COMP20050 – Software Engineering Project II

SPRINT 3 - DOCUMENTATION

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

GROUP 20

Documentation

3.1 A feature implementing a ray that approaches two atoms and is deflected 120deg (two 60deg deflections).
3.2 A feature implementing a ray that approaches two/three atoms and is deflected 180deg.
3.3 A feature implementing a ray that immediately hits a circle of influence and gets reflected straight back.
3.4 A feature that considers another complex path.

Sprint 3 Implementation Details

All the features of this sprint were implemented in a similar way, and were instantiated from a single function called detectPath(). The detectPath() is a recursive function which has the following properties:

two base cases:

- if the possible path for the ray is clear
- if there is an atom present in the possible path which would result in ray being absorbed.

recursion:

- if a 60/120/180 degree deflection encountered, calls detectPath() to detect the new changed path.

The below features describes the basic idea of the algorithm, each sub section of the recursive case uses additional functions to complete their tasks.

3.1 A feature implementing a ray that approaches two atoms and is deflected

This function detects if there is a meeting point of two circles of influence in its path. It takes in the direction of the ray and then traces its path depending on the direction.

-- Ray approaches two atoms and is deflected 120 degrees.

If the ray is not a horizontal ray, it uses similar logic from the 60deg logic implemented last sprint to retrieve its path and exit point. If the hexagons beside it are taken up

-- Ray approaches two atoms and is deflected 180 degrees.

If the ray is a horizontal ray, it gets deflected back at its entry point so therefore both the entry and exit point are the same.

Implemented by Sneha Chhipa & Pallavi Thapliyal

3.2 A feature implementing a ray that approaches three atoms and is reflected 180deg.

-- Ray approaches three atoms and is reflected 180 degrees

This function detects if there is a meeting point of three circles of influences in its path. Once it detects this, it traces its path back to the entry point so that both the entry and exit point are the same.

Implemented by Sneha Chhipa & Pallavi Thapliyal

3.3 A feature implementing a ray that immediately hits a circle of influence and gets reflected straight back.

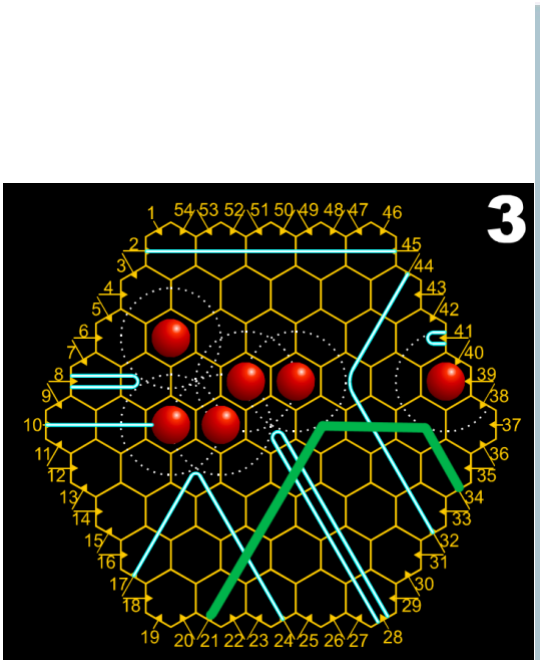
-- Ray that immediately hits a circle of influence and gets reflected straight back.

