# NOBEL Reference Documentation

May 3, 2001

## Contents


**1 NOBEL Class List** 1

**2 NOBEL Class Hierachy** 5

**3 NOBEL Globals** 7

**4 namespace ExternalNS** 10

**5 class ExternalNS::External** 11

**6 class ExternalNS::InputFileNotFoundException** 12

**7 class ExternalNS::InputFileSyntaxException** 13

**8 class ExternalNS::Output** 13

**9 class ExternalNS::OutputRep** 15

**10 class ExternalNS::Settings** 16

**11 class ExternalNS::SettingsRep** 18

**12 class HELVision** 18

**13 struct HELVision::color_info** 21

**14 struct HELVision::earlierRegion** 22

**15 struct HELVision::region** 23

**16 struct HELVision::rle** 24

**17 namespace InputNS** 25

**18 class InputNS::Input** 25

**19 class InputNS::InputInterface** 26

**20 class InputNS::InputRep** 27

**21 class InputNS::ReadError** 28

**22 namespace Localization** 28

**23 namespace LocalizationNS** 29

**24 class LocalizationNS::Bitmap** 33

**25 class LocalizationNS::BitmapRep** 34

**26 class LocalizationNS::CaptureException** 35

**27 class LocalizationNS::LandmarkRecognition** 35

**28 class LocalizationNS::LandmarkRecognitionRep** 37

**29 class LocalizationNS::Landmarks** 37

**30 class LocalizationNS::LandmarksFileNotFoundException** 38

**31 class LocalizationNS::LandmarksInterface** 39

**32 class LocalizationNS::LandmarksRep** 40

**33 class LocalizationNS::Localization** 40

**34 class LocalizationNS::LocalizationException** 41

**35 class LocalizationNS::LocalizationRep** 42

**36 class LocalizationNS::PanTiltException** 44

**37 class LocalizationNS::Panorama** 45

**38 class LocalizationNS::PanoramaRep** 46

**39 class LocalizationNS::Segmentation** 47

**40 class LocalizationNS::SegmentationException** 48

**41 class LocalizationNS::SegmentationRep** 49

**42 class LocalizationNS::View** 50

**43 class LocalizationNS::ViewRep** 51

**44 namespace MapNS** 52

45 class MapNS::BadDestinationException 53

46 class MapNS::BadScaleException 54

47 class MapNS::Grid 54

48 class MapNS::GridCell 57

49 class MapNS::GridCellRep 60

50 class MapNS::GridInterface 61

51 class MapNS::GridIterator 65

52 class MapNS::GridIteratorInterface 67

53 class MapNS::GridIteratorRep 70

54 class MapNS::GridRep 71

55 class MapNS::Map 76

56 class MapNS::MapFileNotFoundException 78

57 class MapNS::MapInterface 79

58 class MapNS::MapRep 81

59 namespace MovementNS 82

60 class MovementNS::Detector 83

61 class MovementNS::Movement 84

62 class MovementNS::RobotMover 85

63 class MovementNS::RobotMoverRep 87

64 class MovementNS::Roller 88

65 class MovementNS::ScoutRoller 89

66 class MovementNS::ScoutRollerRep 91

67 class MovementNS::Sonar 92

68 class MovementNS::SonarRep 95

69 namespace NobelNS 96

70 class NobelNS::Angle 103

71 class NobelNS::AngleRep 106

72 class NobelNS::BlockException 107

73 class NobelNS::BlockExceptionRep 109

74 class NobelNS::InvariantException 110

75 class NobelNS::Length 110

76 class NobelNS::LengthRep 114

77 class NobelNS::Pose 114

78 class NobelNS::PoseRep 118

79 class NobelNS::RobotAcceleration 119

80 class NobelNS::RobotAccelerationRep 121

81 class NobelNS::RobotVelocity 122

82 class NobelNS::RobotVelocityRep 125

83 class NobelNS::SelfTestingObject 126

84 class NobelNS::Vector 126

85 class NobelNS::VectorRep 130

86 class RGB24Image 131

87 class RobotControl 132

88 class RobotControlRep 133

89 class STC 136

90 class STCRep 138

91 class YUV422Image 138

92 struct poly_vertex 140

93 struct polygon 141

94 struct rgb 141

**95 struct rgb24** 142

**96 struct yuv** 142

**97 struct yuv422** 143

**Abstract**

NOBEL Reference Documentation.
Generated: forsblom on sirppi, Thu May 3 11:21:36 2001..

# 1 NOBEL Class List

| | | |
|---|---|---|
| ExternalNS | | 10 |
| External | | 11 |
| InputFileNotFoundException | | 12 |
| InputFileSyntaxException | | 13 |
| Output | Tulostuksen hallinta ja toteutus | 13 |
| OutputRep | | 15 |
| Settings | | 16 |
| SettingsRep | | 18 |
| HELVision | | 18 |
| color_info | | 21 |
| earlierRegion | | 22 |
| region | | 23 |
| rle | | 24 |
| InputNS | | 25 |
| Input | | 25 |
| *InputInterface* | InputInterface määrittelee rajapinnan oliolle, jolta järjestelmä voi navigoinnin aikana kysyä, onko käyttäjä antanut lopetuskomennon. | 26 |
| InputRep | | 27 |
| ReadError | ReadError on poikkeus, joka heitetään, kun käyttäjän syötettä on mahdotonta lukea. | 28 |
| Localization | | 28 |

LocalizationNS Kahdesta limittäisestä kuvasta syntyneiden kaksoismaamerkkien välinen enimmäiserotuskulma, joka johtaa maamerkkien yhdistämiseen. . . . . . . . . . . . . . . . 29

Bitmap . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

BitmapRep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

CaptureException Kuvankaappauspoikkus, joko videokortin initialisointi, tai itse kuvan kaappaus epäonnistui. . . . . . 35

LandmarkRecognition Maamerkkien tunnistus -luokka. . . . . . . . . . . . . . . . . . 35

LandmarkRecognitionRep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

Landmarks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

LandmarksFileNotFoundException . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38

*LandmarksInterface* LandmarksInterface määrittelee rajapinnan säiliöoliolle, joka sisältää tiedot ympäristön maamerkeistä ja niiden sijainneista. Olio tuntee tukivektorikonetiedostot (SVM-tiedostot), joita käytetään merkkien tunnistukseen. Maamerkit nimetään niitä vastaavien SVM-tiedostonimien mukaan. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39

LandmarksRep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40

Localization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40

LocalizationException . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41

LocalizationRep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42

PanTiltException Kameran pantilt-yksikon initialisoinnin epäonnistumiseen liittyvä poikkeus. . . . . . . . . . . . . . 44

Panorama . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

PanoramaRep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46

Segmentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47

SegmentationException Segmentointipoikkeus. . . . . . . . . . . . . . . . . . . . . . . . . . . 48

SegmentationRep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

View Kuvien kaappaus panoraamana. . . . . . . . . . . . . . . . . . 50

ViewRep . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51

MapNS Osoitin sisäiseen toteutukseen. . . . . . . . . . . . . . . . . . . 52

BadDestinationException . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

BadScaleException — Gridin konstruktori heittää BadScaleExceptionin, jos mittakaava on liian pieni. . . . . . . . . . . . . . . . . . . . 54

Grid — Grid-olio on kartta ympäröivästä maailmasta. Se on kaksiulotteinen ruudukko. . . . . . . . . . . . . . . . . . . . . . 54

GridCell — GridCell-oliot ovat kartan alkioita. Niihin tallennetaan tiedot esteistä ja maalipisteeseen johtavista reiteistä. GridCell voi olla joko kartan este, vapaa ruutu tai marginaaliin kuuluva ruutu. . . . . . . . . . 57

GridCellRep — GridCell-luokan toteutus. . . . . . . . . . . . . . . . . . . . . . . . 60

*GridInterface* — GridInterface määrittelee reitinlaskentaosajärjestelmän kartan tietorakenteen rajapinnan: Kartta on kaksiulotteinen ruudukko. Ruudukon alkioita nimitetään tässä soluiksi tai ruuduiksi. . 61

GridIterator — . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65

*GridIteratorInterface* — GridIteratorInterface määrittelee rajapinnan iteraattoreille, joilla voi osoittaa kartan soluihin. Iteraattoreilla voi selata kartan jokaisen solun vierussolut ja merkitä karttaan reittejä. Iteraattori voi osoittaa vain soluun, joka on vapaa ja marginaalin ulkopuolella. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67

GridIteratorRep — GridIteratorin toteutus. . . . . . . . . . . . . . . . . . . . . . . . . 70

GridRep — Grid-luokan toteutus. . . . . . . . . . . . . . . . . . . . . . . . . . . 71

Map — . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76

MapFileNotFoundException — Gridin konstruktori heittää MapFileNotFoundException, jos mapFile-tiedoston avaus epäonnistuu. 78

*MapInterface* — MapInterface määrittelee reitinlaskentaosajärjestelmän tarjoamat palvelut. . . . . . . . . . . 79

MapRep — Luokan Map toteutus. . . . . . . . . . . . . . . . . . . . . . . . . . . 81

MovementNS — Liikkumisosajärjestelmä. . . . . . . . . . . . . . . . . . . . . . . . 82

*Detector* — . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83

*Movement* — . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84

RobotMover — . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85

RobotMoverRep — . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87

*Roller* — . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88

ScoutRoller Robotin liikuttaja. .... 89
ScoutRollerRep .... 91
Sonar .... 92
SonarRep .... 95
NobelNS Pikkuolioita. .... 96
Angle Itsekonvertoiva kulmayksikkö. .... 103
AngleRep .... 106
BlockException BlockException on poikkeusolio, jonka Movement-olio heittää yllättävän esteen ilmestyessä. .... 107
BlockExceptionRep .... 109
InvariantException .... 110
Length Itsekonvertoiva pituusyksikkö. .... 110
LengthRep .... 114
Pose Sijainti+suunta. .... 114
PoseRep .... 118
RobotAcceleration Itsekonvertoiva kiihtyvyysluokka. .... 119
RobotAccelerationRep .... 121
RobotVelocity Itsekonvertoiva nopeusyksikkö. .... 122
RobotVelocityRep .... 125
*SelfTestingObject* .... 126
Vector .... 126
VectorRep .... 130
RGB24Image .... 131
RobotControl Robotin navigoinnista huolehtiva yksikkö. .... 132
RobotControlRep RobotControl:n privaatti osa. .... 133
STC Yllättävien poikkeusten käsittelijän alustaja. .... 136
STCRep Sisäinen esitys poikkeuskäsittelijän alustajalle .... 138
YUV422Image .... 138

poly_vertex ........ 140

polygon ........ 141

rgb ........ 141

rgb24 ........ 142

yuv ........ 142

yuv422 ........ 143

## 2 NOBEL Class Hierachy

*Detector*
  Sonar
External
ExternalNS
*GridInterface*
  Grid
*GridIteratorInterface*
  GridIterator
HELVision
*InputInterface*
  Input
InputNS
*LandmarksInterface*
  Landmarks
LandmarksRep
Localization
LocalizationNS
*MapInterface*
  Map
MapNS
*Movement*
  RobotMover
MovementNS
NobelNS
RGB24Image
*Roller*
  ScoutRoller
SelfTestingObject (unknown)
  Bitmap
  BitmapRep
  LandmarkRecognition
  Panorama
  PanoramaRep
  RobotControl
  RobotControlRep
  Settings
  View
  ViewRep
*SelfTestingObject*

- Angle
- AngleRep
- BlockException
- BlockExceptionRep
- Grid
- GridCell
- GridCellRep
- GridIterator
- GridIteratorRep
- GridRep
- Input
- InputRep
- LandmarkRecognitionRep
- Landmarks
- Length
- LengthRep
- Localization
- LocalizationRep
- Map
- MapRep
- Output
- OutputRep
- Pose
- PoseRep
- RobotAcceleration
- RobotAccelerationRep
- RobotMover
- RobotMoverRep
- RobotVelocity
- RobotVelocityRep
- STC
- STCRep
- ScoutRoller
- ScoutRollerRep
- Segmentation
- SegmentationRep
- SettingsRep
- Sonar
- SonarRep
- Vector
- VectorRep

YUV422Image
color_info
earlierRegion
poly_vertex
polygon
region
rgb
rgb24
rle
std::runtime_error (unknown)
- BlockException

std::runtime_error (unknown)
- InvariantException

std::runtime_error (unknown)

InputFileSyntaxException
std::runtime_error (unknown)
InputFileNotFoundException
std::runtime_error (unknown)
ReadError
std::runtime_error (unknown)
LocalizationException
std::runtime_error (unknown)
SegmentationException
std::runtime_error (unknown)
CaptureException
std::runtime_error (unknown)
LandmarksFileNotFoundException
std::runtime_error (unknown)
PanTiltException
std::runtime_error (unknown)
BadScaleException
std::runtime_error (unknown)
MapFileNotFoundException
std::runtime_error (unknown)
BadDestinationException
yuv
yuv422

## 3 NOBEL Globals

### Index by file

#### Localization/Capture.h

- bool **videoCapture** () ................
- bool **videoConnect** (int width = VIDEO_DEFAULT_WIDTH, int height = VIDEO_DEFAULT_HEIGHT, int format = VIDEO_PALETTE_RGB24) ................
- void **videoDisconnect** () ................
- void **videoCopyBuffer** (unsigned char* buf) ................

#### Localization/clientincludes.h

- typedef double **DataElt** ................
- typedef float **DataElt** ................
- typedef int **DataElt** ................
- typedef short **DataElt** ................
- typedef DataElt * **DataPt** ................
- typedef double **KernVal** ................
- typedef float **KernVal** ................
- typedef int **KernVal** ................

- typedef DataElt * **DataPt**
- typedef SvmBase<DataPt, KernVal> **MySvmBase**
- typedef SvmSmallOpt<DataPt, KernVal> **MySvmSmallOpt**
- typedef SvmLargeOpt<DataPt, KernVal> **MySvmLargeOpt**
- typedef SvmTest<DataPt, KernVal> **MySvmTest**
- typedef SvmKernCache<DataPt, KernVal> **MySvmKernCache**
- enum **KernType** { linear, polynomial, gaussian }
- enum **MachType** {dense, sparse01, sparseN}
- const int **ptSizeFunc** (const DataPt &pt)
- const void **printPoint** (ofstream *os, DataPt pt)
- const void **readPoint** (ifstream *is, DataPt *ptr)
- const int **ptSizeFunc** (const DataPt &pt)
- const void **printPoint** (ofstream *os, DataPt pt)
- const void **readPoint** (ifstream *is, DataPt *ptr)

**Localization/helvision.h**

- inline ostream & **operator <<** (ostream & out, HELVision::rle& r)

**Localization/itos.h**

- inline std::string **my_itos** ( int n )

**Localization/kernelfuncs.h**

- const KernVal **linearProduct** (DataPt p1, DataPt p2)
- const KernVal **polynomialProduct** (DataPt p1, DataPt p2)
- const KernVal **gaussianProduct** (DataPt p1, DataPt p2)
- const KernVal **sparse01InternalLinProd** (DataPt p1, DataPt p2)
- const KernVal **sparse01BinaryProduct** (DataPt p1, DataPt p2)
- const KernVal **sparse01PolynomialProduct** (DataPt p1, DataPt p2)
- const KernVal **sparse01GaussianProduct** (DataPt p1, DataPt p2)
- const KernVal **sparseNInternalLinProd** (DataPt p1, DataPt p2)
- const KernVal **sparseNLinearProduct** (DataPt p1, DataPt p2)
- const KernVal **sparseNPolynomialProduct** (DataPt p1, DataPt p2)
- const KernVal **sparseNGaussianProduct** (DataPt p1, DataPt p2)

**Main/STC.h**


- void **niceAbort** () .......... 10


**rmap/borders.h**


- extern int **AREA_BOTTOM** ..........
- extern int **AREA_LEFT** ..........
- extern int **AREA_RIGHT** ..........
- extern int **AREA_W** ..........


**rmap/parsemap.h**


- struct polygon * **parse_mapfile** (char *) ..........


**rmap/poly.h**


- void **draw_polygon** (struct polygon *, unsigned char color) ..........


**rmap/tri.h**


- void **set_poly_dest** (char *) ..........
- void **set_screen_origo** (int, int) ..........
- void **draw_tri_v** (struct poly_vertex *,struct poly_vertex *,struct poly_vertex *, unsigned char) ..........
- void **draw_xline** (int, int, int, unsigned char) ..........
- void **draw_buffer_d** (int *, int *, int, int, unsigned char) ..........


**rmap/vesa.h**


- void **vesa_window** (int) ..........
- void **set_palette_color** (int, char, char, char) ..........
- void **draw_pixel** (char *, int, int, char) ..........
- void **wait_for_vertical_retrace** (void) ..........
- void **set_video_mode** (int) ..........
- void **flip** (char *, char *) ..........


**scout.h**


- void **init_robot** () ..........
- void **uninit_robot** () ..........
- void **turn_sonars** (int state) ..........
- void **turn_sonars_on** () ..........
- void **turn_sonars_off** () ..........
- bool **isMoving** () ..........
- void **wait_for_stop** () ..........

### Detailed Description

**void** `niceAbort` **()** []

`#include <Main/STC.h>`

Yllättävien poikkeusten käsittelijä. Tulostaa virheilmoituksen (yleisen sekä UNEXPECTED_EXCEPTION_ERROR_MESSAGEn). Lisäksi jos yllättävä poikkeus oli std::exception tai sen aliluokka, tulostaa poikkeuksen sisältämän viestin (std::exception::what()). Lopuksi päättää ohjelman abort()illa.

## 4 namespace ExternalNS

### Public Types


- class **External** ........ 11
- class **OutputRep** ........ 15
- enum **ModuleName** {Mov,Loc,Map,Con} ........ 10
- class **Output** ........ 13
- class **SettingsRep** ........ 18
- class **InputFileNotFoundException** ........ 12
- class **InputFileSyntaxException** ........ 13
- class **Settings** ........ 16


### Detailed Description

### Member Documentation

**ExternalNS::External (class)**

**ExternalNS::OutputRep (class)**

**enum** `ExternalNS::ModuleName` **{Mov,Loc,Map,Con}**

Osajärjestelmien tunnisteet. Mov = Movement, Loc = Localization, Map = Map ja Con = RobotControl.

**ExternalNS::Output (class)**

Tulostuksen hallinta ja toteutus. Output-olio osaa tulostaa muiden osajärjestelmien toimittamia String-olioita ja hallita tulostustasoja.

**ExternalNS::SettingsRep** (class)

**ExternalNS::InputFileNotFoundException** (class)

**ExternalNS::InputFileSyntaxException** (class)

**ExternalNS::Settings** (class)

## 5 class ExternalNS::External

`#include <External/External.h>`

### Public Static Methods

- static void **initExternals** (string filename) .......... 11
- static void **print** (string text, ModuleName module, int verb_level ) .......... 11
- static double **getDoubleSetting** (string setting_id) .......... 12
- static int **getIntSetting** (string setting_id) .......... 12
- static string **getStringSetting** (string setting_id) .......... 12
- static string **itos** (int n) .......... 12

### Private Methods

- **External** () .......... 12

### Detailed Description

### Member Documentation

**void** `ExternalNS::External::initExternals` **(string filename)** [**static** ]

Alustaa Externalin käyttövalmiiksi. Lukee ajotiedoston Settingsiin.
**Parameters**:

| | |
|---|---|
| *filename* | ajotiedoston nimi |

**Throws**: InputFileNotFoundException, InputFileSyntaxException

**void** `ExternalNS::External::print` **(string text, ModuleName module, int verb_level )** [**static** ]

Tulostaa ja kirjoittaa lokiin annetun tekstin ajotiedostossa määriteltyjen tulostusasetusten mukaan.
**Parameters**:

| | |
|---|---|
| *text* | tulostettava teksti |
| *ModuleName* | osajärjestelmän tunniste |
| *verb_level* | tulostustaso |

**double** `ExternalNS::External::getDoubleSetting` **(string setting_id)** [**static** ]

Hakee annettua tunnistetta vastaavan double-arvon.

**Parameters**:

| | |
|---|---|
| *setting_id* | asetustunniste ajotiedostossa |

**Throws**: InputFileSyntaxException

**int** `ExternalNS::External::getIntSetting` **(string setting_id)** [**static** ]

Hakee annettua tunnistetta vastaavan int-arvon.

**Parameters**:

| | |
|---|---|
| *setting_id* | asetustunniste ajotiedostossa |

**Throws**: InputFileSyntaxException

**string** `ExternalNS::External::getStringSetting` **(string setting_id)** [**static** ]

Hakee annettua tunnistetta vastaavan string-arvon.

**Parameters**:

| | |
|---|---|
| *setting_id* | asetustunniste ajotiedostossa |

**Throws**: InputFileSyntaxException

**string** `ExternalNS::External::itos` **(int n)** [**static** ]

Muuntaa kokonaisluvun merkkijonoksi.

**Parameters**:

| | |
|---|---|
| *n* | muunnettava luku |

**Return**:

merkkijonoesitys annetusta luvusta

`ExternalNS::External::External` **()** [**private** ]

## 6 class ExternalNS::InputFileNotFoundException

`#include <External/Settings.h>`

Inherits: std::runtime_error

*public*

(unknown)

### Public Methods

- **InputFileNotFoundException** (const std::string& s) ............ 13
- virtual ∼**InputFileNotFoundException** () ............ 13

### Detailed Description

### Member Documentation

ExternalNS::InputFileNotFoundException::InputFileNotFoundException **(const std::string& s)** []

ExternalNS::InputFileNotFoundException::∼InputFileNotFoundException **()** [ **virtual**]

## 7 class ExternalNS::InputFileSyntaxException

#include <External/Settings.h>
Inherits: std::runtime_error

*public*

(unknown)

### Public Methods

- **InputFileSyntaxException** (const std::string& s) ............ 13
- virtual ∼**InputFileSyntaxException** () ............ 13

### Detailed Description

### Member Documentation

ExternalNS::InputFileSyntaxException::InputFileSyntaxException **(const std::string& s)** []

ExternalNS::InputFileSyntaxException::∼InputFileSyntaxException **()** [ **virtual**]

## 8 class ExternalNS::Output

Tulostuksen hallinta ja toteutus.

#include <External/Output.h>
Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods

- void **print** (const string& text, ModuleName module, int verbosityLevel) ............ 14
- void **init** () ............ 14
- virtual bool **invariant** () const ............ 14
- **Output** () ............ 14
- virtual ∼**Output** () ............ 14

### Private Members


- OutputRep* **rep** .......... 15


### Detailed Description

Tulostuksen hallinta ja toteutus. Output-olio osaa tulostaa muiden osajärjestelmien toimittamia String-olioita ja hallita tulostustasoja.

