HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

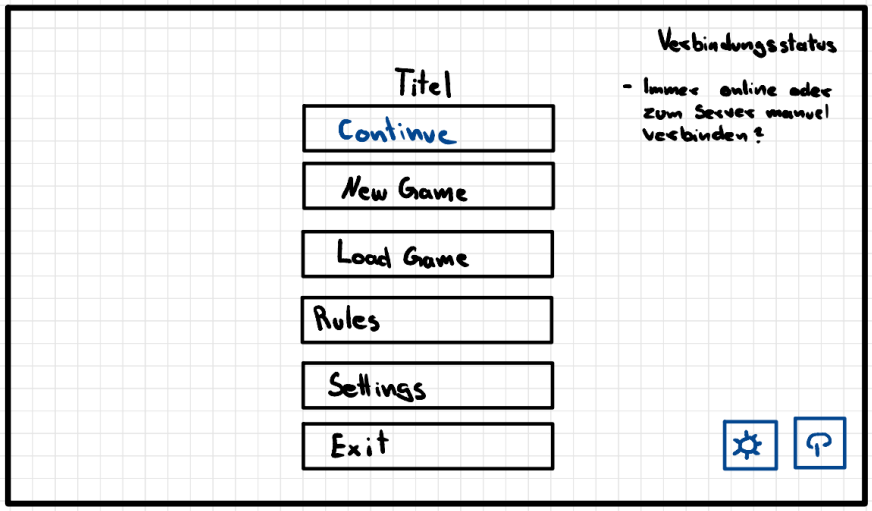
Programmierprojekt WS22/23

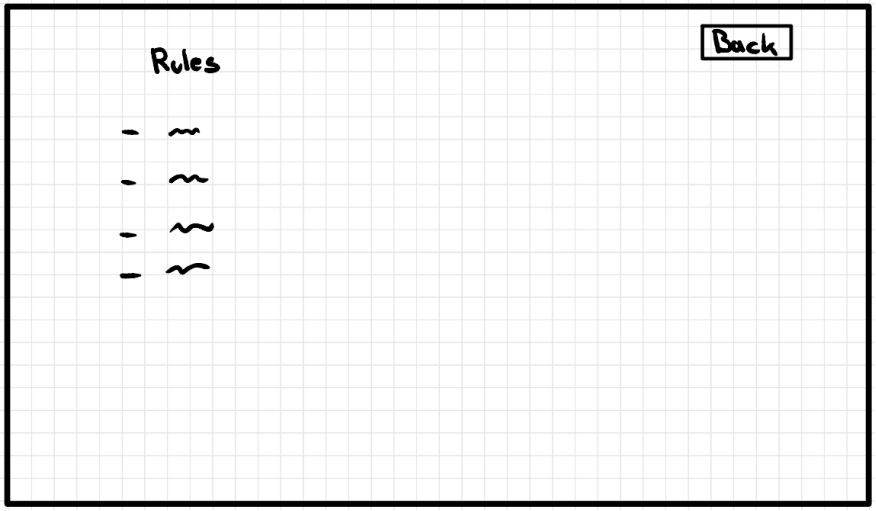
Battle ships

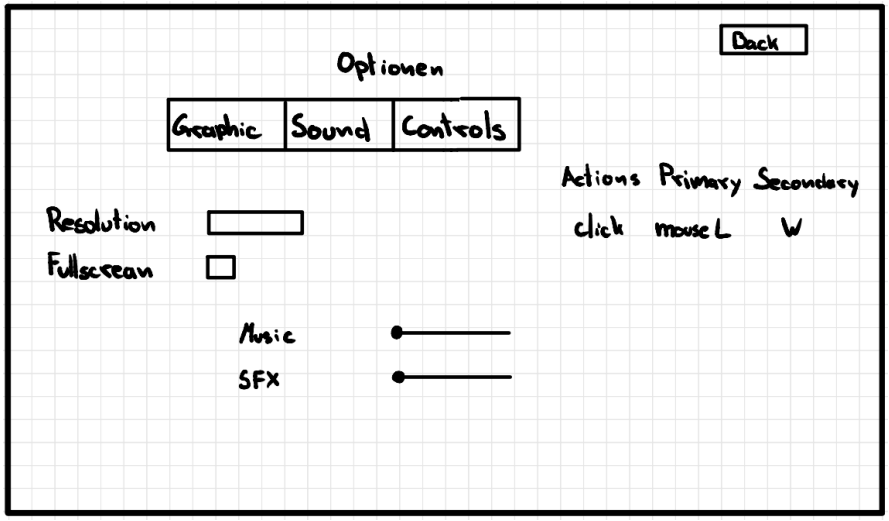
Alexander Fitze, Lukas Müller, Katharina Nolte, Philip