# THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA

## FACULTY OF TECHNOLOGY AND ENGINEERING


**A PROJECT REPORT**

**on**

# Path Finding Algorithm Visualizer

***Submitted by***

**Juvenile Java Pingers (Team of Parth Vaswani)**

***In fulfillment for the course of*** **Java Programming (MCA1202)**

***Of***

**MASTER OF COMPUTER APPLICATIONS**

**in**

**The Department of Computer Science and Engineering**

June 20, 2022.


| Name | PRN |
|---|---|
| Parth Vaswani | 8021070214 |
| Jagriti Singh | 8021069911 |
| Jacquline Khristi | 8021076441 |

# Abstract

This document is intended to report Java mini project by students of Master of Computer Applications – i.e., ours. This report discusses our experience as a software developer in a project from May 2022 to June 2022.

In this project, we have created an application which shows visualizations on 2D graph for various path finding algorithms such as A* (pronounced as A-star), BFS (Breadth First Search) and DFS (Depth First Search) and a maze generation algorithm called Recursive Backtracker.

# Table of Contents

# Preface

After studying Java for a semester in SSMCA, we got to imply our knowledge of coding / programming, teamwork, communication skills, tools, strategies, and creativity into this project.

This project is for a subject of second semester of Master of Computer Applications whose subject code is 'MCA1202'. It must be completed by group of three students.

"Path Finding Algorithm Visualizer" is the project of our group "Juvenile Java Pingers", which includes 3 members (us): Parth Vaswani, Jagriti Singh and Jacquline Khristi. We decided to name our group based on Initials of our First Name, which is J, J and P, respectively. Hence, the name of our group is Juvenile Java Pingers.

This project is designed to gain visual understanding of how the path finding algorithms are working. Which will be used by teachers for teaching students about path finding algorithms.

As the term of semester 2 is coming to an end, we are submitting this document to our course teacher Ms. Mamta Padole, which includes diagrams, strategies, features and limitations of an application.

In this document, we tried to make everything i.e., sections and diagrams with explanations, consistent, subsequent, and easy to interpret by anyone who knows at least basics of software engineering.

As this document will not be read by only faculties but also by junior students of our succeeding batch to document their project like this. With this concern in mind, we are putting strategies and methods along with appropriate graphics, which we have considered while working.

# Acknowledgment

Foremost, we would like to express our sincere gratitude…

To Dean of the Faculty of Technology and Engineering, Prof. (Dr.) Chivukula N. Murthy and our Head of the Department, Prof. (Dr.) Apurva M. Shah for providing us an opportunity to show our knowledge which we gained here; And for sharing their personal experiences about system and software development on-field, too.

To our subject teacher, Ms. Mamta Padole for advising us throughout the project with her personal experience and professional ideas, who did not only recognize our mistakes but helped to solve them. We feel blessed because of her.

# About Project

**What was the need of this application?**

Algorithms and data structures as an essential part of knowledge in a framework of computer science have their stable position in computer science curricular, since every computer scientist and every professional programmer should have the basic knowledge from the area. With the increasing number of students in computer science field nowadays, introduction of appropriate methods into the process of their education is also required.

We discuss the extension of standard methods of teaching algorithms, using the whiteboard or slides, with the algorithm visualizations. According to they can be used to attract students' attention during the lecture, explain concepts in visual terms, encourage a practical learning process, and facilitate better communication between students and instructors. Interactive algorithm visualizations allow students to experiment and explore the ideas with respect to their individual needs. Extensive studies on algorithm visualization effectiveness are available nowadays, and results are quite encouraging.

Algorithm visualization, as part of software visualization, could be described as "graphical representation of an algorithm or program that dynamically changes as the algorithm runs"[1].

**How did we define the project? And what is the scope & purpose of it?**

We decided to start developing our own algorithm visualization application. The motivation behind the decision is detailed in and it includes the fact, that the application is intended to be used as a support tool within the subject Artificial Intelligence, taught in a master study program at MS University, Baroda. The selection of topics within the scope of the subject is quite wide and it could probably be changed over the time. To cover the scope of the subject, probably more tools would be used, or quite big interventions to selected tool would be required. But in the time being we start from path finding algorithms.

As a development platform for the project, we had to work on Java, ensuring high portability and very good support by available tools, libraries, etc. Another important decision to make was the selection of software framework. After the analysis of available solutions, the processing was chosen, which is software sketchbook designed to be flexible and for ease of development.