### Member Documentation

**void** `ExternalNS::Output::print` **(const string& text, ModuleName module, int verbosityLevel)** []

Tulostaa tekstin lokitiedostoon ja/tai tulostusvirtaan, jos loki- ja tulostusasetukset oikeuttavat.

**Parameters**:

| | |
|---|---|
| *text* | Tulostettava teksti |
| *module* | Tulostava moduuli |
| *verbosityLevel* | Tulostettavan tekstin tulostustaso |

**void** `ExternalNS::Output::init` **()** []

Hakee tulostusasetukset External-luokalta sekä

**bool** `ExternalNS::Output::invariant` **()** [**const virtual**]

Tarkista olion eheys.

**Return**:

olion eheys

`ExternalNS::Output::Output` **()** []

Konstruktori.

`ExternalNS::Output::~Output` **()** [ **virtual**]

**OutputRep*** `ExternalNS::Output::rep` [**private**]

## 9 class ExternalNS::OutputRep

`#include <External/OutputRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods


- int **getPrintLevel** (ModuleName module) .......... 15

- int **getLogLevel** (ModuleName module) ........ 16
- void **createLogFile** () ........ 16
- virtual bool **invariant** () const ........ 16
- **OutputRep** () ........ 16
- virtual **~OutputRep** () ........ 16


### Public Static Methods


- static std::string **toString** (int i) ........ 16


### Public Members


- int **printLevel**[4] ........ 15
- int **logLevel**[4] ........ 15
- ofstream **logfile** ........ 15


### Detailed Description

### Member Documentation

**int** `ExternalNS::OutputRep::printLevel[4]`

**int** `ExternalNS::OutputRep::logLevel[4]`

**ofstream** `ExternalNS::OutputRep::logfile`

**int** `ExternalNS::OutputRep::getPrintLevel` **(ModuleName module)** []

Tutkii moduulin tulostustason tulostusvirtaan.
**Parameters**:

*module* Moduuli, jonka tulostustaso tutkitaan

**Return**:

Moduulin tulostustaso

**int** `ExternalNS::OutputRep::getLogLevel` **(ModuleName module)** []

Tutkii moduulin tulostustason lokitiedostoon.
**Parameters**:

*module* Moduuli, jonka tulostustaso tutkitaan

**Return**:

Moduulin tulostustaso

**void** `ExternalNS::OutputRep::createLogFile` **()** []

Luo lokitiedoston ajotiedostossa anetun nimen mukaisesti. Jos sen niminen tiedosto on jo olemassa, lisätään nimen perään ensimmäinen järjestysnumero, joka on vapaa (2 tai suurempi).

**std::string** `ExternalNS::OutputRep::toString` **(int i)** [**static** ]

**bool** `ExternalNS::OutputRep::invariant` **()** [**const virtual**]

Invariantin tarkistus.

**Return**:

true

`ExternalNS::OutputRep::OutputRep` **()** []

`ExternalNS::OutputRep::~OutputRep` **()** [ **virtual**]

## 10 class ExternalNS::Settings

`#include <External/Settings.h>`
Inherits: SelfTestingObject

*public*

(unknown)

### Public Methods


- virtual void **init** (string filename) .......... 17
- virtual double **getDoubleSetting** (string setting_id) .......... 17
- virtual int **getIntSetting** (string setting_id) .......... 17
- virtual string **getStringSetting** (string setting_id) .......... 17
- **Settings** () .......... 17
- virtual bool **invariant** () const .......... 17


### Private Members


- SettingsRep* **rep** .......... 18


### Detailed Description

### Member Documentation

**void** `ExternalNS::Settings::init` **(string filename)** [ **virtual**]

Alustaa Settingsin, lukee ja jäsentää ajotiedoston.

**Parameters**:

*filename* ajotiedoston nimi.

**double** `ExternalNS::Settings::getDoubleSetting` **(string setting_id)** [ **virtual**]

Palauttaa annettua asetustunnistetta vastaavan double-arvon.

**Parameters**:

*setting_id* asetustunniste.

**int** `ExternalNS::Settings::getIntSetting` **(string setting_id)** [ **virtual**]

Palauttaa annettua asetustunnistetta vastaavan int-arvon.

**Parameters**:

*setting_id* asetustunniste.

**string** `ExternalNS::Settings::getStringSetting` **(string setting_id)** [ **virtual**]

Palauttaa annettua asetustunnistetta vastaavan string-arvon.

**Parameters**:

*setting_id* asetustunniste.

`ExternalNS::Settings::Settings` **()** []

Konstruktori.

**bool** `ExternalNS::Settings::invariant` **()** [**const virtual**]

**SettingsRep*** `ExternalNS::Settings::rep` [**private**]

## 11 class ExternalNS::SettingsRep

`#include <External/SettingsRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods


- virtual bool **invariant** () const ........ 18
- **SettingsRep** () ........ 18
- virtual ~**SettingsRep** () ........ 18


### Public Members


- map<string, string> **s_data** ........ 18

### Detailed Description

### Member Documentation

**map<string, string>** `ExternalNS::SettingsRep::s_data`

**bool** `ExternalNS::SettingsRep::invariant` **()** [**const virtual**]

`ExternalNS::SettingsRep::SettingsRep` **()** []

`ExternalNS::SettingsRep::~SettingsRep` **()** [ **virtual**]

## 12 class HELVision

`#include <Localization/helvision.h>`

### Public Types


- struct **rle** .......... 24
- struct **color_info** .......... 21
- struct **region** .......... 23


### Public Methods


- **HELVision** () .......... 21
- ~**HELVision** () .......... 21
- bool **initialize** (int nwidth,int nheight) .......... 21
- bool **loadOptions** (char *filename) .......... 21
- bool **saveOptions** (char *filename) .......... 21
- void **close** () .......... 21
- bool **testClassify** (rgb *out,yuv422 *image) .......... 21
- bool **getThreshold** (int color, int &y_low,int &y_high, int &u_low,int &u_high, int &v_low,int &v_high) 21
- bool **setThreshold** (int color, int y_low,int y_high, int u_low,int u_high, int v_low,int v_high) .......... 21
- char * **getColorName** (int color) .......... 21
- color_info * **getColorInfo** (int color) .......... 21
- void **getColorInfo** (int color,color_info &info) .......... 21
- void **setColorInfo** (int color,color_info &info) .......... 21
- bool **processFrame** (yuv422 *image) .......... 21
- int **numRegions** (int color_id) .......... 21
- region * **getRegions** (int color_id) .......... 21

## Protected Types

- struct **earlierRegion** ... 22

## Protected Methods

- void **classifyFrame** (yuv422 *img,unsigned *map) ... 21
- int **encodeRuns** (rle *out,unsigned *map) ... 21
- void **connectComponents** (rle *map,int num) ... 21
- int **extractRegions** (region *reg,rle *rmap,int num) ... 21
- int **separateRegions** (region *reg,int num) ... 21
- region * **sortRegionListByArea** (region *list,int passes) ... 21
- void **sortRegions** (int max_area) ... 21
- void **clear** () ... 21
- void **clearDataBuffers** () ... 21

## Protected Members

- unsigned **yclass[CMV_COLOR_LEVELS]** ... 21
- unsigned **uclass[CMV_COLOR_LEVELS]** ... 21
- unsigned **vclass[CMV_COLOR_LEVELS]** ... 21
- region **region_table[CMV_MAX_REGIONS]** ... 21
- region * **region_list[CMV_MAX_COLORS]** ... 21
- int **region_count[CMV_MAX_COLORS]** ... 21
- rle **rmap[CMV_MAX_RUNS]** ... 21
- color_info **colors[CMV_MAX_COLORS]** ... 21
- int **width** ... 21
- int **height** ... 21
- unsigned * **map** ... 21

**Detailed Description**

**Member Documentation**

**HELVision::rle** (struct)

**HELVision::color_info** (struct)

**HELVision::region** (struct)

**unsigned** `HELVision::yclass[CMV_COLOR_LEVELS]` [protected]

**unsigned** `HELVision::uclass[CMV_COLOR_LEVELS]` [protected]

**unsigned** `HELVision::vclass[CMV_COLOR_LEVELS]` [protected]

**region** `HELVision::region_table[CMV_MAX_REGIONS]` [protected]

**region *** `HELVision::region_list[CMV_MAX_COLORS]` [protected]

**int** `HELVision::region_count[CMV_MAX_COLORS]` [protected]

**rle** `HELVision::rmap[CMV_MAX_RUNS]` [protected]

**color_info** `HELVision::colors[CMV_MAX_COLORS]` [protected]

**int** `HELVision::width` [protected]

**int** `HELVision::height` [protected]

**unsigned *** `HELVision::map` [protected]

**HELVision::earlierRegion** (struct) [protected]

**void** `HELVision::classifyFrame` **(yuv422 *img,unsigned *map)** [protected ]

**int** `HELVision::encodeRuns` **(rle *out,unsigned *map)** [protected ]

**void** `HELVision::connectComponents` **(rle *map,int num)** [protected ]

**int** `HELVision::extractRegions` **(region *reg,rle *rmap,int num)** [protected ]

**int** `HELVision::separateRegions` **(region *reg,int num)** [protected ]

**region *** `HELVision::sortRegionListByArea` **(region *list,int passes)** [protected ]

**void** `HELVision::sortRegions` **(int max_area)** [protected ]

**void** `HELVision::clear` **()** [protected ]

**void** `HELVision::clearDataBuffers` **()** [protected ]

`HELVision::HELVision` **()** []

`HELVision::~HELVision` **()** []

**bool** `HELVision::initialize` **(int nwidth,int nheight)** []

**bool** `HELVision::loadOptions` **(char *filename)** []

**bool** `HELVision::saveOptions` **(char *filename)** []

**void** `HELVision::close` **()** []

**bool** `HELVision::testClassify` **(rgb *out,yuv422 *image)** []

**bool** `HELVision::getThreshold` **(int color, int &y_low,int &y_high, int &u_low,int &u_high, int &v_low,int &v_high)** []

**bool** `HELVision::setThreshold` **(int color, int y_low,int y_high, int u_low,int u_high, int v_low,int v_high)** []

**char *** `HELVision::getColorName` **(int color)** []

**color_info *** `HELVision::getColorInfo` **(int color)** []

**void** `HELVision::getColorInfo` **(int color,color_info &info)** []

**void** `HELVision::setColorInfo` **(int color,color_info &info)** []

**bool** `HELVision::processFrame` **(yuv422 *image)** []

**int** `HELVision::numRegions` **(int color_id)** []

**region *** `HELVision::getRegions` **(int color_id)** []

## 13 struct HELVision::color_info

```
#include <Localization/helvision.h>
```

### Public Members


- rgb **color** .......... 22
- char * **name** .......... 22
- double **merge** .......... 22
- int **y_low** .......... 22
- int **y_high** .......... 22
- int **u_low** .......... 22
- int **u_high** .......... 22
- int **v_low** .......... 22
- int **v_high** .......... 22


### Detailed Description

### Member Documentation

**rgb** `HELVision::color_info::color`

**char *** `HELVision::color_info::name`

**double** `HELVision::color_info::merge`

**int** `HELVision::color_info::y_low`

**int** `HELVision::color_info::y_high`

**int** `HELVision::color_info::u_low`

**int** `HELVision::color_info::u_high`

**int** `HELVision::color_info::v_low`

**int** `HELVision::color_info::v_high`

## 14 struct HELVision::earlierRegion

`#include <Localization/helvision.h>`

### Public Methods


- bool **operator()** (region r1, region r2) const .......... 23

#### Detailed Description

#### Member Documentation

**bool** `HELVision::earlierRegion::operator()` **(region r1, region r2)** [**const** ]

## 15 struct HELVision::region

`#include <Localization/helvision.h>`

#### Public Members


- int **color** .......... 24
- int **area** .......... 24
- int **x1** .......... 24
- int **y1** .......... 24
- int **x2** .......... 24
- int **y2** .......... 24
- float **cen_x** .......... 24
- float **cen_y** .......... 24
- int **sum_x** .......... 24
- int **sum_y** .......... 24
- region * **next** .......... 24
- vector<rle> **pixels** .......... 24

### Detailed Description

### Member Documentation

**int** `HELVision::region::color`

**int** `HELVision::region::area`

**int** `HELVision::region::x1`

**int** `HELVision::region::y1`

**int** `HELVision::region::x2`

**int** `HELVision::region::y2`

**float** `HELVision::region::cen_x`

**float** `HELVision::region::cen_y`

**int** `HELVision::region::sum_x`

**int** `HELVision::region::sum_y`

**region *** `HELVision::region::next`

**vector<rle>** `HELVision::region::pixels`

## 16 struct HELVision::rle

`#include <Localization/helvision.h>`

### Public Members


- unsigned **color** .......... 25
- int **x** .......... 25
- int **y** .......... 25
- int **length** .......... 25
- int **parent** .......... 25

### Detailed Description

### Member Documentation

**unsigned** `HELVision::rle::color`

**int** `HELVision::rle::x`

**int** `HELVision::rle::y`

**int** `HELVision::rle::length`

**int** `HELVision::rle::parent`

## 17 namespace InputNS

### Public Types


- class **InputRep** .......... 27
- class **Input** .......... 25
- class **ReadError** .......... 28
- class ***InputInterface*** .......... 26


### Detailed Description

### Member Documentation

**InputNS::InputRep (class)**

**InputNS::Input (class)**

**InputNS::ReadError (class)**

**InputNS::InputInterface (class)**

**Author**:

Jarkko Salminen

## 18 class InputNS::Input

`#include <Input/Input.h>`
Inherits: InputNS::InputInterface
*public*
(), NobelNS::SelfTestingObject
*public*
()

## Public Methods


- virtual bool **isHaltSet** () const ........ 26
- **Input** () ........ 26
- virtual ~**Input** () ........ 26
- virtual bool **invariant** () const ........ 26


## Private Members


- InputRep* **rep** ........ 26


## Detailed Description

## Member Documentation

**bool** `InputNS::Input::isHaltSet` **()** [**const virtual**]

isHaltSet kertoo, onko käyttäjä antanut lopetuskomennon. Funktio on toteutettu siten, että se ei blokkaa prosessia, jos mitään syötettä ei ole annettu.

Jos käyttäjän syötteen lukeminen epäonnistuu heitetään ReadError. Poikkeusta ei heitetä, jos käyttäjä ei ole antanut mitään syötettä.

**Return**:

Palauttaa true, jos käyttäjä on antanut lopetuskomennon. Muuten false.

`InputNS::Input::Input` **()** []

Konstruktori heittää ReadError-olion, jos syötteen lukeminen on mahdotonta.

`InputNS::Input::~Input` **()** [ **virtual**]

**bool** `InputNS::Input::invariant` **()** [**const virtual**]

Tarkistaa olion tilan eheyden.

**InputRep*** `InputNS::Input::rep` [**private**]

# 19 class InputNS::InputInterface

InputInterface määrittelee rajapinnan oliolle, jolta järjestelmä voi navigoinnin aikana kysyä, onko käyttäjä antanut lopetuskomennon.

**Abstract class.**

`#include <Input/InputInterface.h>`

Input

### Public Methods


- virtual bool ***isHaltSet*** () const ........ 27
- virtual ~**InputInterface** () ........ 27


### Detailed Description

**Author**:

Jarkko Salminen

### Member Documentation

**bool** *`InputNS::InputInterface::isHaltSet`* **()** [**const pure virtual**]

Palauttaa true, jos käyttäjä on antanut lopetuskomennon. Muuten false. Kun käyttäjä on kerran antanut lopetuskomennon, funktio palauttaa uudelleen kutsuttaessa aina true, vaikka lopetuskomentoa ei olisi annettu uudestaan.

Jos käyttäjän syötteen lukeminen epäonnistuu heitetään ReadError. Poikkeusta ei heitetä, jos käyttäjä ei ole antanut mitään syötettä.

`InputNS::InputInterface::~InputInterface` **()** [ **virtual**]

## 20 class InputNS::InputRep

`#include <Input/InputRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods


- explicit **InputRep** (int) ........ 28
- virtual ~**InputRep** () ........ 28
- virtual bool **invariant** () const ........ 28


### Public Members


- int **fd** ........ 28
- bool **halt** ........ 28

### Detailed Description

### Member Documentation

**int** `InputNS::InputRep::fd`

**bool** `InputNS::InputRep::halt`

**explicit** `InputNS::InputRep::InputRep` **(int)** []

`InputNS::InputRep::~InputRep` **()** [ **virtual**]

**bool** `InputNS::InputRep::invariant` **()** [**const virtual**]

## 21 class InputNS::ReadError

ReadError on poikkeus, joka heitetään, kun käyttäjän syötettä on mahdotonta lukea.

`#include <Input/InputInterface.h>`

Inherits: std::runtime_error

*public*

(unknown)

### Public Methods

- **ReadError** (const std::string& s) ........ 28
- virtual **~ReadError** () ........ 28

### Detailed Description

### Member Documentation

`InputNS::ReadError::ReadError` **(const std::string& s)** []

Konstruktori saa parametrinaan virheilmoituksen.

`InputNS::ReadError::~ReadError` **()** [ **virtual**]

## 22 namespace Localization

### Public Methods

- void **sorter** (std::vector< std::pair<std::string, NobelNS::Angle> >& v) ........ 29

### Detailed Description

### Member Documentation

**void** `Localization::sorter` **(std::vector< std::pair<std::string, NobelNS::Angle> >& v)** []

## 23 namespace LocalizationNS

Kahdesta limittäisestä kuvasta syntyneiden kaksoismaamerkkien välinen enimmäiserotuskulma, joka johtaa maamerkkien yhdistämiseen.

### Public Types


- class **BitmapRep** ........ 34
- class **Bitmap** ........ 33
- class **LandmarkRecognitionRep** ........ 37
- class **LandmarkRecognition** ........ 35
- class **LandmarksRep** ........ 40
- class **Landmarks** ........ 37
- class ***LandmarksInterface*** ........ 39
- class **LocalizationRep** ........ 42
- class **LocalizationException** ........ 41
- class **Localization** ........ 40
- class **SegmentationException** ........ 48
- class **CaptureException** ........ 35
- class **LandmarksFileNotFoundException** ........ 38
- class **PanTiltException** ........ 44
- class **PanoramaRep** ........ 46
- class **Panorama** ........ 45
- class **SegmentationRep** ........ 49
- class **Segmentation** ........ 47
- class **ViewRep** ........ 51
- class **View** ........ 50

### Public Methods


- bool **initSegmenter** (string colorFileName) ........ 30
- void **closeSegmenter** () ........ 30
- std::vector< pair<Bitmap,double> > **getSegments** (YUV422Image img) ........ 30
- void **initCapture** () ........ 30
- YUV422Image **getImage** () ........ 31
- void **releaseCapture** () ........ 31
- void **panTilt** (const Angle& horizontal, const Angle& vertical) ........ 32
- bool **initMachines** (const std::vector<std::string> landmarks) ........ 32
- std::string **recognize** (Bitmap shape) ........ 32


### Detailed Description

Kahdesta limittäisestä kuvasta syntyneiden kaksoismaamerkkien välinen enimmäiserotuskulma, joka johtaa maamerkkien yhdistämiseen...(huh?!).

### Member Documentation

**bool** `LocalizationNS::initSegmenter` **(string colorFileName)** []

Alustaa segmentoijan.

**Parameters**:

*colorFileName* väritiedoston nimi.

**void** `LocalizationNS::closeSegmenter` **()** []

**std::vector**< **pair**<**Bitmap,double**> > `LocalizationNS::getSegments` **(YUV422Image img)** []

Segmentoi annetun kuvan.

**Return**:

vektori, joka sisältää <Bitmap, double> -pareja. Näissä Bitmap on yksittäinen segmentti ja double ko. segmentin keskipisteen sijainti alkuperäisessä kuvassa normalisoituna välille [0-1].

**LocalizationNS::BitmapRep (class)**

**LocalizationNS::Bitmap (class)**

**void** `LocalizationNS::initCapture` **()** []

Initialisoi kuvankaappauskortin.

**Return**:

true, jos onnistui

**Throws**: CaptureException

**YUV422Image** `LocalizationNS::getImage` **()** []

Ottaa kuvan ja luo YUV422Image-olion, joka sisältää ko. kuvan.

**Return**:

Otettu kuva.

**Throws**: CaptureException

**void** `LocalizationNS::releaseCapture` **()** []

Vapauttaa kaappauskortin.

**LocalizationNS::LandmarkRecognitionRep (class)**

**LocalizationNS::LandmarkRecognition (class)**

Maamerkkien tunnistus -luokka. Luokka tunnistaa Segmentation-luokan palauttamia hahmoja hahmontunnistuskoneilla.

**LocalizationNS::LandmarksRep (class)**

**LocalizationNS::Landmarks (class)**

**LocalizationNS::LandmarksInterface (class)**

**Author**:

Jarkko Salminen

**LocalizationNS::LocalizationRep (class)**

**LocalizationNS::LocalizationException (class)**

**LocalizationNS::Localization (class)**

**LocalizationNS::SegmentationException (class)**

Segmentointipoikkeus. Tarkoittaa, ettei segmentointikirjastoa pystytty alustamaan.

**LocalizationNS::CaptureException (class)**

Kuvankaappauspoikkus, joko videokortin initialisointi, tai itse kuvan kaappaus epäonnistui.

**LocalizationNS::LandmarksFileNotFoundException (class)**

**LocalizationNS::PanTiltException (class)**

Kameran pantilt-yksikon initialisoinnin epäonnistumiseen liittyvä poikkeus.

**void** `LocalizationNS::panTilt` **(const Angle& horizontal, const Angle& vertical)** []

Kääntää kameran jalustaa haluttuun suuntaan.

**Parameters**:

| | |
|---|---|
| *horizontal* | Horisontaalinen kulma, 0 astetta on suoraan eteen. |
| *vertical* | Vertikaalinen kulma, 0 astetta vaakasuoraan. |

**Throws**: PanTiltException

**LocalizationNS::PanoramaRep (class)**

**LocalizationNS::Panorama (class)**

**LocalizationNS::SegmentationRep (class)**

**LocalizationNS::Segmentation (class)**

**bool** `LocalizationNS::initMachines` **(const std::vector<std::string> landmarks)** []

Initialisoi hahmontunnistuskoneet. Käytännössä vain tallettaa niiden nimet tunnistamista varten.

