

p0 Design Document (pyuan)

PotentialField class

Private members

- `int width, height` : Dimensions of the grid.
 - The user should not be able to change the size of the field after creating it, so it is private.
- `double k` : Constant for potential calculations.
 - Should only be changed through the public method `update()` so that we can update the potentials immediately afterward. Otherwise it should be inaccessible, so it is private.
- `double*** grid` : 3D dynamically allocated array to represent the field.
 - User should only be able to change the field through the public methods such as `addPoint()` and not directly, so it is private.
 - The array structure will be: `[row][col][0, 1]` . `0` will hold the potential at that point (the x and y components are always equal, so we only need to store a single value). `1` will hold the object type at that point (goal or obstacle or nothing).
 - Goal: `1.0` , Obstacle: `2.0` , Nothing: `0.0`
- `static const double GOAL_TYPE, OBSTACLE_TYPE` : Constants to represent goals and obstacles.
 - `GOAL_TYPE = 1.0`
 - `OBSTACLE_TYPE = 2.0`

Public methods

These methods are public since they need to be accessed from `main()` .

- `PotentialField()` : Trivial constructor.
 - Set all values to `0` or `nullptr` .
 - The assignment of member variables is handled in the `create()` member function.
- `~PotentialField()` : Destructor.
 - Call the `deleteGrid()` member function to handle deallocation of `grid` .
- `void create(int w, int h)` : Initializes values of `PotentialField` object.
 - If `grid` already exists, call `deleteGrid()` .
 - Set values for `w` and `h` and sets `k` to 0. Dynamically allocate 3D array for `grid` with all values set to `0.0` .
- `void point(char type, int xG, int yG)` : Add goal or obstacle to the field at the specific coordinates.
 - If the same type of obstacle already exists at the given coordinates, do nothing.

- Iterate through each cell in `grid`. Skip `(xG, yG)`. Update the potential with `recomputePotential()` and the object type.
- `void move(int x, int y)` : Print potential at given coordinates.
 - Return potential at given coordinates.
 - Time complexity: `O(1)` since accessing an array is a constant-time operation.
- `void clear()` : Reset all values in `grid`.
 - Iterate through each cell in `grid`. Set potential to 0 and object type to none.
- `void updateK(double newK)` : Update value of `K`.
 - Update `K`. Iterate through each cell in `grid`. Multiply potential by `newK / oldK`.

Private methods

These methods are private since they do not need to be called from `main`. They are used as helper functions for the public methods.

- `void deleteGrid()` : Handles deallocation of `grid`.
 - Iterate through innermost array and deallocate. Repeat with second layer. Deallocate outer pointer.
- `bool isValidPosition(int x, int y)` : Return `true` if the given coordinates are within `grid`.
- `void recomputePotential(int x, int y, int xG, int yG, char type)` : Recomputes the potential for a cell given a newly placed object.
 - Calculates the potential `P` at `(x, y)` due to the newly placed object at `(xG, yG)`.
 - `P` is multiplied by `-1.0` if the object is a goal.
 - If the existing object at `(xG, yG)` is the opposite of `type`, then we need to multiply `P` by 2 to compensate.
 - Time complexity: `O(m * n)` since each cell is visited exactly once, and the operations done are arithmetic calculations and adding the new `P` to the existing value, which are `O(1)`.