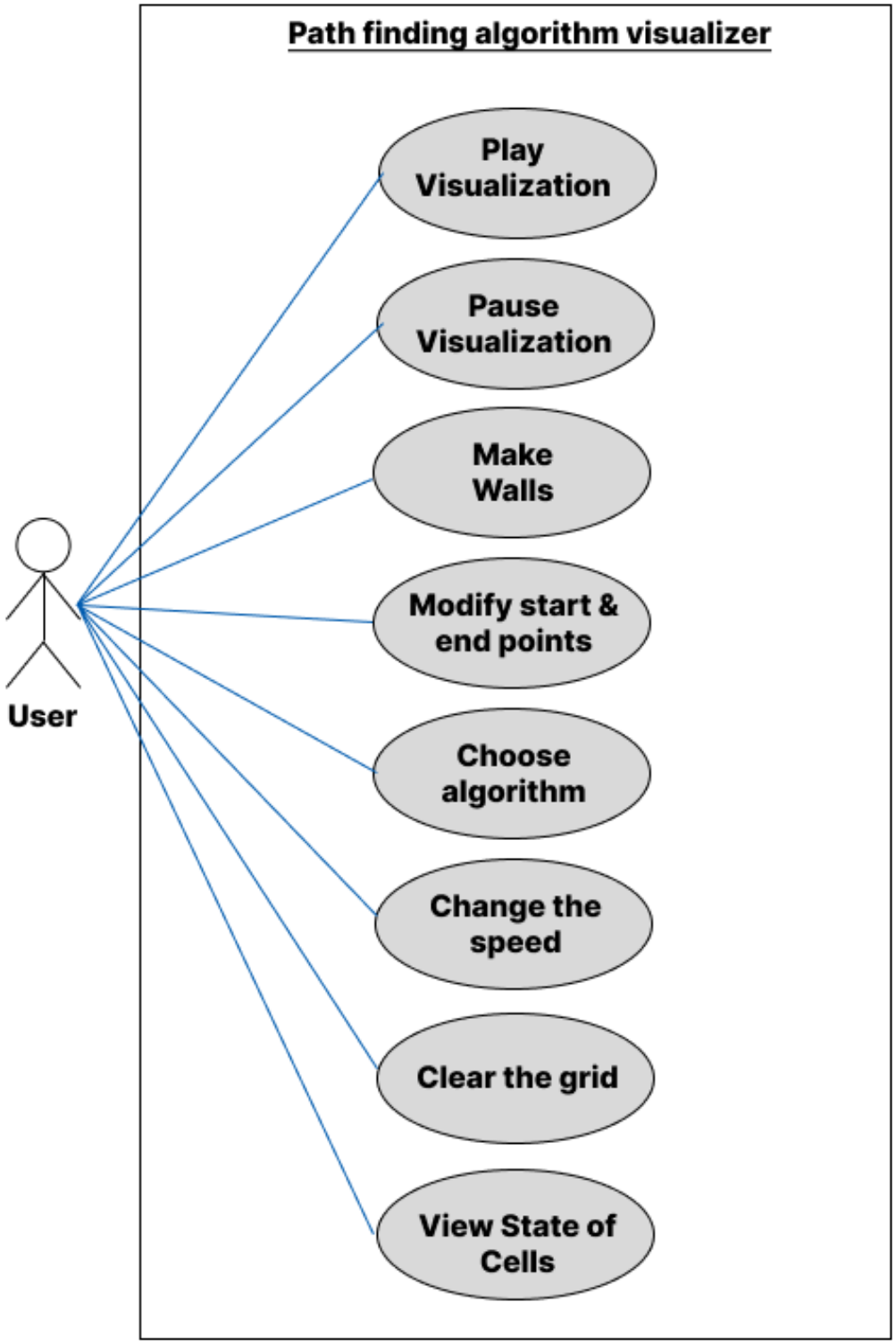
Path finding algorithms' visualizer aims to simplify and deepen the understanding of algorithms operation by showing the process of path finding algorithm step-by-step, allowing the users to pause the execution of an algorithm at any point of time, and profile the state of algorithm & the data related to it as well.

# Tools, Technologies, Libraries & Features

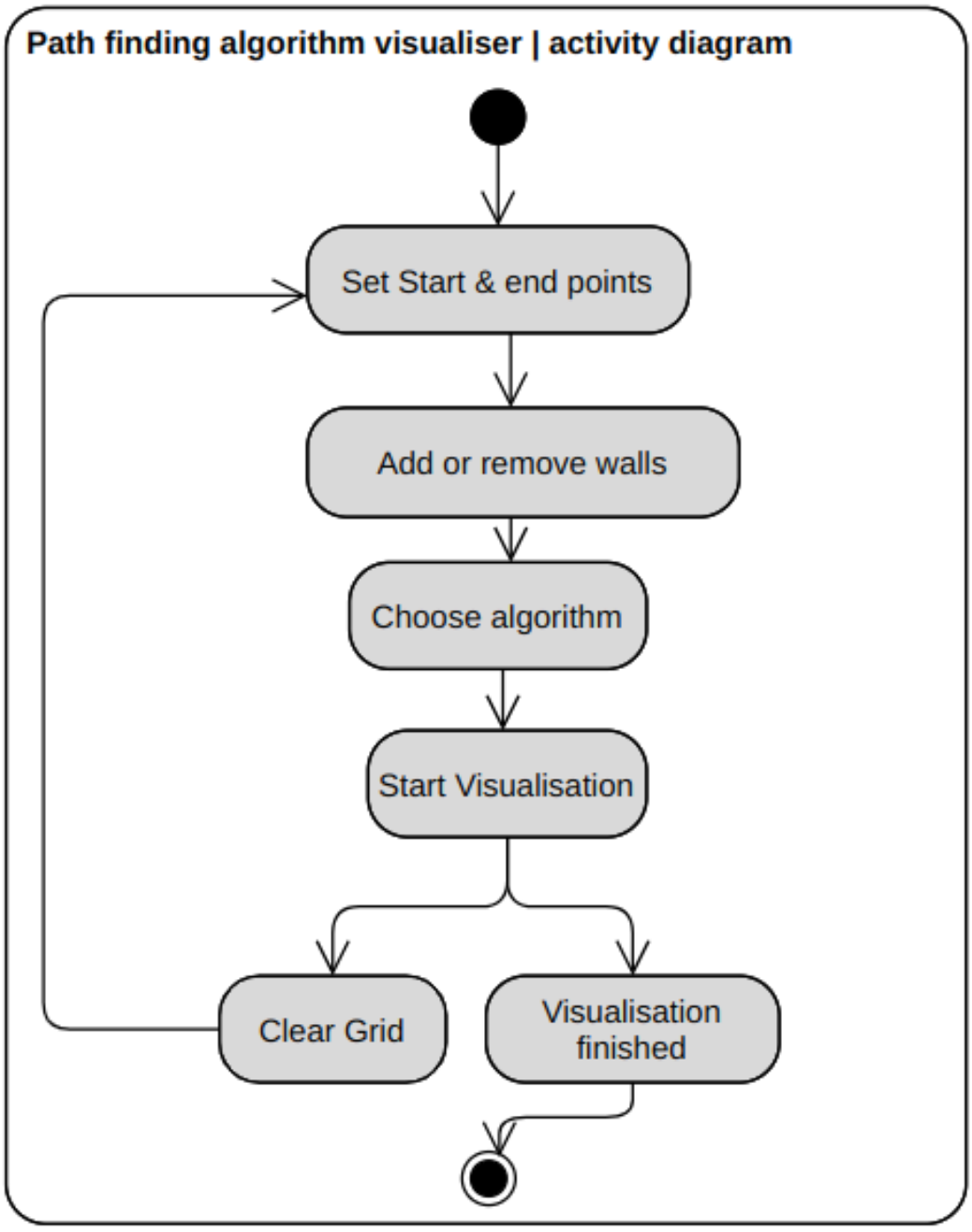
- IDE (Integrated Development Environment)
    - **Processing IDE**: It includes a text editor, a compiler, and a display window. It enables the creation of software within a carefully designed set of constraints.
- Libraries
    - **Processing**: It is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology.
    - **Controlp5**: It offers a range of controllers that allow us to easily change and adjust values while our sketch is running. Each controller is identified by unique name assigned when creating a controller. ControlP5 locates variables and functions inside our sketch and will link controllers to matching variables or functions automatically. Controller changes can easily be captured within our sketch by implementing the controlEvent function.
- Features of Java we employed within our project are as follows:
    - **Applet**: It is a small program that is intended not to be run on its own, but rather to be embedded inside another application.
    - **Swing**: Java Swing is a lightweight Java graphical user interface (GUI) widget toolkit that includes a rich set of widgets.
    - **Collections**: It is a framework that provides an architecture to store and manipulate the group of objects.
    - **OOPs**: It is about creating objects that contain both data and methods. It is a paradigm that provides many concepts, such as inheritance, data binding, polymorphism, etc.