**Parameters**:

| | |
|---|---|
| *landmarks* | Maamerkkien/tunnistuskoneiden nimet vektorissa. |

**Return**:

True, jos onnistui, muuten false.

**std::string** `LocalizationNS::recognize` **(Bitmap shape)** []

Yrittää tunnistaa hahmon kaikilla initialisoiduilla koneilla.

**Parameters**:

| | |
|---|---|
| *shape* | Hahmo, joka yritetään tunnistaa. |

**Return**:

Tunnistavan koneen nimi, tai tyhjä string-arvo, jos ei tunnistettu.

**LocalizationNS::ViewRep (class)**

**LocalizationNS::View (class)**

Kuvien kaappaus panoraamana. Kuvia tarvitaan maamerkkien tunnistamiseen ja paikanmääritykseen.

## 24 class LocalizationNS::Bitmap

`#include <Localization/Bitmap.h>`
Inherits: SelfTestingObject

*public*

(unknown)

### Public Methods


- int **getWidth** () ....33
- int **getHeight** () ....33
- unsigned char* **getDataPtr** () ....33
- bool **save** (char* name) ....33
- int **getDepth** () ....34
- **Bitmap** () ....34
- **Bitmap** (int width, int height, int depth, unsigned char *data_ptr) ....34
- **~Bitmap** () ....34
- virtual bool **invariant** () const ....34


### Private Members


- BitmapRep* **rep** ....34


### Detailed Description

### Member Documentation

**int** `LocalizationNS::Bitmap::getWidth` **()** []

Palauttaa kuvan leveyden.

**int** `LocalizationNS::Bitmap::getHeight` **()** []

Palauttaa kuvan korkeuden.

**unsigned char*** `LocalizationNS::Bitmap::getDataPtr` **()** []

Palauttaa osoittimen kuvadataan.

**bool** `LocalizationNS::Bitmap::save` **(char* name)** []

Tallentaa kuvan levylle.
**Parameters**:

| | |
|---|---|
| *name* | kuvatiedoston nimi |

**int** `LocalizationNS::Bitmap::getDepth` **()** []

Palauttaa kuvan värisyvyyden bitteinä/pikseli.

`LocalizationNS::Bitmap::Bitmap` **()** []

Parametriton konstruktori.

`LocalizationNS::Bitmap::Bitmap` **(int width, int height, int depth, unsigned char *data_ptr)** []

Konstruktori.

`LocalizationNS::Bitmap::~Bitmap` **()** []

Destruktori.

**bool** `LocalizationNS::Bitmap::invariant` **()** [**const virtual**]

**BitmapRep*** `LocalizationNS::Bitmap::rep` [**private**]

## 25 class LocalizationNS::BitmapRep

`#include <Localization/BitmapRep.h>`
Inherits: SelfTestingObject
*public*
(unknown)

### Public Methods

- **BitmapRep** () ........ 35
- **~BitmapRep** () ........ 35
- virtual bool **invariant** () const ........ 35

### Public Members

- int **width** ........ 35
- int **height** ........ 35
- int **depth** ........ 35
- unsigned char * **data_ptr** ........ 35

### Detailed Description

### Member Documentation

**int** `LocalizationNS::BitmapRep::width`

**int** `LocalizationNS::BitmapRep::height`

**int** `LocalizationNS::BitmapRep::depth`

**unsigned char *** `LocalizationNS::BitmapRep::data_ptr`

`LocalizationNS::BitmapRep::BitmapRep` **()** []

`LocalizationNS::BitmapRep::~BitmapRep` **()** []

**bool** `LocalizationNS::BitmapRep::invariant` **()** [**const virtual**]

## 26 class LocalizationNS::CaptureException

Kuvankaappauspoikkus, joko videokortin initialisointi, tai itse kuvan kaappaus epäonnistui.

`#include <Localization/LocalizationExceptions.h>`

Inherits: std::runtime_error

*public*

(unknown)

### Public Methods


- **CaptureException** (const std::string& s) ........ 35
- virtual ~**CaptureException** () ........ 35


### Detailed Description

Kuvankaappauspoikkus, joko videokortin initialisointi, tai itse kuvan kaappaus epäonnistui.

### Member Documentation

`LocalizationNS::CaptureException::CaptureException` **(const std::string& s)** []

Konstruktori.

**Parameters**:

*s* virheilmoitus

`LocalizationNS::CaptureException::~CaptureException` **()** [ **virtual**]

## 27 class LocalizationNS::LandmarkRecognition

Maamerkkien tunnistus -luokka.

`#include <Localization/LandmarkRecognition.h>`

Inherits: SelfTestingObject

*public*

(unknown)

## Public Methods


- std::vector< std::pair<std::string, Angle> > **getRecognizedLandmarks** () const .................. 36
- virtual bool **invariant** () const .................................................................. 36
- **LandmarkRecognition** (std::vector<std::string> landmarks) ...................................... 36
- virtual ~**LandmarkRecognition** () ................................................................ 37


## Private Members


- LandmarkRecognitionRep* **rep** .................................................................... 37


## Detailed Description

Maamerkkien tunnistus -luokka. Luokka tunnistaa Segmentation-luokan palauttamia hahmoja hahmontunnistuskoneilla.

## Member Documentation

**std::vector**< **std::pair**<**std::string, Angle**> > `LocalizationNS::LandmarkRecognition::getRecognizedLandmarks` **()** [**const** ]

Yrittää tunnistaa robottia ympäröivät maamerkit. Tunnistamisen luotettavuus on kiinni Segmentation-luokan palauttamien hahmojen laadusta. Hahmojen laatu on taas riippuvainen kuvan laadusta, valaistuksesta, maamerkkien etäisyydestä ja asennosta suhteessa robottiin.

**Return**:

Vektori, joka sisältää pair-tyyppisiä alkioita. Jokainen pair sisältää maamerkin nimen ja havaintokulman.

**bool** `LocalizationNS::LandmarkRecognition::invariant` **()** [**const virtual**]

Invariantin tarkistus.

**Return**:

true, jos olio on eheä, false muuten

`LocalizationNS::LandmarkRecognition::LandmarkRecognition` **(std::vector**<**std::string**> **landmarks)**
[]

Konstruktori.

**Parameters**:

*Vektori* , joka sisältää maamerkkien/hahmontunnistuskoneiden nimet.

`LocalizationNS::LandmarkRecognition::~LandmarkRecognition` **()** [ **virtual**]

**LandmarkRecognitionRep*** `LocalizationNS::LandmarkRecognition::rep` [**private**]

## 28 class LocalizationNS::LandmarkRecognitionRep

`#include <Localization/LandmarkRecognitionRep.h>`
Inherits: NobelNS::SelfTestingObject
*public*
()

### Public Methods

- virtual bool **invariant** () const ....37
- **LandmarkRecognitionRep** () ....37
- virtual ~**LandmarkRecognitionRep** () ....37

### Public Members

- Segmentation **segm** ....37

### Detailed Description

### Member Documentation

**Segmentation** `LocalizationNS::LandmarkRecognitionRep::segm`

**bool** `LocalizationNS::LandmarkRecognitionRep::invariant` **()** [**const virtual**]

Invariantin tarkistus.
**Return**:

true, jos olio on eheä, false muuten.

`LocalizationNS::LandmarkRecognitionRep::LandmarkRecognitionRep` **()** []

`LocalizationNS::LandmarkRecognitionRep::~LandmarkRecognitionRep` **()** [ **virtual**]

## 29 class LocalizationNS::Landmarks

`#include <Localization/Landmarks.h>`
Inherits: LandmarksInterface
*public*

(), NobelNS::SelfTestingObject

*public*

()

### Public Methods


- virtual std::vector<NobelNS::Vector> **getLandmarkLocation** (const std::string& landmark) const . 38
- virtual std::vector<std::string> **getLandmarks** () const ............ 38
- **Landmarks** (std::string landmarksFile) ............ 38
- virtual ~**Landmarks** () ............ 38
- virtual bool **invariant** () const ............ 38


### Private Members


- LandmarksRep* **rep** ............ 38


### Detailed Description

### Member Documentation

**std::vector<NobelNS::Vector>** `LocalizationNS::Landmarks::getLandmarkLocation` **(const std::string& landmark)** [**const virtual**]

**std::vector<std::string>** `LocalizationNS::Landmarks::getLandmarks` **()** [**const virtual**]

`LocalizationNS::Landmarks::Landmarks` **(std::string landmarksFile)** []

`LocalizationNS::Landmarks::~Landmarks` **()** [ **virtual**]

**bool** `LocalizationNS::Landmarks::invariant` **()** [**const virtual**]

**LandmarksRep*** `LocalizationNS::Landmarks::rep` [**private**]

## 30 class LocalizationNS::LandmarksFileNotFoundException

`#include <Localization/LocalizationExceptions.h>`

Inherits: std::runtime_error

*public*

(unknown)

### Public Methods


- **LandmarksFileNotFoundException** (const std::string& s) ............ 39
- virtual ~**LandmarksFileNotFoundException** () ............ 39

### Detailed Description

### Member Documentation

`LocalizationNS::LandmarksFileNotFoundException::LandmarksFileNotFoundException` **(const std::string& s)** []

`LocalizationNS::LandmarksFileNotFoundException::~LandmarksFileNotFoundException` **()** [ **virtual**]

## 31 class LocalizationNS::LandmarksInterface

LandmarksInterface määrittelee rajapinnan säiliöoliolle, joka sisältää tiedot ympäristön maamerkeistä ja niiden sijainneista. Olio tuntee tukivektorikonetiedostot (SVM-tiedostot), joita käytetään merkkien tunnistukseen. Maamerkit nimetään niitä vastaavien SVM-tiedostonimien mukaan.

**Abstract class.**

`#include <Localization/LandmarksInterface.h>`

Landmarks

### Public Methods


- virtual std::vector<NobelNS::Vector> ***getLandmarkLocation*** (const std::string& landmark) const . 39
- virtual std::vector<std::string> ***getLandmarks*** () const . . . . . . . . . . 39
- virtual **~LandmarksInterface** () . . . . . . . . . . 40


### Detailed Description

**Author**:

Jarkko Salminen

### Member Documentation

**std::vector<NobelNS::Vector>** *LocalizationNS::LandmarksInterface::getLandmarkLocation* **(const std::string& landmark)** [**const pure virtual**]

Funktio palauttaa parametrinä annetun maamerkin sijainnin. Jos merkin sijainti ei ole yksikäsitteinen, palautetaan kaikki mahdolliset sijainnit.

**Parameters**:

*string* & landmark on maamerkin nimi.

**Return**:

std::vector<NobelNS::Vector> on taulukko maamerkin mahdollisia sijainteja. Jos parametri on tuntematon maamerkki, palautetaan vector, jonka koko on nolla.

**std::vector<std::string>** *LocalizationNS::LandmarksInterface::getLandmarks* **()** [**const pure virtual**]

Funktio palauttaa SVM-tiedostojen nimet. Maamerkkien tunnisteet ovat samat kuin tiedostonimet.

**Return**:

std::vector<std::string> on taulukko olion tuntemista maamerkeistä.

`LocalizationNS::LandmarksInterface::~LandmarksInterface` **()** [ **virtual**]

## 32 class LocalizationNS::LandmarksRep

`#include <Localization/LandmarksRep.h>`

### Public Types

- typedef std::map<const std::string, NobelNS::Vector>::const_iterator **const_iterator** ................40

### Public Methods

- **LandmarksRep** () ................40
- virtual **~LandmarksRep** () ................40

### Public Members

- std::map<const std::string, NobelNS::Vector> **marks** ................40

### Detailed Description

### Member Documentation

**typedef std::map<const std::string, NobelNS::Vector>::const_iterator** `LocalizationNS::LandmarksRep::const_iterator`

**std::map<const std::string, NobelNS::Vector>** `LocalizationNS::LandmarksRep::marks`

`LocalizationNS::LandmarksRep::LandmarksRep` **()** []

`LocalizationNS::LandmarksRep::~LandmarksRep` **()** [ **virtual**]

## 33 class LocalizationNS::Localization

`#include <Localization/Localization.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods

- NobelNS::Pose **getPose** () const .......... 41
- NobelNS::Pose **getPose** (NobelNS::Pose X) const .......... 41
- virtual bool **invariant** () const .......... 41
- **Localization** () .......... 41
- virtual **~Localization** () .......... 41

### Private Members

- LocalizationRep* **rep** .......... 41

### Detailed Description

### Member Documentation

**NobelNS::Pose** `LocalizationNS::Localization::getPose` **()** [**const** ]

**NobelNS::Pose** `LocalizationNS::Localization::getPose` **(NobelNS::Pose X)** [**const** ]

**bool** `LocalizationNS::Localization::invariant` **()** [**const virtual**]

`LocalizationNS::Localization::Localization` **()** []

`LocalizationNS::Localization::~Localization` **()** [ **virtual**]

**LocalizationRep*** `LocalizationNS::Localization::rep` [**private**]

## 34 class LocalizationNS::LocalizationException

`#include <Localization/Localization.h>`

Inherits: std::runtime_error

*public*

(unknown)

### Public Methods

- int **getLandmarkCount** () const .......... 42
- std::vector< std::string > **getSeenLandmarks** () const .......... 42
- **LocalizationException** (string s) .......... 42
- **LocalizationException** (const string ilmo, vector< pair<std::string, NobelNS::Angle> > X, vector< pair<std::string, NobelNS::Angle> >::iterator p_it1, vector< pair<std::string, NobelNS::Angle> >::iterator p_it2, vector< pair<std::string, NobelNS::Angle> >::iterator p_it3) .......... 42
- virtual **~LocalizationException** () .......... 42

### Private Members


- std::vector< std::pair<std::string, NobelNS::Angle> > **seenlms** ....................................42
- std::vector< std::pair<std::string, NobelNS::Angle> >::const_iterator **it1** ............................42
- std::vector< std::pair<std::string, NobelNS::Angle> >::const_iterator **it2** ............................42
- std::vector< std::pair<std::string, NobelNS::Angle> >::const_iterator **it3** ............................42


### Detailed Description

### Member Documentation

**int** `LocalizationNS::LocalizationException::getLandmarkCount` **()** [**const** ]

**std::vector**< **std::string** > `LocalizationNS::LocalizationException::getSeenLandmarks` **()** [**const** ]

`LocalizationNS::LocalizationException::LocalizationException` **(string s)** []

`LocalizationNS::LocalizationException::LocalizationException` **(const string ilmo, vector< pair<std::string, NobelNS::Angle> > X, vector< pair<std::string, NobelNS::Angle> >::iterator p_it1, vector< pair<std::string, NobelNS::Angle> >::iterator p_it2, vector< pair<std::string, NobelNS::Angle> >::iterator p_it3)** []

`LocalizationNS::LocalizationException::~LocalizationException` **()** [ **virtual**]

**std::vector**< **std::pair<std::string, NobelNS::Angle>** > `LocalizationNS::LocalizationException::seenlms`
[**private**]

**std::vector**< **std::pair<std::string, NobelNS::Angle>** >**::const_iterator** `LocalizationNS::LocalizationException::it1`
[**private**]

**std::vector**< **std::pair<std::string, NobelNS::Angle>** >**::const_iterator** `LocalizationNS::LocalizationException::it2`
[**private**]

**std::vector**< **std::pair<std::string, NobelNS::Angle>** >**::const_iterator** `LocalizationNS::LocalizationException::it3`
[**private**]

## 35 class LocalizationNS::LocalizationRep

`#include <Localization/LocalizationRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods


- NobelNS::Pose **getPose** () ..........................................................................43
- NobelNS::Pose **getPose** (NobelNS::Pose start) ......................................................43
- **LocalizationRep** () ..................................................................................43
- bool **invariant** () const ..............................................................................43
- **~LocalizationRep** () ..................................................................................43

## Private Methods


- void **stripDoubles** (std::vector< std::pair<std::string, Angle> >& seenlms) . . . . . . . . . . . . . . . . . . . . . . . 43
- NobelNS::Pose **fixLocation** (NobelNS::Pose p) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
- NobelNS::Pose **triangulate** (NobelNS::Angle a1, NobelNS::Angle a2, NobelNS::Angle a3, pair<double,double> locus1, pair<double,double> locus2, pair<double,double> locus3) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44


## Private Members


- Landmarks* **LM** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
- LandmarkRecognition* **LMrec** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43


## Detailed Description

## Member Documentation

**NobelNS::Pose** `LocalizationNS::LocalizationRep::getPose` **()** []

Pyrkii määrittämään robotin sijainnin ilman aikaisempaa tietoa sijainnista.

**Return**:

Pose-olion, jossa määrittety robotin sijainti ja suunta

**NobelNS::Pose** `LocalizationNS::LocalizationRep::getPose` **(NobelNS::Pose start)** []

Pyrkii määrittämään robotin nykyisen sijainnin iteratiivisesti aloittaen annetusta oletussijainnista EI TOTEUTETTU

**Parameters**:

*start* Pose-olio, jossa nykyinen oletussijainti

**Return**:

Pose-olion, jossa määritetty robotin sijainti ja suunta

`LocalizationNS::LocalizationRep::LocalizationRep` **()** []

**bool** `LocalizationNS::LocalizationRep::invariant` **()** [**const** ]

`LocalizationNS::LocalizationRep::~LocalizationRep` **()** []

**Landmarks*** `LocalizationNS::LocalizationRep::LM` [**private**]

**LandmarkRecognition*** `LocalizationNS::LocalizationRep::LMrec` [**private**]

**void** `LocalizationNS::LocalizationRep::stripDoubles` **(std::vector< std::pair<std::string, Angle> >& seenlms)** [**private** ]

osoitin LandmarkRecognition-olioon, joka luodaan konstruktorissa

**See also**: LandmarkRecognition

**NobelNS::Pose** `LocalizationNS::LocalizationRep::fixLocation` **(NobelNS::Pose p)** [**private** ]

Paikanmäärityksessä pitää ottaa huomioon, että kamera ei ole täysin keskellä robottia.

**Parameters**:

| | |
|---|---|
| *triangulaation* | laskema sijainti kameran mukaan |

**Return**:

sijainti robotin keskipisteen mukaan

**NobelNS::Pose** `LocalizationNS::LocalizationRep::triangulate` **(NobelNS::Angle a1, NobelNS::Angle a2, NobelNS::Angle a3, pair<double,double> locus1, pair<double,double> locus2, pair<double,double> locus3)** [**private** ]

getPosen käyttöön. Laskee geometrista triangulaatiota käyttäen robotin aseman kolmen maamerkin avulla

**Parameters**:

| | |
|---|---|
| *a1* | 1. maamerkin havaintokulma Angle-oliona |
| *a2* | 2. maamerkin havaintokulma Angle-oliona |
| *a3* | 3. maamerkin havaintokulma Angle-oliona |
| *locus1* | 1. maamerkin sijainti kartalla pair<double x,double y> muodossa |
| *locus2* | 2. maamerkin sijainti kartalla pair<double x,double y> muodossa |
| *locus2* | 3. maamerkin sijainti kartalla pair<double x,double y> muodossa |

**Return**:

laskettu robotin asema kartalla Pose-oliona

## 36 class LocalizationNS::PanTiltException

Kameran pantilt-yksikon initialisoinnin epäonnistumiseen liittyvä poikkeus.

`#include <Localization/LocalizationExceptions.h>`

Inherits: std::runtime_error

*public*

(unknown)

### Private Methods

- **PanTiltException** (const std::string& s) ....45
- virtual ~**PanTiltException** () ....45

### Detailed Description

Kameran pantilt-yksikon initialisoinnin epäonnistumiseen liittyvä poikkeus.

### Member Documentation

`LocalizationNS::PanTiltException::PanTiltException` **(const std::string& s)** [**private** ]

`LocalizationNS::PanTiltException::~PanTiltException` **()** [**private virtual**]

## 37 class LocalizationNS::Panorama

`#include <Localization/Panorama.h>`
Inherits: SelfTestingObject

*public*

(unknown)

### Public Methods


- virtual int **getImageCount** () .......45
- virtual YUV422Image **getImage** (int) .......45
- virtual Angle **getAngleOfView** () .......46
- virtual void **setAngleOfView** (Angle) .......46
- virtual Angle **getImageDirection** (int) .......46
- virtual void **insertImage** (YUV422Image, Angle) .......46
- virtual bool **invariant** () const .......46
- **~Panorama** () .......46
- **Panorama** () .......46


### Private Members


- PanoramaRep * **rep** .......46


### Detailed Description

### Member Documentation

**int** `LocalizationNS::Panorama::getImageCount` **()** [ **virtual**]

Palauttaa panoraaman sisältämien kuvien määrän.
**Return**:

kuvien määrä.

**YUV422Image** `LocalizationNS::Panorama::getImage` **(int)** [ **virtual**]

Palauttaa annettua indeksiä vastaavan kuvan.

**Parameters**:

| | |
|---|---|
| *index* | indeksi |

**Angle** `LocalizationNS::Panorama::getAngleOfView` **()** [ **virtual**]

Palauttaa kuvien optisen laajuuskulman.

**void** `LocalizationNS::Panorama::setAngleOfView` **(Angle)** [ **virtual**]

**Angle** `LocalizationNS::Panorama::getImageDirection` **(int)** [ **virtual**]

Palauttaa annettua indeksiä vastaavan kuvan keskipisteen kulman suhteessa robotin suuntaan.

**void** `LocalizationNS::Panorama::insertImage` **(YUV422Image, Angle)** [ **virtual**]

Lisää kuvan panoraamaan.

