Division of work on this assignment

	Kresten s235103	Peter s235069	Jonathan s235115	Oscar s224752	Jamie s236939	Martin s214406
Workload	1.4	1.1	1.4	1.1	1.0	0.0

Contents

Analysis and preparation

Glossary

- Racing Course (map): The playing area of the game. There are various racing courses available, each with its own layout and features.
    - Starter Course: A beginner-level racing course designed for players new to the game.
    - Intermediate: more challenging board elements and obstacles compared to beginner courses.
    - Advanced: higher level of strategic challenge, featuring cramped spaces and intense interaction with board elements and rival robots.
    - Robots Must Die: Hardest level of challenge
- Board
    - Timer: Used to track time during the game rounds.
    - Priority Antenna: A game element that determines the turn order for players.
    - Board Elements: Elements on the game board that activate at the end of each register and affect robots sitting on them.
    - Checkpoints: Marked locations on the racing course that robots must reach to progress.
    - Reboot Token: Indicates where robots can reboot after taking certain types of damage.
- Robot: An entity that is controlled by players.
- Player mat: A mat used by players to organise their robot’s programming deck and other resources
- Programming deck: A deck of cards that represent actions/ movements that a robot can take
- Energy Cubes: Resources placed on the racing course that robots can collect and utilize
- Rebooting: A process where a robot resets after falling off the board, into a pit, or activating a worm card.
- Discard pile: A pile where used or discarded cards are placed.
- Upgrade Phase: Players use energy cubes to purchase upgrades for their robot, based on priority.
- Programming Phase: Players program their robots simultaneously

- Activation Phase: Players activate their robots, and carry out its programming. Board elements activate as well.
- Special Programming Cards: Cards with unique effects that are not specific to any particular robot.
- Damage Cards: Cards representing different types of damage that can affect robots in the game.
    - SPAM: acquired when a robot is shot by a board or robot laser.
    - VIRUS: forces nearby robots to take a virus card.
    - WORM: requires the immediate reboot of the player's robot.
    - TROJAN HORSE: causes the player to take two SPAM damage cards immediately.
- Upgrade Cards: Cards representing upgrades or enhancements that robots can acquire during the game.

Game elements for domain model

Progress/round:

Upgrade phase
Programming phase
Activation phase (robots move "simultaneously", move 1 for all after priority, then move 2 and so on)

Player:

Robot
Player mat (energy cubes, the two kinds of upgrade cards, card registers)
Programming deck
Discard pile

Board:

Priority antenna
Blank spaces
Start spaces (gears)
Walls
Pits
Checkpoints
Reboot tokens
Activation spaces(still part of board):
Lasers(1,2,3)
Gears(turn 90 degrees)
Push panels
Conveyor belts

Cards:

Individual programming cards
Special programming cards
Damage cards (split in SPAM, Worm, Virus and Trojan Horse)
Upgrade cards (permanent and temporary)

Other:

Card piles
Draw pile
Timer
Checkpoint tokens
Energy cubes
Upgrade shop

Taxonomy

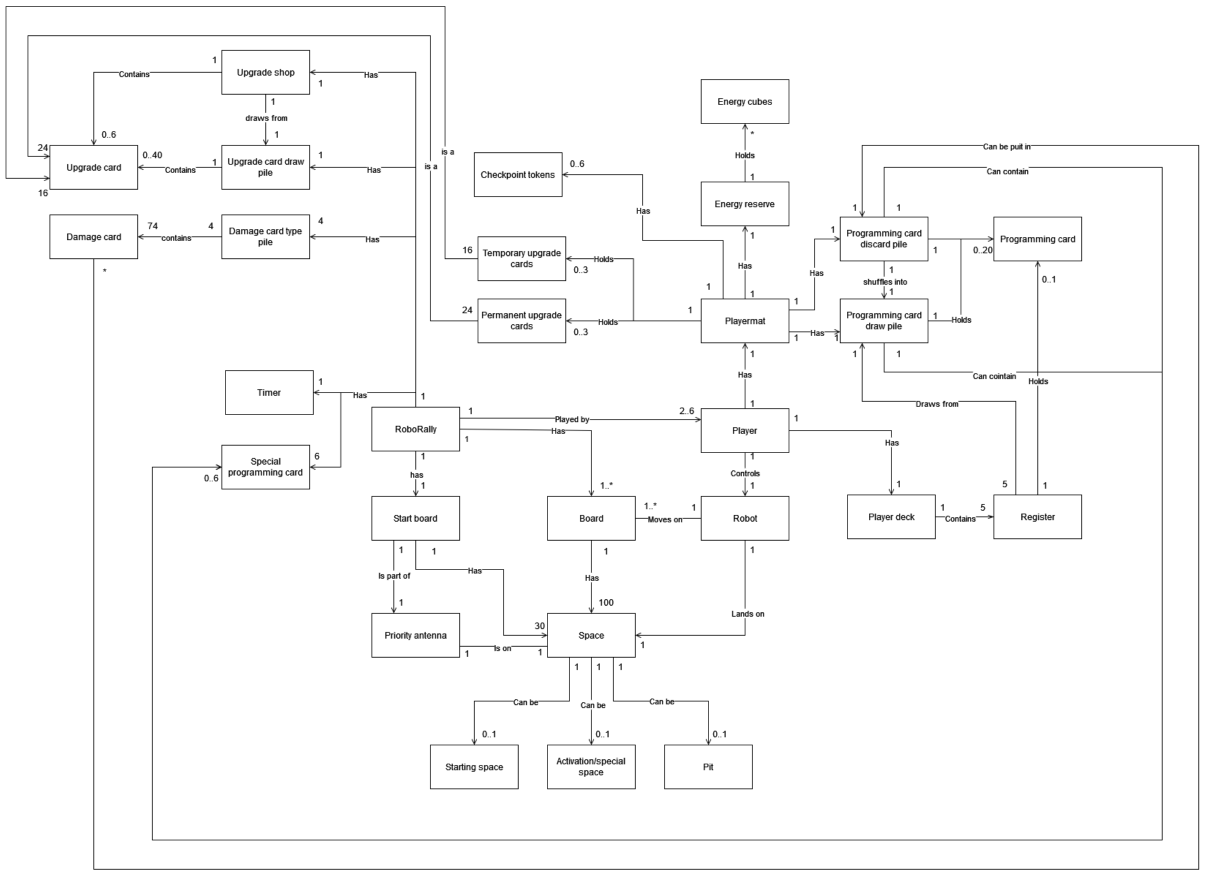
Nouns:

Board elements(walls, laser, conveyer belt, pit, turning gear, pushing panels, energy space)
Card piles
Checkpoints
Damage card
Energy bank
Energy cubes
Energy reserve
Gameboard
Player
Player mat
Priority antenna
Programming deck
Racing course (map)
Reboot tokens
Registers
Round
Robot (figure)
Startboard
Spaces
Timer
Upgrade shop
Upgrade slots
Upgrade (permanent and temporary)

Verbs:

Start game
Move robot (according with commands)
Draw card
Discard card
Discard pile
Shuffle deck
Use card (upgrade or programming)
Execute board-elements
Power up (robots)
Spend energy
Collect energy
Collect checkpoints
End game
Reboot robot
Pick map
Pick starting position
Push (other) robots
Start timer
Buy upgrade-cards

Domain model



From RoboRally, a connection goes to a group of boards with the starter board as well as other boards that a race course is made of. They each contain fields/spaces that the robots can move on, and the spaces can have different attributes as special spaces like conveyors and checkpoints, or pits and the starting space, where a robot starts. The starting field also contains a priority antenna that is placed on a space/field.

Up to 6 players can play a game of RoboRally at a time. Each player has a mat that contains their programming cards, checkpoint tokens, energy reserve and upgrade cards. Each player has an individual pile of programming cards and 5 registers to place those cards within. These card piles may sometimes contain special programming cards or damage cards depending on the current game.