# UML Diagrams

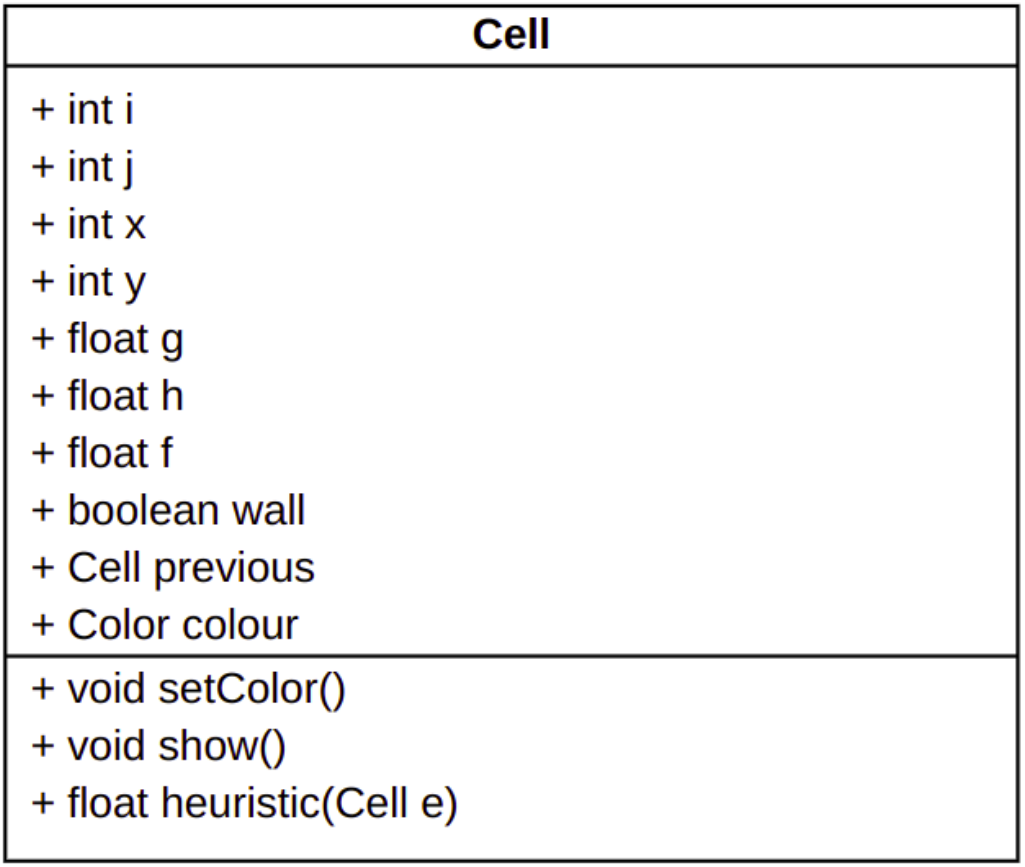
**Use Case Diagram:**



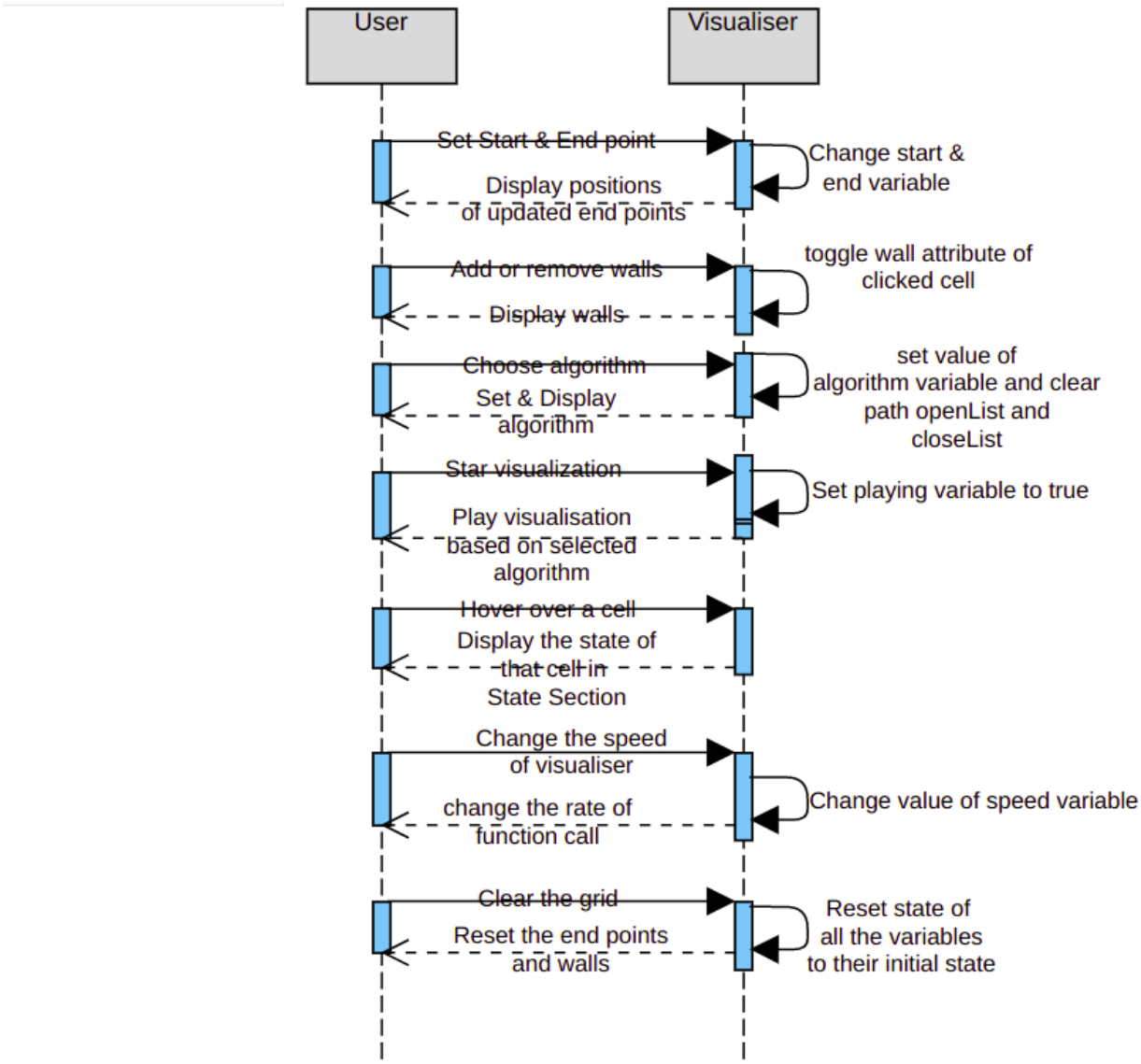
**Activity Diagram:**



**Class Diagram:**



**Sequence Diagram:**



# Test Cases

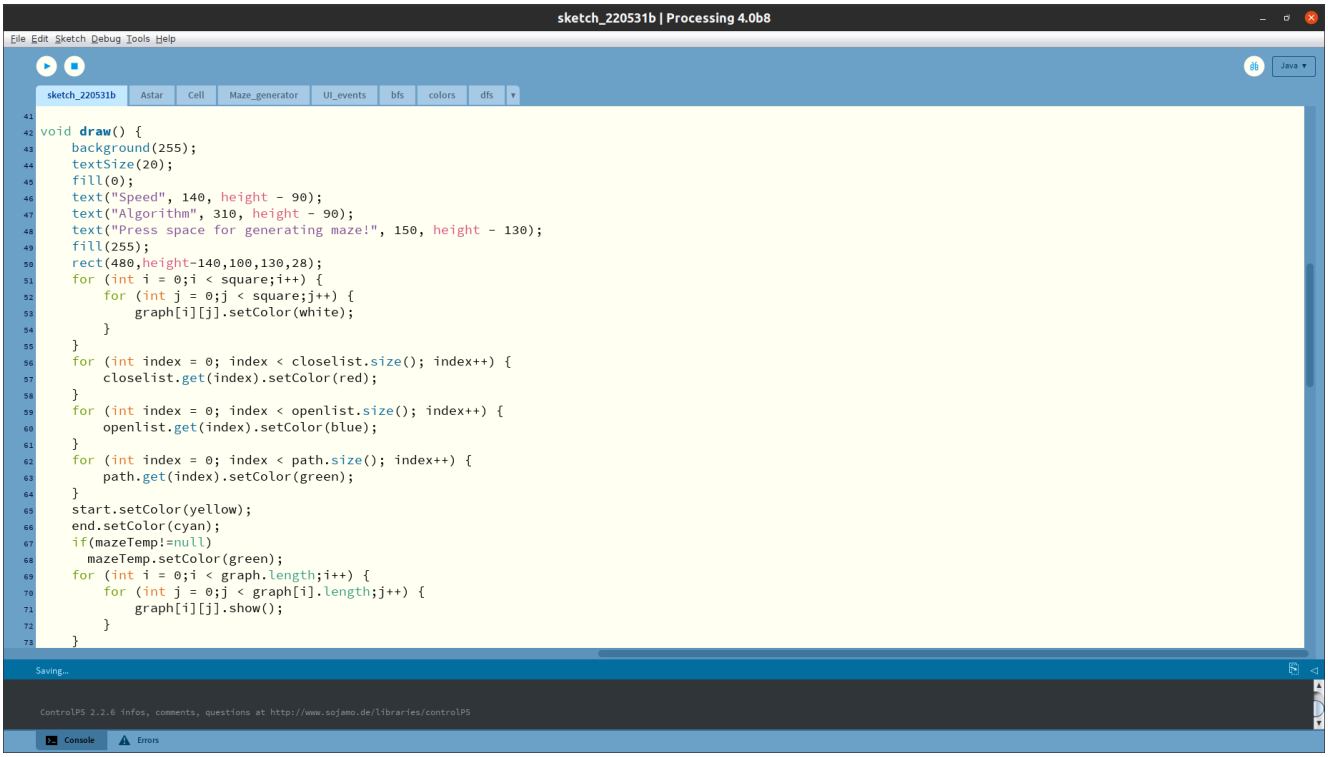
*"Discovering the unexpected is more important than confirming the known"*

*– George E. P. Box*

"Test cases" in our project can also be called "User scenarios" as we are not considering every kind of test.