**Parameters**:

| | |
|---|---|
| *im* | lisättävä kuva |
| *Angle* | kuvan kulma |

**bool** `LocalizationNS::Panorama::invariant` **()** [**const virtual**]

`LocalizationNS::Panorama::~Panorama` **()** []

`LocalizationNS::Panorama::Panorama` **()** []

**PanoramaRep *** `LocalizationNS::Panorama::rep` [**private**]

## 38 class LocalizationNS::PanoramaRep

`#include <Localization/PanoramaRep.h>`

Inherits: SelfTestingObject

*public*

(unknown)

### Public Methods


- virtual bool **invariant** () const ........ 47
- **PanoramaRep** () ........ 47
- **~PanoramaRep** () ........ 47

### Public Members


- std::vector<pair<YUV422Image, Angle> > **im_vector** .......... 47
- Angle **angle_of_view** .......... 47


### Detailed Description

### Member Documentation

**std::vector<pair<YUV422Image, Angle> >** `LocalizationNS::PanoramaRep::im_vector`

**Angle** `LocalizationNS::PanoramaRep::angle_of_view`

**bool** `LocalizationNS::PanoramaRep::invariant` **()** [**const virtual**]

`LocalizationNS::PanoramaRep::PanoramaRep` **()** []

`LocalizationNS::PanoramaRep::~PanoramaRep` **()** []

## 39 class LocalizationNS::Segmentation

`#include <Localization/Segmentation.h>`
Inherits: NobelNS::SelfTestingObject
*public*
()

### Public Methods


- virtual bool **invariant** () const .......... 47
- std::vector< pair<Bitmap, NobelNS::Angle> > **getSegments** () .......... 48
- **Segmentation** () .......... 48
- **Segmentation** (const Segmentation& a) .......... 48
- virtual void **swap** (Segmentation& a) .......... 48
- Segmentation& **operator=** (const Segmentation& a) .......... 48
- virtual **~Segmentation** () .......... 48


### Private Members


- SegmentationRep* **rep** .......... 48


### Detailed Description

### Member Documentation

**bool** `LocalizationNS::Segmentation::invariant` **()** [**const virtual**]

Tarkista invariantin paikkaansapitävyys.
**Return**:

true, jos olio on eheässä tilassa, false muuten

**std::vector< pair<Bitmap, NobelNS::Angle> >** `LocalizationNS::Segmentation::getSegments` **()** []

Segmenttien haku. Ottaa panoraaman (kuvasarjan robotin ympäristöstä), segmentoi ne ja palauttaa saadut segmentit ja segmenttien suunnan.

**Return**:

taulukko, jossa pareja (segmentti, kulma)

**Throws**: CaptureException, PanTiltException

`LocalizationNS::Segmentation::Segmentation` **()** []

`LocalizationNS::Segmentation::Segmentation` **(const Segmentation& a)** []

Kopiokonstruktori.

**Parameters**:

*a* kopioitava segmentoija

**void** `LocalizationNS::Segmentation::swap` **(Segmentation& a)** [ **virtual**]

Vaihda kaksi oliota keskenään.

**Parameters**:

*a* vaihdettava olio

**Segmentation&** `LocalizationNS::Segmentation::operator=` **(const Segmentation& a)** []

Sijoitusoperaattori.

**Parameters**:

*a* sijoitettava olio.

`LocalizationNS::Segmentation::~Segmentation` **()** [ **virtual**]

**SegmentationRep*** `LocalizationNS::Segmentation::rep` [**private**]

## 40 class LocalizationNS::SegmentationException

Segmentointipoikkeus.

`#include <Localization/LocalizationExceptions.h>`

Inherits: std::runtime_error

*public*

(unknown)

### Public Methods

- **SegmentationException** (const std::string& s) .......... 49
- virtual **~SegmentationException** () .......... 49

### Detailed Description

Segmentointipoikkeus. Tarkoittaa, ettei segmentointikirjastoa pystytty alustamaan.

### Member Documentation

`LocalizationNS::SegmentationException::SegmentationException` **(const std::string& s)** []

Konstruktori.
**Parameters**:

*s* virheilmoitus

`LocalizationNS::SegmentationException::~SegmentationException` **()** [ **virtual**]

## 41 class LocalizationNS::SegmentationRep

`#include <Localization/SegmentationRep.h>`
Inherits: NobelNS::SelfTestingObject
*public*
()

### Public Methods

- virtual bool **invariant** () const .......... 49
- **SegmentationRep** (const View& newV) .......... 50
- virtual **~SegmentationRep** () .......... 50

### Public Members

- View **v** .......... 49

### Detailed Description

### Member Documentation

**View** `LocalizationNS::SegmentationRep::v`

**bool** `LocalizationNS::SegmentationRep::invariant` **()** [**const virtual**]

Invariantin tarkistus.
**Return**:

invariantin paikkaansapitävyys

`LocalizationNS::SegmentationRep::SegmentationRep` **(const View& newV)** []

`LocalizationNS::SegmentationRep::~SegmentationRep` **()** [ **virtual**]

## 42 class LocalizationNS::View

Kuvien kaappaus panoraamana.

`#include <Localization/View.h>`

Inherits: SelfTestingObject

*public*

(unknown)

### Public Methods


- Panorama* **getPanorama** () ........ 50
- virtual bool **invariant** () const ........ 50
- **View** (const View& v) ........ 51
- **View** () ........ 51
- virtual ~**View** () ........ 51


### Private Members


- ViewRep* **rep** ........ 51


### Detailed Description

Kuvien kaappaus panoraamana. Kuvia tarvitaan maamerkkien tunnistamiseen ja paikanmääritykseen.

### Member Documentation

**Panorama*** `LocalizationNS::View::getPanorama` **()** []

Ottaa n (IMAGECOUNT) kpl kuvia tasaisin välein robotin ympäristöstä ja sijoittaa ne Panorama-ilmentymään. Kuvaus on optimoitu siten, että panoraama kuvataan joka toinen kerta vasemmalta oikealle ja joka toinen oikealta vasemmalle.

**Return**:

Kuvat panoraamassa.

**Throws**: CaptureException, PanTiltException

**bool** `LocalizationNS::View::invariant` **()** [**const virtual**]

Tarkista olion eheys.

**Return**:

olion eheys

`LocalizationNS::View::View` **(const View& v)** []

`LocalizationNS::View::View` **()** []

`LocalizationNS::View::~View` **()** [ **virtual**]

**ViewRep*** `LocalizationNS::View::rep` [**private**]

## 43 class LocalizationNS::ViewRep

`#include <Localization/ViewRep.h>`
Inherits: SelfTestingObject

*public*

(unknown)

### Public Methods


- virtual bool **invariant** () const ....... 51
- **ViewRep** () ....... 51
- virtual **~ViewRep** () ....... 52


### Public Members


- int **direction** ....... 51
- int **panoramaCount** ....... 51


### Detailed Description

### Member Documentation

**int** `LocalizationNS::ViewRep::direction`

**int** `LocalizationNS::ViewRep::panoramaCount`

**bool** `LocalizationNS::ViewRep::invariant` **()** [**const virtual**]

Invariantin tarkistus.
**Return**:

true

`LocalizationNS::ViewRep::ViewRep` () []

`LocalizationNS::ViewRep::~ViewRep` () [ **virtual**]

# 44 namespace MapNS

Osoitin sisäiseen toteutukseen.

## Public Types


- class **BadScaleException** ........ 54
- class **MapFileNotFoundException** ........ 78
- class **GridRep** ........ 71
- class **Grid** ........ 54
- class **GridIteratorRep** ........ 70
- class **GridIterator** ........ 65
- class **GridCellRep** ........ 60
- class **GridCell** ........ 57
- class ***GridIteratorInterface*** ........ 67
- class ***GridInterface*** ........ 61
- class **MapRep** ........ 81
- class **BadDestinationException** ........ 53
- class **Map** ........ 76
- class ***MapInterface*** ........ 79


## Detailed Description

Osoitin sisäiseen toteutukseen.

## Member Documentation

**MapNS::BadScaleException (class)**

**MapNS::MapFileNotFoundException (class)**

**MapNS::GridRep (class)**

**MapNS::Grid (class)**

**MapNS::GridIteratorRep (class)**

**MapNS::GridIterator (class)**

**MapNS::GridCellRep (class)**

**Author**:

Jarkko Salminen

**MapNS::GridCell (class)**

**MapNS::GridIteratorInterface (class)**

**Author**:

Jarkko Salminen

**MapNS::GridInterface (class)**

**Author**:

Jarkko Salminen

**MapNS::MapRep (class)**

**Author**:

Jarkko Salminen

**MapNS::BadDestinationException (class)**

**MapNS::Map (class)**

**MapNS::MapInterface (class)**

**Author**:

Jarkko Salminen

## 45 class MapNS::BadDestinationException

`#include <Map/Map.h>`
Inherits: std::runtime_error

*public*

(unknown)

### Public Methods

- **BadDestinationException** (const std::string& s) ....54
- virtual **~BadDestinationException** () ....54

### Detailed Description

### Member Documentation

`MapNS::BadDestinationException::BadDestinationException` **(const std::string& s)** []

`MapNS::BadDestinationException::~BadDestinationException` **()** [ **virtual**]

## 46 class MapNS::BadScaleException

Gridin konstruktori heittää BadScaleExceptionin, jos mittakaava on liian pieni.

`#include <Map/Grid.h>`

Inherits: std::runtime_error

*public*

(unknown)

### Public Methods

- **BadScaleException** (const std::string& s) ........ 54
- virtual ~**BadScaleException** () ........ 54

### Detailed Description

### Member Documentation

`MapNS::BadScaleException::BadScaleException` **(const std::string& s)** []

`MapNS::BadScaleException::~BadScaleException` **()** [ **virtual**]

## 47 class MapNS::Grid

Grid-olio on kartta ympäröivästä maailmasta. Se on kaksiulotteinen ruudukko.

`#include <Map/Grid.h>`

Inherits: GridInterface

*public*

(), NobelNS::SelfTestingObject

*public*

()

### Public Methods

- virtual bool **isFree** (const NobelNS::Vector& location) const ........ 56
- virtual bool **isInMargin** (const NobelNS::Vector& location) const ........ 56
- virtual bool **isSameCell** (const NobelNS::Vector& location1, const NobelNS::Vector& location2) const 56
- virtual bool **isRouteGood** (const NobelNS::Vector& begin, const NobelNS::Vector& routeVector) const 56

- virtual bool **isRouteFree** (const NobelNS::Vector& begin, const NobelNS::Vector& routeVector) const 56
- virtual int **count** () const ........ 56
- virtual NobelNS::Vector **getRoute** (const NobelNS::Vector& location) const ........ 56
- virtual NobelNS::Vector **getRouteOutOfMargin** (const NobelNS::Vector& location, const NobelNS::Angle& direction) const ........ 56
- virtual NobelNS::Vector **getRouteOutOfMargin** (const NobelNS::Vector& location) const ........ 56
- virtual GridIteratorInterface* **getIterator** (const NobelNS::Vector& location) const ........ 56
- virtual GridIteratorInterface* **copyIterator** (GridIteratorInterface* iterator) const ........ 56
- **Grid** (NobelNS::Length scale, std::string mapFile, NobelNS::Length margin = NobelNS::Length(20, NobelNS::CM)) ........ 56
- virtual ~**Grid** () ........ 56
- virtual bool **invariant** () const ........ 56

### Private Members

- GridRep* **rep** ........ 57

## Detailed Description

## Member Documentation

**bool** `MapNS::Grid::isFree` **(const NobelNS::Vector& location)** [const virtual]

**bool** `MapNS::Grid::isInMargin` **(const NobelNS::Vector& location)** [const virtual]

**bool** `MapNS::Grid::isSameCell` **(const NobelNS::Vector& location1, const NobelNS::Vector& location2)** [const virtual]

**bool** `MapNS::Grid::isRouteGood` **(const NobelNS::Vector& begin, const NobelNS::Vector& routeVector)** [const virtual]

**bool** `MapNS::Grid::isRouteFree` **(const NobelNS::Vector& begin, const NobelNS::Vector& routeVector)** [const virtual]

**int** `MapNS::Grid::count` **()** [const virtual]

**NobelNS::Vector** `MapNS::Grid::getRoute` **(const NobelNS::Vector& location)** [const virtual]

**NobelNS::Vector** `MapNS::Grid::getRouteOutOfMargin` **(const NobelNS::Vector& location, const NobelNS::Angle& direction)** [const virtual]

**NobelNS::Vector** `MapNS::Grid::getRouteOutOfMargin` **(const NobelNS::Vector& location)** [const virtual]

**GridIteratorInterface*** `MapNS::Grid::getIterator` **(const NobelNS::Vector& location)** [const virtual]

**GridIteratorInterface*** `MapNS::Grid::copyIterator` **(GridIteratorInterface* iterator)** [const virtual]

`MapNS::Grid::Grid` **(NobelNS::Length scale, std::string mapFile, NobelNS::Length margin = NobelNS::Length(20, NobelNS::CM))** []

Konstruktori

Konstruktori heittää BadScaleExceptionin, jos mittakaava on liian pieni. MapFileNotFoundException heitetään, jos mapFile-tiedoston avaus epäonnistuu.

**Parameters**:

| | |
|---|---|
| *scale* | on mittakaava, eli ruudun sivun todellinen pituus. mapFile on karttatiedoston nimi. margin on marginaalin leveys. Oletusarvo on 20 cm. |

`MapNS::Grid::~Grid` **()** [ virtual]

Destruktori

**bool** `MapNS::Grid::invariant` **()** [**const virtual**]

**GridRep*** `MapNS::Grid::rep` [**private**]

## 48 class MapNS::GridCell

GridCell-oliot ovat kartan alkioita. Niihin tallennetaan tiedot esteistä ja maalipisteeseen johtavista reiteistä. GridCell voi olla joko kartan este, vapaa ruutu tai marginaaliin kuuluva ruutu.

`#include <Map/GridCell.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods


- virtual int **getName** () const ........ 58
- virtual int **getShortestRoute** () const ........ 58
- virtual double **getRouteLength** () const ........ 58
- virtual bool **isFree** () const ........ 58
- virtual bool **isInMargin** () const ........ 58
- virtual bool **flag** () const ........ 58
- virtual bool **operator<** (const GridCell&) const ........ 58
- virtual void **setName** (int) ........ 58
- virtual void **setShortestRoute** (int) ........ 59
- virtual void **setRouteLength** (double) ........ 59
- virtual void **setBlock** () ........ 59
- virtual void **setMargin** () ........ 59
- virtual void **setFlag** () ........ 59
- **GridCell** () ........ 59
- **GridCell** (const GridCell&) ........ 59
- virtual void **swap** (GridCell&) ........ 59
- GridCell& **operator=** (const GridCell&) ........ 60
- virtual ~**GridCell** () ........ 60
- virtual bool **invariant** () const ........ 60


### Private Members


- GridCellRep* **rep** ........ 60

### Detailed Description

### Member Documentation

**int** `MapNS::GridCell::getName` **()** [**const virtual**]

Funktio palauttaa olion nimen, joka on asetettu setName(int)-funktiolla.

**int** `MapNS::GridCell::getShortestRoute` **()** [**const virtual**]

Funktio palauttaa sen GridCell-olion nimen, jonne lyhin reitti kulkee.

**double** `MapNS::GridCell::getRouteLength` **()** [**const virtual**]

Palauttaa maalipisteeseen johtavan reitin pituuden.

**bool** `MapNS::GridCell::isFree` **()** [**const virtual**]

Funktio tutkii, onko olio merkitty vapaaksi kartan soluksi.

**Return**:

Palauttaa true, jos GridCell on vapaa tai se kuuluu marginaaliin. Jos GridCell on merkitty esteeksi (setBlock), palautetaan false.

**bool** `MapNS::GridCell::isInMargin` **()** [**const virtual**]

Funktio tutkii, kuuluuko olio kartan marginaaliin.

**Return**:

Palauttaa true, jos solu kuuluu marginaaliin. Muuten false.

**bool** `MapNS::GridCell::flag` **()** [**const virtual**]

flag() palauttaa true, jos solulle on jo löydetty lyhin reitti. Muuten false.

**See also**: MapNS::GridCell#setFlag()

**bool** `MapNS::GridCell::operator<` **(const GridCell&)** [**const virtual**]

Operaattori < vertailee GridCell-olioihin merkittyjen reittien pituuksia.

**Parameters**:

*GridCell* -olio, jonka reitin pituuteen verrataan.

**Return**:

Palauttaa true, jos parametrina annetun GridCell-olioon merkityn reitin pituus on pienempi. Muuten false.

**void** `MapNS::GridCell::setName` **(int)** [ **virtual**]

setName() merkitsee solun nimeksi integerin.

**Parameters**:

| | |
|---|---|
| *solun* | nimi |

**void** `MapNS::GridCell::setShortestRoute` **(int)** [ **virtual**]

Funktio merkitsee, minne lyhin reitti kulkee. Reitti asetetaan vain GridCell-oliolle, joka ei kuulu marginaaliin eikä ole este. Jos lyhin reitti merkitään kulkevaksi soluun itseensä, kyseisen reitin pituudeksi merkitään nolla.

**Parameters**:

| | |
|---|---|
| *solun* | nimi, jonne lyhin reitti johtaa. |

**void** `MapNS::GridCell::setRouteLength` **(double)** [ **virtual**]

Asettaa reitin pituuden. Pituus asetetaan vain vapaalle GridCell-oliolle, joka ei kuulu marginaaliin eikä ole este. Reitin pituus ei voi koskaan olla negatiivinen. Jos parametri on nolla, lyhin reitti merkitään kulkevaksi soluun itseensä.

**Parameters**:

| | |
|---|---|
| *double* | reitin pituus. Jos parametri on negatiivinen, asetetaan reitin pituudeksi nolla. |

**void** `MapNS::GridCell::setBlock` **()** [ **virtual**]

setBlock() merkitsee solun esteeksi.

**void** `MapNS::GridCell::setMargin` **()** [ **virtual**]

Funktio setMargin() merkitsee vapaan solun marginaalisoluksi.

**void** `MapNS::GridCell::setFlag` **()** [ **virtual**]

Funktio asettaa lyhimmän reitin löydetyksi.

`MapNS::GridCell::GridCell` **()** []

Konstruktori luo GridCell-olion, jolle pätee: reitin pituus on ääretön, lyhintä reittiä ei ole löydetty ja solu on vapaa.

`MapNS::GridCell::GridCell` **(const GridCell&)** []

Kopiokonstruktori

**void** `MapNS::GridCell::swap` **(GridCell&)** [ **virtual**]

swap vaihtaa olioiden arvot keskenään

**GridCell&** `MapNS::GridCell::operator=` **(const GridCell&)** []

Sijoitus

`MapNS::GridCell::~GridCell` **()** [ **virtual**]

Destruktori

**bool** `MapNS::GridCell::invariant` **()** [**const virtual**]

invariant tarkistaa olion tilan oikeellisuuden.

**GridCellRep*** `MapNS::GridCell::rep` [**private**]

# 49 class MapNS::GridCellRep

GridCell-luokan toteutus.

`#include <Map/GridCellRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

## Public Methods


- **GridCellRep** () ..... 61
- **GridCellRep** (const GridCellRep&) ..... 61
- GridCellRep& **operator=** (const GridCellRep&) ..... 61
- virtual ~**GridCellRep** () ..... 61
- virtual bool **invariant** () const ..... 61


## Public Members


- int **name** ..... 61
- int **route** ..... 61
- double **routeLength** ..... 61
- int **state** ..... 61
- bool **flag** ..... 61


## Detailed Description

**Author**:

### Member Documentation

**int** `MapNS::GridCellRep::name`

**int** `MapNS::GridCellRep::route`

**double** `MapNS::GridCellRep::routeLength`

**int** `MapNS::GridCellRep::state`

**bool** `MapNS::GridCellRep::flag`

`MapNS::GridCellRep::GridCellRep` **()** []

GridCell voi olla kolmessa tilassa: 0 = vapaa solu, 1 = marginaalisolu, 2 = estesolu

`MapNS::GridCellRep::GridCellRep` **(const GridCellRep&)** []

**GridCellRep&** `MapNS::GridCellRep::operator=` **(const GridCellRep&)** []

`MapNS::GridCellRep::~GridCellRep` **()** [ **virtual**]

**bool** `MapNS::GridCellRep::invariant` **()** [**const virtual**]

## 50 class MapNS::GridInterface

GridInterface määrittelee reitinlaskentaosajärjestelmän kartan tietorakenteen rajapinnan: Kartta on kaksiulotteinen ruudukko. Ruudukon alkioita nimitetään tässä soluiksi tai ruuduiksi.

**Abstract class.**

`#include <Map/GridInterface.h>`

Grid

### Public Methods

- virtual bool ***isFree*** (const NobelNS::Vector& location) const ........ 62
- virtual bool ***isInMargin*** (const NobelNS::Vector& location) const ........ 62
- virtual bool ***isSameCell*** (const NobelNS::Vector& location1, const NobelNS::Vector& location2) const 62
- virtual bool ***isRouteGood*** (const NobelNS::Vector& begin, const NobelNS::Vector& routeVector) const 63
- virtual bool ***isRouteFree*** (const NobelNS::Vector& begin, const NobelNS::Vector& routeVector) const 63
- virtual int ***count*** () const ........ 63
- virtual NobelNS::Vector ***getRoute*** (const NobelNS::Vector& location) const ........ 63
- virtual NobelNS::Vector ***getRouteOutOfMargin*** (const NobelNS::Vector& location, const NobelNS::Angle& direction) const ........ 64

- virtual NobelNS::Vector ***getRouteOutOfMargin*** (const NobelNS::Vector& location) const ......... 64
- virtual GridIteratorInterface* ***getIterator*** (const NobelNS::Vector& location) const ................ 64
- virtual GridIteratorInterface* ***copyIterator*** (GridIteratorInterface* iterator) const ................. 65
- virtual ~**GridInterface** () ........................................................................ 65


## Detailed Description

**Author**:

Jarkko Salminen

## Member Documentation

**bool *MapNS::GridInterface::isFree* (const NobelNS::Vector& location)** [**const pure virtual**]

Kertoo, onko parametrina annettu sijainti esteen ulkopuolella.

**Parameters**:

*location* on sijainti kartalla.

**Return**:

Palauttaa true, jos location ei ole este. Muuten false. Jos location on kartan ulkopuolella palautetaan false.

**bool *MapNS::GridInterface::isInMargin* (const NobelNS::Vector& location)** [**const pure virtual**]

Kertoo, onko parametrina annettu sijainti marginaalissa.

**Parameters**:

*location* on sijainti kartalla.

**Return**:

Palauttaa true, jos location on marginaalissa. Muuten false. Jos location on kartan ulkopuolella palautetaan false.

**bool *MapNS::GridInterface::isSameCell* (const NobelNS::Vector& location1, const NobelNS::Vector& location2)** [**const pure virtual**]

Funktio tutkii, sijaitsevatko location1 ja location2 samassa kartan ruudussa, eli ovatko parametrit sama sijainti kartan resoluution tarkkuudella.

**Parameters**:

*location1* ja location2 ovat kartan pisteitä.

**Return**:

Funktio palauttaa true, jos parametrit ovat samassa kartan ruudussa. Muuten false. Jos toinen tai molemmat parametreista ovat kartan ulkopuolisia pisteitä, palautetaan false.