The game also includes other elements that aren’t part of the board or in a player’s control. Those elements are a timer as well as card piles for upgrade cards with the upgrade card shop, damage cards and special programming cards. The players can then interact with these card piles to draw cards and use them in certain situations.

Game Description

With this project we hope to bring the revolutionary and amazing board game RoboRally to a pc version just like you remember it from the board version. If you don't know, the game "RoboRally" pits 2 to 6 players against each other on a map with various hazards, racing to reach all the checkpoints in order first. To accomplish this, players are dealt "programming cards" that they use to move their chosen champion across 5 turns, where they can choose to advance their own agenda or choose to interrupt and sabotage other players. Repeat this until the ultimate racer robot is found, and you have a winner.

Use cases

Draw programming cards: When the programming phase begins (precondition), all players (actors) must draw 9 cards from their programming card deck. If there aren't enough cards in the pile, shuffle the pile. If 9 cards are successfully drawn from the correct pile, it succeeds.

Place programming cards: After having drawn 9 programming cards (precondition), the player (actor) must place 1 card in each of their 5 registers as the moves for their robot. After all 5 are placed, a timer begins, and the player discards the rest of their cards. If not all 5 registers are filled when the timer runs out, discard current cards and draw a new card for each empty register and place them there directly.

Shuffle programming cards: If the programming card pile runs out, while trying to draw a card (precondition), the discard pile must be shuffled and placed in the draw pile.

Determine player order: When a round begins (precondition), count the distance from the priority antenna to all players. The order then goes from the player closest to the antenna to the player furthest from the antenna. In case of a tie, a line from the front of the antenna turns clockwise to hit the players, and that determines their order.

Buy upgrade cards: At the start of each round (precondition), a player (actor) can purchase a single upgrade card using energy cubes as currency. The order in which players can purchase is determined by proximity to the priority antenna. If the cards can be used during the programming and activation phase it is a success (success scenario). Temporary cards should be single use and permanent cards should be permanent. Cards must only be usable at the designated times (failure scenario).

Activate upgrade card: During the programming and activation phases (precondition) players (actors) can activate their upgrade card, according to the rules specified on that card (success scenario). Players should not be able to activate cards in ways not specified by the card (failure scenario).

Replenish shop: At the beginning of the upgrade phase (precondition) the game (actor) replenishes the upgrade shop. After a replenish there should be as many cards as there are players (success scenario). If no cards were purchased in the last upgrade phase, replace all cards. (success scenario). If there are more or less cards than players something went wrong (failure scenario).

Move robot/take turn(register): When the programming phase is done (precondition) the game (actor) starts executing the first register move in the determined player order (success scenario). If a player activates an upgrade card it has priority over standard rules (success scenario). If a player moves twice before another player moves once (failure scenario). If the robot ignores the environment/field when moving (failure scenario).

End of register: After the end of each register (precondition) the board elements activate in order of element activation priority. If a player (actor) is on an affected field the player moves or gets punished accordingly (success scenario). If a player is affected in the wrong order (failure scenario).