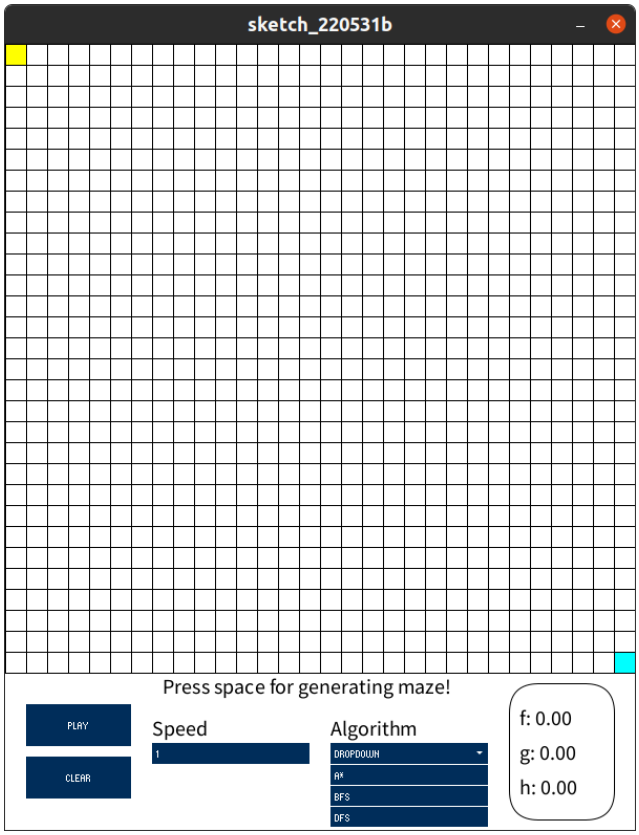
| | Action | Output |
|---|---|---|
| Scene – 1 | Click on Play button | Starts the visualizer |
| Scene – 2 | Click on Pause button | Pauses the visualizer at its current state |
| Scene – 3 | Drag the slider of speed | Speed changes can be seen accordingly |
| Scene – 4 | Change the algorithm from the drop-down list | Grid gets refreshed and visualizer gets started only if the state is playing. |
| Scene – 5 | Hover over any particular cell | State Section shows up-to-date information about the cell on which mouse pointer lies |
| Scene – 6 | Click on clear button | Grid gets reset, by means, all the walls will be removed and starting and ending positions will be set to their initial position |
| Scene – 7 | Make walls so that there is no path to end point | Program should not generate any error. |
| Scene – 8 | Attempt to drag start or end point outside the grid | Start or end point should remain on the cell from where mouse pointer left the grid. |
| Scene – 9 | Pause the algorithm in between its execution and attempt to change the starting or ending point or to add walls | Grid and settings should remain unchanged. |
| Scene – 10 | Mouse pointer leaves the grid | State section gets cleared. |
| Scene – 11 | Space is pressed | Maze generation gets started and other actions are locked until it finishes. |

# Screenshots & Explanation

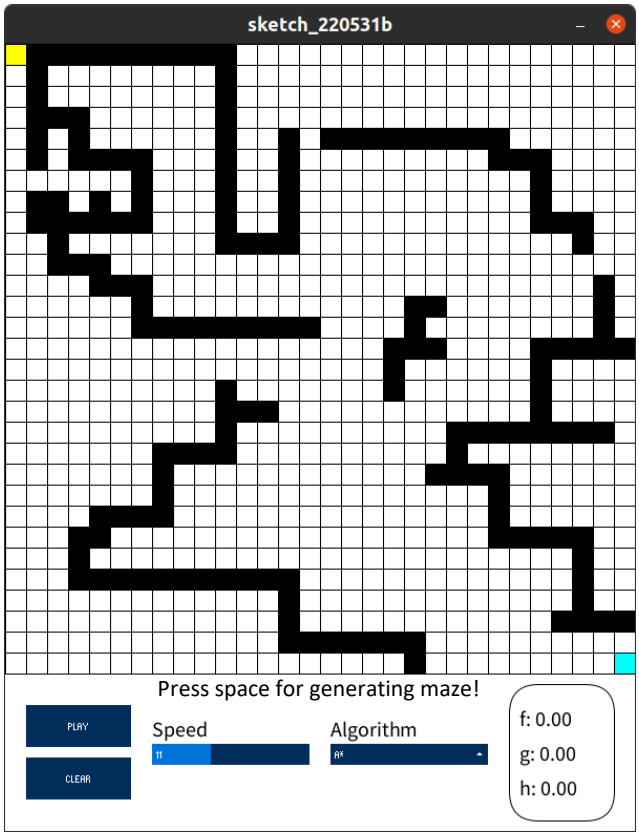



```
void draw() {
    background(255);
    textSize(20);
    fill(0);
    text("Speed", 140, height - 90);
    text("Algorithm", 310, height - 90);
    text("Press space for generating maze!", 150, height - 130);
    fill(255);
    rect(480,height-140,100,130,28);
    for (int i = 0;i < square;i++) {
        for (int j = 0;j < square;j++) {
            graph[i][j].setColor(white);
        }
    }
    for (int index = 0; index < closelist.size(); index++) {
        closelist.get(index).setColor(red);
    }
    for (int index = 0; index < openlist.size(); index++) {
        openlist.get(index).setColor(blue);
    }
    for (int index = 0; index < path.size(); index++) {
        path.get(index).setColor(green);
    }
    start.setColor(yellow);
    end.setColor(cyan);
    if(mazeTemp!=null)
      mazeTemp.setColor(green);
    for (int i = 0;i < graph.length;i++) {
        for (int j = 0;j < graph[i].length;j++) {
            graph[i][j].show();
        }
    }
```