Ohizu, Stefan Rifel, Jannik Rosendahl, 21.10.2022

Table of Contents

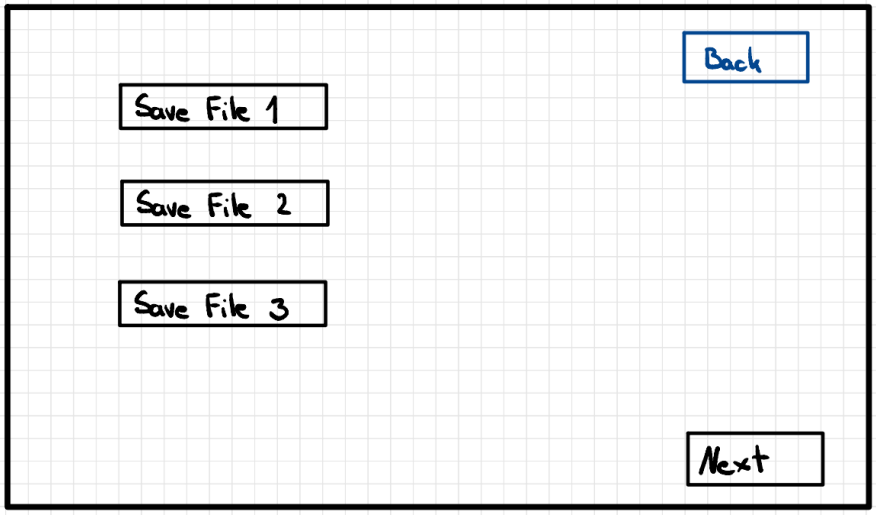
GUI

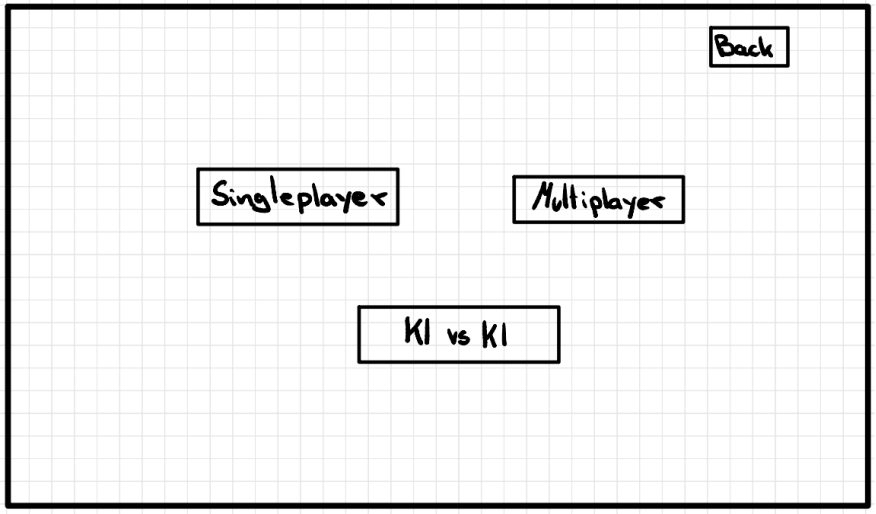

Titel
Continue
New Game
Load Game
Rules
Settings
Exit
Verbindungsstatus
- Immer online oder zum Server manuel verbinden?

