Prishi Amin

Design Document - Project 1: Potential Field Navigation

FieldMap **Class:**

FieldMap is the main class for this project. This class controls a 2D grid map that represents a potential field. It calculates and updates the potential value of each cell in the grid depending on the goals and obstacles that are added to the field.

Data Structures:

The class uses a 2D array to hold the values of the potential field at each cell because a 2D array allows for values at specific indexes to be accessed easily. Additionally, the desired grid size is unknown until run time, so to provide flexibility and avoid putting a constraint that may be too small for the input grid size, it is dynamically allocated. The goals_array and obstacles_array are used to keep track of where goals and obstacles are placed on the grid, so that their coordinates can be accessed easily when calculating potential values for all the grid cells. These two arrays are also dynamically allocated to once again provide flexibility for various grid sizes since the number of goals and obstacles are unknown until run time.

Member Variables:

The important private member variables are N_rows, M_columns, and K. N_rows and M_columns determine the size of the grid which enables the allocation of the exact amount of memory needed to function. K is a variable that is needed in the calculation of the potential values throughout the grid whenever necessary. These variables play a crucial part in the grid size and potential values, so they are private to avoid incorrect changes to their values (i.e. changing them to negative values) which can lead to significant errors at runtime. By making them private variables, it ensures that any changes to these values are possible, but limited, as they can only be altered through member functions that check the input before making changes to these variables.

Member Functions:

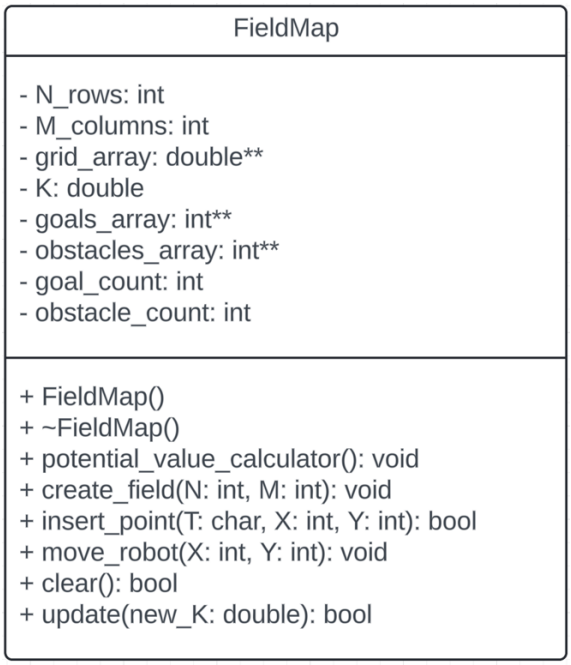
void potential_value_calculator() [helper function]: This is a helper function that is called by the other functions to calculate and update the potential values of all the grid cells. It uses a formula for this calculation to account for the effect that the obstacles and goals in the grid have on each individual grid cell. The function operates by iterating through the array of goals and obstacles then it calculates the effect that each of these has on the current cell. These individual values are then summed together and stored as a total potential value for each cell in the grid_array.

void create_field(int N, int M): This function creates a new grid of size N (rows) x M (columns). When the grid is created, memory is allocated for an array of N pointers. Each pointer in this array represents a row. Then, each pointer (row) points to another array of size M which is the columns of the array. If a grid already exists, this function will deallocate the memory for the

previous grid to prevent memory leaks, and then it will allocate memory for the new grid that is specified.

bool insert_point(char T, int X, int Y): This function populates the grid with goals and obstacles as requested, but it is limited by the bounds of the grid. If the requested goal/obstacle's index falls within the bounds, its value will be added to the grid_array, with goals represented by 1.0 and obstacles by 2.0. At the same time, the coordinates of these goals and obstacles will be added to a dynamic array to be stored for use when calculating the potential values. Keeping track of the goals and obstacles is crucial because they are all used by the potential_value_calculator() helper function. The helper function is called after every goal/obstacle is added to keep the grid updated with the changing potential values due to the effects of additional goals and obstacles on the potential. The function then returns true or false depending on whether the goal/obstacle was inserted correctly within the bounds.

UML Diagram:



Runtime:

Prove that your MOVE command has runtime O(NXM)

The MOVE command outputs the potential values at the specific coordinate by getting the potential value from the grid_array. However, the potential value that is in this array is calculated using the potential_value_calculator() helper function which has a runtime of O(NXM). This function iterates through each cell in the grid using nested for loops. The outer loop iterates over the rows N times, while the inner loop iterates over the columns M times for each row. Thus, this means that to go through all of the cells in the grid, it takes NXM iterations, resulting in a runtime of O(NXM).