Porthos

An-embedded-linux-robot

0.0.0-cmake

generated on Wed Nov 30 2016 22:51:05

Contents

1 Main Page 1

2 Licensing 3

2.1 Licensing of documentation . . . 3

2.2 Licensing of source code . . . 3

3 System Description 5

4 Testing 7

4.1 Unittests . . . 7

5 Workflow 9

5.1 Workflow . . . 9

5.2 Setting up . . . 9

5.3 Starting on your feature . . . 9

5.4 Finish work . . . 10

5.5 Cleanup . . . 10

6 Todo List 11

7 Module Index 13

7.1 Modules . . . 13

8 Class Index 15

8.1 Class List . . . 15

9 Module Documentation 17

9.1 Mapper-library . . . 17

9.1.1 Detailed Description . . . 17

9.1.2 Function Documentation . . . 17

9.1.2.1 _mapper_add_children() . . . 17

9.1.2.2 mapper_add_point() . . . 19

9.1.2.3 mapper_get_point() . . . 19

9.1.2.4 mapper_get_xy_from_z_order() . . . 19

9.1.2.5 mapper_get_z_order() 20
9.1.2.6 mapper_init() 20
9.1.2.7 mapper_print_map() 21
9.1.2.8 mapper_set_area() 21

10 Class Documentation **23**

10.1 Node Struct Reference 23
10.1.1 Detailed Description 23

Index **25**

Chapter 1

Main Page

Porthos

This project tries to create an embedded linux robot system.

A high-level description of the system is given in the system_description. Some requirements have been created, but these are still very much open to discussion.

The compiled documentation can be read at http://spoorcc.github.io/porthos/ And the PDF at http://spoorcc.github.io/porthos/Porthos.pdf

Compiling

```
mkdir bld
cd bld
cmake ..
make
```

Generating documentation

```
cd bld
make doc
```

Chapter 2

Licensing

This page describes the licencing for the Porthos system.

2.1 Licensing of documentation

Todo determine licensing

2.2 Licensing of source code

Todo determine licensing

Chapter 3

System Description

Describes the robot system on high-level

Chapter 4

Testing

This page describes the testing procedures for the Porthos project.

4.1 Unittests

This project uses check for testing the C-code In order to run the tests do the following

```
cd bld
cmake ..
make
make test
```

To have more output for analyzing failing tests use follwoing command instead of make test:

```
ctest --verbose
```

Chapter 5

Workflow

This page describes the workflow for the Porthos project.

5.1 Workflow

This project works following the git-flow branching model. Each feature is developed on a feature branch, branched of of develop. Check out http://nvie.com/posts/a-successful-git-branching-model/ for more info.

The below workflow is based on http://qq.is/tutorial/2011/10/23/git-flow-on-github.html

5.2 Setting up

First clone the repository

```
git clone https://github.com/spoorcc/porthos.git
```

Go into the repo

```
cd porthos
```

Setup the origin

```
git remote add upstream git@github.com:spoorcc@porthos
```

Setup git flow (first install git flow if you haven't got it)

```
git flow init
```

And accept all the defaults

5.3 Starting on your feature

Create a new branch for your awesome feature

```
git flow feature start <my_great_feature>
```

Push the branch remote.

```
git flow feature publish <my_great_feature>
```

Commit your changes reguraly locally with descriptive messages.

Also push the changes back up to GitHub.

```
git push origin HEAD
```

5.4 Finish work

Create a pull request in the GitHub interface. In the pull request add usefull info. Click the send pull request to confirm you think you're done.

When your awesome feature is reviewed, sometimes additional changes are needed. Make them locally, commit and push them up to your branch.

Make sure your on your feature branch:

```
git checkout feature/<my_awesome_feature>
```

Do your development, commit and push the changes again. (see Starting on your feature).

5.5 Cleanup

When all your changes are agreed upon and merged by the project, your feature branch will be deleted. Locally you can finish your feature as well.

```
git flow feature finish
```

Chapter 6

Todo List

Page Licensing

determine licensing

determine licensing

Chapter 7

Module Index

7.1 Modules

Here is a list of all modules:

Mapper-library 17

Chapter 8

Class Index

8.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Node 23

Chapter 9

Module Documentation

9.1 Mapper-library

Functions

- int mapper_init (float x_size, float y_size, unsigned int max_level)

  Initializes the mapper library.
- int mapper_clear_map ()

  Clears entire map.
- int mapper_add_point (float x, float y, const MaptileValueEnum value)

  Sets value at given coordinate.
- int mapper_set_area (float x1, float y1, float x2, float y2, MaptileValueEnum value)
- int mapper_get_point (float x, float y, MaptileValueEnum *value)

  Gets value at given coordinate.
- int _mapper_add_children (Node *node)

  Adds 4 new children to given node.
- int _mapper_flatten_node (Node *node)

  If all children have same value remove children and set parent to same value.
- int _mapper_remove_children (Node *node)

  Frees memory of all non-NULL children pointers.
- int mapper_get_z_order (const int x, const int y, int *z)

  Calculates Z-order from x,y coordinates.
- int mapper_get_xy_from_z_order (const int z, int *x, int *y)

  Calculates X,Y coordinates based on Z-order.
- int mapper_print_map (bool with_depth)

  Print the map on stderr.