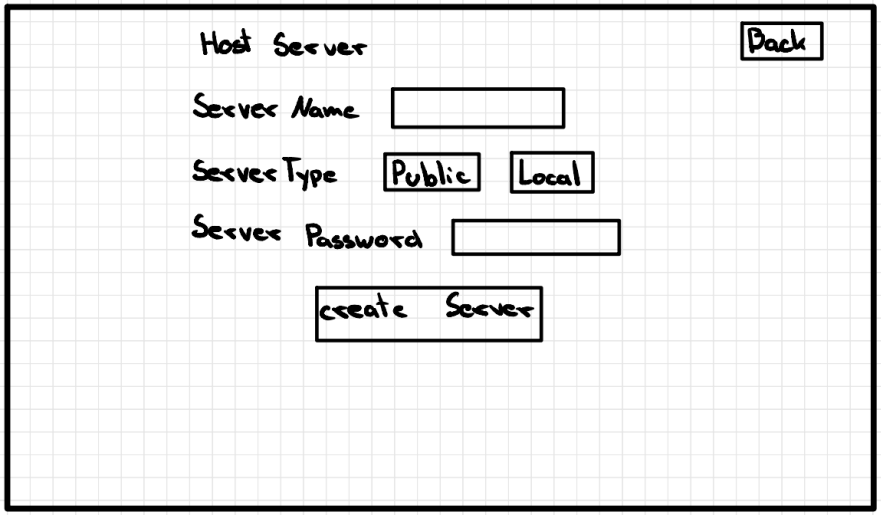

Rules
Back


Optionen
Back
Graphic
Sound
Controls
Actions Primary Secondary
click mouse L W
Resolution
Fullscreen
Music
SFX

