Release Notes for NaviBot Version 1

Overview

Welcome to Version 1 of *NaviBot*! When running this application for the first time you will notice that the User Interface is very simple. This is because pretty user interface and design was not a focus for the first sprint - we opted for a mediocre looking yet functional product over a visually stunning yet non-functional one.

NaviBot Version 1 is a very simple, yet functional app that implements features that are core to the vision statement. Instead of cramming everything into a single screen, *NaviBot* uses two screen - one for the programming area, and one for the maze and control area. You can switch between them using the buttons found on the bottom of each screen.

Users can create programs by dragging and dropping code blocks from the block area into the program area. At the moment there are four block available:

1. Move the robot forward one tile
2. Turn it anticlockwise once
3. Turn it clockwise once
4. Detect the number of tiles between the robot and the wall in front of it

Of course, this is a very small subset of the programming language that we hope to implement by the end of the project. We have focused on designing *NaviBot* so that creating new program block types and program statements is easy. Next iteration we plan on completing the programming language so that users can create proper programs.

At the moment, blocks are added to program in a sequential order - i.e. the program will execute statements in the order they are added. Furthermore, it is not currently possible to edit your program, or remove blocks from the program (you need to reset the entire application and start again). This are goals for the next iteration.

On the second screen (the “Maze” screen) you will see a control panel on the left hand column, and a maze taking up the rest of the screen. The control panel has two buttons - Run and Stop. The former makes your program run to completion, and the latter has no functionality yet (it is just a placeholder). The maze layout and the robot location has been hard coded for this iteration, so it is not very interesting. The way we have designed *NaviBot* allows for dynamic generation of the maze, however it was not implemented in this sprint. Furthermore, the graphics are very basic - the tiles and the robot are actually using Kivy Button Widgets as graphical placeholders. In the next iteration they will be proper images.

Finally, *NaviBot* has a win state! If you manage to navigate the Robot to the Goal tile (the blue tile) then you will receive a “You Won!” message. There is only one path to the Goal tile, so the app in its current state presents no real challenge.

Below is a list of all the features implemented based on the user and tech stories from our backlog:

1. Added three different tiles Goal, Wall and Clear.
    - Tile Goal: the tile where the robot aims at.
    - Tile Wall: the tile which restricts the movement of the robot.
    - Tile Clear: the tile which is the road for the robot to move through.

2. Added 5x5 2D Maze.
    - A fixed maze containing three different tiles Goal, Wall and Clear.

3. Added detect_wall code block:
    - This function can detect whether there is a wall in front of the robot.
    - If there is wall in front of the robot, the robot cannot move forward;
    - If not, the robot can move forward.

4. Added detect_win code block:
    - This function can detect whether the robot reaches the Goal tile or not.
    - If the robot reaches the Goal tile, the program will display a win message.

5. Added distance_to_wall code block:
    - This function can show the distance from the robot to the wall which the robot is currently facing.

6. Added move code block:
    - This function can move the robot one step forward to which the robot is currently facing.

7. Added the feature of drag and drop to create the program:
    - The user can drag the code block on the left part of the Programming screen to the right part, and drop the code block at the programming area.

8. Added RUN button to create the program:
    - After drag and drop the code blocks to the programming area in Programming screen, switch to Maze screen and click RUN button.
    - The user can see how robot moves after starting the program.

9. [INCOMPLETE] Added turn 90-degree clockwise code block:
    - This function can turn the robot 90 degrees in the clockwise direction.

- IMPORTANT: As of now, this code block only makes the robot turn once, not “a specific number of times.”

10. [INCOMPLETE] Added turn 90-degree anti-clockwise code block:
    - This function can turn the robot 90 degrees in the anti-clockwise direction.
    - IMPORTANT: As of now, this code block only makes the robot turn once, not “a specific number of times.”