9.1.1 Detailed Description

Library for keeping map of surroundings

9.1.2 Function Documentation

9.1.2.1 _mapper_add_children()

```
int _mapper_add_children (
            Node * node )
```

Adds 4 new children to given node.

Parameters

in	*node*	Pointer node to add children to

Return values

MAPPER_OK	Everything went OK
MAPPER_PARAMETER_ERROR	One of the children already exists
MAPPER_MEMORY_ERROR	Allocating memory went wrong

9.1.2.2 mapper_add_point()

```
int mapper_add_point (
            float x,
            float y,
            const MaptileValueEnum value )
```

Sets value at given coordinate.

Parameters

in	*x*	X-coordinate
in	*y*	Y-coordinate
in	*value*	The X,Y coordinate must be within map_size set during mapper_init

Precondition

mapper_init has been called

9.1.2.3 mapper_get_point()

```
int mapper_get_point (
            float x,
            float y,
            MaptileValueEnum * value )
```

Gets value at given coordinate.

Parameters

in	*x*	X-coordinate
in	*y*	Y-coordinate
out	*value*	The X,Y coordinate must be within map_size set during mapper_init

Precondition

mapper_init has been called

9.1.2.4 mapper_get_xy_from_z_order()

```
int mapper_get_xy_from_z_order (
```

```
            const int z,
            int * x,
            int * y )
```

Calculates X,Y coordinates based on Z-order.

Parameters

in	*z*	Z-order
out	*x*	X-coordinate
out	*y*	Y-coordinate

Based on provided Z-order calculates the X,Y coordinates. For more information see https://en.wikipedia.org/wiki/Z-order_curve

Precondition

level must be set using mapper_init

9.1.2.5 mapper_get_z_order()

```
int mapper_get_z_order (
            const int x,
            const int y,
            int * z )
```

Calculates Z-order from x,y coordinates.

Parameters

in	*x*	X-coordinate
in	*y*	Y-coordinate
out	*z*	Z-order

Based on provided X,Y coordinates calculates the Z-order. For more information see https://en.wikipedia.org/wiki/Z-order_curve

Precondition

level must be set using mapper_init

9.1.2.6 mapper_init()

```
int mapper_init (
            float x_size,
            float y_size,
            unsigned int max_level )
```

Initializes the mapper library.

Parameters

in	*x_size*	Size in X-direction
in	*y_size*	Size in Y-direction
in	*max_level*	Two to the max_level of detail

First mapper_init must be called to configure the library. With this call the real-world size that the map spans is set. The max_level sets the 2 to the Nth maximum level of detail. For example max_level of 2 means the map will be divided in 4x4 grid.

9.1.2.7 mapper_print_map()

```
int mapper_print_map (
            bool with_depth )
```

Print the map on stderr.

Parameters

in	*with_depth*	If enabled, depth of graph-node is shown

9.1.2.8 mapper_set_area()

```
int mapper_set_area (
            float x1,
            float y1,
            float x2,
            float y2,
            MaptileValueEnum value )
```

Parameters

in	*x1*	
in	*y1*	
in	*x2*	
in	*y2*	
in	*value*	Sets the region with x1,y1 −> x2,y2 to given value

Return values

MAPPER_OK	All went well
MAPPER_PARAMETER_ERROR	Coordinates are not OK

Chapter 10

Class Documentation

10.1 Node Struct Reference

`#include <mapper_internal.h>`

10.1.1 Detailed Description

Mapper works with quadtree internally

The documentation for this struct was generated from the following file:

- mapper_internal.h

Index

_mapper_add_children
    Mapper-library, 17

Mapper-library, 17
    _mapper_add_children, 17
    mapper_add_point, 19
    mapper_get_point, 19
    mapper_get_xy_from_z_order, 19
    mapper_get_z_order, 20
    mapper_init, 20
    mapper_print_map, 21
    mapper_set_area, 21
mapper_add_point
    Mapper-library, 19
mapper_get_point
    Mapper-library, 19
mapper_get_xy_from_z_order
    Mapper-library, 19
mapper_get_z_order
    Mapper-library, 20
mapper_init
    Mapper-library, 20
mapper_print_map
    Mapper-library, 21
mapper_set_area
    Mapper-library, 21

Node, 23