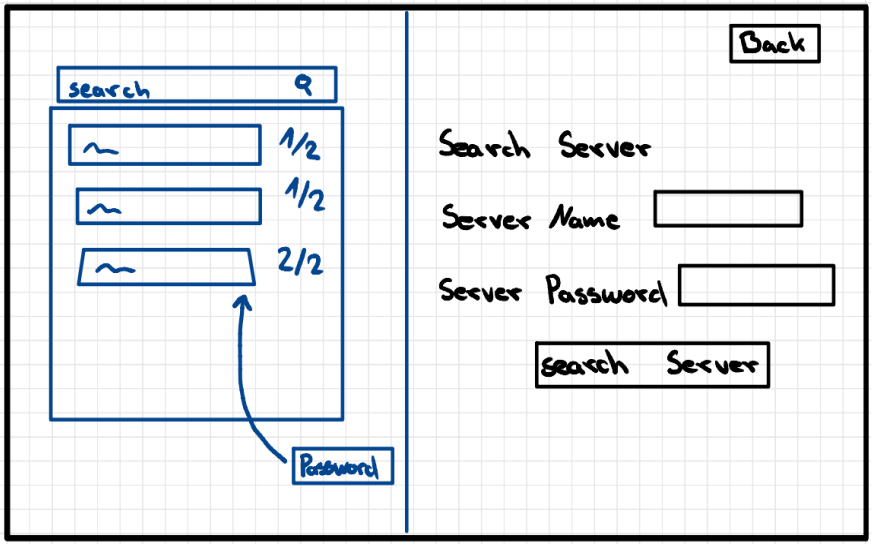
GUI

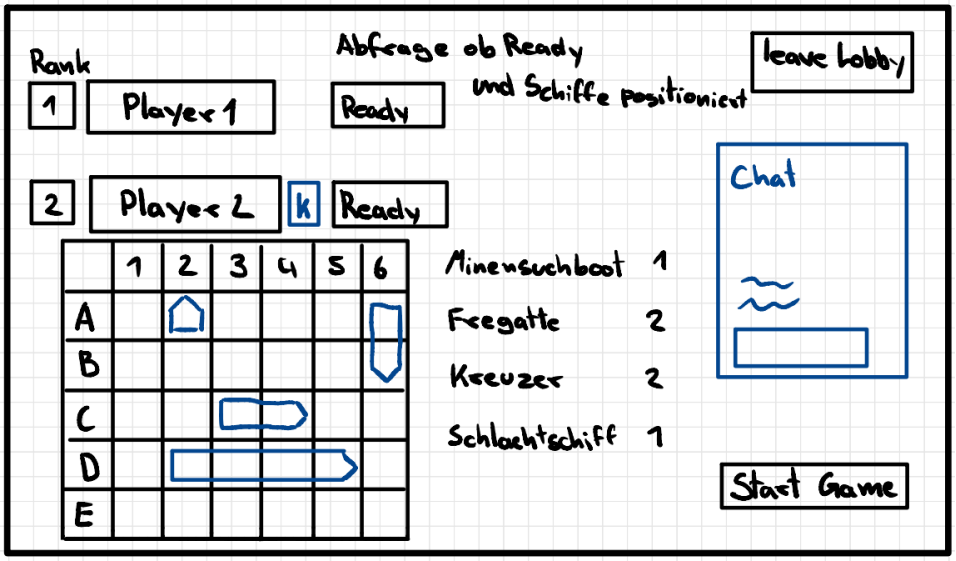

Back
Save File 1
Save File 2
Save File 3
Next

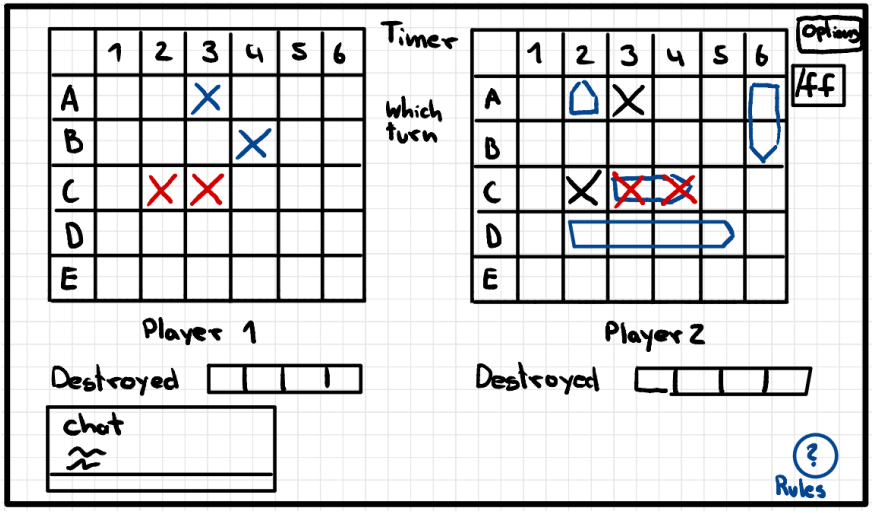

Back
Singleplayer
Multiplayer
KI vs KI


Host Server
Back
Server Name
Server Type
Public
Local
Server Password
create Server

