RobotSimDocumentation

Elie Chelly

July 2023

1 Geometry Module

1.1 Typedef

The `Coordinates` type defined in includes/geometry.h is the base data structure to store coordinates. It stores three `double` in an `array` to represent respectively X, Y and Z carthesian coordinates of three dimensional space.

1.2 Functions

`double dist(Coordinates const& p1, Coordinates const& p2)`
Description: This function computes the euclidian distances between two poits.
Arguments: `Coordinates const& p1` first point to compute distance. `Coordinates const& p1` second point to compute distance.
Return: Computed distance, returned as a `double`.

`void display_coordinates(Coordinates const& pos)`
Description: This function displays coordinates in the terminal.
Arguments: `Coordinates const& pos` the coordinates to be displayed.
Return: None.

`bool check_termination(Coordinates const& p1, Coordinates const& p2)`
Description: Verifies if the both coordinates given in argument are the same.
Arguments: `Coordinates const& p1` and `Coordinates const& p2` are the coordinates to be verified.
Return: Returns `true` if both points are the same. If not, it returns `false`.

`void get_unit_vect(Coordinates const& p1, Coordinates const& p2, Coordinates& u)`
Description: Computes the unitary direction vector of the segment defined by two points. The computed vector's direction is from p1 to p2.
Arguments: `Coordinates const& p1` and `Coordinates const& p2` are the points that define the line segment. The result is stored in `Coordinates u`.
Return: None.

`void assign_pos(Coordinates& p1, Coordinates const& p2)`
Description: Sets a point's values.
Arguments: `Coordinates& p1` is the point to be set and `Coordinates const& p2` provides the values.
Return: None.

2 Input Module

2.1 Functions

`bool isNumber(string const& s)`
Description: Checks if a string is a number (integer or decimal).

Arguments: `string const& s` is the input string to be verified. `Coordinates const& p1` second point to compute distance.
Return: A boolean, `true` if the string is a number, `false` otherwise.

`bool get_coordinates(Coordinates& pos, string input, size_t cnt = 0)`
Description: Extracts coordinates from a string in a recursive fashion. Coordinates should be numbers separated by semi-columns.
Arguments: `Coordinates& pos` is the variable in which the coordinates are stored. `string input` is the string from which the coordinates are extracted `size_t cnt` is a counter used for the recursion.
Return: A boolean, `true` if coordinates were extracted properly, `false` otherwise.

`bool get_target_pos(Coordinates& pos)`
Description: Asks the user for coordinates until the user's input is considered correct. If the user inputs `stop`, it returns `false` in order to kill the main event loop.
Arguments: `Coordinates& pos` is the variable in which the coordinates are stored.
Return: A boolean, `true` if coordinates are extracted, `false` if the user inputs `stop`.

`double positive_input(string name)`
Description: Verifies if a string is a strictly positive number. If not, prints an error message and asks again.
Arguments: `string name` is an argument that modifies the error message.
Return: A `double` which is the number extracted from the string.

3 robotClass Module

3.1 Class

The `class Robot` is defined. It encapsulates the simulated robot by storing its position, its maximum speed and the timestep. It has the ability to move from its position to a new given position.

3.1.1 Data values

`Coordinates pos` stores the robot's position.
`double max_vel` stores the robot's maximum velocity.
`double dt` stores the simulation's timestep.

3.1.2 Methods

`void initialize()`
Description: Initializes the robot by asking the user to give it an initial position, a maximum velocity and a time step.
Arguments: None.
Return: None.

`void move(Coordinates const& target)`
Description: Moves the robot from its actual position to the target position. The robot always moves at its maximum velocity and slows down at the end to avoid missing the target. At each time setp, the robot's position is printed in the terminal and a message notifies the user when the robot has reached its target position.
Arguments: `Coordinates const& target` is the target position that the robot should reach.
Return: None.