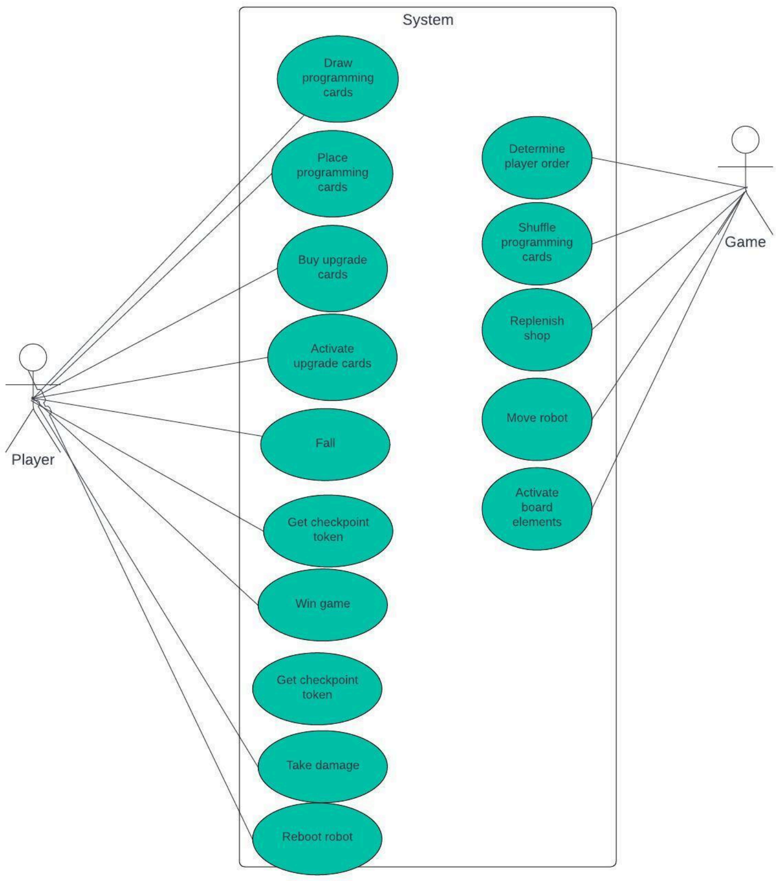
Falling: If during a turn a player (actor) falls into a pit or falls off the board (precondition) the player must reboot their robot at the reboot token (success scenario). If player continues off the map (failure scenario). If the player moves through the fall pit (failure scenario).

Get checkpoint token: When a player lands on a checkpoint field and if they don’t have the corresponding checkpoint token, while also having the previous checkpoint token (they are counted in numbers), they will receive that checkpoint token and add it to their player mat.

Win game: When a player (actor) has collected all the checkpoint tokens in the right order (precondition), the game ends, and they win(success scenario).

Take damage: When falling into pits, being shot or getting knocked off the board or affected by upgrade cards (precondition) players (actors) robot must take damage. When taking damage, draw damage cards and put them in your programming discard pile, take damage when the card is activated in one of the registers (success scenario).

Reboot robot: When a player’s robots needs a reboot because of falling off the course, falling into a pit or activating a worm damage card (preconditions), the player’s robot (actor) takes 2 SPAM damage, the rest of their registers are emptied and the robot is moved to the reboot field on the same board. If the player rebooted on the start board, reboot on their start field. The player can turn the robot in any direction just after the reboot.



The use case diagram depicts the various interactions within the RoboRally game system. It illustrates the functionalities available to both players and the game itself, providing a high-level overview of how users and the system interact during different phases of the game, including upgrade, programming and activation phases.

Features

Moscow

Must have: Critical functions for the game to be playable.
Should have: Essentiel parts of the game.
Could have: Smaller details that improve the game or somewhat significant parts, that are very time consuming to implement
Won’t have: Excessive features that are too time consuming to implement.

Must have:

- Board: A general board with fields, for the robots to move on
- Robot movement: Robot movement on the board according to instructions as well as landing on fields

- Player mat with registers: A player specific area with 5 register, as well as space for upgrade cards and programming card piles
- Checkpoints and checkpoint tracker/tokens: Checkpoint fields on the board with numbers and a counter for players to keep track of checkpoints reached
- Programming cards and piles: Programming card decks, draw piles and discard piles

Should have:

- Special fields (Wall, Laser, Push Panels, Gears, Conveyors, Pits, Checkpoints, Energy Spaces, Priority antenna): Fields with special attributes, that affect the robots in various ways when landed on
- Determine priority (player order): Determines the order of play for robots based on position relative to the priority antenna
- Damage cards: Worm, Trojan, SPAM and Virus damage cards, that is mixed into discard pile for programming cards, so they are drawn later and affect the robot in various ways
- Programming card shuffle: Shuffle programming discard pile to randomise drawing during the programming phase
- Multiple maps: Multiple different maps for the players to choose from
- Menu: Menu for starting the game, selecting amount of players and map and maybe changing certain settings
- General user feedback (log or something else): Log of event or animations to show what is happening clearly to the players

