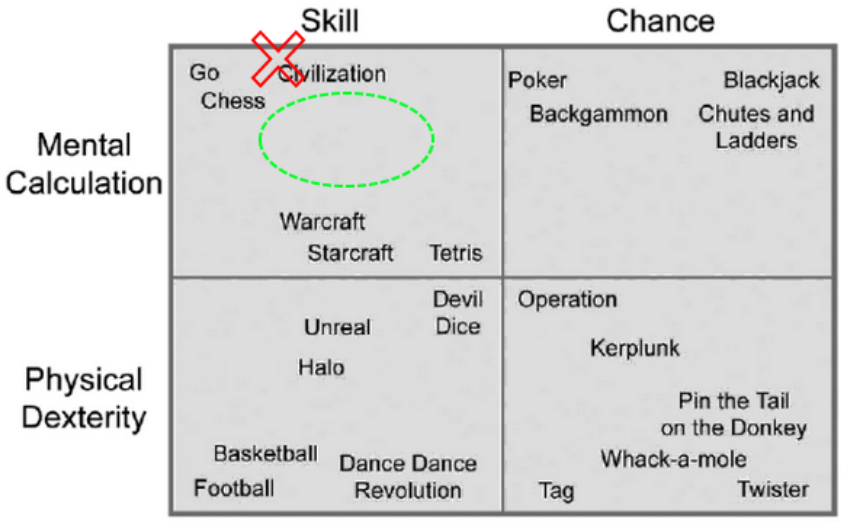


Figure 19: We think the game is located at the red cross. The average of the testers placed it inside the green circle.

5.5 Changes

5.5.1 User interface

Main menu We will change the new game button behaviour. Currently it starts the new game directly. We intend to implement a proper session configuration screen, where you can choose the players and your seed.

Tutorial The tutorial has to be reworked. We plan to remove the transparent background and increase the font size.
Additionally we will have to explain the ingame controls and improve our explanation of the effectiveness system.
We might add visualizations for all units. This could include their graphics, their movement behaviour and their attack patterns.

Ingame UI The phase display needs to be reworked since it is currently hard to read. We might reposition it to couple it stronger with the next phase button. Also we will have to improve the turn order display, it was sometimes confused with a player name. We also want to make it more clear if a player is two times in a row.
We have to improve the currents players highlighting. Specifically it has to be made clearer when a player gets two turns in a row. Especially in the beginning

players found it difficult to discern which units were theirs.
We will introduce an ingame helper for our effectiveness system.

5.5.2 Gameplay

We want better attack feedback, players sometimes did not notice an attack and are missing the satisfaction of an attack. This might include adding sounds and effects (We are planning for a splashscreen showing the two opponents fighting).
To decrease effectiveness of the artillery we will add more mountains to the map generation, this should also add to the infantry.

5.5.3 Artificial intelligence

We might use the provided feedback to improve our AIs. This includes making a new iteration of our trained AI as mentioned in previous chapters.

5.5.4 Design status after playtesting

Shortly after collecting data from playtesting, the enemy characters were finished. In addition, the tutorial screen has been updated from a simple textbox (as can be seen in 20) to multiple pages of instructions with examples of the game elements, so the player can get a better understanding of the game.