**bool *MapNS::GridInterface::isRouteGood* (const NobelNS::Vector& begin, const NobelNS::Vector& routeVector)** [**const pure virtual**]

Funktio tutkii, voiko kartan pisteestä begin kulkea reitin routeVector. Reitti on sallittu, jos: 1. begin on kartalla 2. begin ei ole este 3. reitin päätepiste on kartalla 4. päätepiste ei ole este 5. päätepiste ei ole marginaalissa 6. reitin pisteet eivät sijaitse esteiden sisällä 7. reitin pisteet eivät ole marginaalissa. Piste begin ja sitä seuraavat reitin pisteet saavat kuitenkin olla marginaalissa, kunhan reitti päättyy marginaalin ulkopuolelle ja reitti ei kulje uudelleen marginaalialueen kautta.

**Parameters**:

*begin* on reitin lähtöpiste, ja routeVector lähtöpisteestä alkava reitti.

**Return**:

Palauttaa true, jos reitti on sallittu, muuten false.

**bool *MapNS::GridInterface::isRouteFree* (const NobelNS::Vector& begin, const NobelNS::Vector& routeVector)** [**const pure virtual**]

Funktio tutkii, voiko kartan pisteestä begin kulkea reitin routeVector. Reitti on sallittu, jos reitin pisteet ovat kartalla ja vapaita. Pisteet saavat sijaita marginaalissa.

**Parameters**:

*begin* on reitin lähtöpiste kartalla, ja routeVector pisteestä begin alkava reitti.

**Return**:

Palauttaa true, jos reitti on sallittu, muuten false.

**int *MapNS::GridInterface::count* ()** [**const pure virtual**]

Funktio laskee kartan solut, jotka ovat vapaita ja marginaalin ulkopuolella.

**Return**:

palauttaa solujen lukumäärän, joille pätee: isFree() ja !isInMargin().

**NobelNS::Vector *MapNS::GridInterface::getRoute* (const NobelNS::Vector& location)** [**const pure virtual**]

Funktio palauttaa suoran reitin, joka alkaa kartan pisteestä location ja seuraa karttaan tallennettua reittiä, kunnes kartan reitin suunta muuttuu tai reitti päättyy.

**Parameters**:

*location* on kartan sijainti, josta reitti alkaa.

**Return**:

Palauttaa karttaan tallennetun mahdollisimman pitkän suoran reitinpätkän, joka alkaa sijainnista location. Jos location on kartan ulkopuolella, esteen sisällä, marginaalissa tai reitin päätepiste, palautetaan Vector, jonka pituus on nolla. Palautettava Vector on reitin päätepisteen ja parametrin location välinen erotus.

**NobelNS::Vector** *MapNS::GridInterface::getRouteOutOfMargin* **(const NobelNS::Vector& location, const NobelNS::Angle& direction)** [**const pure virtual**]

Funktio laskee vektorin, joka johtaa ulos marginaalista.

**Parameters**:

*location* on kartan marginaalin piste, josta etsitään reitti marginaalin ulkopuolelle suuntaan direction.

**Return**:

Palauttaa Vector-olion, joka ulottuu juuri marginaalin ulkopuolelle suunnassa direction. Vector alkaa sijainnista location. Jos location on kartan tai marginaalin ulkopuolella, palautetaan Vector, jonka pituus on nolla. Jos sijainnista location ei ole suuntaan direction reittiä marginaalin ulkopuolelle, palautetaan myös nollavektori.

**NobelNS::Vector** *MapNS::GridInterface::getRouteOutOfMargin* **(const NobelNS::Vector& location)** [**const pure virtual**]

Funktio laskee lyhimmän vektorin marginaalin ulkopuolelle pisteestä location.

**Parameters**:

*location* on kartan marginaalin piste, josta etsitään reitti marginaalin ulkopuolelle.

**Return**:

Palauttaa lyhimmän Vector-olion, joka ulottuu juuri marginaalin ulkopuolelle. Vector alkaa sijainnista location. Jos location on kartan tai marginaalin ulkopuolella, palautetaan Vector, jonka pituus on nolla. Jos sijainnista location ei ole reittiä marginaalin ulkopuolelle, palautetaan myös nollavektori.

**GridIteratorInterface*** *MapNS::GridInterface::getIterator* **(const NobelNS::Vector& location)** [**const pure virtual**]

Funktio luo iteraattorin, joka osoittaa kartan pisteeseen location. Käyttäjän on huolehdittava iteraattorin deletoinnista.

**Parameters**:

*location* on kartan piste, jota vastaavaan kartan soluun palautettava iteraattori osoittaa. Parametrin pitää olla kartan sisällä, vapaa ja marginaalin ulkopuolella.

**Return**:

Luo abstraktin luokan GridIteratorInterface toteuttavan luokan olion, joka osoittaa kartan pisteeseen location. Funktio palauttaa osoittimen kyseiseen olioon. Jos parametri ei ole kelvollinen, palautetaan nolla.

**GridIteratorInterface*** *MapNS::GridInterface::copyIterator* **(GridIteratorInterface* iterator)** [**const pure virtual**]

Funktio palauttaa kopion parametrina annetusta iteraattorista. Käyttäjän on huolehdittava palautettavan iteraattorin deletoinnista.

**Parameters**:

*Parametri* iterator on iteraattori, josta tehdään kopio.

**Return**:

Palauttaa osoittimen luotuun iteraattoriin, joka on luokan GridIteratorInterface aliluokka.

`MapNS::GridInterface::~GridInterface` **()** [ **virtual**]

## 51 class MapNS::GridIterator

`#include <Map/Grid.h>`

Inherits: GridIteratorInterface

*public*

(), NobelNS::SelfTestingObject

*public*

()

### Public Methods


- virtual bool **isFlagSet** () const .......... 66
- virtual double **getDistanceTo** (const GridIteratorInterface* iterator) const .......... 66
- virtual double **getRouteLength** () const .......... 66

- virtual bool **operator<** (const GridIteratorInterface& iterator) const ........ 66
- virtual bool **operator!=** (const GridIteratorInterface& iterator) const ........ 66
- virtual GridIteratorInterface& **operator++** () ........ 66
- virtual void **setRouteLength** (double length) ........ 66
- virtual void **setShortestRoute** (const GridIteratorInterface* iterator) ........ 66
- virtual void **setFlag** () ........ 66
- **GridIterator** (const GridIterator&) ........ 66
- virtual void **swap** (GridIterator&) ........ 66
- GridIterator& **operator=** (const GridIterator&) ........ 67
- virtual ~**GridIterator** () ........ 67
- virtual bool **invariant** () const ........ 67

### Private Methods

- **GridIterator** (GridRep* gr, int i, int j) ........ 67

### Private Members

- GridIteratorRep* **rep** ........ 67
- friend class **Grid** ........ 67

### Detailed Description

### Member Documentation

**bool** `MapNS::GridIterator::isFlagSet` **()** [**const virtual**]

**double** `MapNS::GridIterator::getDistanceTo` **(const GridIteratorInterface* iterator)** [**const virtual**]

**double** `MapNS::GridIterator::getRouteLength` **()** [**const virtual**]

**bool** `MapNS::GridIterator::operator<` **(const GridIteratorInterface& iterator)** [**const virtual**]

**bool** `MapNS::GridIterator::operator!=` **(const GridIteratorInterface& iterator)** [**const virtual**]

**GridIteratorInterface&** `MapNS::GridIterator::operator++` **()** [ **virtual**]

**void** `MapNS::GridIterator::setRouteLength` **(double length)** [ **virtual**]

**void** `MapNS::GridIterator::setShortestRoute` **(const GridIteratorInterface* iterator)** [ **virtual**]

**void** `MapNS::GridIterator::setFlag` **()** [ **virtual**]

`MapNS::GridIterator::GridIterator` **(const GridIterator&)** []

Kopiokonstruktori

**void** `MapNS::GridIterator::swap` **(GridIterator&)** [ **virtual**]

Swap

**GridIterator&** `MapNS::GridIterator::operator=` **(const GridIterator&)** []

Sijoitus

`MapNS::GridIterator::~GridIterator` **()** [ **virtual**]

Destruktori

**bool** `MapNS::GridIterator::invariant` **()** [**const virtual**]

`MapNS::GridIterator::GridIterator` **(GridRep* gr, int i, int j)** [**private** ]

Konstruktori

**GridIteratorRep*** `MapNS::GridIterator::rep` [**private**]

**friend class** `MapNS::GridIterator::Grid` [**private**]

## 52 class MapNS::GridIteratorInterface

GridIteratorInterface määrittelee rajapinnan iteraattoreille, joilla voi osoittaa kartan soluihin. Iteraattoreilla voi selata kartan jokaisen solun vierussolut ja merkitä karttaan reittejä. Iteraattori voi osoittaa vain soluun, joka on vapaa ja marginaalin ulkopuolella.

**Abstract class.**

`#include <Map/GridInterface.h>`

GridIterator

### Public Methods


- virtual bool ***isFlagSet*** () const ........ 69
- virtual double ***getDistanceTo*** (const GridIteratorInterface* iterator) const ........ 69
- virtual double ***getRouteLength*** () const ........ 69
- virtual bool ***operator\ensuremath\lbrace<\rbrace*** (const GridIteratorInterface& iterator) const ..69
- virtual bool ***operator!=*** (const GridIteratorInterface& iterator) const ........ 69
- virtual GridIteratorInterface& ***operator++*** () ........ 69
- virtual void ***setRouteLength*** (double length) ........ 69
- virtual void ***setShortestRoute*** (const GridIteratorInterface* iterator) ........ 70
- virtual void ***setFlag*** () ........ 70
- virtual ~**GridIteratorInterface** () ........ 70

**Detailed Description**

**Author**:

Jarkko Salminen

### Member Documentation

**bool** *MapNS::GridIteratorInterface::isFlagSet* **()** [**const pure virtual**]

isFlagSet() kertoo, onko iteraattorin osoittamalle kartan solulle tehty kutsu setFlag().

**double** *MapNS::GridIteratorInterface::getDistanceTo* **(const GridIteratorInterface* iterator)** [**const pure virtual**]

Funktio laskee iteraattoreiden välisen etäisyyden.

**Parameters**:

*et* äisyys lasketaan parametrin iterator osoittamaan kartan soluun

**Return**:

Palauttaa iteraattoreiden välisen etäisyyden. Palautusarvon yksikkö on kartan ruutu.

**double** *MapNS::GridIteratorInterface::getRouteLength* **()** [**const pure virtual**]

Funktio palauttaa kartan ruutuun tallennetun reitin pituuden.

**bool** *MapNS::GridIteratorInterface::operator\ensuremath\lbrace<\rbrace* **(const GridIteratorInterface& iterator)** [**const pure virtual**]

Vertailuoperaattori < palauttaa true, jos parametrina annetun iteraattorin osoittamaan kartan soluun tallennetun reitin pituus on pienempi.

**Parameters**:

*iterator* on osoitin kartan soluun, johon tallennettua reittiä halutaan vertailla

**bool** *MapNS::GridIteratorInterface::operator!=* **(const GridIteratorInterface& iterator)** [**const pure virtual**]

Funktio palauttaa true, jos iteraattorit osoittavat kartan eri soluihin. Muuten false.

**GridIteratorInterface&** *MapNS::GridIteratorInterface::operator++* **()** [**pure virtual**]

Operaattori ++ siirtää iteraattorin osoittamaan seuraavaa vierussolua. Vierussoluja ovat ympäröivät vapaat marginaalin ulkopuoliset solut. Operaattorilla kierretään sitä solua, johon iteraattori alunperin konstruoitaessa osoitti. Jos kaikki vierussolut on käyty läpi, iteraattori siirtyy juuri tähän keskimmäiseen soluun.

**void** *MapNS::GridIteratorInterface::setRouteLength* **(double length)** [**pure virtual**]

Funktio asettaa iteraattorin osoittamalle ruudulle reitin pituudeksi parametrin length.

**void** *MapNS::GridIteratorInterface::setShortestRoute* **(const GridIteratorInterface* iterator)** [**pure virtual**]

Funktio merkitsee iteraattorin osoittamalle ruudulle, että lyhin reitti siihen kulkee parametrina annetun iteraattorin osoittaman ruudun kautta.

**void** *MapNS::GridIteratorInterface::setFlag* **()** [**pure virtual**]

Funktio merkitsee, että iteraattorin osoittamalle ruudulle on löydetty lyhin reitti

`MapNS::GridIteratorInterface::~GridIteratorInterface` **()** [ **virtual**]

## 53 class MapNS::GridIteratorRep

GridIteratorin toteutus.

`#include <Map/GridRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods


- **GridIteratorRep** (GridRep* gr, int i, int j) ........ 71
- **GridIteratorRep** (const GridIteratorRep&) ........ 71
- GridIteratorRep& **operator=** (const GridIteratorRep&) ........ 71
- virtual **~GridIteratorRep** () ........ 71
- virtual bool **invariant** () const ........ 71


### Public Members


- GridRep* **grep** ........ 71
- int **i** ........ 71
- int **j** ........ 71
- int **nexti** ........ 71
- int **nextj** ........ 71

### Detailed Description

### Member Documentation

**GridRep*** `MapNS::GridIteratorRep::grep`

**int** `MapNS::GridIteratorRep::i`

**int** `MapNS::GridIteratorRep::j`

**int** `MapNS::GridIteratorRep::nexti`

**int** `MapNS::GridIteratorRep::nextj`

`MapNS::GridIteratorRep::GridIteratorRep` **(GridRep* gr, int i, int j)** []

Konstruktori

**Parameters**:

| | |
|---|---|
| *gr* | on osoitin cells-taulukkoon ja pari (i, j) on taulukon alkio, johon iteraattori osoittaa |

`MapNS::GridIteratorRep::GridIteratorRep` **(const GridIteratorRep&)** []

Kopiokonstruktori

**GridIteratorRep&** `MapNS::GridIteratorRep::operator=` **(const GridIteratorRep&)** []

Sijoitus

`MapNS::GridIteratorRep::~GridIteratorRep` **()** [ **virtual**]

Destruktori

**bool** `MapNS::GridIteratorRep::invariant` **()** [**const virtual**]

## 54 class MapNS::GridRep

Grid-luokan toteutus.

`#include <Map/GridRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods

- virtual bool **isInGrid** (const NobelNS::Vector& location) const ...................................... 72
- virtual bool **isInGrid** (int i, int j) const ...................................... 73
- virtual bool **isInXAxis** (const NobelNS::Length& x) const ...................................... 73

- virtual bool **isInYAxis** (const NobelNS::Length& y) const ............ 73
- virtual bool **isOnEdge** (int i, int j) const ............ 73
- virtual int **getRowIndex** (const NobelNS::Vector& location) const ............ 74
- virtual int **getColumnIndex** (const NobelNS::Vector& location) const ............ 74
- virtual int **getRowIndex** (int cellName) const ............ 74
- virtual int **getColumnIndex** (int cellName) const ............ 74
- virtual NobelNS::Vector **parseVector** (int i, int j) const ............ 75
- virtual void **addExtraBlocks** (int i, int j) ............ 75
- virtual void **addMargin** (int i, int j, int radius) ............ 75
- **GridRep** (NobelNS::Length scale, std::ifstream& grid, NobelNS::Length margin) ............ 75
- virtual **~GridRep** () ............ 75
- virtual bool **invariant** () const ............ 75


## Public Members


- NobelNS::Length **scale** ............ 72
- GridCell** **cells** ............ 72
- int **rows** ............ 72
- int **columns** ............ 72
- NobelNS::Length **margin** ............ 72
- static const NobelNS::Length **robotSize** ............ 75
- static const NobelNS::Length **minScale** ............ 76
- static const NobelNS::Length **minMargin** ............ 76


## Detailed Description

## Member Documentation

**NobelNS::Length** `MapNS::GridRep::scale`

**GridCell**** `MapNS::GridRep::cells`

**int** `MapNS::GridRep::rows`

**int** `MapNS::GridRep::columns`

**NobelNS::Length** `MapNS::GridRep::margin`

**bool** `MapNS::GridRep::isInGrid` **(const NobelNS::Vector& location)** [**const virtual**]

isInGrid tutkii, onko location kartan rajojen sisäpuolella.
**Parameters**:

*location* on koordinaattipiste

**Return**:

Palauttaa true, jos location on kartalla. Muuten false

**bool** `MapNS::GridRep::isInGrid` **(int i, int j)** [**const virtual**]

isInGrid tarkistaa cells-taulukon indeksoinnin

**Parameters**:

*i* on rivin ja j on sarakkeen indeksi

**Return**:

Palauttaa true, jos indeksointi on oikein. Muuten false

**bool** `MapNS::GridRep::isInXAxis` **(const NobelNS::Length& x)** [**const virtual**]

isInXAxis tutkii, onko x kartan x-akselin piste

**Parameters**:

*x* on x-akselin koordinaatti

**Return**:

Palauttaa true, jos x on x-akselin piste. Muuten false.

**bool** `MapNS::GridRep::isInYAxis` **(const NobelNS::Length& y)** [**const virtual**]

isInYAxis tutkii, onko y kartan y-akselin piste

**Parameters**:

*y* on y-akselin koordinaatti

**Return**:

Palauttaa true, jos y on y-akselin piste. Muuten false.

**bool** `MapNS::GridRep::isOnEdge` **(int i, int j)** [**const virtual**]

Funktio etsii esteiden reunoja

**Parameters**:

*i* on rivin ja j on sarakkeen indeksi cells-taulukossa

**Return**:

Palauttaa true, jos (i, j) on este ja jokin sen vierusalkioista on vapaa. Muuten false.

**int** `MapNS::GridRep::getRowIndex` **(const NobelNS::Vector& location)** [**const virtual**]

Palauttaa kartan pistettä location vastaavan cells-taulukon rivi-indeksin

**Parameters**:

*location* on kartan piste

**Return**:

Palauttaa parametria location vastaavan rivi-indeksin. Jos parametri ei ole kartalla, palautetaan -1.

**int** `MapNS::GridRep::getColumnIndex` **(const NobelNS::Vector& location)** [**const virtual**]

Palauttaa kartan pistettä location vastaavan cells-taulukon sarakeindeksin

**Parameters**:

*location* on kartan piste

**Return**:

Palauttaa parametria location vastaavan sarake-indeksin. Jos parametri ei ole kartalla, palautetaan -1.

**int** `MapNS::GridRep::getRowIndex` **(int cellName)** [**const virtual**]

Palauttaa solua vastaavan cells-taulukon rivi-indeksin

**Parameters**:

*int* cellName on GridCell-olion nimi

**Return**:

Palauttaa parametria cellName vastaavan rivi-indeksin. Jos parametri ei ole kartalla, palautetaan -1.

**int** `MapNS::GridRep::getColumnIndex` **(int cellName)** [**const virtual**]

Palauttaa solua vastaavan cells-taulukon sarakeindeksin

**Parameters**:

*int* cellName on GridCell-olion nimi

**Return**:

Palauttaa parametria cellName vastaavan sarakeindeksin. Jos parametri ei ole kartalla, palautetaan -1.

**NobelNS::Vector** `MapNS::GridRep::parseVector` **(int i, int j)** [**const virtual**]

Palauttaa cells-taulukon alkiota (i, j) vastaavan paikkavektorin

**Parameters**:

| | |
|---|---|
| *pari* | (i, j) on cells-taulukon alkio |

**Return**:

Palauttaa cells-taulukon alkiota (i, j) vastaavan paikkavektorin. Jos indeksointi on virheellinen, palautetaan nollavektori.

**void** `MapNS::GridRep::addExtraBlocks` **(int i, int j)** [ **virtual**]

Merkitsee estesoluja cells-taulukkoon alkion (i, j) ympärille. Estesoluja asetetaan robotin säteen peittämälle alalle. Jos (i, j) ei ole kartalla tai on vapaa solu, esteitä ei lisätä.

**Parameters**:

| | |
|---|---|
| *Pari* | (i, j) on cells-taulukon alkio |

**void** `MapNS::GridRep::addMargin` **(int i, int j, int radius)** [ **virtual**]

Merkitsee marginaalisoluja cells-taulukkoon alkion (i, j) ympärille. Marginaali asetetaan säteen int radius peittämälle alalle. Alueella olevia estesoluja ei muuteta marginaalisoluiksi. Jos (i, j) ei ole kartalla tai on vapaa solu, marginaalia ei lisätä.

**Parameters**:

| | |
|---|---|
| *Pari* | (i, j) on cells-taulukon alkio. radius on lisättävän marginaalin säde soluina. |

`MapNS::GridRep::GridRep` **(NobelNS::Length scale, std::ifstream& grid, NobelNS::Length margin)** []

Konstruktori

`MapNS::GridRep::~GridRep` **()** [ **virtual**]

Destruktori

**bool** `MapNS::GridRep::invariant` **()** [**const virtual**]

Testaa olion tilan oikeellisuude

**static const NobelNS::Length** `MapNS::GridRep::robotSize`

**static const NobelNS::Length** `MapNS::GridRep::minScale`

**static const NobelNS::Length** `MapNS::GridRep::minMargin`

## 55 class MapNS::Map

`#include <Map/Map.h>`
Inherits: MapInterface
*public*
(), NobelNS::SelfTestingObject
*public*
()

### Public Methods


- virtual bool **isInMargin** (const NobelNS::Vector& location) const .................................. 76
- virtual bool **isDestination** (const NobelNS::Vector& location) const ................................ 76
- virtual bool **isGoodLocation** (const NobelNS::Vector& location) const ............................ 77
- virtual bool **routeExists** (const NobelNS::Vector& location) const .................................. 77
- virtual NobelNS::Vector **getRouteVector** (const NobelNS::Vector& location) ....................... 77
- virtual NobelNS::Vector **getReverseVector** (const NobelNS::Vector& location) ..................... 78
- **Map** (GridInterface* grid, const NobelNS::Vector& destination) ..................................... 78
- virtual **~Map** () ..................................................................................... 78
- virtual bool **invariant** () const ...................................................................... 78


### Protected Members


- MapRep* **rep** ......................................................................................... 78


### Detailed Description

### Member Documentation

**bool** `MapNS::Map::isInMargin` **(const NobelNS::Vector& location)** [**const virtual**]

Kertoo, onko parametrina annettu sijainti marginaalissa.
**Parameters**:

*Vector* & on robotin sijainti kartalla

**Return**:

Palauttaa true, jos robotti on marginaalissa. Muuten false.

**bool** `MapNS::Map::isDestination` **(const NobelNS::Vector& location)** [**const virtual**]

Kertoo, onko parametrina annettu sijainti maalipiste.

