Project 0 Design Document

Kelly Kim
ECE 250
24 September 2024

This is the design document for ECE250 Project 0 Potential Fields.

Using a single header file for the project keeps things organized and straightforward, reducing complexity. Since the components are closely related, multiple header files can create confusion about which methods interact with which member variables. Additionally, this approach can lead to faster compilation times, as the compiler processes fewer files.

<u>Public:</u>

- board(int n, int m)
- ~Board()

Both constructor and destructor are <u>public</u> because users have to create and destroy Board objects. The constructor sets up the board with given nRows and mColumns and the destructor cleans up the memory when the board is done being used.

- void create(int n, int m)
- void point (char t, int x, int y)
- void move (int x, int y)
- void clear();
- void update(double k)
- bool exit();

By having these commands be private, the user would not have direct access to them. The user would not be able to create or modify the board directly, which would limit the class's functionality and usability.
Create/Clear function will have a void return type because it will not be returning anything (will print success or failure), just initializing the board/clearing the board.
Point/Update function will have a void return type because it will be updating points/kConst on the board without returning anything.
Move function will have a void return type, not returning but printing the potential value based on given coordinates.
Exit function will be used to control when the program stops. Returning true for the program to exit, returning false for the program to keep going.

- bool exitsign = false;

This variable changes to true when the exit function is called to exit the program.

- double calculatePotential(int x, int y, int xG, int yG, char t)
- void potentials (int xG, int yG, char t)
- Void calculateAll()
- clean()

calculatePotential function will be used to calculate the potential at (x,y). This function will have a double return type, since the value will be in decimals.

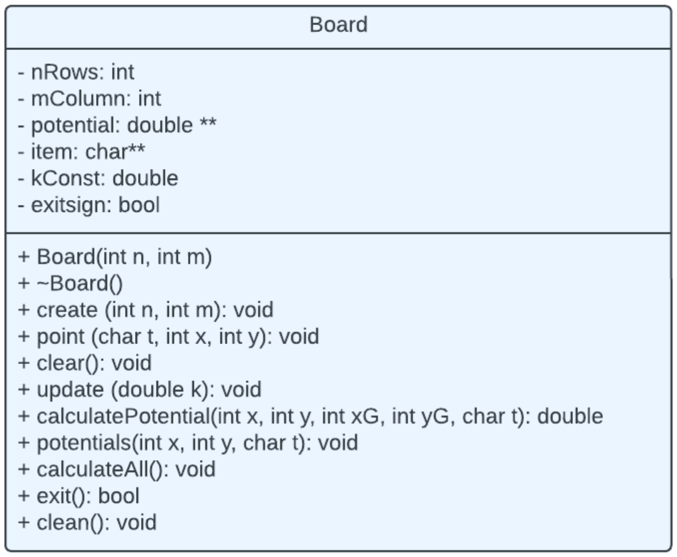
potentials function calculates how the specified coordinates and item (goal, obstacle, or nothing) affect all other points on the board, updating the potential array accordingly. It has a void return type since it does not return a value, only modifies the array.
calculateAll() calculates the whole board, most likely will be used to update potential values.
clean() function cleans everything on the board.

Private:

- int nRows;
- int mColumns;
- double **potential
- char **item
- double kConst

nRows, mColumns and kConst will be private to prevent the user from changing it to an invalid value. By having this private, it makes sure that any changes happen through controlled methods that check the input.
potential/item will be 2D arrays containing potential values/item(Goal, Obstacle, Nothing) at each point on the board. It will be private because the user should not be able to change it directly. In order to change these values, they will have to go through other public functions.

UML DIAGRAM:



```
Board
- nRows: int
- mColumn: int
- potential: double **
- item: char**
- kConst: double
- exitsign: bool
+ Board(int n, int m)
+ ~Board()
+ create (int n, int m): void
+ point (char t, int x, int y): void
+ clear(): void
+ update (double k): void
+ calculatePotential(int x, int y, int xG, int yG, char t): double
+ potentials(int x, int y, char t): void
+ calculateAll(): void
+ exit(): bool
+ clean(): void
```

Running Time:

Move function does not include any loops or recursive calls, it only checks the bounds and accesses potential values directly. To execute the function, it takes C_1(constant time). $f(n) = C_1$
There exists positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$
$0 \leq f(n) \leq cg(n) \rightarrow C_1 \leq c(MxN)$
By substituting $c = \frac{C_1}{MxN}$ into the equation above
$C_1 \leq \frac{C_1}{MxN}(MxN)$
$C_1 \leq C_1$
Therefore, $f(n) = O(MxN)$