# SNAKE

## Software Development Methods

Lucas Jakin
Francesco Rumiz
Lorenzo Giaccari
Francesca Craievich

# THE GAME

The goal is to control the snake and collect food to make it grow.

## Controls

Move in 4 directions (up, down, left, right)

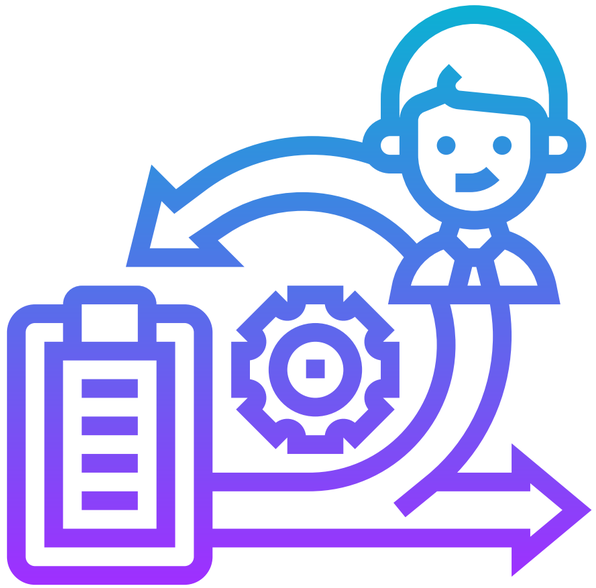
## Game over

The game ends if the snake hits the walls or itself

## Score

Each food collected increases the snake’s length and score

# WORK FLOW



## Planning

We defined how Snake works and identified all possible tests

## Setup

Created a GitHub repository and set up CI for the project

## First steps

Our first step was writing tests and creating a basic test board

# TESTS & FUNCTIONALITY

We followed the Scrum method, using a Product Backlog with features and tests for Snake. Weekly sprints and daily standups helped track progress and improve our process.

## ParseBoard
## ParseCellParser

- Board’s structure
- Correct placement of walls, snake, and food
- Verify cell status

## GameOverConditions

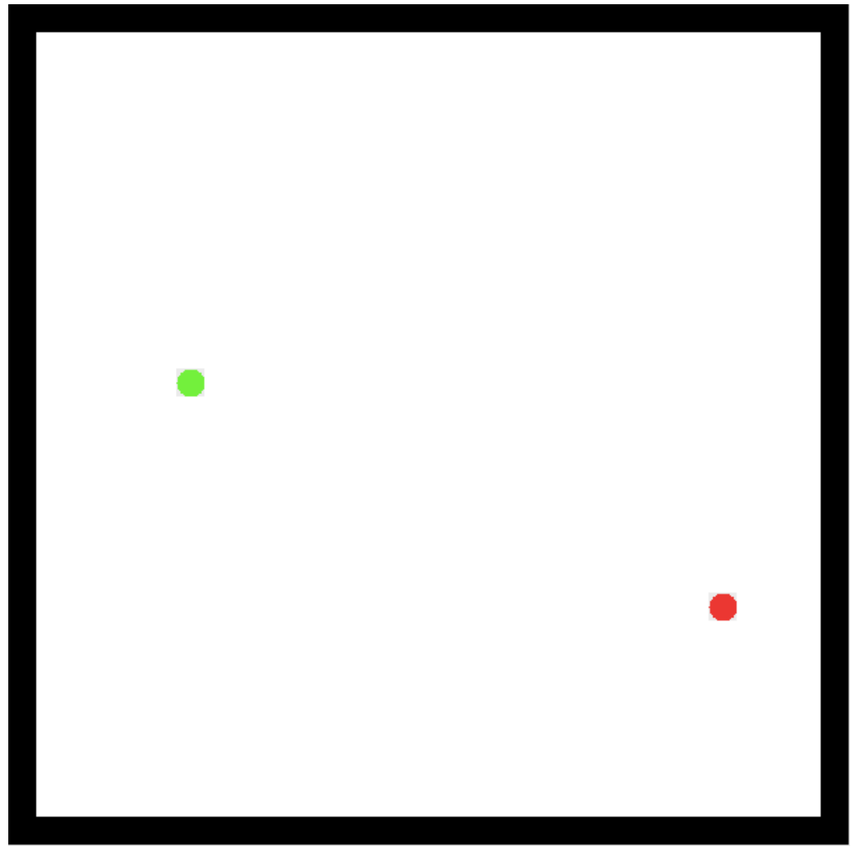
- Wall collision
- Self-collision

## ParseSnake
## ParseSnakeMovement

- Correct snake position updates
- Body growth
- Movement in all directions

# BOARD

- It’s the **game environment**
- Two private fields: **BOARD_SIZE** and **board**, a matrix of Cells (BLANK, SNAKE, FOOD, WALL)
- Two constructors: one standard, one for tests
- Food generation