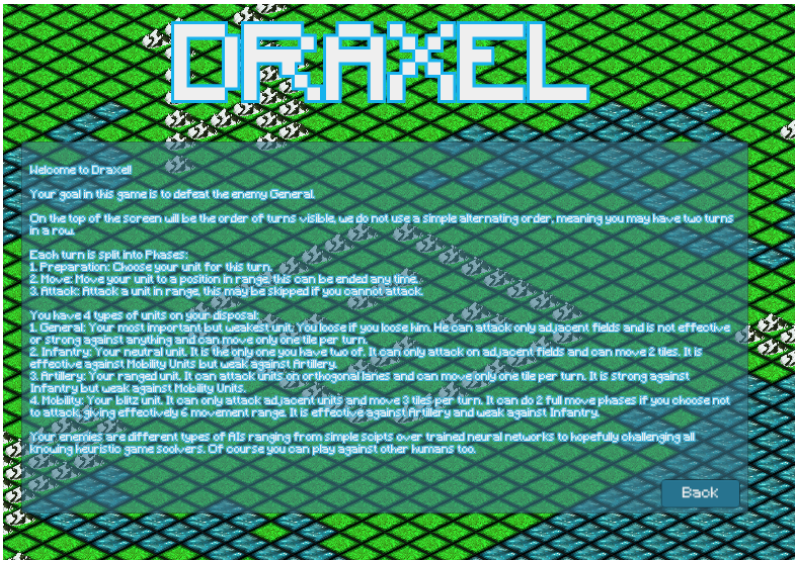


Figure 20: The tutorial screen that was used in the playtesting phase.

Figure 21: The enemy character sprites.

The new enemy characters can be seen in 21 and try to resemble extraterrestrial and sci-fi-esque lifeforms and machinery. The enemy general takes inspiration from Jesus as a biblical figure. We've applied some scientific changes to Jesus' character to make him fit into the role of an alien that is enhanced by extraterrestrial technology, such as metal wings. All of the characters that were designed show signs of being improved by technology, with cables running through machinery or armor and blue lights glowing in an unnatural way on all enemy characters. The hostile cavalry unit resembles a cheetah with its body exchanged for a mechanical one, the infantry unit is a purple, four-armed alien wearing power armor. The artillery unit is a glowing cannon.

5.5.5 Design goals for the final release

Going forth from the current status, the next big goal design-wise is to adjust all sprites to look balanced when put together.

Apart from that, some variety tiles have been and will be made (such as grass with flowers) and may be incorporated for the final release.

At this point in time, the design goal of animations seems highly unlikely to be reached in its entirety, as eight different characters exist that would have to have different attack animations, with multiple hand-drawn frames for each animation. This is simply not feasible in the given time frame (and without specialized software). Instead, attack screens will be drawn for the final release, that at least show an interaction between characters. A very rough example sketch for such an attack scene can be seen in 22 and shall serve as a rough guideline for going forth

with this idea.

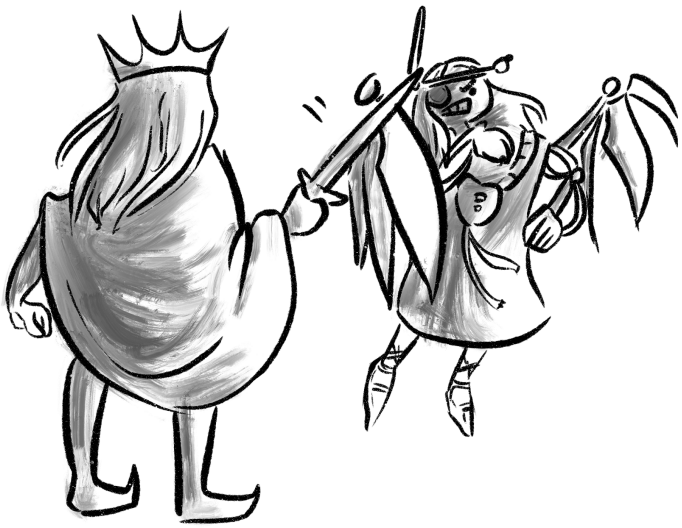


Figure 22: A rough sketch of an attack scene between the player general, the king (left), and the enemy general, cyber Jesus (right).

Figure 23: First screen of the new tutorial

6 Release

6.1 Summary

We managed to implement a turn-based chess like strategy game and implemented our own neural-networked AI for it. Using very simple rules the game is very challenging. This is improved by using a thue-morse sequence for alternating the players.
The game is currently in the high target. We have multiple different AIs that each provide a different difficulty.