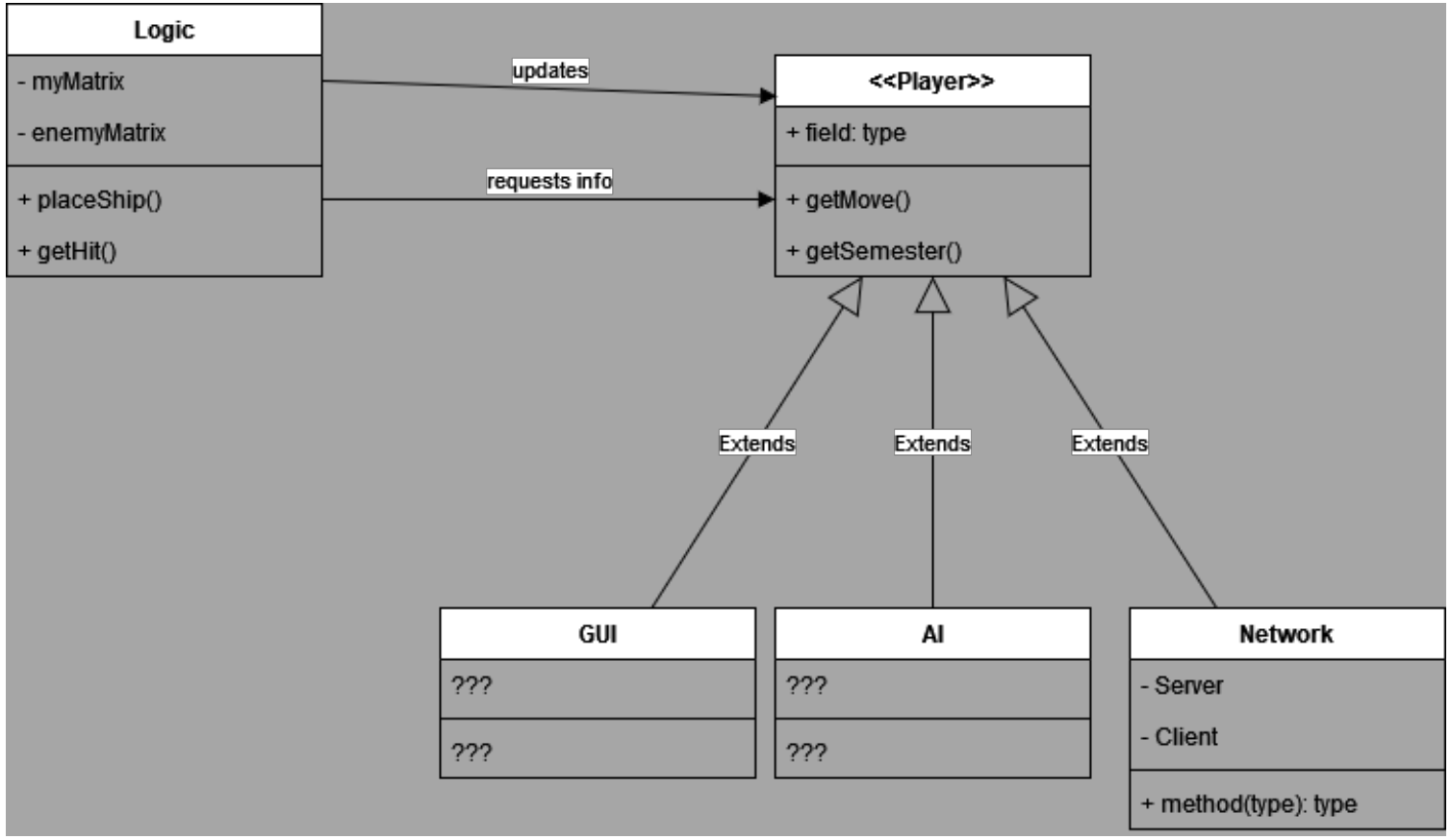
GUI


search
1/2
1/2
2/2
Password
Back
Search Server
Server Name
Server Password
search Server


Rank
1
Player 1
Ready
Abfrage ob Ready und Schiffe positioniert
leave Lobby
2
Player 2
k
Ready
Chat
1 2 3 4 5 6
A
B
C
D
E
Minensuchboot 1
Fregatte 2
Kreuzer 2
Schlachtschiff 1
Start Game


1 2 3 4 5 6
A
B
C
D
E
Timer
which turn
Player 1
Destroyed
chat
Player 2
Destroyed
Rules

UML


Logic
- myMatrix
- enemyMatrix
+ placeShip()
+ getHit()
updates
requests info
<<Player>>
+ field: type
+ getMove()
+ getSemester()
Extends
Extends
Extends
GUI
???
???
AI
???
???
Network
- Server
- Client
+ method(type): type

Project Analysis

- Completely captured use cases+GUI prototype.
- Layered architecture (min. 2 layers; GUI and logic separated!).
- Six semesters (game stages/levels)
- The game must have Sound.
- A "surprise" in the game.
- Project planning.
- Careful documentation.
- Error-free running, crash and application-safe game!
- Runnable in a window (JavaFX).
- The game must be playable against the games of the other project groups.
- Binaries as JAR file.
- if necessary, with start script for Linux and Windows.