Could have:

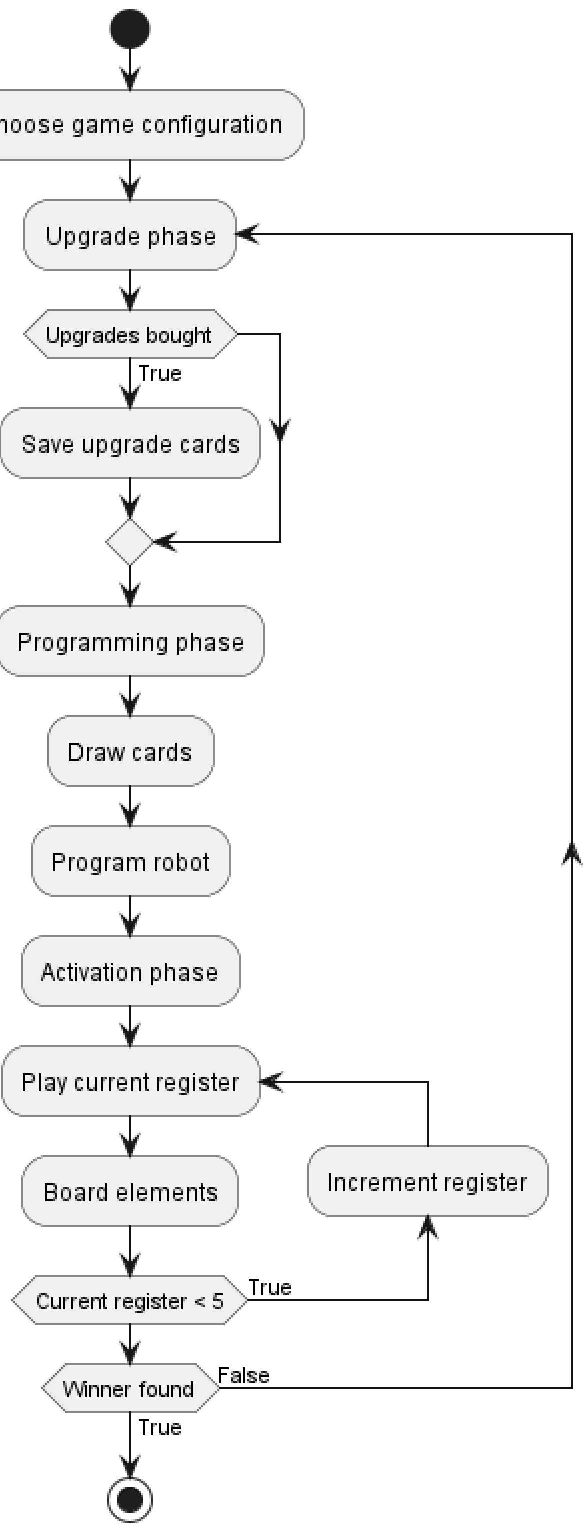
- Upgrade cards: Permanent and temporary upgrade cards, that affect the player during the programming or activation phase
- Energy cubes: Collecting energy cubes as currency to buy upgrade cards
- Upgrade shop: Upgrade card pile and shop for buying upgrade cards with energy cubes during the upgrade phase
- Special programming cards: Special programming cards, that can be collected with upgrade cards, that works as programming cards, but with special effects
- Timer for programming phase: A timer that activates, when the first player finishes programming. When the timer finishes, any players that haven't finished programming, will have random cards assigned to the remaining registers
- Custom maps: An option to make custom maps in a map creator
- Music: Background music
- Sound effects: Sound effects for various actions in-game
- Save/load: Being able to save the state of the game to open and continue playing after having closed the game

