COMP20050 – Software Engineering Project II

HIGH LEVEL DESIGN

Sneha Chhipa - 22320616
Pallavi Thapliyal - 22206412
Neha Abey - 22342831

GROUP 20

High Level Design

Tasks are assigned but all members will be helping out each other if needed and to ensure everyone is contributing equally.
Anything with "*" next to it means it may be implemented differently according to new game features to account for portability.

Sprint 1 (12th Feb - 23rd Feb)

Feature 1.1 - Board layout with coordinates marked out

- Constructed using JavaFx GUI.
- Implementing visibility logic of the board – (using the set visibility function in JavaFx to show/hide components of the board when required.)
- Having two separate boards running with the logic. (one to be used for each round.)

Collaboration on this feature by: Neha & Sneha

Feature 1.2 - Implementing "Welcome" screen to the user.

- Prompts option to play against the computer (opt 1) or separate player (opt 2)
- If against the computer, the user is automatically set as the experimenter and is asked to input in their name. 2 rounds are played with each player getting a turn to be experimenter,
- If against separate player, the game prompts the two players to insert their name.

Collaboration on this feature by: Pallavi

Feature 1.3 - Placing the atoms and game commencement.

- If (opt 2) chosen - Setter chooses placement of atoms by using coordinates/ by clicking a numbered hexagon.*
- Also to implement the display of dotted circles around each of the atoms - to determine how rays will bounce off/back.
- If (opt 1) chosen - Computer will randomly generate coordinates.
- Board clears and blank board is shown to the experimenter.

Collaboration on this feature by: Pallavi

Feature 1.4 - A feature that displays the in-game options.

- Show - shows the hidden location of atoms and the circle of influence. (Useful for testing purposes)
- Check - shows the location of atoms and ends the round.
- Quit - exits the current game and transfers to the "Welcome" screen.
- New Ray - prompts user to input location of next ray.
- Place Atom - prompts user to input location of where they think an atom is.

Collaboration on this feature by: Neha & Sneha

Sprint 2 (26th Feb - 8th March)

Feature 2.1 - Implementing a ray that doesn't encounter any atoms.

- Exit point is announced to experimenter
- Entry and exit point is marked by numbers/colours.
- The ray data and ray result used by the score method to update the score board.

Collaboration on this feature by: Neha

Feature 2.2 - Implementing a ray that makes a direct hit to an atom.

- Message ‘absorbed’ is printed
- Entry points are marked by numbers/colours. 
- The ray data and ray result used by the score method to update the score board.

Collaboration on this feature by: Sneha

Feature 2.3 - Implementing a ray that approaches one atom and is deflected 60deg.

- Exit point is announced to experimenter
- Entry and exit point is marked by numbers/colours. 
- The ray data and ray result used by the score method to update the score board.

Collaboration on this feature by: Pallavi

Sprint 3 (25th March - 5th April)

Feature 3.1 - Implementing a ray that approaches two atoms and is deflected 120deg (two 60deg deflections)

- Exit point is announced to experimenter
- Entry and exit point is marked by numbers/colours. 
- The ray data and ray result used by the score method to update the score board.

Collaboration on this feature by: Pallavi

Feature 3.2 - Implementing a ray that approaches two atoms and is deflected 180deg

- Message ‘reflected’ is printed
- Exit point is announced to experimenter
- Entry/exit point is marked by numbers/colours. 
- The ray data and ray result used by the score method to update the score board.

Collaboration on this feature by: Neha

Feature 3.3 - Implementing a ray that immediately hits a circle of influence and gets deflected straight back

- Message ‘reflected’ is printed
- Exit point is announced to experimenter
- Entry/exit point is marked by numbers/colours. 
- The ray data and ray result used by the score method to update the score board.

Collaboration on this feature by: Sneha

Feature 3.4 - Considering complex path.

- Exit point is announced to experimenter
- Entry and exit point is marked by numbers/colours. 
- The ray data and ray result used by the score method to update the score board.

Collaboration on this feature by: Pallavi, Sneha & Neha

Sprint 4 (8th April - 19th April)

Feature 4.1 - Displays the full board.

- Atoms - Location of atoms are revealed
- Circles of influence – Location of circles of influences are revealed
- Rays - Rays are displayed.