**Parameters**:

*Vector* & on robotin sijainti

**Return**:

Palauttaa true, jos maalipiste sijaitsee samassa kartan solussa kuin parametri. Muuten false.

**bool** `MapNS::Map::isGoodLocation` **(const NobelNS::Vector& location)** [**const virtual**]

Funktio tarkistaa, onko parametri järkevä sijainti robotille.

**Parameters**:

*Vector* & on robotin sijainti kartalla

**Return**:

Palauttaa false, jos parametri sijaitsee kartan ulkopuolella tai esteen sisällä. Muuten true. Jos parametri on marginaalipiste, palautetaan myös true.

**bool** `MapNS::Map::routeExists` **(const NobelNS::Vector& location)** [**const virtual**]

Funktio tarkistaa, voiko maalipistettä saavuttaa parametrina annetusta sijainnista.

**Parameters**:

*Vector* & on robotin sijainti kartalla

**Return**:

Palauttaa true, jos parametrina annetusta sijainnista on olemassa reitti maalipisteeseen. Muuten false.

**NobelNS::Vector** `MapNS::Map::getRouteVector` **(const NobelNS::Vector& location)** [ **virtual**]

Funktio palauttaa optimoidun reittivektorin. Tämä vektori on mahdollisimman pitkä suoranpätkä lyhimmästä reitistä maalipisteeseen. Jos funktion parametri on marginaalipiste, palautetaan myös optimoitu reittivektori. Katso getReverseVector().

**Parameters**:

*Vector* & on robotin sijainti kartalla.

**Return**:

Palauttaa vektorin, joka esittää mahdollisimman pitkää suoraa, jonka robotti voi kulkea. Jos parametri on maalipiste tai sijaintina mahdoton (katso isGoodLocation() määritelmä), palautetaan Vector-olio, jonka pituus on nolla. Jos parametrin määrittelemästä sijainnista ei ole olemassa reittiä maalipisteeseen, palautetaan Vector, jonka pituus on nolla.

**See also**: MapNS::Map#getReverseVector(), MapNS::Map#isGoodLocation(), MapNS::Map#routeExists()

**NobelNS::Vector** `MapNS::Map::getReverseVector` **(const NobelNS::Vector& location)** [ **virtual**]

Funktio palauttaa vektorin, joka johtaa ulos marginaalista. Se ulottuu juuri marginaalin ulkopuolelle.

Vektori ei ole optimaalinen maalipisteen saavuttamisen kannalta. Se ei noudata lyhintä reittiä maalipisteeseen.

Funktion käyttö on järkevää, jos robotti on niin lähellä kartalle merkittyä estettä, että sonarit havaitsevat kyseisen esteen.

**Parameters**:

*Vector* & on robotin sijainti kartalla.

**Return**:

Vector on reitti ulos marginaalista. Sen pituus on nolla, jos parametri on marginaalin ulkopuolella. Jos parametri on maalipiste tai sijaintina mahdoton (katso isGoodLocation()), palautetaan myös nollavektori. Samoin jos parametrina annetusta sijainnista ei voi saavuttaa maalipistettä, palautetaan nollavektori.

`MapNS::Map::Map` **(GridInterface* grid, const NobelNS::Vector& destination)** []

`MapNS::Map::~Map` **()** [ **virtual**]

**bool** `MapNS::Map::invariant` **()** [**const virtual**]

**MapRep*** `MapNS::Map::rep` [**protected**]

## 56 class MapNS::MapFileNotFoundException

Gridin konstruktori heittää MapFileNotFoundException, jos mapFile-tiedoston avaus epäonnistuu.

`#include <Map/Grid.h>`

Inherits: std::runtime_error

*public*

(unknown)

### Public Methods


- **MapFileNotFoundException** (const std::string& s) ..... 79
- virtual ~**MapFileNotFoundException** () ..... 79

### Detailed Description

### Member Documentation

`MapNS::MapFileNotFoundException::MapFileNotFoundException` **(const std::string& s)** []

`MapNS::MapFileNotFoundException::~MapFileNotFoundException` **()** [ **virtual**]

## 57 class MapNS::MapInterface

MapInterface määrittelee reitinlaskentaosajärjestelmän tarjoamat palvelut.

**Abstract class.**

`#include <Map/MapInterface.h>`

Map

### Public Methods


- virtual bool ***isInMargin*** (const NobelNS::Vector&) const .......... 79
- virtual bool ***isDestination*** (const NobelNS::Vector&) const .......... 79
- virtual bool ***isGoodLocation*** (const NobelNS::Vector&) const .......... 80
- virtual bool ***routeExists*** (const NobelNS::Vector&) const .......... 80
- virtual NobelNS::Vector ***getRouteVector*** (const NobelNS::Vector&) .......... 80
- virtual NobelNS::Vector ***getReverseVector*** (const NobelNS::Vector&) .......... 81
- virtual **~MapInterface** () .......... 81


### Detailed Description

**Author**:

Jarkko Salminen

### Member Documentation

**bool** *`MapNS::MapInterface::isInMargin`* **(const NobelNS::Vector&)** [**const pure virtual**]

Kertoo, onko parametrina annettu robotin sijainti marginaalissa.

**Parameters**:

*Vector* & on robotin keskipisteen sijainti kartalla

**Return**:

Palauttaa true, jos robotti on marginaalissa. Muuten false. Robotti on marginaalissa, vaikka vain robotin reuna olisi alueen sisällä.

**bool** *MapNS::MapInterface::isDestination* **(const NobelNS::Vector&)** [**const pure virtual**]

Kertoo, onko parametrina annettu sijainti maalipiste.

**Parameters**:

*Vector* & on robotin sijainti

**Return**:

Palauttaa true, jos maalipiste sijaitsee samassa kartan solussa kuin parametri. Muuten false.

**bool** *MapNS::MapInterface::isGoodLocation* **(const NobelNS::Vector&)** [**const pure virtual**]

Funktio tarkistaa, onko parametri järkevä sijainti robotille.

**Parameters**:

*Vector* & on robotin sijainti kartalla

**Return**:

Palauttaa false, jos parametri sijaitsee kartan ulkopuolella tai esteen sisällä. Muuten true. Jos parametri on marginaalipiste, palautetaan myös true.

**bool** *MapNS::MapInterface::routeExists* **(const NobelNS::Vector&)** [**const pure virtual**]

Funktio tarkistaa, voiko maalipistettä saavuttaa parametrina annetusta sijainnista.

**Parameters**:

*Vector* & on robotin sijainti kartalla

**Return**:

Palauttaa true, jos parametrina annetusta sijainnista on olemassa reitti maalipisteeseen. Muuten false.

**NobelNS::Vector** *MapNS::MapInterface::getRouteVector* **(const NobelNS::Vector&)** [**pure virtual**]

Funktio palauttaa optimoidun reittivektorin. Tämä vektori on mahdollisimman pitkä suoranpätkä lyhimmästä reitistä maalipisteeseen. Jos funktion parametri on marginaalipiste, palautetaan myös optimoitu reittivektori. Katso getReverseVector().

**Parameters**:

*Vector* & on robotin sijainti kartalla.

**Return**:

Palauttaa vektorin, joka esittää mahdollisimman pitkää suoraa, jonka robotti voi kulkea. Jos parametri on maalipiste tai sijaintina mahdoton (katso isGoodLocation() määritelmä), palautetaan Vector-olio, jonka pituus on nolla. Jos parametrin määrittelemästä sijainnista ei ole olemassa reittiä maalipisteeseen, palautetaan Vector, jonka pituus on nolla.

**See also**: MapNS::MapInterface#getReverseVector(), MapNS::MapInterface#isGoodLocation(), MapNS::MapInterface#routeExists()

**NobelNS::Vector** *MapNS::MapInterface::getReverseVector* **(const NobelNS::Vector&)** [**pure virtual**]

Funktio palauttaa vektorin, joka johtaa ulos marginaalista. Se ulottuu juuri marginaalin ulkopuolelle.

Vektori ei ole optimaalinen maalipisteen saavuttamisen kannalta. Se ei noudata lyhintä reittiä maalipisteeseen.

Funktion käyttö on järkevää, jos robotti on niin lähellä kartalle merkittyä estettä, että sonarit havaitsevat kyseisen esteen.

**Parameters**:

*Vector* & on robotin sijainti kartalla.

**Return**:

Vector on reitti ulos marginaalista. Sen pituus on nolla, jos parametri on marginaalin ulkopuolella. Jos parametri on maalipiste tai sijaintina mahdoton (katso isGoodLocation()), palautetaan myös nollavektori. Jos parametrina annetusta sijainnista ei voi saavuttaa maalipistettä, palautetaan silti reitti ulos marginaalista.

`MapNS::MapInterface::~MapInterface` **()** [ **virtual**]

## 58 class MapNS::MapRep

Luokan Map toteutus.

`#include <Map/MapRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods

- **MapRep** (GridInterface* grid, const NobelNS::Vector& destination) ........ 82
- virtual **~MapRep** () ........ 82
- virtual bool **invariant** () const ........ 82

### Public Members

- GridInterface* **grid** ........ 82
- NobelNS::Vector **destination** ........ 82

### Detailed Description

**Author**:

Jarkko Salminen

### Member Documentation

**GridInterface*** `MapNS::MapRep::grid`

**NobelNS::Vector** `MapNS::MapRep::destination`

`MapNS::MapRep::MapRep` **(GridInterface* grid, const NobelNS::Vector& destination)** []

Konstruktori

`MapNS::MapRep::~MapRep` **()** [ **virtual**]

Destruktori

**bool** `MapNS::MapRep::invariant` **()** [**const virtual**]

## 59 namespace MovementNS

Liikkumisosajärjestelmä.

### Public Types


- class ***Detector*** .......... 83
- class ***Movement*** .......... 84
- class **RobotMoverRep** .......... 87
- class **RobotMover** .......... 85
- class ***Roller*** .......... 88
- class **ScoutRollerRep** .......... 91
- class **ScoutRoller** .......... 89
- class **SonarRep** .......... 95
- class **Sonar** .......... 92


### Detailed Description

Liikkumisosajärjestelmä. Sisältää robotin liikkumisen sekä sonarien käyttöön liittyviä luokkia.

### Member Documentation

**MovementNS::Detector** **(class)**

**MovementNS::Movement** **(class)**

**MovementNS::RobotMoverRep** **(class)**

**MovementNS::RobotMover** **(class)**

**MovementNS::Roller** **(class)**

**MovementNS::ScoutRollerRep** **(class)**

**MovementNS::ScoutRoller** **(class)**

Robotin liikuttaja. Sisältää palvelut, joilla liikuttaa Nomad Super Scout robottia.

**MovementNS::SonarRep** **(class)**

**MovementNS::Sonar** **(class)**

## 60 class MovementNS::Detector

**Abstract class.**
`#include <Movement/Detector.h>`
Sonar

### Public Methods


- virtual NobelNS::Vector ***getBlockPos*** () const ........ 83
- virtual NobelNS::Angle ***getFreeDirection*** () const ........ 83
- virtual bool ***detect*** () ........ 84
- virtual Detector& ***reverse*** () ........ 84
- virtual **~Detector** () ........ 84


### Detailed Description

### Member Documentation

**NobelNS::Vector** *MovementNS::Detector::getBlockPos* **()** [**const pure virtual**]

Palauta viimeksi havaitun esteen suhteellinen sijainti.
**Return**:

viimeksi havaitun esteen suhteellinen sijainti

**NobelNS::Angle** *MovementNS::Detector::getFreeDirection* **()** [**const pure virtual**]

Palauta suunta, jossa on (eniten) tilaa liikkua.

**Return**:

kulma, johon suuntaan laite voi liikkua

**bool** *MovementNS::Detector::detect* **()** [**pure virtual**]

Havaitse esteet.

**Return**:

true, jos este havaittu, false muuten

**Detector&** *MovementNS::Detector::reverse* **()** [**pure virtual**]

Vaihda havaintopolitiikkaa vastaamaan päinvastaiseen suuntaan liikkumista.

**Return**:

viite käännettyyn havaitsijaolioon

`MovementNS::Detector::~Detector` **()** [ **virtual**]

## 61 class MovementNS::Movement

**Abstract class.**

`#include <Movement/Movement.h>`

RobotMover

### Public Methods


- virtual NobelNS::Pose ***move*** (const NobelNS::Vector& route) const ... 84
- virtual NobelNS::Pose ***randomWalk*** (const NobelNS::Length& dist) const ... 85
- virtual ~**Movement** () ... 85


### Detailed Description

### Member Documentation

**NobelNS::Pose** *MovementNS::Movement::move* **(const NobelNS::Vector& route)** [**const pure virtual**]

Siirrä laitetta suhteellisen vektorin verran törmäämättä esteisiin. Jos laite ei voi kulkea käskettyä matkaa, heittää metodi poikkeuksen.

**Parameters**:

*route* suhteellinen vektori, joka laitteen on kuljettava.

**Return**:

pose, jossa laite on liikkumisen jälkeen (suhteellinen)

**Throws**: NobelNS::BlockException

**NobelNS::Pose** *MovementNS::Movement::randomWalk* **(const NobelNS::Length& dist)** [**const pure virtual**]

Kulje ennalta määräämättömään suuntaan tietty matka.

**Parameters**:

| | |
|---|---|
| *dist* | kuljettava matka |

**Return**:

suhteellinen pose

**Throws**: NobelNS::BlockException

`MovementNS::Movement::~Movement` **()** [ **virtual**]

## 62 class MovementNS::RobotMover

`#include <Movement/RobotMover.h>`

Inherits: Movement

*public*

(), NobelNS::SelfTestingObject

*public*

()

### Public Methods


- virtual NobelNS::Pose **move** (const NobelNS::Vector& target) const ........ 86
- virtual NobelNS::Pose **randomWalk** (const NobelNS::Length& dist) const ........ 86
- virtual bool **invariant** () const ........ 86
- **RobotMover** (Detector& d, const Roller& r) ........ 86
- **RobotMover** (const RobotMover& r) ........ 86
- virtual void **swap** (RobotMover& r) ........ 87
- RobotMover& **operator=** (const RobotMover& r) ........ 87
- virtual **~RobotMover** () ........ 87


### Private Members


- RobotMoverRep* **rep** ........ 87

### Detailed Description

### Member Documentation

**NobelNS::Pose** `MovementNS::RobotMover::move` **(const NobelNS::Vector& target)** [**const virtual**]

Liikkumismetodi. Liikuta Marttia määrätty matka. Jos este tulee eteen, heitä poikkeus. Kääntyminen on optimoitu: jos on nopeampaa kääntyä vasemmalle ja liikkua takaperin kuin kääntyä oikealle ja liikkua eteenpäin, valitaan vasemmalle kääntyminen ja takaperin liikkuminen. Tapauksella on symmetrinen vastakohta.

**Parameters**:

| | |
|---|---|
| *target* | suhteellinen vektori, joka liikutaan |

**Return**:

pose (suhteellinen), johon liikuttiin

**Throws**: NobelNS::BlockException

**NobelNS::Pose** `MovementNS::RobotMover::randomWalk` **(const NobelNS::Length& dist)** [**const virtual**]

Kulje tietty matka suuntaan, jossa on eniten tilaa.

**Parameters**:

| | |
|---|---|
| *dist* | kuljettava matka |

**Return**:

suhteellinen pose, johon liikuttiin

**Throws**: NobelNS::BlockException

**bool** `MovementNS::RobotMover::invariant` **()** [**const virtual**]

Invariantin tarkistus.

**Return**:

true, jos olio on eheä, false muuten

`MovementNS::RobotMover::RobotMover` **(Detector& d, const Roller& r)** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *d* | havaitsija, jota käytetään esteiden havaitsemiseen |
| *rv* | robotin nopeus |
| *ra* | robotin kiihtyvyys |

`MovementNS::RobotMover::RobotMover` **(const RobotMover& r)** []

Kopiokonstruktori.

**Parameters**:

*r* kopioitava olio

**void** `MovementNS::RobotMover::swap` **(RobotMover& r)** [ **virtual**]

Vaihto-operaatio.

**Parameters**:

*r* olio, jonka kanssa vaihdetaan

**RobotMover&** `MovementNS::RobotMover::operator=` **(const RobotMover& r)** []

Sijoitusoperaattori.

**Parameters**:

*r* sijoitettava olio

**Return**:

viite muutettuun olioon

`MovementNS::RobotMover::~RobotMover` **()** [ **virtual**]

**RobotMoverRep*** `MovementNS::RobotMover::rep` [**private**]

# 63 class MovementNS::RobotMoverRep

`#include <Movement/RobotMoverRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

## Public Methods

- virtual bool **invariant** () const ........ 88
- **RobotMoverRep** (Detector& nd, const Roller& nr, bool b) ........ 88
- virtual **~RobotMoverRep** () ........ 88

## Public Members

- Detector& **d** ........ 88
- const Roller& **r** ........ 88

- bool **inReverse** .......... 88


### Detailed Description

### Member Documentation

**Detector&** `MovementNS::RobotMoverRep::d`

**const Roller&** `MovementNS::RobotMoverRep::r`

**bool** `MovementNS::RobotMoverRep::inReverse`

**bool** `MovementNS::RobotMoverRep::invariant` **()** [**const virtual**]

Invariantin tarkistus.

**Return**:

true

`MovementNS::RobotMoverRep::RobotMoverRep` **(Detector& nd, const Roller& nr, bool b)** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *nd* | havaitsinolio |
| *nr* | liikuttajaolio |
| *b* | pakin asetus |

`MovementNS::RobotMoverRep::~RobotMoverRep` **()** [ **virtual**]

## 64 class MovementNS::Roller

**Abstract class.**

`#include <Movement/Roller.h>`

ScoutRoller

### Public Methods


- virtual void ***zeroCoords*** () const .......... 89
- virtual void ***forward*** (const NobelNS::Length& dist, bool blocking) const .......... 89
- virtual void ***rotate*** (const NobelNS::Angle& angle, bool blocking) const .......... 89
- virtual bool ***isRolling*** () const .......... 89
- virtual NobelNS::Pose ***getRelativePose*** () const .......... 89
- virtual **~Roller** () .......... 89

### Detailed Description

### Member Documentation

**void** *MovementNS::Roller::zeroCoords* **()** [**const pure virtual**]

**void** *MovementNS::Roller::forward* **(const NobelNS::Length& dist, bool blocking)** [**const pure virtual**]

Liiku eteenpäin annettu matka.

**Parameters**:

| | |
|---|---|
| *dist* | kuljettava matka, jos negatiivinen, kuljetaan taaksepäin. |
| *blocking* | jos true, kutsu ei palaa ennen kuin laite on pysähtynyt, muuten palataan samantien |

**void** *MovementNS::Roller::rotate* **(const NobelNS::Angle& angle, bool blocking)** [**const pure virtual**]

Käännä robottia.

**Parameters**:

| | |
|---|---|
| *angle* | käännyttävä kulma |
| *blocking* | jos true, kutsu ei palaa ennen kuin laite on pysähtynyt, muuten palataan samantien |

**bool** *MovementNS::Roller::isRolling* **()** [**const pure virtual**]

Hae robotin liiketila.

**Return**:

true, jos laite liikkuu vielä, false muuten

**NobelNS::Pose** *MovementNS::Roller::getRelativePose* **()** [**const pure virtual**]

`MovementNS::Roller::~Roller` **()** [ **virtual**]

## 65 class MovementNS::ScoutRoller

Robotin liikuttaja.

`#include <Movement/ScoutRoller.h>`

Inherits: NobelNS::SelfTestingObject

*public*

(), Roller

*public*

()

### Public Methods


- virtual bool **invariant** () const ....................90
- virtual void **zeroCoords** () const ....................90
- virtual void **forward** (const NobelNS::Length& dist, bool blocking) const ....................90
- virtual void **rotate** (const NobelNS::Angle& angle, bool blocking) const ....................90
- virtual bool **isRolling** () const ....................91
- virtual NobelNS::Pose **getRelativePose** () const ....................91
- **ScoutRoller** (const NobelNS::RobotVelocity& vel, const NobelNS::RobotAcceleration& acc) ........91
- virtual **~ScoutRoller** () ....................91


### Private Members


- ScoutRollerRep* **rep** ....................91


### Detailed Description

Robotin liikuttaja. Sisältää palvelut, joilla liikuttaa Nomad Super Scout robottia.

### Member Documentation

**bool** `MovementNS::ScoutRoller::invariant` **()** [**const virtual**]

Invariantin tarkistus.

**Return**:

true, jos olio on eheässä tilassa, false muuten

**void** `MovementNS::ScoutRoller::zeroCoords` **()** [**const virtual**]

Koordinaattien nollaus. Nollaa robotin sisäiset koordinaatit pisteeseen (0,0) ja suunnaksi 0 astetta.

**void** `MovementNS::ScoutRoller::forward` **(const NobelNS::Length& dist, bool blocking)** [**const virtual**]

Eteenpäin liikkuminen. Liikuttaa robottia eteenpäin (taaksepäin, jos dist on negatiivinen).

**Parameters**:

| | |
|---|---|
| *dist* | kuljettava matka (jos negatiivinen, robotti liikkuu taaksepäin |
| *blocking* | jos true, metodi palaa vasta kun robotti on liikkunut annetun matkan |

**void** `MovementNS::ScoutRoller::rotate` **(const NobelNS::Angle& angle, bool blocking)** [**const virtual**]

Käännä robottia paikallaan.

**Parameters**:

| | |
|---|---|
| *angle* | kulma, jonka robotti kääntyy (jos negatiivinen, robotti kääntyy oikealle) |
| *blocking* | jos true, metodi palaa vasta kun robotti on kääntynyt annetun kulman. |

**bool** `MovementNS::ScoutRoller::isRolling` **()** [**const virtual**]

Tutki, liikkuuko robotti.

**Return**:

true, jos robotti liikkuu, false muuten

**NobelNS::Pose** `MovementNS::ScoutRoller::getRelativePose` **()** [**const virtual**]

Palauta suhteellinen pose. Suhteellinen pose on pose, johon robotti on siirtynyt viimeisen zeroCoords() -metodin suorituksen jälkeen. Jos zeroCoords()ia ei ole suoritettu kertaakaan, lopputulos on määrittelemätön (eli näennäisesti satunnainen).