6.1.1 Changes

We overhauled the tutorial completely. It is now divided into multiple screens explaining different topics. Additionally we changed the main menu to keep it more in line with what players are used to from other strategy games. The font was also changed to a more readable one which also fits better with our art style.

The in-game user interface was changed as well. It now includes an image displaying the damage system together with new icons for the different unit types (these icons are also displayed next to the units on the game board). Since players struggled with the next phase display we re-positioned it to be closer to the next phase button which archives better positional correlation.
We also added new art assets for the enemy units.

Figure 24: Example of the new in-game UI. Noteworthy changes include the damage system display in the lower right corner, the new unit icons, and the re-positioned next phase text display.

To improve the feedback for users when attacking a unit, we added a battle screen similar to what players see in the game Pokemon or Tekken. It shows the attacking and the attacked unit in their colors and changes based on received damage.

Finally we fixed one game breaking bug in the scripted AI were it tried to attack units that it could not attack. This resulted in it doing nothing for several turns.

6.2 Evaluation

Our overall Game idea was completely preserved over the whole development process. The turn-based mechanic with a fair share sequence works well together with simple units and individual AIs. Our initial development schedule took a big hit due to early unplanned refactoring of big parts of our code base, but we were mostly able to stick to the plan in the end after we cut some expandable content. The prototype phase was very helpful for our game because it transferred perfectly into our digital version. Especially the play testing phase was very useful for us since we got great feedback and were able to fix most of the problems to get a more playable, fun and engaging game.