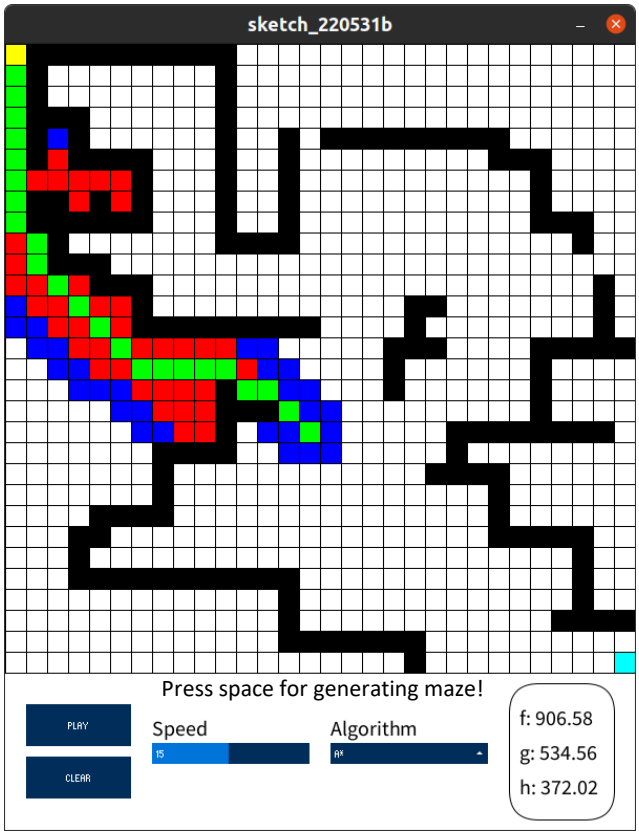
Fig.: Processing IDE Window



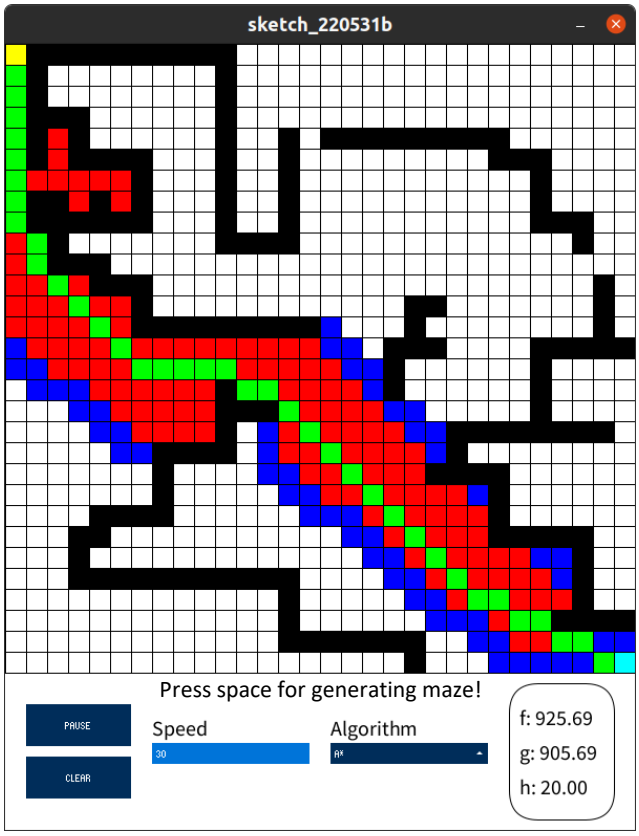

User will get above screen as soon as the application starts. On which user can make walls as we made, you can see them in the following screenshot.



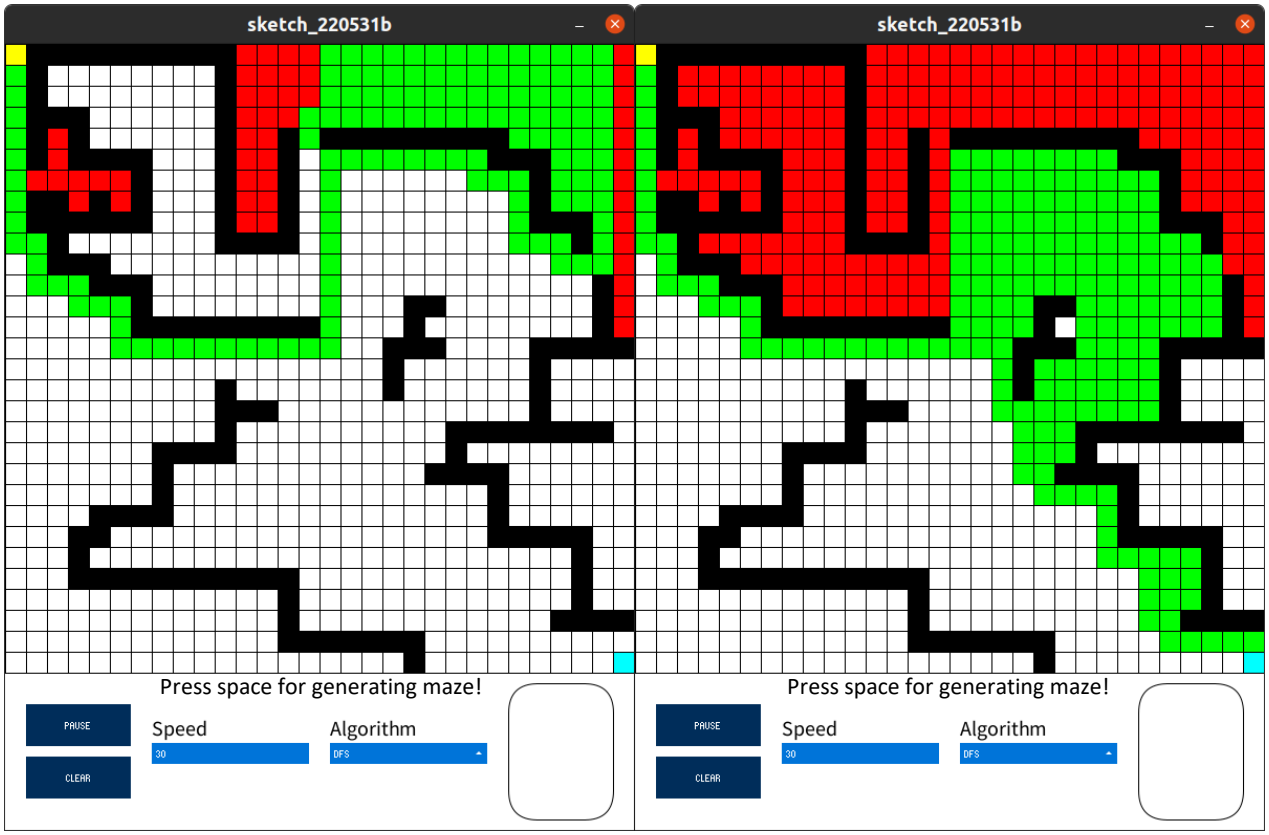

After creating walls, changing start & end point and selecting algorithm, user can start visualizing it.