**Return**:

suhteellinen pose

`MovementNS::ScoutRoller::ScoutRoller` **(const NobelNS::RobotVelocity& vel, const NobelNS::RobotAcceleration& acc)** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *vel* | robotin maksiminopeus |
| *acc* | robotin kiihtyvyys |

`MovementNS::ScoutRoller::~ScoutRoller` **()** [ **virtual**]

**ScoutRollerRep*** `MovementNS::ScoutRoller::rep` [**private**]

## 66 class MovementNS::ScoutRollerRep

`#include <Movement/ScoutRollerRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

## Public Methods

- virtual bool **invariant** () const ....92
- virtual ~**ScoutRollerRep** () ....92

## Detailed Description

## Member Documentation

**bool** `MovementNS::ScoutRollerRep::invariant` **()** [**const virtual**]

`MovementNS::ScoutRollerRep::~ScoutRollerRep` **()** [ **virtual**]

# 67 class MovementNS::Sonar

`#include <Movement/Sonar.h>`

Inherits: Detector

*public*

(), NobelNS::SelfTestingObject

*public*

()

## Public Methods

- virtual bool **invariant** () const ....93
- virtual NobelNS::Vector **getBlockPos** () const ....94
- virtual NobelNS::Angle **getFreeDirection** () const ....94
- virtual Sonar& **reverse** () ....94
- virtual bool **detect** () ....94
- **Sonar** (const std::vector<NobelNS::Length>& distances) ....94
- **Sonar** (const Sonar& s) ....94
- virtual void **swap** (Sonar& s) ....95
- Sonar& **operator=** (const Sonar& s) ....95
- virtual ~**Sonar** () ....95

## Private Members

- SonarRep* **rep** ....95

### Detailed Description

### Member Documentation

**bool** `MovementNS::Sonar::invariant` () [**const virtual**]

Tarkista invariantti.

**Return**:

invariantin eheys

**NobelNS::Vector** `MovementNS::Sonar::getBlockPos` **()** [**const virtual**]

Hae viimeksi havaitun esteen suhteellinen sijainti.

**Return**:

viimeksi havaitun esteen suhteellinen sijainti

**NobelNS::Angle** `MovementNS::Sonar::getFreeDirection` **()** [**const virtual**]

Palauta suunta, jossa on sonareiden mukaan eniten tilaa.

**Return**:

suunta, jossa on sonareiden mukaan eniten tilaa

**Sonar&** `MovementNS::Sonar::reverse` **()** [ **virtual**]

Käännä sonareiden pysäytysetäisyydet päinvastaisiksi (takaperin/etuperin liikkumista varten).

**Return**:

viite muutettuun olioon

**bool** `MovementNS::Sonar::detect` **()** [ **virtual**]

Havaitse hälytysrajan sisällä olevat esteet.

**Return**:

true, jos löytyi este liian läheltä, false muuten

`MovementNS::Sonar::Sonar` **(const std::vector<NobelNS::Length>& distances)** []

Konstruktori.

**Parameters**:

*distances* pysäytysetäisyyksien taulukko. vektorissa tulee olla tasan SONARS määrä alkioita. Alkiot ovat pysäytysetäisyyksiä (1 per sonari, sonarit numeroitu edestäpäin vastapäivään). Martin tapauksessa SONARS on 16 (määritelty Nclients.h tiedostossa).

`MovementNS::Sonar::Sonar` **(const Sonar& s)** []

Kopiokonstruktori.

**Parameters**:

*s* kopioitava olio

**void** `MovementNS::Sonar::swap` **(Sonar& s)** [ **virtual**]

Vaihto-operaatio.

**Parameters**:

*s* vaihdettava olio

**Sonar&** `MovementNS::Sonar::operator=` **(const Sonar& s)** []

Sijoitusoperaattori.

**Parameters**:

*s* sijoitettava olio

`MovementNS::Sonar::~Sonar` **()** [ **virtual**]

**SonarRep*** `MovementNS::Sonar::rep` [**private**]

## 68 class MovementNS::SonarRep

`#include <Movement/SonarRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods

- virtual bool **invariant** () const ........ 96
- **SonarRep** (const std::vector<int>& d, const NobelNS::Vector& v) ........ 96
- virtual ~**SonarRep** () ........ 96

### Public Members

- std::vector<int> **distsInch** ........ 96
- NobelNS::Vector **blockLocation** ........ 96

### Detailed Description

### Member Documentation

**std::vector<int>** `MovementNS::SonarRep::distsInch`

**NobelNS::Vector** `MovementNS::SonarRep::blockLocation`

**bool** `MovementNS::SonarRep::invariant` **()** [**const virtual**]

Invariantin tarkistus.

**Return**:

true jos distsInch-taulukko on oikean kokoinen ja blockLocationin invariantti on voimassa, false muuten

`MovementNS::SonarRep::SonarRep` **(const std::vector<int>& d, const NobelNS::Vector& v)** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *d* | etäisyydet |
| *v* | este |

`MovementNS::SonarRep::~SonarRep` **()** [ **virtual**]

## 69 namespace NobelNS

Pikkuolioita.

### Public Types


- class **AngleRep** .......... 106
- enum **AngleUnit** { Deg, Deg10th, Rad, PTU } .......... 97
- class **Angle** .......... 103
- class **BlockExceptionRep** .......... 109
- class **BlockException** .......... 107
- class **LengthRep** .......... 114
- enum **LengthUnit** { M, CM, In, In10th } .......... 100
- class **Length** .......... 110
- class **PoseRep** .......... 118
- class **Pose** .......... 114
- class **RobotAccelerationRep** .......... 121
- enum **AccelerationUnit** { Ms2, CMs2, IN10THs2 } .......... 102

- class **RobotAcceleration** ........ 119
- class **RobotVelocityRep** ........ 125
- enum **VelocityUnit** { Ms, CMs, IN10THs } ........ 102
- class **RobotVelocity** ........ 122
- class ***SelfTestingObject*** ........ 126
- class **InvariantException** ........ 110
- class **VectorRep** ........ 130
- class **Vector** ........ 126


## Public Methods


- Angle **operator+** (const Angle& a, const Angle& b) ........ 98
- Angle **operator-** (const Angle& a, const Angle& b) ........ 98
- Angle **operator*** (const Angle& a, double b) ........ 98
- Angle **operator/** (const Angle& a, double b) ........ 98
- bool **operator**< (const Angle& a, const Angle& b) ........ 99
- bool **operator==** (const Angle& a, const Angle& b) ........ 99
- inline double **normalize360** (double a) ........ 99
- inline double **normalize180** (double a) ........ 99
- Length **operator+** (const Length& a, const Length& b) ........ 100
- Length **operator-** (const Length& a, const Length& b) ........ 100
- Length **operator*** (const Length& a, double b) ........ 100
- Length **operator/** (const Length& a, double b) ........ 101
- bool **operator**< (const Length& a, const Length& b) ........ 101
- bool **operator==** (const Length& a, const Length& b) ........ 101
- Vector **operator+** (const Vector& a, const Vector& b) ........ 102
- Vector **operator-** (const Vector& a, const Vector& b) ........ 102


## Detailed Description

Pikkuolioita. Nimiavaruus NobelNS sisältää erinäköisiä hyödyllisiä pikku- olioita, joita käytetään ohjelmiston toteutuksen apuna, erityisesti raja- pintojen välillä kulkevina viestiolioina, mutta myös muuna toteutuksen apuna.

## Member Documentation

**NobelNS::AngleRep (class)**

**enum** `NobelNS::AngleUnit` **{ Deg, Deg10th, Rad, PTU }**

Kulman mahdolliset mittayksiköt. PTU tarkoittaa pan-tilt-yksikön käyttämää mittayksikköä, Rad radiaaneja, Deg ja Deg10th asteita ja asteiden kymmenesosia.

**NobelNS::Angle** (class)

Itsekonvertoiva kulmayksikkö. Angle-luokan ilmentymät ovat kulmia, jotka osaavat itse kertoa kokonsa eri mittayksiköissä. Kulma on normalisoiva siten, että palautettu kulman koko on aina välillä ]-180, 180] astetta.

**Angle** `NobelNS::operator+` **(const Angle& a, const Angle& b)** []

Kahden kulman yhteenlasku.

**Parameters**:

| | |
|---|---|
| *a* | ensimmäinen yhteenlaskettava |
| *b* | toinen yhteenlaskettava |

**Return**:

yhteenlaskettu kulma

**Angle** `NobelNS::operator-` **(const Angle& a, const Angle& b)** []

Kahden kulman vähennyslasku.

**Parameters**:

| | |
|---|---|
| *a* | kulma, josta vähennetään |
| *b* | kulma, joka vähennetään |

**Return**:

vähennyslaskun tulos

**Angle** `NobelNS::operator*` **(const Angle& a, double b)** []

Kulman kertominen luvulla.

**Parameters**:

| | |
|---|---|
| *a* | kulma joka kerrotaan |
| *b* | luku jolla kerrotaan |

**Return**:

kerrottu kulma

**Angle** `NobelNS::operator/` **(const Angle& a, double b)** []

Kulman jakaminen luvulla.

**Parameters**:

| | |
|---|---|
| *a* | kulma joka jaetaan |
| *b* | luku jolla jaetaan |

**Return**:

jaettu kulma

**bool** `NobelNS::operator<` **(const Angle& a, const Angle& b)** []

Pienempi-kuin operaattori kulmille.

**Parameters**:

*a* vasen puoli
*b* oikea puoli

**Return**:

a < b

**bool** `NobelNS::operator==` **(const Angle& a, const Angle& b)** []

Yhtäsuuruusvertailu kulmille.

**Parameters**:

*a* vasen puoli
*b* oikea puoli

**Return**:

a==b

**inline double** `NobelNS::normalize360` **(double a)** []

Normalisoi niin, että alue on [0, 360[

**Parameters**:

*a* normalisoitava

**Return**:

normalisoitu

**inline double** `NobelNS::normalize180` **(double a)** []

Normalisoi niin, että alue on ]-180, 180].

**Parameters**:

*a* normalisoitava

**Return**:

normalisoitu

**NobelNS::BlockExceptionRep (class)**

**NobelNS::BlockException (class)**

**See also**: Nobel::Movement#Pose, move(const, Vector&, const, Pose&)

**Author**:

Jarkko Salminen

**NobelNS::LengthRep (class)**

**enum** `NobelNS::LengthUnit` **{ M, CM, In, In10th }**

Pituuden mahdolliset mittayksiköt. M ja CM ovat metri ja senttimetri, In ja In10th tuuma ja tuuman kymmenesosa vastaavasti.

**NobelNS::Length (class)**

Itsekonvertoiva pituusyksikkö. Length-oliot osaavat kertoa pituutensa eri mittayksiköissä.

**Length** `NobelNS::operator+` **(const Length& a, const Length& b)** []

Yhteenlaskuoperaattori pituuksille.

**Parameters**:

*a* ensimmäinen yhteenlaskettava
*b* toinen yhteenlaskettava

**Return**:

summa

**Length** `NobelNS::operator-` **(const Length& a, const Length& b)** []

Vähennyslaskuoperaattori pituuksille.

**Parameters**:

*a* se, josta vähennetään
*b* se, joka vähennetään

**Return**:

erotus

**Length** `NobelNS::operator*` **(const Length& a, double b)** []

Kertolaskuoperaattori. Pituuden kertominen liukuluvulla.

**Parameters**:

| | |
|---|---|
| *a* | pituus |
| *b* | liukuluku, jolla pituus kerrotaan |

**Return**:

tulo

**Length** `NobelNS::operator/` **(const Length& a, double b)** []

Jakolaskuoperaattori. Pituuden jakaminen liukuluvulla.

**Parameters**:

| | |
|---|---|
| *a* | pituus |
| *b* | liukuluku, jolla pituus jaetaan |

**Return**:

jaettu pituus

**bool** `NobelNS::operator<` **(const Length& a, const Length& b)** []

Pienempi-kuin operaattori pituuksille.

**Parameters**:

| | |
|---|---|
| *a* | vasen puoli |
| *b* | oikea puoli |

**Return**:

a < b

**bool** `NobelNS::operator==` **(const Length& a, const Length& b)** []

Yhtäsuuruusvertailu pituuksille.

**Parameters**:

| | |
|---|---|
| *a* | vasen puoli |
| *b* | oikea puoli |

**Return**:

a==b

**NobelNS::PoseRep** (class)

**NobelNS::Pose** (class)

Sijainti+suunta. Luokka, joka ilmaisee (robotin) sijainnin sekä suunnan.

**NobelNS::RobotAccelerationRep** (class)

**enum** `NobelNS::AccelerationUnit` **{ Ms2, CMs2, IN10THs2 }**

**NobelNS::RobotAcceleration** (class)

Itsekonvertoiva kiihtyvyysluokka. Luokan ilmentymiä luotaessa ja muutettaessa tarkistetaan, että kiihtyvyys on sallituissa rajoissa. Robotin kiihtyvyyden rajat ovat [0, 800[ tuuman kymmenesosaa/s**2 (800 tuuman kymmenesosaa = 203.2 senttimetriä)

**NobelNS::RobotVelocityRep** (class)

**enum** `NobelNS::VelocityUnit` **{ Ms, CMs, IN10THs }**

**NobelNS::RobotVelocity** (class)

Itsekonvertoiva nopeusyksikkö. Luokan oliot osaavat itse konvertoida itsensä toiseen mittayksikköön. Luokan olioita luotaessa ja muutettaessa tarkistetaan, että nopeus on robotin sallimissa rajoissa. Robotin nopeus rajoittuu alueelle [0, 400[ tuuman kymmenesosaa sekunnissa (400 tuuman kymmenesosaa = 101.6 senttimetriä).

**NobelNS::SelfTestingObject** (class)

**NobelNS::InvariantException** (class)

**NobelNS::VectorRep** (class)

**NobelNS::Vector** (class)

**Vector** `NobelNS::operator+` **(const Vector& a, const Vector& b)** []

Yhteenlaskuoperaattori vektoreille.

**Parameters**:

*a* summattava
*b* summattava

**Return**:

summa

**Vector** `NobelNS::operator-` **(const Vector& a, const Vector& b)** []

Vähennyslaskuoperaattori vektoreille.

**Parameters**:

*a* mistä vähennetään
*b* mikä vähennetään

**Return**:

erotus

## 70 class NobelNS::Angle

Itsekonvertoiva kulmayksikkö.

`#include <Nobel/Angle.h>`

Inherits: SelfTestingObject

*public*

()

### Public Methods


- double **getAngle** (AngleUnit unit, bool norm360=false) const ..... 103
- virtual std::string **asString** () const ..... 104
- virtual bool **invariant** () const ..... 104
- Angle& **setAngle** (double newAngle, AngleUnit unit) ..... 104
- Angle& **operator+=** (const Angle& a) ..... 104
- Angle& **operator-=** (const Angle& a) ..... 104
- Angle& **operator*=** (double b) ..... 105
- Angle& **operator/=** (double b) ..... 105
- **Angle** (double angle=0, AngleUnit unit=Deg) ..... 105
- **Angle** (const Angle& a) ..... 105
- virtual void **swap** (Angle& a) ..... 106
- Angle& **operator=** (const Angle& a) ..... 106
- virtual **~Angle** () ..... 106


### Private Members


- AngleRep* **rep** ..... 106


### Detailed Description

Itsekonvertoiva kulmayksikkö. Angle-luokan ilmentymät ovat kulmia, jotka osaavat itse kertoa kokonsa eri mittayksiköissä. Kulma on normalisoiva siten, että palautettu kulman koko on aina välillä ]-180, 180] astetta.

### Member Documentation

**double** `NobelNS::Angle::getAngle` **(AngleUnit unit, bool norm360=false)** [**const** ]

Palauta kulma halutussa mittayksikössä.

**Parameters**:

| | |
|---|---|
| *unit* | haluttu mittayksikkö |
| *norm360* | jos true, palauttaa kulman alueella [0, 360[ |

**Return**:

kulman koko

**std::string** `NobelNS::Angle::asString` **()** [**const virtual**]

Palauta kulman esitys tekstimuodossa.

**Return**:

kulman suuruus asteissa

**bool** `NobelNS::Angle::invariant` **()** [**const virtual**]

Tarkista onko olio eheässä tilassa.

**Return**:

olion eheys totuusarvona

**Angle&** `NobelNS::Angle::setAngle` **(double newAngle, AngleUnit unit)** []

Aseta kulman suuruus.

**Parameters**:

| | |
|---|---|
| *newAngle* | kulman uusi koko |
| *unit* | koon mittayksikkö |

**Return**:

viite muutettuun kulmaan

**Angle&** `NobelNS::Angle::operator+=` **(const Angle& a)** []

Laske kulma yhteen sijoittamalla.

**Parameters**:

| | |
|---|---|
| *a* | yhteenlaskettava kulma |

**Return**:

viite muutettuun olioon

**Angle&** `NobelNS::Angle::operator-=` **(const Angle& a)** []

Vähennä kaksi kulmaa toisistaan sijoituksella.

**Parameters**:

| | |
|---|---|
| *a* | vähennettävä kulma |

**Return**:

viite muutettuun olioon

**Angle&** `NobelNS::Angle::operator*=` **(double b)** []

Kulman kertominen luvulla (sijoittava versio).

**Parameters**:

| | |
|---|---|
| *b* | luku jolla kerrotaan |

**Return**:

viite muutettuun kulmaan

**Angle&** `NobelNS::Angle::operator/=` **(double b)** []

Kulman jakaminen luvulla (sijoittava versio).

**Parameters**:

| | |
|---|---|
| *b* | luku jolla jaetaan |

**Return**:

viite muutettuun kulmaan

`NobelNS::Angle::Angle` **(double angle=0, AngleUnit unit=Deg)** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *angle* | kulman koko |
| *unit* | koon mittayksikkö |

`NobelNS::Angle::Angle` **(const Angle& a)** []

Kopiokonstruktori.

**Parameters**:

| | |
|---|---|
| *a* | kopioitava kulma |

**void** `NobelNS::Angle::swap` **(Angle& a)** [ **virtual**]

Vaihda kaksi kulmaa keskenään.

**Parameters**:

*a* vaihdettava kulma

**Angle&** `NobelNS::Angle::operator=` **(const Angle& a)** []

Sijoitusoperaattori.

**Parameters**:

*a* sijoitettava kulma.

`NobelNS::Angle::~Angle` **()** [ **virtual**]

**AngleRep*** `NobelNS::Angle::rep` [**private**]

## 71 class NobelNS::AngleRep

`#include <Nobel/AngleRep.h>`

Inherits: SelfTestingObject

*public*

()

### Public Methods

- virtual bool **invariant** () const .......... 106
- void **normalize** () .......... 106
- virtual ~**AngleRep** () .......... 107

### Public Members

- double **angleDeg** .......... 106

### Detailed Description

### Member Documentation

**double** `NobelNS::AngleRep::angleDeg`

**bool** `NobelNS::AngleRep::invariant` **()** [**const virtual**]

Invariantin tarkistus.

**Return**:

true, jos angleDeg kuuluu alueelle ]-180, 180]

**void** `NobelNS::AngleRep::normalize` **()** []

`NobelNS::AngleRep::~AngleRep` **()** [ **virtual**]

## 72 class NobelNS::BlockException

BlockException on poikkeusolio, jonka Movement- olio heittää yllättävän esteen ilmestyessä.

`#include <Nobel/BlockException.h>`

Inherits: SelfTestingObject

*public*

(), std::runtime_error

*public*

(unknown)

### Public Methods

- virtual std::string **getMessage** () const ....... 107
- virtual Vector **getUnfinishedMovement** () const ....... 108
- virtual Vector **getBlockLocation** () const ....... 108
- virtual Pose **getRelativePose** () const ....... 108
- explicit **BlockException** (const Vector& a, const Vector& b, const Pose& c) ....... 108
- **BlockException** (const BlockException&) ....... 108
- virtual void **swap** (BlockException&) ....... 108
- BlockException& **operator=** (const BlockException&) ....... 108
- virtual ~**BlockException** () ....... 108
- virtual bool **invariant** () const ....... 109

### Private Members

- BlockExceptionRep* **rep** ....... 109

### Detailed Description

**See also**: Nobel::Movement#Pose, move(const, Vector&, const, Pose&)

**Author**:

Jarkko Salminen

### Member Documentation

**std::string** `NobelNS::BlockException::getMessage` **()** [**const virtual**]

Esittää poikkeuksen stringinä.

**Return**:

Olion esitys stringinä.

**Vector** `NobelNS::BlockException::getUnfinishedMovement` **()** [**const virtual**]

Palauttaa robotin kulkematta jääneen matkan

**Return**:

kulkematta jäänyt matka

**Vector** `NobelNS::BlockException::getBlockLocation` **()** [**const virtual**]

Aksessori, joka palauttaa poikkeuksen aiheuttaneen esteen sijainnin.

**Return**:

Esteen sijainti

**Pose** `NobelNS::BlockException::getRelativePose` **()** [**const virtual**]

Palauta robotin uusi suhteellinen pose.

**Return**:

uusi suhteellinen pose

**explicit** `NobelNS::BlockException::BlockException` **(const Vector& a, const Vector& b, const Pose& c)** []

Konstruktori

**Parameters**:

*a* liikkumatta jäänyt matka
*b* esteen suhteellinen sijainti robottiin nähden
*c* robotin uusi suhteellinen pose

`NobelNS::BlockException::BlockException` **(const BlockException&)** []

Kopiokonstruktori

**void** `NobelNS::BlockException::swap` **(BlockException&)** [ **virtual**]

Swap

**BlockException&** `NobelNS::BlockException::operator=` **(const BlockException&)** []

Sijoitus

`NobelNS::BlockException::~BlockException` **()** [ **virtual**]

Destruktori

**bool** `NobelNS::BlockException::invariant` **()** [**const virtual**]

Testaa olion tilan oikeellisuuden.

**Return**:

Palauttaa true, jos olion attribuutit ovat eheässä tilassa. Muuten false.