# BOARD TESTS

## ParseBoard

- Focus: Board class
- Ensures correct construction of Board

## ParseCellParser

- Focus: Cell enum
- Uses the test constructor of Board to ensure validity of cell assignment

# COORDINATE

- Identifies a **specific cell of the Board**
- **Clearer code**
- Two private fields: **x** and **y**
- Three constructors: coordinate creation from different data types
- plus method: sum two Coordinates
- Designed to be **intuitive** and **futureproof**

# THE SNAKE

How the Snake is handled in the program.

## Class Snake

- ArrayList of Coordinates
- Getters and Setters

## Class SnakeMovement

- Movement of the Snake
- Uses various classes

# SNAKE-MOVEMENT

- **Constructor**
  - Board as input
  - Initializes Snake
- **GameOver**
- **EatFood**
- **CheckDirection**

- **MoveSnake**
  - Direction as input
  - Different behavior if encounters food
  - Updates Board and Snake

# SNAKE TESTS

- **PARSE SNAKE MOVEMENT**
  - Test for each direction
    - Asserting Cell values and CoordSnake length
  - Test for eating food

- **PARSE SNAKE**
  - Test for getting CoordSnake
  - Test for getting Coord tail of Snake

# GRAPHICAL USER INTERFACE

- Straightforward and user-friendly flow
- Design uses simple retro aesthetic
- Three main panels:


## Main Menu

Features two buttons:

- **New game**
- **Exit game**

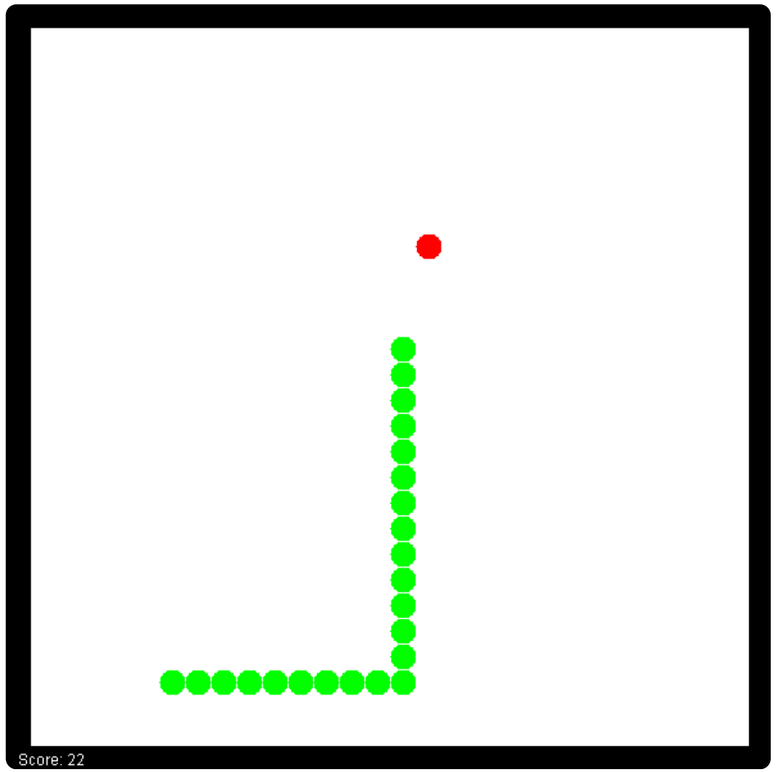
## Game Board

Game board consists of black bordered white screen composed by cells

## Game Over

Always checking Snake’s movement and collision cases

# GAME PANEL




## Interface transition

Once player starts the game, transition to **game panel**

## Movement

Player controls the Snake towards food using **arrow keys**

## Gameplay Thread

- Separate **game loop** from UI
- Handling speed adjustments
- Continuously updating **game board**
- Updating score and increasing snake length

# GAME OVER

## Game Termination

- isGameOver() - collision detection
- **GameOver** or **Victory** screen

## New Game

- UI dynamically updates to final panel
- **“Back”** button allows player to restart game
- New **game session** is established ensuring fresh start

# THANK YOU FOR THE ATTENTION!