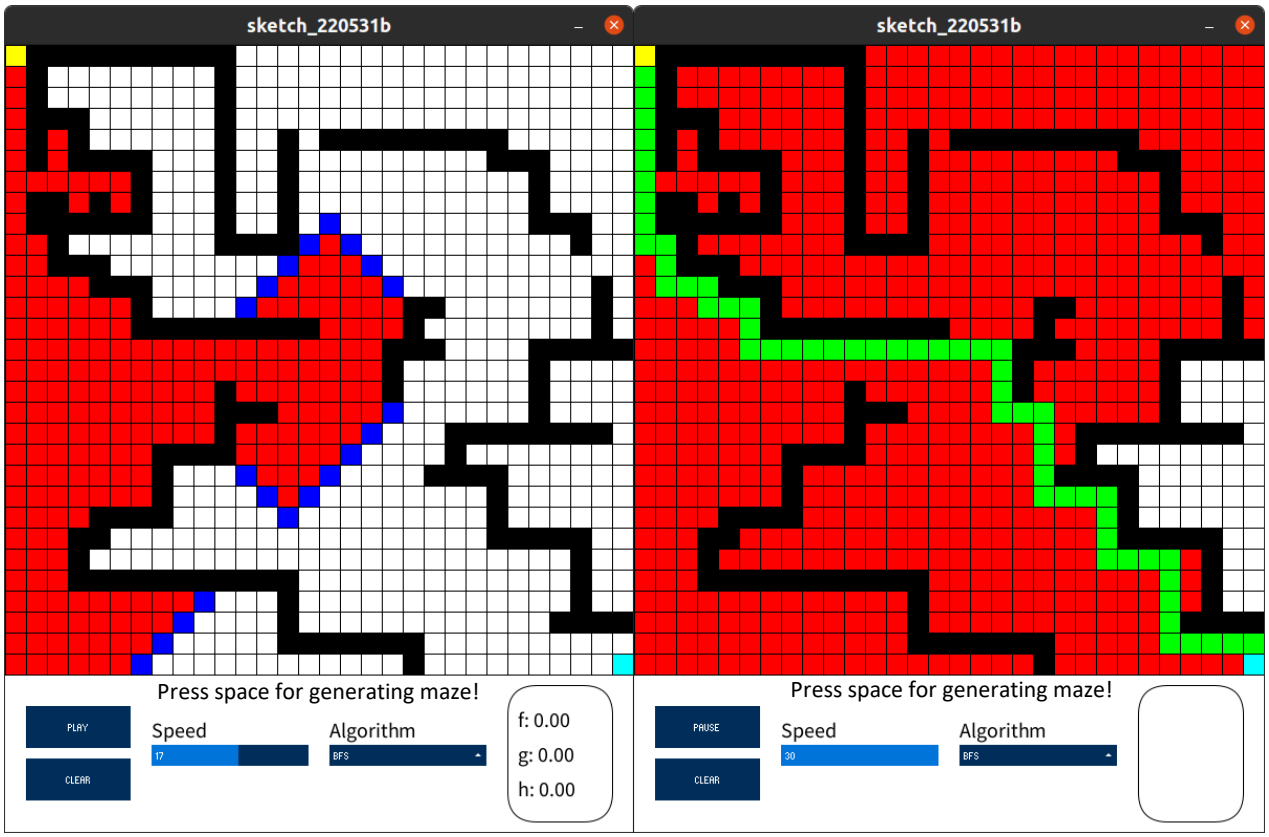

User can pause the visualization and can check the state of the cells by hovering over them in the “state section”. In the state section, ‘g’ represents the distance between start and current cell; ‘h’ represents heuristics (distance between current cell and end cell); And ‘f’ is the summation of both ‘h’ and ‘g’.