Project Requirements

- A ship can be placed horizontally or vertically.
- The matrix is at least 14 x 14 fields and increases with each further semester by one row/column.
- 1st semester: 14x14, 2nd semester: 15x15 etc
- The game represents all six semesters of the BIN course (i.e., there are six levels)
- Opponents in the game can be both human and computer.
- Possible pairings Human vs human, human vs. computer or computer against computer.
- Courses are represented by ships.

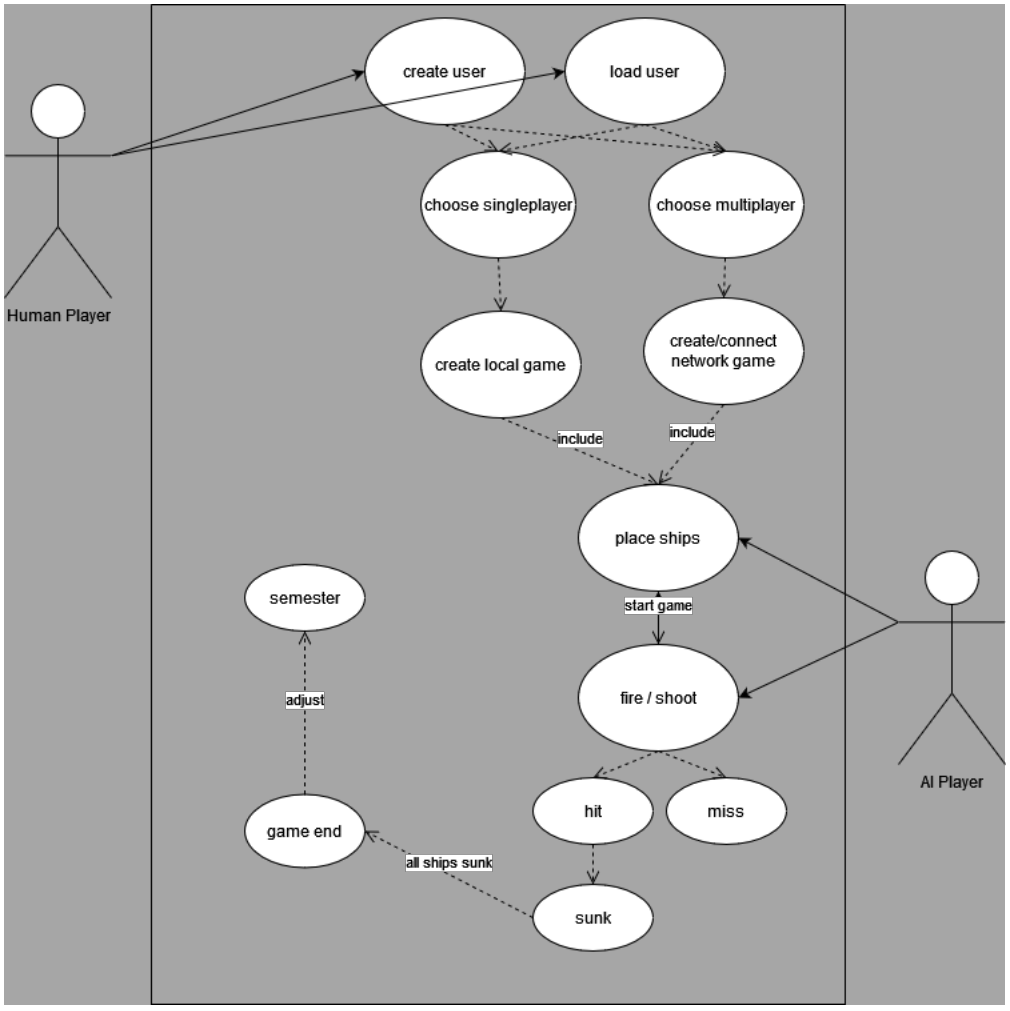
Project Analysis

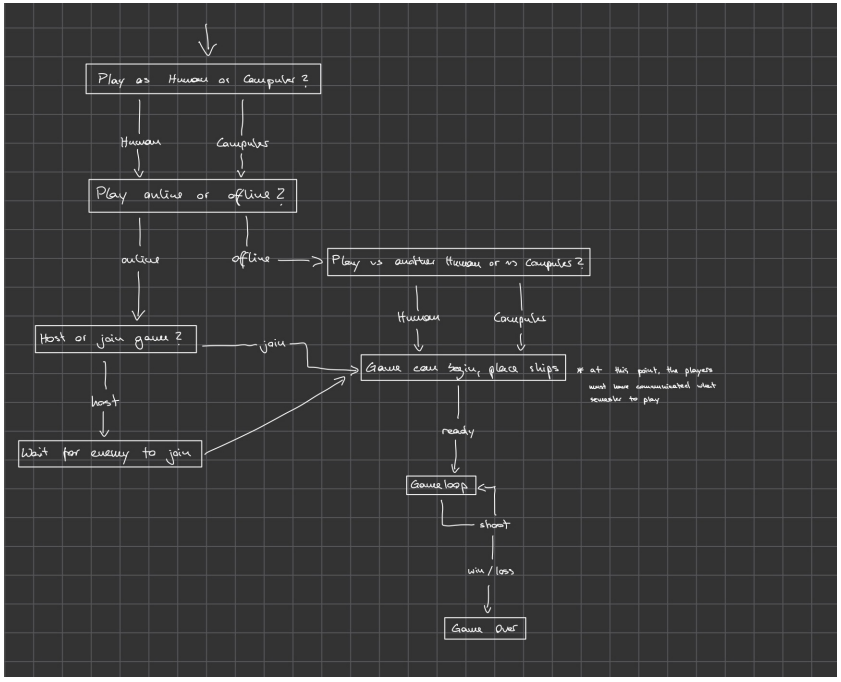
Use Case

The use case diagram consists of the two users, player and computer. The human player has the actions create and load user. The human player also has the options to create or join a server from the multiplayer action. It is necessary to place the ships at the beginning of the game and both players have the fire and shoot actions. This action shoot, itself, is extended by the states hit and miss. The state hit is extended by the state sink. When all ships are sunk the result is the game end state, which in turn leads to a higher semester state for the winner of the level.

Project Analysis

Use Case/Flow Diagram





Schedule

Requirements→ 04.11

finish defining communication interfaces on how to communicate between modules/teams

- ai: random agent, no intelligence, only able to take part in game
- network: provide all needed communication for basic game
- gui: 1st gui frame able to display gamestate (2 matrices, turn, input (not yet working))

- all teams: build one proof of concept prototype together, limited features but able to create games with 2 players (human vs. human/ai/network), simple limited features gui, not yet "playable" - logic: support basic game loop communication (shoot, answer, repeat), being able to set ships, but without basic rules (being one field apart other ships)

Schedule Requirements → 11.11, →18.11

finish defining communication interfaces on how to communicate between modules/teams

- logic: add basic rules
- logic: add basic rules, add levelselect

- gui: working gui frame for main game input and display

- network: finish v1 protocol

- all teams: build first working prototype, playable game with all parties (human/ai/network) support semester functions

- logic: support all rules

- logic: support all rules / being able to save and load savestate

- gui: add gui to create/join etc

- ai: pseudo random agent, "hunts" hit ships

- - add all documents to git

Schedule Requirements → 16.12

finish defining communication interfaces on how to communicate between modules/teams

- finish fixing bugs - finish fixing bugs