Won't have:

- Online multiplayer: Multiple people playing on different devices
- Multiple language support: Translation to other languages

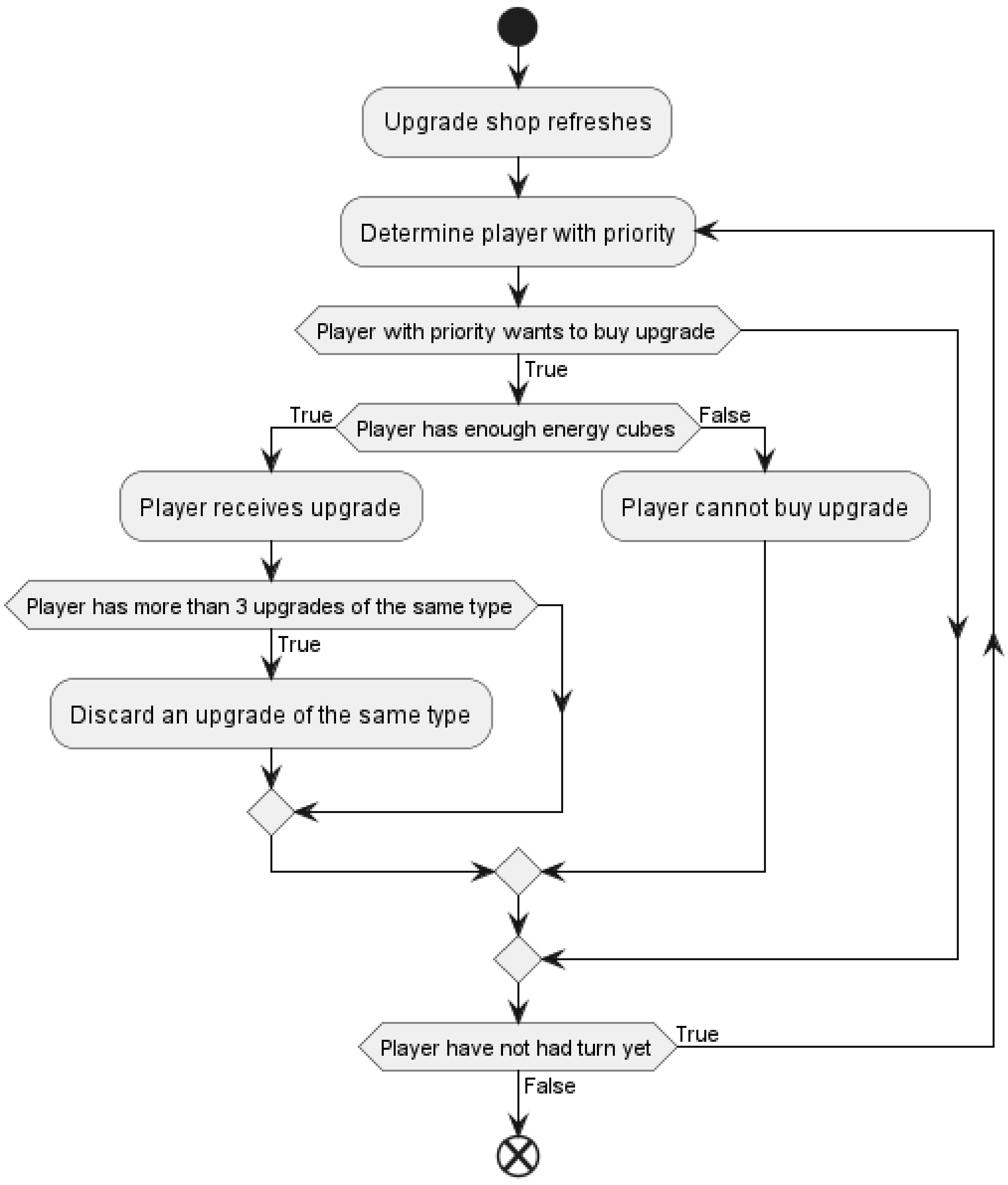
State and activity diagrams

Gameflow activity diagram



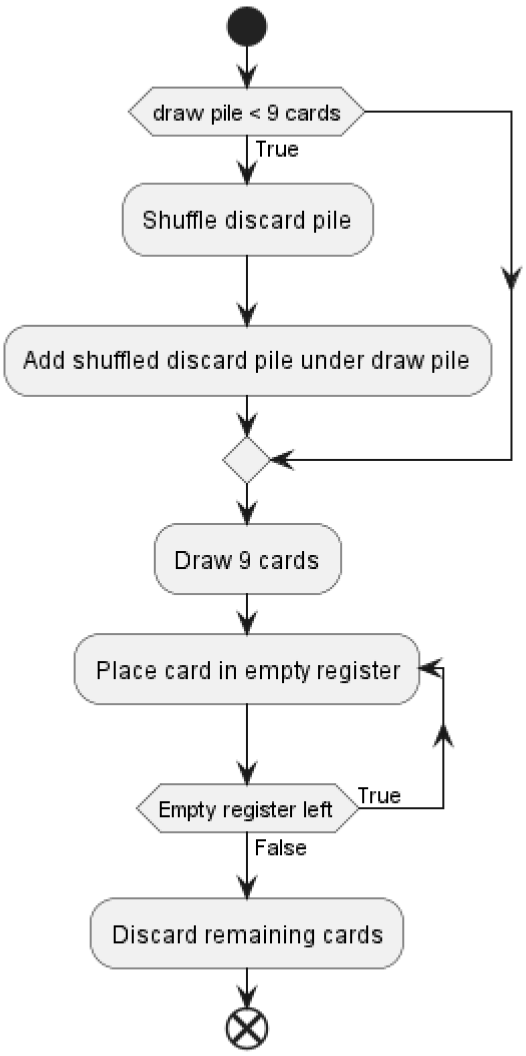
The gameflow activity diagram outlines the sequential flow of actions and phases within the game. It begins with the upgrade phase, during which players have the option to purchase upgrade cards. Next, the programming phase allows players to draw cards and program their robots with specific actions. Subsequently, the activation phase executes the current register's actions and activates board elements accordingly for every player. This continues for all five registers of each player. The game continues until a winner emerges, at which point the game ends, or else the loop continues for the next round.

Upgrade phase activity diagram


Upgrade shop refreshes
Determine player with priority
Player with priority wants to buy upgrade
True
Player has enough energy cubes
True
False
Player receives upgrade
Player cannot buy upgrade
Player has more than 3 upgrades of the same type
True
Discard an upgrade of the same type
Player have not had turn yet
True
False