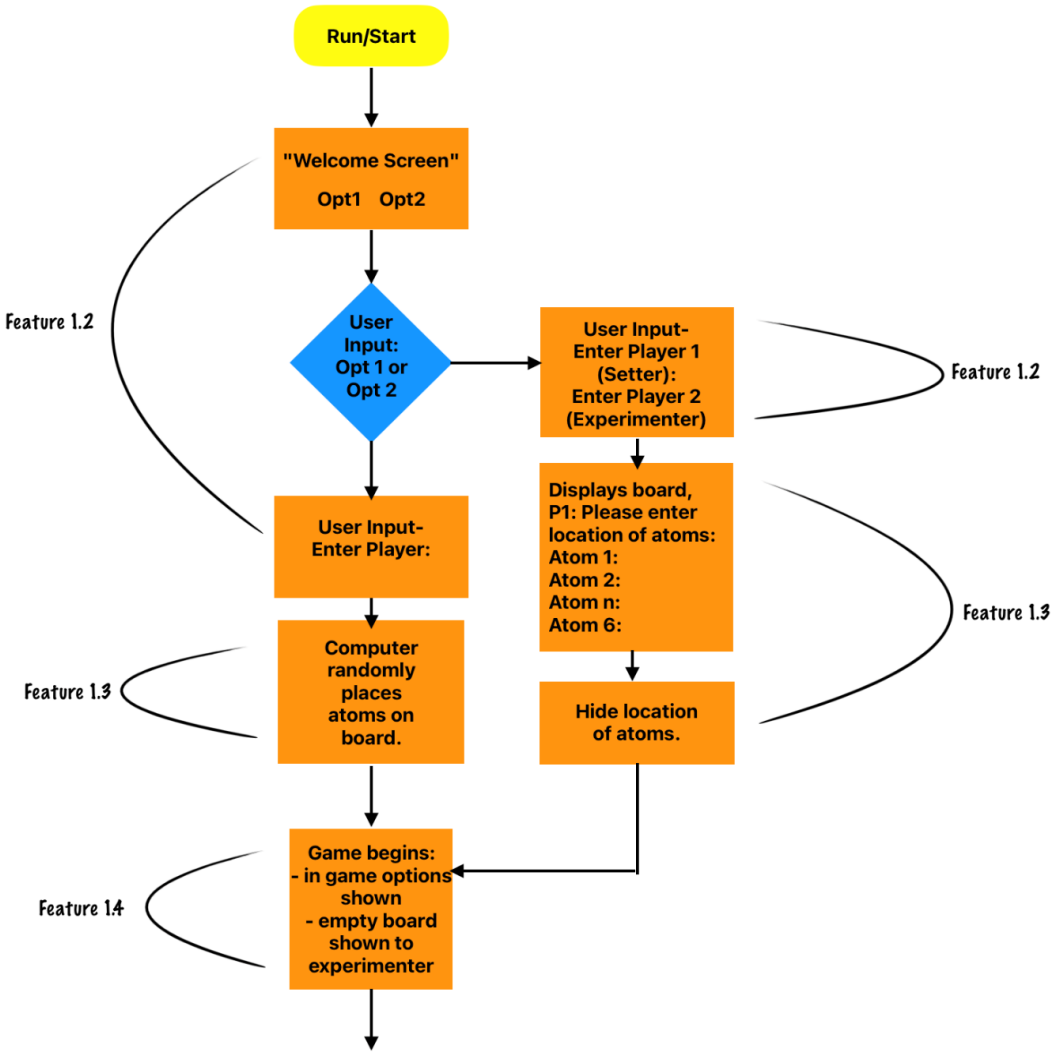
Collaboration on this feature by: Pallavi, Sneha & Neha

Feature 4.2 - Displays a detailed breakdown of scores.

- Score = number of rays + 5(misplaced atom).
- After each round, the score is displayed - including the number of rays shot and the number of atoms placed.
- After two rounds a detailed breakdown is recorded - including the number of rays shot and the number of atoms placed.
- Score is compared and the winner is announced on screen.

