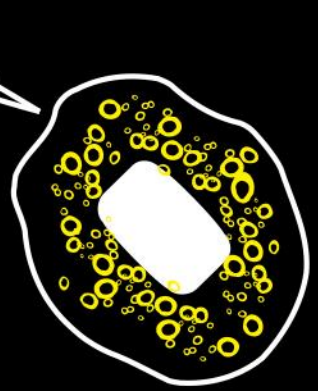
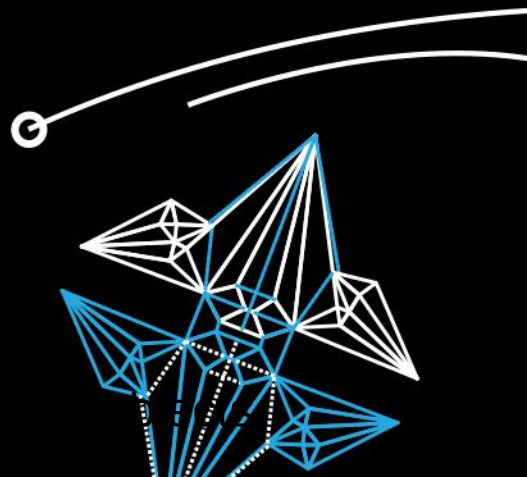
UNIVERSITY OF TWENTE.



Module 2 Project Reveal & Kick-Off



OVERVIEW

1. What is the game?
2. Organisation & grading
3. How to get started

WHAT IS THE GAME?

WHERE TO GET THE GAME

- Marktplaats
- 3D print

ORGANISATION

Boring but important

READING TO DO

- Description: manual pages **17 – 23**
- Submission & grading: manual pages **21 – 23** and **133 – 148**
- Activities & deadlines: manual page **23**

- Game rules: on Canvas
- Communication protocol: on Canvas

- **Task #1: READ!!**

WHAT ARE YOU MAKING

Functionality

- Server that can support many clients and games at once
- Client with a user friendly text interface (your family&friends should be able to play)
- AI that can play games automatically

Functionality is only 20% of the grade

WHAT ARE YOU MAKING

Software

- Testing!
- Application of OOP principles
- Good design and use of patterns
- Clean code, Javadoc, comments, JML

WHAT ARE YOU MAKING

Report

- Testing! (also in the report)
- Explanation (with justifications!) of design and concurrency
- Reflection

OVERVIEW

- Week 6 + 7 + 8: programming exercises during lab sessions (deadline Wednesday week 8)

- Week 7: **planning** with a TA (**P-7.1**)
  - the more detailed your planning is, the better the feedback

- Week 8: **submit initial design** via Canvas (Wednesday)
  - completely up to you, but the more you do, the better your **reflection**
  - no initial design = no points for design reflection

OVERVIEW

- Week 7: **planning** with a TA (**P-7.1**)
- Week 8: **submit initial design** via Canvas (Wednesday)

- Week 9: **midway submission** via Canvas (Wednesday, for 0.5 bonus points)
  - All game rules are implemented
  - Javadoc: all public game logic classes/methods
  - Comments: all complex methods
  - Unit tests for game logic
  - Initial report with test plan

OVERVIEW

- Week 7: **planning** with a TA (**P-7.1**)
- Week 8: **submit initial design** via Canvas (Wednesday)
- Week 9: **midway submission** via Canvas (Wednesday for 0.5 bonus points)

- Week 10: **final submission** via Canvas (Wednesday)
  - Each day late is -1.0 max project grade
  - **Check your submission after you submit!**

OVERVIEW

- Week 7: **planning** with a TA (**P-7.1**)
- Week 8: **submit initial design** via Canvas (Wednesday)
- Week 9: **midway submission** via Canvas (Wednesday for 0.5 bonus points)
- Week 10: **final submission** via Canvas (Wednesday)

- Week 10: **tournament** (Friday to win 1 or 0.5 bonus points)
  - participation mandatory (*unless you resit calculus*)
  - tournament per house (minor and resit separate houses)
  - winners of each house get 1.0 bonus point; runners-up get 0.5 points
  - *probably*: double-bracket best-of-5
  - **rule: your AI has to be your own**

PROGRAMMING PAIRS

- Project is done by pairs: you work with your programming partner
- **No more changes after week 7**

PAIR PROGRAMMING

- Not mandatory
- Recommended: you will both know the code better and there will be fewer bugs

GIT REPOSITORIES

- Mandatory use of **private gitlab.utwente.nl repositories** created by your group TA

- Commit often
- **Push commits daily**
  - including report!

- Reasons:
  - TAs can monitor progress
  - Backup in case you lose data
  - Good practice to commit often
  - Quality of commits is **not graded**

BONUS POINTS

Bonus points earned by:

- Ranked leaderboard (+0.3)
- Chatting (+0.3)
- Encryption (+0.4)
- Authentication (+0.4)
- Midway submission (+0.5)
- Tournament (+1.0 / +0.5)

Bonus points only count after passing the project (>= 5.5)

QUESTIONS

- Daily project sessions 13:45 – 15:30
- Discord #qa-programming-project

GETTING STARTED

How to approach the **design** of the project?

REQUIREMENTS

- Requirements are given (no interview etc.)

- Think about:
  - How will you test/verify that you fulfill the requirements
  - Client ≠ Server
  - Making a simple AI is not hard

- Make use of what you learned in the lab exercises

WANT TO CODE IMMEDIATELY?

- “Design as you code” is bad
- “Think before you code” is good

INITIAL DESIGN

- Some “design decisions” are truly bad

- Design decisions have consequences
  - some are easier to program
  - some are easier to test
  - some are easier to adapt

- Design pros and cons in report

GETTING STARTED

- Things to consider:
  - "how am I going to check if a move is valid?“
  - "how am I going to check if there is a five-in-a-row?“
  - “how am I going to do the rotation?”

REPRESENTING THE GAME BOARD

- How to represent the Board?
  - A two-dimensional array of Cells?
  - A one-dimensional array of Cells?
  - A Map<Marble, Cell> or Map<Cell, Marble>?
  - Maybe just a list of Marbles and each Marble has an int position?
  - Maybe four Subboards with 9 Cells each?
- How to do subboard rotation? Swap the Cells explicitly? Or as a Subboard property?
- How to represent an empty space?
- Is the marble an enum Marble.BLACK, Marble.WHITE, a subclass BlackMarble is-a Marble, an int?

GETTING STARTED

- How to represent players vs a game
  - Player has-a Board?
  - Board has-a Player?
  - Something else?
- Is AutoPlayer a Player?
- Is Move a class or something else?
- How do you want other parts of the program to use your game implementation?

- These are all possible choices. Each choice has consequences

TESTING WHILE DEVELOPING

- Reference server: 130.89.253.64:55555
- Reference client (week 7)

SUCCEEDING

If you want to succeed…

- Read before you think
- Think before you code
- Consider the bigger picture (all parts of the program, everything needed to pass)