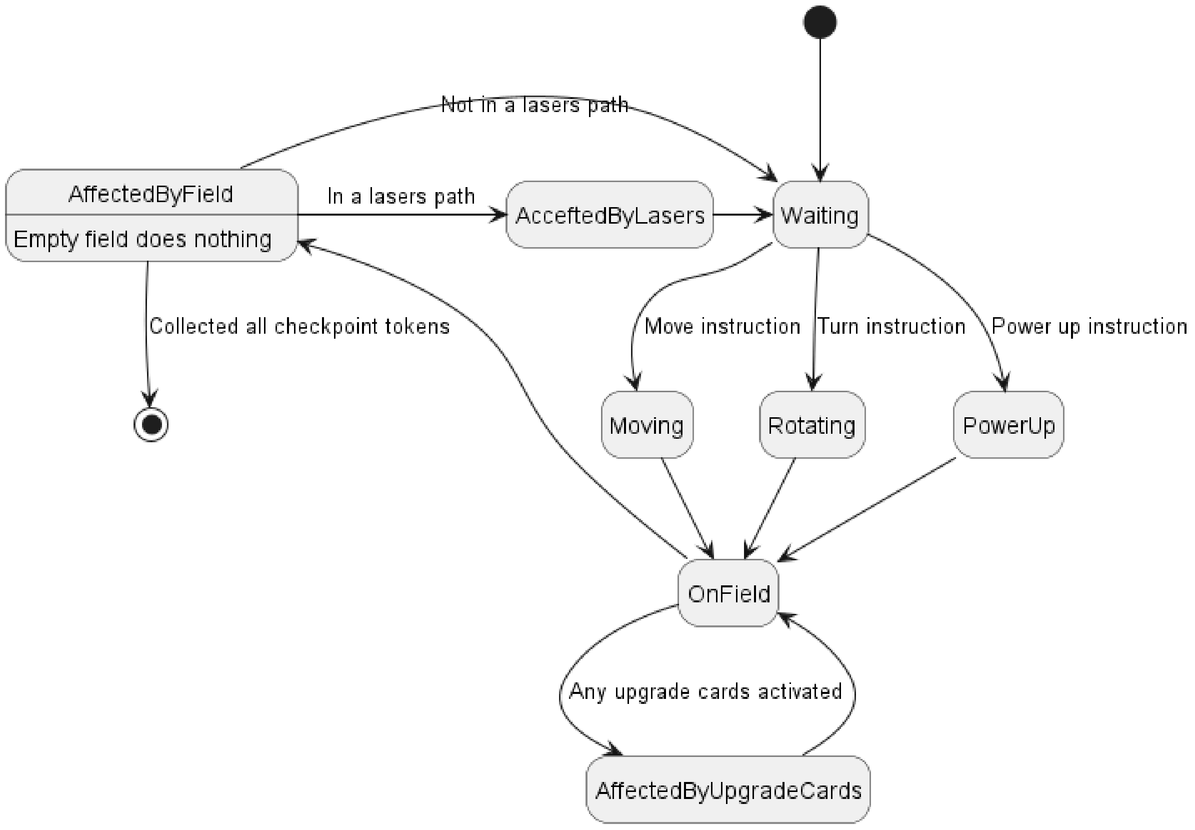
At the beginning of every upgrade phase the shop replenishes its items, each player in order determined by proximity to the priority antenna gets a chance to buy one upgrade card. If a player has enough energy cubes they are allowed to buy the selected upgrade card. After a purchase or refusal to buy the turn passes to the next player. If a player has more than 3 of a permanent or temporary upgrade they must discard one card of that type.

Programming phase activity diagram



At the start of the programming phase each player must draw 9 cards. If there are less than 9 cards left in the draw pile the discard pile is shuffled and added to the bottom of the draw pile. Then 9 cards are drawn. The cards will then be placed in empty registers one by one until all empty registers are filled. Then the rest of the 9 programming cards are discarded, and that player’s programming phase ends.

State diagram for the Robots



The state diagram visualises the state of a robot during the game, and how it behaves in the activation phase. It starts by waiting for instructions from the registers, which it gets at the start of the activation phase, it then powers up, moves or turns based on the instruction.
It ends up on a field, where it starts by checking for activation of any upgrade cards, and are affected by them. When no upgrade cards are left, it is affected by the field it is on, there might not be any effect, and then it is affected by lasers, if the robot is in a laser's path. At the end, it moves back to the waiting state to wait for the next instruction.
If the field it is affected by is a checkpoint, and all checkpoints are collected, then the game ends.

Non-functional requirements

Performance:

- There shouldn't be delays on user actions of more than 0,5 seconds

Scalability:

- It should be possible to add new upgrade cards based on a generic template
- It should be possible to add new programming cards based on a generic template

Usability:

- There should be a tutorial for how the UI works in-game and a reference to the game rules
- The users should be able to play the game without having game breaking misunderstandings (Tested with user test and conversation with the tester)

Maintainability:

- It should be possible to implement multiplayer support
- It should be possible to implement a server to through a server instead of through a client

Documentation

Javadoc

Java doc can be found at this path: src/main/resources/javadoc in the repository.