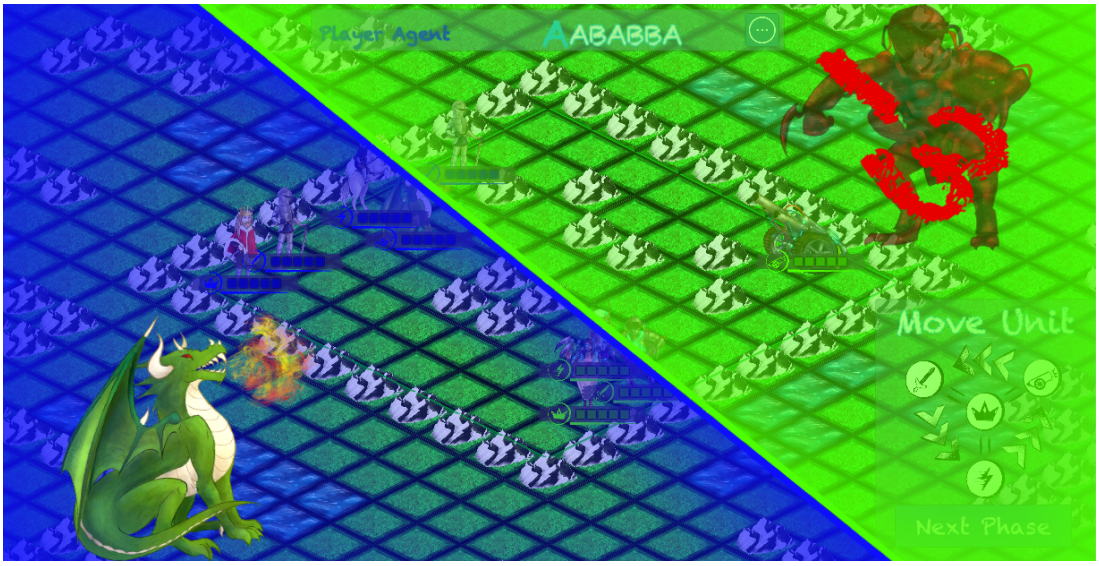


Figure 25: Example for a battle screen

6.3 Questions

6.3.1 Maximilian

The biggest technical challenge was the interaction of our game architecture and the structure of the ML-Agents framework. We started without a clear software design, and the efforts to refactor it in a way that would make it usable for the upcoming problems in the end turned out to be unsuccessful. The theme was a useful inspiration for ideas, although finding a game concept could maybe be easier with total freedom and would probably lead to more passion for the concept. In future game projects it will be important to assess the group dynamics and probably elect a lead developer / designer to avoid some of the communication (or lack-of-communicatoni-) overhead. Establishing a workflow to make simultaneous work (even during refactoring) easier also seems important in hindsight. I learned a lot about the ML-Agents framework and state of the art chess AI, but I was not able to bring these two worlds together in a way that would be useful for our project – so in the end I am happy that the team could compensate for that. I think we achieved what we wanted, as part of the idea was to get used to the framework and find out it's strengths and weaknesses which we definitely did. We hit most of the milestones but had to readjust more often than it would have been necessary if we would had planned everything better from the beginning.

6.3.2 Moritz

My biggest challenge was to work with the structure of ML-Agents and to bring my own AI ideas into their format.
The theme felt a little broad and we had to narrow it further down to get sophisticated ideas. Without the the theme, thought, our game wouldn't be as it is because it is a central core of our game.
Our biggest problem was the re-factoring in the middle of the prototype phase, I would have liked to avoid it and it turned out to be of not much use in the end. With more time on Software Design while prototyping we may could have prevented this and I will keep that in future projects in mind (especially if working in a team).
The hardest nut to crack was to get our MultiVisualAgent to work. I had to fix a bug within the ML-Agents framework to be able to train the network.
I am quite happy with how our game turned out. We managed to realize almost everything we planned. Only thing we not got as we intended was the strength of our AIs, but that's to be expected with effective one instead of two AI-programmers after one plan for an AI did not turn out well.
We hit most of the milestones or had to push back features by just a few days.

6.3.3 Phillip

I really enjoyed this course. It allows for a lot of personal freedom and is a nice change to the usual university life with regular exams.

The given theme was really cool. It could be interpreted in various different ways and restricted the idea gathering process in the beginning just enough to improve creativity. If i had had to choose between total freedom and a given theme i would choose the theme any time.

Overall i am very happy and proud of our final result. It is fun and engaging and can provide a real challenge when played against a good opponent. We managed to complete everything from our desired target and even got some things from the high target done.
If i would do this course again i would definitely improve the initial planning. Especially in the beginning we suffered from it During this period some tasks could not be started because earlier tasks proved to be more challenging than expected and delayed everything else.

6.3.4 Sonja

Going into the project, we weren't 100% sure what art style to follow and half-heartedly settled on pixel art from the get go. While doing the first assets, it became clear that tiles like grass or water were not too hard to make, however making character sprites in a pixel style proved to be very difficult, as I haven't had any prior experience with making pixel art. During the design process, it was then decided to scrap the pixel style and do what feels comfortable to the artist, so all following sprites and tiles are in a much higher resolution. It was definitely a big time loss to only properly settle for an art style halfway through the project phase and for upcoming projects, this decision needs to be made before starting.
In the end, we somewhat reached our high target for the design part of the project. Initially, we wanted to make animations for our units, but since high quality animations are very time consuming to make (especially fluent ones), this idea was exchanged for a simpler version of animations. We now use "battle screens'' for when units attack each other, effectively having 16 images for the battle scenes (8 different character sprites with an attack and a defence sprite each).
Although we had some difficult phases throughout the realization of the project, all in all I think it is still a success.

References

[1] Subset Games | Into the Breach. https://subsetgames.com/itb.html, Feb. 2018.

[2] Negamax. *Wikipedia* (Feb. 2019). Page Version ID: 881512142.

[3] Unity Machine Learning Agents Toolkit. Unity Technologies, Apr. 2019.

[4] Headbomb. Thue–Morse sequence. *Wikipedia* (Apr. 2019). Page Version ID: 894938569.

[5] Shannon, C. E. XXII. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 41*, 314 (Mar. 1950), 256–275.