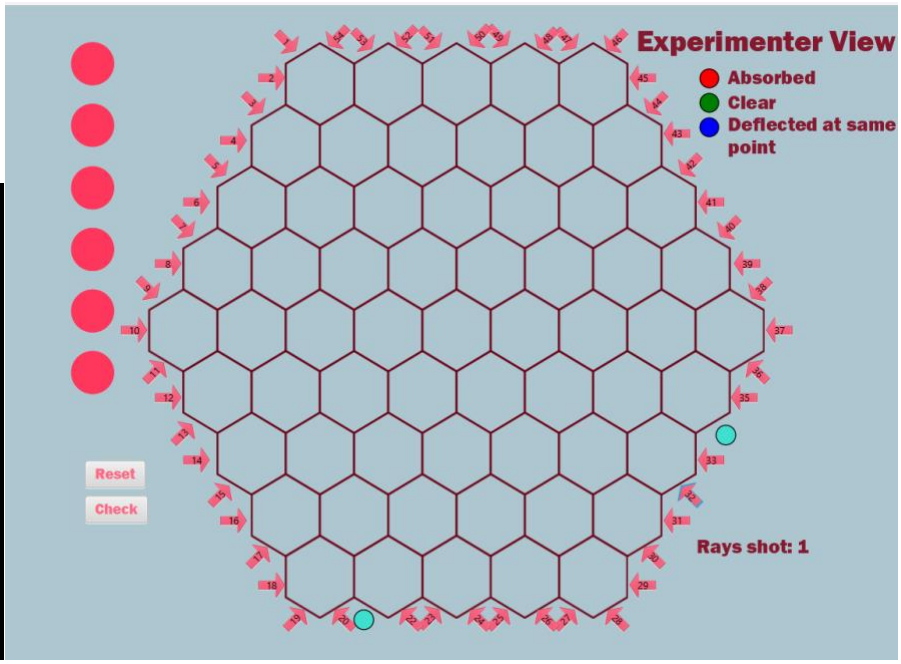
This function detects if a circle of influence is present in the first hexagon in the ray path. If it is, the ray is reflected straight back to its entry point.

Implemented by Sneha Chhipa & Pallavi Thapliyal

3.4 A feature that consider a complex path

Since recursion is being used, it is able to easily detect complex paths. However some paths are quiet complex and therefore the function cannot detect it all. An example of it working is below (path highlighted in green) and it working in our implemented version (the turquoise ray markers show the entry/ exit point of the ray):





Additional Implements

A feature that implements ray markers

-- Implementing ray markers

'placeRayMarker' method was created which takes in the ray as well as the button array. It sets the colour for each ray depending on it's rayType. If the rayType is CLEAR, the colour to green, if the rayType is ABSORBED, colour to red, and if it gets deflected at the same point, the colour is set to blue. For any other rays, the generateRandomColour method was called. Then the drawRayMarker method is called for each the entry and exit points.

The drawRayMarker takes in the entry/ exit point of the ray, the buttonArray and the colour. It creates a circle at the point of the button, by taking in it's x and y values and setting the button to invisible.

The generateRandomColor uses an array of predefined colours and returns the first colour in the array and also removes it so that colours are not repeating.

Implemented by Neha Abey

Other Changes

An approach towards Object Oriented Programming…

In this sprint our focus was to implement classes in order to easily define object components of the game and reduce the working with redundant structures like 3D ArrayList which can lead to confusion when implementing edge case. Therefore, even simple self-define classes like Coordinate, Atom, and Path helps code the ray logic comprehensively. The relationship of inheritance between the Ray class (child) and the Factory class (parent) also allows us to implement clean code practices by avoiding to rewrite methods as the child can access them from the parent class

Implemented by Sneha Chhipa

Simplified Controller Method

A Polygon array was created to hold the hexagons. By doing this, we were able to shorten and simplify the controller method and avoid repetition; by iterating through the array with loops in both the 'getHexID' method and the

methods to draw and reset the atoms. Before it called by each of the hexagons individually and stored its fx:id which was fed to another function to gets its x and y coordinates.

The getButton method was changed the same way. Instead of calling each of the buttons and feeding it the shootRay function with the individual entry points, an array was created to store the buttons and was iterated through using loops.

Implemented by Neha Abey

Testing

According to the project plan, the completed features include features relating to the ray logic, features, and features that are related to implementing the graphical user interface components. The ray logic and correct storage of game data is tested in the file TestGame.java. The testing of the functions is implemented using J-Unit 4. Some of the data structures and logic were also tested using print statements on the command line, to see the coordinates of the ray and if its collision points (where it hit a circle of influence) was right.

Trello

Link to Trello Board:
https://trello.com/invite/b/Pzp4HSxK/ATTI03a5fb51b606b3d9661fb5bd078c2c5d99A9B9EC/sprint-3-blackbox

Below is a screenshot of our Trello Board.