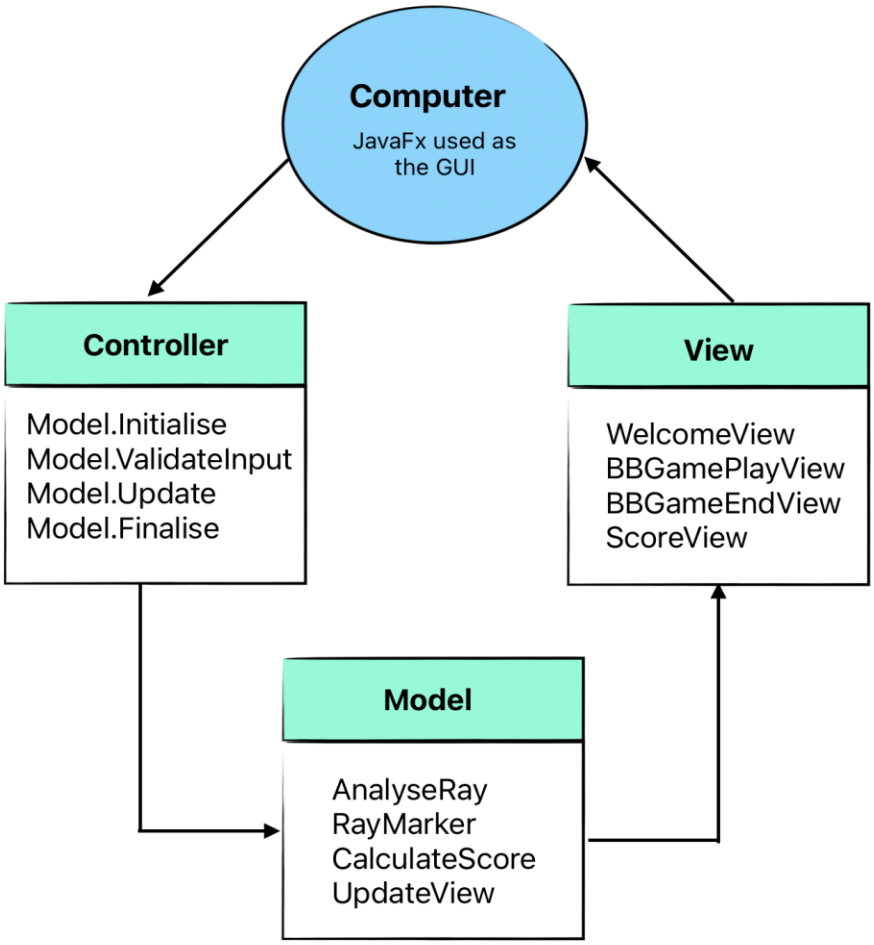
Collaboration on this feature by: Pallavi

Flowchart describing implementation of Sprint 1:



Software Architecture Design

Model-View-Controller (MVC) pattern:



Below is a detailed description of the stages of MVC model.

Controller

Initialise

- *When new game is clicked, Model.Initialise is called*

ValidateInput

- *Checks for valid string name (no blanks/null)*

Update

- *When an input ray button is clicked, Mode.Update is called*

Finalise

- *When experimenter has placed all 6* atoms, Model.Finalise is called*
    - *After round 1, player 1's score and board with all elements is shown. Round 2 commences.*
    - *After round 2, player 2's score and board with all elements is shown.*
    - *Both scores are shown including detailed breakdown of rays and atoms.*

Model

AnalyseRay

- *This looks at the the ray path and calls appropriate decision method ie. directHit, 60degDeflection etc.*
- *As long as the ray path is not clear, the analyseRay method is called recursively.*

RayMarker

- *Ray markers in the form of numbers/colours are placed to indicate entry and/or exit point.*

CalculateScore

- *Calculates individual score for players.*
- *Compares score and announces winner.*

UpdateView

- *Updates view after the setter places atoms and after the experimenter has finished their turn (ie after computer logic has been completed)*

View

WelcomeView:

- *Displays options for players to choose game mode and/or roles.*

BBGamePlayView:

- *Blank board shown to either setter/experimenter depending on gamemode.*
- *In-game options are displayed - place atom/ shoot ray.*
- *Has ongoing details of ray markers during the gameplay.*

BBEndGameView:

- *Details of the game displayed: atoms, circles of influence, rays, ray markers etc.*

ScoreView:

- *displayScore method is called.*
- *Shows basic score after each round.*
- *Shows detailed breakdown of scores after both rounds are played.*

Data Structures

- ArrayList of Integers: Ray paths for every entry point.
- 2 ArrayList (1 for each round) of Lists of Integers: Atom locations along with corresponding circles of influence.
- 2 ArrayList (1 for each round) Integers: Containing locations of meeting points of circles of influence.
- ArrayList Integers: Keep track of separate scores.

Classes and Interfaces

Interfaces

board	***game***
drawBoard clearBoard hideBoard showBoard getBoard	startGame showScore quitGame newGame endGame

Classes

BlackBoard *implements* ***board***	***Score***	***Ray***	***Blackbox*** *implements* ***game***
getCell isEmptyCell setCell	displayScore hideScore CalculateScore	shootRay analyseRay noHit directHit 60dgDeflect 120dgDeflect reflection	genSetterAtoms generateAtoms gameUtility newRay endGame