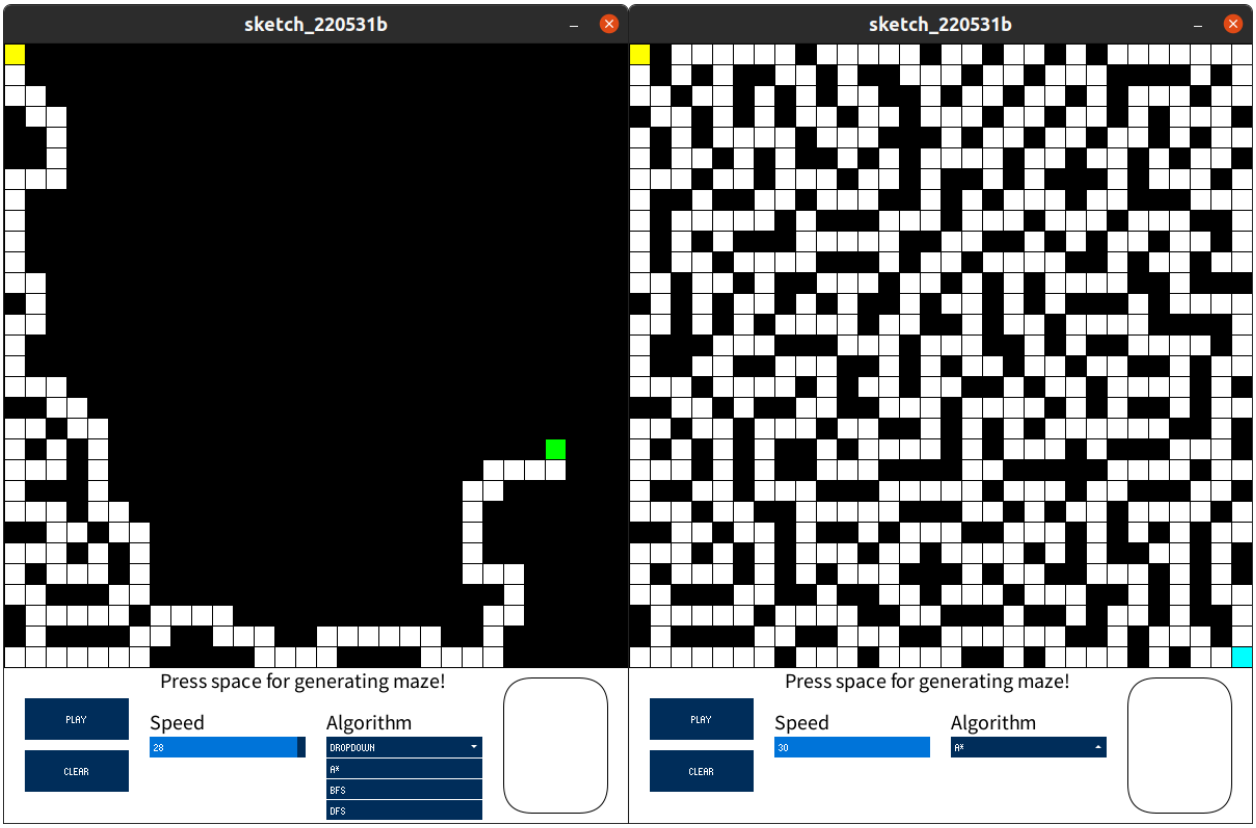

After the algorithm is finished finding the path between start and end point, it will look something like the above figure.




Both of the above figures are showing working scenarios of DFS algorithm.




Both of the above figures are showing working scenarios of BFS algorithm.




By pressing space user can generate a maze using recursive backtracking algorithm.

# Reports

Due to the summer holidays, we were not obliged to show our regular updates of the project to the course teacher, hence, we do not have any reports to put here.

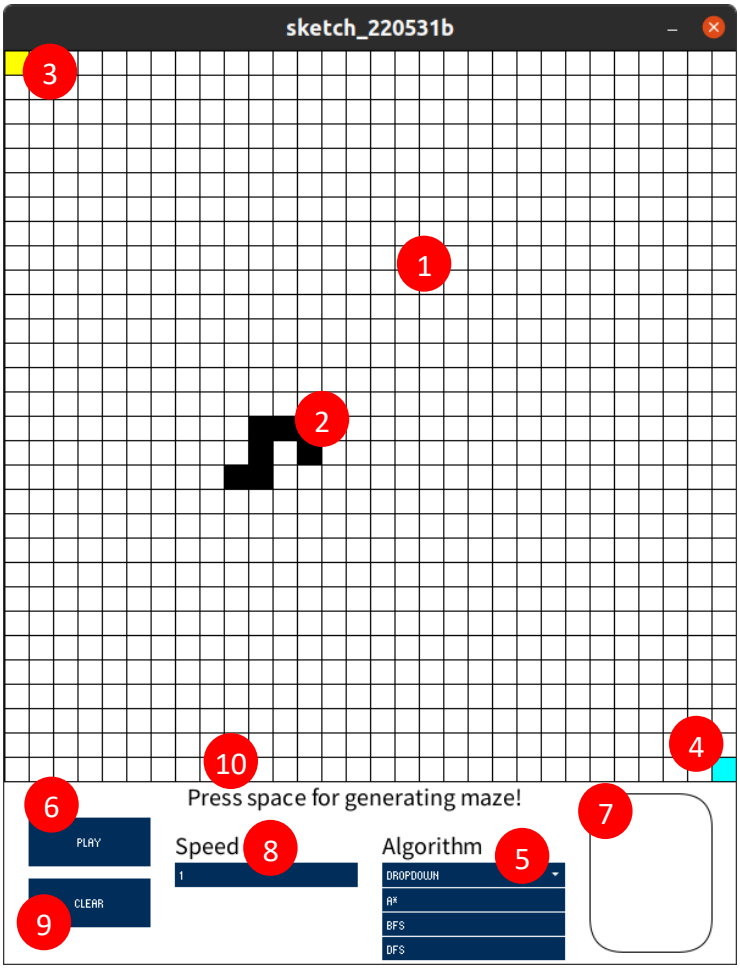
# User Manual

This section of the report is meant for the users to interact with our application, using which they can easily interact with it.

In our application, we have provided canvas which is a grid, actually. Where user can choose starting and ending point for the algorithm, by dragging the points around on the grid. Here, yellow and cyan are the starting and ending points, respectively. User can also create or remove a wall from grid by pressing any cell other than start and end point. I.e., wall represents an obstacle on the grid. User can also press space to generate maze using the recursive backtracking algorithm.

After that, user can choose which algorithm to visualize. Available choices are A*, DFS and BFS given in the list. After selecting the algorithm, user can start the visualization using “Play” button; and can pause at any moment using the same button.

User can control the speed of visualization using the speed slider at any moment. User can view the state of a cell by hovering over it in the state section which is represented by the (7) in the following figure. User can clear the grid at any point of time, by clicking on “Clear” button.




1 – Grid (Canvas)
2 – Walls (Obstacles)
3 – Starting Point
4 – End Point
5 – Selection from Available Algorithms
6 – Start and/or Pause
7 – Current state of particular cell
8 – Speed of the animation
9 – Clear grid
10 – Press space to generate maze

# References and Bibliography


1. https://www.researchgate.net/
2. https://processing.org/reference
3. https://www.sojamo.de/libraries/controlP5/
4. https://docs.oracle.com/en/java/
5. https://en.wikipedia.org/wiki/Maze_generation_algorithm
6. https://en.wikipedia.org/wiki/A*_search_algorithm

# Conclusion

This project boosted our knowledge and skills required in a corporate world specifically. Also, it exposed us to new technologies like processing and its whole environment of libraries and allowed us to improve on our previous knowledge of JAVA. We learnt teamwork, planning and human resource management.

In our team, there was a vast difference between the knowledge of the members, so, it was hard to keep everyone on the same page, but we learnt to overcome those limitations and finally achieved our goal.