NFQ Web App

1.0

Generated by Doxygen 1.8.8

Mon May 25 2015 21:20:48

Contents

1 File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

app.js
Javascript application for visualisation of NFQ algorithm **1**

2 File Documentation

2.1 app.js File Reference

Javascript application for visualisation of NFQ algorithm.

Functions

- function **connectRobot** ()
  Initializes new websocket connection to server.
- function **serverResponse** (response)
  Parses received message object from websocket connection.
- function **showIntro** (msg)
  Displays a introduction page.
- function **showControls** ()
  Displays a NFQ controls and state information.
- function **saveTask** ()
  Request saving current task progress to the database.
- function **loadTasks** (tasks)
  Creates a table of all available tasks from DB.
- function **runTask** (task)
  Requests a server to perform a task on a robot.
- function **doAction** (action_num)
  Order a robot to do selected action.
- function **enableAllButtons** ()
  Enables all buttons on page.
- function **disableAllButtons** ()
  Disables all buttons on the page to block request until the current action is finished.
- function **updateState** (distance, angle)
  Updates current state information with a new, received over Websocket.

2.1.1 Detailed Description

Javascript application for visualisation of NFQ algorithm.

Author

Jan Gamec

Date

24 May 2015

This application is served as a webpage when running the server application.

2.1.2 Function Documentation

2.1.2.1 function connectRobot ( )

Initializes new websocket connection to server.

Establishes a connection to robot through opening a new websocket connection to server. In addition it binds corresponding callbacks for websocket events. When a new message is received, function serverResponse is called, with a received object.

2.1.2.2 function disableAllButtons ( )

Disables all buttons on the page to block request until the current action is finished.

2.1.2.3 function doAction (*action_num*)

Order a robot to do selected action.

Handles click events on each movement arrows in order to move the "real" robot.

Parameters

action_num	Number of action from action set

2.1.2.4 function enableAllButtons ( )

Enables all buttons on page.

2.1.2.5 function loadTasks (*tasks*)

Creates a table of all available tasks from DB.

Parameters

tasks	List of task names from database

2.1.2.6 function runTask (*task*)

Requests a server to perform a task on a robot.

Parameters

task	Name of the task to be performed

2.1.2.7 function saveTask ( )

Request saving current task progress to the database.

Takes a name from corresponding input box, requests a server to save the task in DB.

2.1.2.8 function serverResponse (*response*)

Parses received message object from websocket connection.

This function handles basically 4 types of responses and reactions to them:

- Confirmation about succesful connection to robot - display a control UI.
- Warning about connection error - warns about an error.
- State update - refreshes the UI.
- Result of the performed task - displays final cumulative reward after completion of the task.

Parameters

response	A message object received over websocket

2.1.2.9 function showControls ( )

Displays a NFQ controls and state information.

2.1.2.10 function showIntro (*msg*)

Displays a introduction page.

Parameters

msg	If any warning is received, display it

2.1.2.11 function updateState (*distance, angle*)

Updates current state information with a new, received over Websocket.

Parameters

distance	New distance state
angle	New angle state