**See also**: Nobel::SelfTestingObject

**BlockExceptionRep*** `NobelNS::BlockException::rep` [**private**]

## 73 class NobelNS::BlockExceptionRep

`#include <Nobel/BlockExceptionRep.h>`

Inherits: SelfTestingObject

*public*

()

### Public Methods


- virtual std::string **asString** () const ....... 110
- explicit **BlockExceptionRep** (const Vector&, const Vector&, const Pose&) ....... 110
- **BlockExceptionRep** (const BlockExceptionRep&) ....... 110
- BlockExceptionRep& **operator=** (const BlockExceptionRep&) ....... 110
- virtual **~BlockExceptionRep** () ....... 110
- virtual bool **invariant** () const ....... 110


### Public Members


- Vector **unmoved** ....... 110
- Vector **blockLocation** ....... 110
- Pose **newRelativePose** ....... 110

**Detailed Description**

**Member Documentation**

**Vector** `NobelNS::BlockExceptionRep::unmoved`

**Vector** `NobelNS::BlockExceptionRep::blockLocation`

**Pose** `NobelNS::BlockExceptionRep::newRelativePose`

**std::string** `NobelNS::BlockExceptionRep::asString` **()** [**const virtual**]

**explicit** `NobelNS::BlockExceptionRep::BlockExceptionRep` **(const Vector&, const Vector&, const Pose&)** []

`NobelNS::BlockExceptionRep::BlockExceptionRep` **(const BlockExceptionRep&)** []

**BlockExceptionRep&** `NobelNS::BlockExceptionRep::operator=` **(const BlockExceptionRep&)** []

`NobelNS::BlockExceptionRep::~BlockExceptionRep` **()** [ **virtual**]

**bool** `NobelNS::BlockExceptionRep::invariant` **()** [**const virtual**]

## 74 class NobelNS::InvariantException

`#include <Nobel/SelfTestingObject.h>`
Inherits: std::runtime_error

*public*

(unknown)

**Public Methods**

- **InvariantException** (std::string s, int line) .......... 110
- virtual **~InvariantException** () .......... 110

**Private Static Methods**

- static std::string **toString** (int i) .......... 110

**Detailed Description**

**Member Documentation**

`NobelNS::InvariantException::InvariantException` **(std::string s, int line)** []

`NobelNS::InvariantException::~InvariantException` **()** [ **virtual**]

**std::string** `NobelNS::InvariantException::toString` **(int i)** [**private static** ]

## 75 class NobelNS::Length

Itsekonvertoiva pituusyksikkö.

`#include <Nobel/Length.h>`
Inherits: SelfTestingObject

*public*

()

## Public Methods


- double **getLength** (LengthUnit unit) const ........ 111
- virtual std::string **asString** () const ........ 111
- virtual bool **invariant** () const ........ 112
- Length& **setLength** (double newLength, LengthUnit unit) ........ 112
- Length& **operator+=** (const Length& a) ........ 112
- Length& **operator-=** (const Length& a) ........ 112
- Length& **operator*=** (double a) ........ 113
- Length& **operator/=** (double a) ........ 113
- **Length** (double length=0, LengthUnit unit=CM) ........ 113
- **Length** (const Length& a) ........ 113
- virtual void **swap** (Length& a) ........ 113
- Length& **operator=** (const Length& a) ........ 113
- virtual ~**Length** () ........ 114


## Private Members


- LengthRep* **rep** ........ 114


## Detailed Description

Itsekonvertoiva pituusyksikkö. Length-oliot osaavat kertoa pituutensa eri mittayksiköissä.

## Member Documentation

**double** `NobelNS::Length::getLength` **(LengthUnit unit)** [**const** ]

Hae pituus.
**Parameters**:

*unit* haluttu mittayksikkö

**Return**:

pituus pyydetyissä yksiköissä

**std::string** `NobelNS::Length::asString` **()** [**const virtual**]

Hae pituuden esitys tekstinä.

**Return**:

pituus senttimetreinä

**bool** `NobelNS::Length::invariant` **()** [**const virtual**]

Tarkista olion eheys.

**Return**:

olion eheys

**Length&** `NobelNS::Length::setLength` **(double newLength, LengthUnit unit)** []

Aseta pituus.

**Parameters**:

| | |
|---|---|
| *newLength* | uusi pituus |
| *unit* | pituuden mittayksikkö |

**Return**:

viite muutettuun olioon

**Length&** `NobelNS::Length::operator+=` **(const Length& a)** []

Yhteenlaskuoperaattori sijoituksella.

**Parameters**:

| | |
|---|---|
| *a* | yhteenlaskettava pituus |

**Return**:

viite muutettuun olioon

**Length&** `NobelNS::Length::operator-=` **(const Length& a)** []

Vähennyslaskuoperaattori sijoituksella.

**Parameters**:

| | |
|---|---|
| *a* | vähennettävä pituus |

**Return**:

viite muutettuun olioon

**Length&** `NobelNS::Length::operator*=` **(double a)** []

Kertolaskuoperaattori sijoituksella.

**Parameters**:

| | |
|---|---|
| *a* | luku, jolla pituus kerrotaan |

**Return**:

viite muutettuun olioon

**Length&** `NobelNS::Length::operator/=` **(double a)** []

Jakolaskuoperaattori sijoituksella.

**Parameters**:

| | |
|---|---|
| *a* | luku, jolla pituus jaetaan |

**Return**:

viite muutettuun olioon

`NobelNS::Length::Length` **(double length=0, LengthUnit unit=CM)** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *length* | pituus |
| *unit* | mittayksikkö |

`NobelNS::Length::Length` **(const Length& a)** []

Kopiokonstruktori.

**Parameters**:

| | |
|---|---|
| *a* | kopioitava |

**void** `NobelNS::Length::swap` **(Length& a)** [ **virtual**]

Vaihto-operaatio.

**Parameters**:

| | |
|---|---|
| *a* | vaihdettava |

**Length&** `NobelNS::Length::operator=` **(const Length& a)** []

Sijoitusoperaattori.

**Parameters**:

*a* sijoitettava

`NobelNS::Length::~Length` **()** [ **virtual**]

**LengthRep*** `NobelNS::Length::rep` [**private**]

## 76 class NobelNS::LengthRep

`#include <Nobel/LengthRep.h>`

Inherits: SelfTestingObject

*public*

()

### Public Methods

- virtual bool **invariant** () const ....114
- virtual **~LengthRep** () ....114

### Public Members

- double **lengthCM** ....114

### Detailed Description

### Member Documentation

**double** `NobelNS::LengthRep::lengthCM`

**bool** `NobelNS::LengthRep::invariant` **()** [**const virtual**]

Invariantin tarkistus.

**Return**:

true

`NobelNS::LengthRep::~LengthRep` **()** [ **virtual**]

## 77 class NobelNS::Pose

Sijainti+suunta.

`#include <Nobel/Pose.h>`

Inherits: SelfTestingObject

*public*

()

### Public Methods


- Vector **getPosition** () const ........ 115
- Angle **getAngle** () const ........ 115
- Vector **getProportionalTo** (const Vector& v) const ........ 116
- virtual bool **invariant** () const ........ 116
- virtual std::string **asString** () const ........ 116
- Pose& **setPosition** (const Vector& p) ........ 116
- Pose& **updatePose** (const Pose& relPose) ........ 116
- Pose& **setAngle** (const Angle& a) ........ 117
- Pose& **setPose** (const Vector& p, const Angle& a) ........ 117
- **Pose** (const Vector& p=Vector(), const Angle& a=Angle()) ........ 117
- **Pose** (const Pose& p) ........ 117
- virtual void **swap** (Pose& p) ........ 117
- Pose& **operator=** (const Pose& p) ........ 118
- virtual **~Pose** () ........ 118


### Private Members


- PoseRep* **rep** ........ 118


### Detailed Description

Sijainti+suunta. Luokka, joka ilmaisee (robotin) sijainnin sekä suunnan.

### Member Documentation

**Vector** `NobelNS::Pose::getPosition` **()** [**const** ]

Hae sijainti.
**Return**:

sijainti

**Angle** `NobelNS::Pose::getAngle` **()** [**const** ]

Hae suunta.
**Return**:

suunta

**Vector** `NobelNS::Pose::getProportionalTo` **(const Vector& v)** [**const** ]

Suhteellisen vektorin laskenta.

**Parameters**:

*v* paikkavektori, johon nähden lasketaan

**Return**:

suhteellinen vektori v:hen nähden

**Author**:

Maarit

**bool** `NobelNS::Pose::invariant` **()** [**const virtual**]

Invariantin tarkistus.

**Return**:

invariantin paikkaansapitävyys

**std::string** `NobelNS::Pose::asString` **()** [**const virtual**]

Hae sijainti ja suunta tekstimuodossa.

**Return**:

location: X, Y orientation: alpha

**Pose&** `NobelNS::Pose::setPosition` **(const Vector& p)** []

Aseta sijainti.

**Parameters**:

*p* uusi sijainti

**Return**:

viite muutettuun oliooon

**Pose&** `NobelNS::Pose::updatePose` **(const Pose& relPose)** []

Päivitä pose uuden suhteellisen posen mukaan (suhteellinen pose on pose, johon robotti on siirtynyt päivitettävästä posesta se origona).

**Parameters**:

*relPose* suhteellinen pose

**Return**:

päivitetty pose

**Pose&** `NobelNS::Pose::setAngle` **(const Angle& a)** []

Aseta suunta.

**Parameters**:

*a* uusi suunta

**Return**:

viite muutettuun olioon

**Pose&** `NobelNS::Pose::setPose` **(const Vector& p, const Angle& a)** []

Aseta sijainti ja suunta.

**Parameters**:

*p* uusi sijainti
*a* uusi suunta

**Return**:

viite muutettuun olioon

`NobelNS::Pose::Pose` **(const Vector& p=Vector(), const Angle& a=Angle())** []

Konstruktori.

**Parameters**:

*p* sijainti
*a* suunta

`NobelNS::Pose::Pose` **(const Pose& p)** []

Kopiokonstruktori.

**Parameters**:

*p* kopioitava

**void** `NobelNS::Pose::swap` **(Pose& p)** [ **virtual**]

Vaihto-operaatio.

**Parameters**:

*p* vaihdettava

**Pose&** `NobelNS::Pose::operator=` **(const Pose& p)** []

Sijoitusoperaattori.

**Parameters**:

*p* sijoitettava

**Return**:

viite sijoituksen kohteeseen

`NobelNS::Pose::~Pose` **()** [ **virtual**]

**PoseRep*** `NobelNS::Pose::rep` [**private**]

## 78 class NobelNS::PoseRep

`#include <Nobel/PoseRep.h>`

Inherits: SelfTestingObject

*public*

()

### Public Methods

- virtual bool **invariant** () const ............ 119
- **PoseRep** (const Vector& v, const Angle& a) ............ 119
- virtual ~**PoseRep** () ............ 119

### Public Members

- Vector **position** ............ 119
- Angle **orientation** ............ 119

### Detailed Description

### Member Documentation

**Vector** `NobelNS::PoseRep::position`

**Angle** `NobelNS::PoseRep::orientation`

**bool** `NobelNS::PoseRep::invariant` **()** [**const virtual**]

Invariantin tarkistus. Tarkistaa attribuuttien invariantit.
**Return**:

invariantin paikkaansapitävyys

`NobelNS::PoseRep::PoseRep` **(const Vector& v, const Angle& a)** []

Konstruktori.
**Parameters**:

| | |
|---|---|
| *v* | sijainti |
| *a* | suunta |

`NobelNS::PoseRep::~PoseRep` **()** [ **virtual**]

## 79 class NobelNS::RobotAcceleration

Itsekonvertoiva kiihtyvyysluokka.
`#include <Nobel/RobotAcceleration.h>`
Inherits: SelfTestingObject
*public*
()

### Public Methods


- double **getAccel** (AccelerationUnit unit) const ........ 120
- virtual std::string **asString** () const ........ 120
- virtual bool **invariant** () const ........ 120
- RobotAcceleration& **setAccel** (double newAccel, AccelerationUnit unit) ........ 120
- **RobotAcceleration** (double acc, AccelerationUnit unit) ........ 121
- **RobotAcceleration** (const RobotAcceleration& r) ........ 121
- virtual void **swap** (RobotAcceleration& r) ........ 121
- RobotAcceleration& **operator=** (const RobotAcceleration& r) ........ 121
- virtual **~RobotAcceleration** () ........ 121

### Private Members

- RobotAccelerationRep* **rep** ........................................................................121

### Detailed Description

Itsekonvertoiva kiihtyvyysluokka. Luokan ilmentymiä luotaessa ja muutettaessa tarkistetaan, että kiihtyvyys on sallituissa rajoissa. Robotin kiihtyvyyden rajat ovat [0, 800[ tuuman kymmenesosaa/s**2 (800 tuuman kymmenesosaa = 203.2 senttimetriä)

### Member Documentation

**double** `NobelNS::RobotAcceleration::getAccel` **(AccelerationUnit unit)** [**const** ]

Hae kiihtyvyys.

**Parameters**:

*unit* haluttu mittayksikkö

**Return**:

kiihtyvyys

**std::string** `NobelNS::RobotAcceleration::asString` **()** [**const virtual**]

Muodosta merkkijonoesitys.

**Return**:

kiihtyvyys (cm/s**2)

**bool** `NobelNS::RobotAcceleration::invariant` **()** [**const virtual**]

Tarkista invariantti.

**Return**:

invariantin eheys

**RobotAcceleration&** `NobelNS::RobotAcceleration::setAccel` **(double newAccel, AccelerationUnit unit)** []

Aseta kiihtyvyys.

**Parameters**:

| | |
|---|---|
| *newAccel* | uusi kiihtyvyys |
| *unit* | kiihtyvyyden mittayksikkö |

**Return**:

viite muutettuun olioon

**Throws**: std::range_error

`NobelNS::RobotAcceleration::RobotAcceleration` **(double acc, AccelerationUnit unit)** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *acc* | kiihtyvyys |
| *unit* | mittayksikkö |

**Throws**: std::range_error

`NobelNS::RobotAcceleration::RobotAcceleration` **(const RobotAcceleration& r)** []

Kopiokonstruktori.

**Parameters**:

| | |
|---|---|
| *r* | kopioitava olio |

**void** `NobelNS::RobotAcceleration::swap` **(RobotAcceleration& r)** [ **virtual**]

Vaihto-operaatio.

**Parameters**:

| | |
|---|---|
| *r* | olio, jonka kanssa vaihdetaan |

**RobotAcceleration&** `NobelNS::RobotAcceleration::operator=` **(const RobotAcceleration& r)** []

Sijoitusoperaattori.

**Parameters**:

| | |
|---|---|
| *r* | sijoitettava olio |

**Return**:

viite muutettuun olioon

`NobelNS::RobotAcceleration::~RobotAcceleration` **()** [ **virtual**]

**RobotAccelerationRep*** `NobelNS::RobotAcceleration::rep` [**private**]

## 80 class NobelNS::RobotAccelerationRep

`#include <Nobel/RobotAccelerationRep.h>`

Inherits: SelfTestingObject

*public*

()

### Public Methods


- virtual bool **invariant** () const ....................122
- bool **allowedAcceleration** () const ....................122
- **RobotAccelerationRep** (const Length& l) ....................122
- virtual ~**RobotAccelerationRep** () ....................122


### Public Members


- Length **accS2** ....................122


### Detailed Description

### Member Documentation

**Length** `NobelNS::RobotAccelerationRep::accS2`

**bool** `NobelNS::RobotAccelerationRep::invariant` **()** [**const virtual**]

Invariantti. Tarkistaa accS2:n invariantin ja nopeuden sallittavuuden.

**bool** `NobelNS::RobotAccelerationRep::allowedAcceleration` **()** [**const** ]

Tarkista, onko kiihtyvyys sallituissa rajoissa.
**Return**:

true, jos kiihtyvyys OK, muuten false

`NobelNS::RobotAccelerationRep::RobotAccelerationRep` **(const Length& l)** []

Konstruktori.
**Parameters**:

*l* kiihtyvyys yksikköä/s**2

`NobelNS::RobotAccelerationRep::~RobotAccelerationRep` **()** [ **virtual**]

## 81 class NobelNS::RobotVelocity

Itsekonvertoiva nopeusyksikkö.
`#include <Nobel/RobotVelocity.h>`
Inherits: SelfTestingObject
*public*
()

### Public Methods


- double **getVelocity** (VelocityUnit unit) const ........ 123
- virtual std::string **asString** () const ........ 123
- virtual bool **invariant** () const ........ 123
- RobotVelocity& **setVelocity** (double vel, VelocityUnit unit) ........ 124
- **RobotVelocity** (double velocity, VelocityUnit unit) ........ 124
- **RobotVelocity** (const RobotVelocity& r) ........ 124
- virtual void **swap** (RobotVelocity& r) ........ 124
- RobotVelocity& **operator=** (const RobotVelocity& r) ........ 124
- virtual ~**RobotVelocity** () ........ 125


### Private Members


- RobotVelocityRep* **rep** ........ 125


### Detailed Description

Itsekonvertoiva nopeusyksikkö. Luokan oliot osaavat itse konvertoida itsensä toiseen mittayksikköön. Luokan olioita luotaessa ja muutettaessa tarkistetaan, että nopeus on robotin sallimissa rajoissa. Robotin nopeus rajoittuu alueelle [0, 400[ tuuman kymmenesosaa sekunnissa (400 tuuman kymmenesosaa = 101.6 senttimetriä).

### Member Documentation

**double** `NobelNS::RobotVelocity::getVelocity` **(VelocityUnit unit)** [**const** ]

Hae nopeus.

**Parameters**:

*unit* haluttu nopeusyksikkö

**Return**:

nopeus

**std::string** `NobelNS::RobotVelocity::asString` **()** [**const virtual**]

Muodosta merkkijonoesitys.

**Return**:

nopeus senttimetreinä sekunnissa

**bool** `NobelNS::RobotVelocity::invariant` **()** [**const virtual**]

Tarkista invariantti.

**Return**:

invariantin paikkaansapitävyys

**RobotVelocity&** `NobelNS::RobotVelocity::setVelocity` **(double vel, VelocityUnit unit)** []

Aseta nopeus.

**Parameters**:

| | |
|---|---|
| *vel* | uusi nopeus |
| *unit* | nopeuden yksikkö |

**Return**:

viite muutettuun olioon

**Throws**: std::range_error

`NobelNS::RobotVelocity::RobotVelocity` **(double velocity, VelocityUnit unit)** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *velocity* | nopeus |
| *unit* | nopeuden mittayksikkö |

**Throws**: std::range_error

`NobelNS::RobotVelocity::RobotVelocity` **(const RobotVelocity& r)** []

Kopiokonstruktori.

**Parameters**:

| | |
|---|---|
| *r* | kopioitava olio |

**void** `NobelNS::RobotVelocity::swap` **(RobotVelocity& r)** [ **virtual**]

Vaihto-operaatio.

**Parameters**:

| | |
|---|---|
| *r* | olio, jonka kanssa vaihdetaan |

**RobotVelocity&** `NobelNS::RobotVelocity::operator=` **(const RobotVelocity& r)** []

Sijoitusoperaattori.

**Parameters**:

| | |
|---|---|
| *r* | sijoitettava |

**Return**:

viittaus muutettuun olioon

`NobelNS::RobotVelocity::~RobotVelocity` **()** [ **virtual**]

**RobotVelocityRep*** `NobelNS::RobotVelocity::rep` [**private**]

## 82 class NobelNS::RobotVelocityRep

`#include <Nobel/RobotVelocityRep.h>`
Inherits: SelfTestingObject
*public*
()

### Public Methods


- virtual bool **invariant** () const ........ 125
- bool **allowedSpeed** () const ........ 125
- **RobotVelocityRep** (const Length& l) ........ 125
- virtual ~**RobotVelocityRep** () ........ 126


### Public Members


- Length **speedS** ........ 125


### Detailed Description

### Member Documentation

**Length** `NobelNS::RobotVelocityRep::speedS`

**bool** `NobelNS::RobotVelocityRep::invariant` **()** [**const virtual**]

Tarkista invariantti. Tarkistaa attribuutin invariantin ja nopeuden rajat.
**Return**:

invariantin eheys

**bool** `NobelNS::RobotVelocityRep::allowedSpeed` **()** [**const** ]

Tarkista, onko nopeus sallituissa rajoissa.
**Return**:

true, jos nopeus on sallittu, false muuten

NobelNS::RobotVelocityRep::RobotVelocityRep **(const Length& l)** []

Konstruktori.

**Parameters**:

*l* nopeus sekunnissa

NobelNS::RobotVelocityRep::~RobotVelocityRep **()** [ **virtual**]

## 83 class NobelNS::SelfTestingObject

**Abstract class.**

`#include <Nobel/SelfTestingObject.h>`

AngleAngleRepBlockExceptionBlockExceptionRepGridGridCellGridCellRepGridIteratorGridIteratorRepGridRepInputInputRepLandmarkRecognitionRepLandmarksLengthLengthRepLocalizationLocalizationRepMapMapRepOutputOutputRepPosePoseRepRobotAccelerationRobotAccelerationRepRobotMoverRobotMoverRepRobotVelocityRobotVelocityRepSTCSTCRepScoutRollerScoutRollerRepSegmentationSegmentationRepSettingsRepSonarSonarRepVectorVectorRep

### Public Methods


- virtual bool ***invariant*** () const ....126
- virtual **~SelfTestingObject** () ....126


### Detailed Description

### Member Documentation

**bool** *NobelNS::SelfTestingObject::invariant* **()** [**const pure virtual**]

NobelNS::SelfTestingObject::~SelfTestingObject **()** [ **virtual**]

## 84 class NobelNS::Vector

`#include <Nobel/Vector.h>`

Inherits: SelfTestingObject

*public*

()

### Public Methods


- Angle **getAngle** () const ....127
- Length **getLength** () const ....127
- Length **getX** () const ....127
- Length **getY** () const ....127
- virtual std::string **asString** () const ....128
- virtual bool **invariant** () const ....128
- Vector& **setVector** (const Angle& a, const Length& l) ....128

- Vector& **setVector** (const Length& x, const Length& y) ....128
- Vector& **operator+=** (const Vector& v) ....128
- Vector& **operator-=** (const Vector& v) ....129
- **Vector** (const Length& x=Length(), const Length& y=Length()) ....129
- **Vector** (const Angle& a, const Length& l) ....129
- **Vector** (const Vector& v) ....129
- virtual void **swap** (Vector& v) ....129
- Vector& **operator=** (const Vector& v) ....129
- virtual ~**Vector** () ....130


### Private Members


- VectorRep* **rep** ....130


### Detailed Description

### Member Documentation

**Angle** `NobelNS::Vector::getAngle` **()** [**const** ]

Hae vektorin kulma suhteessa X-akseliin.
**Return**:

vektorin kulma ( ]-180, 180] astetta )

**Length** `NobelNS::Vector::getLength` **()** [**const** ]

Hae vektorin pituus.
**Return**:

vektorin pituus (aina ei-negatiivinen)

**Length** `NobelNS::Vector::getX` **()** [**const** ]

Hae vektorin x-komponentin pituus.
**Return**:

x-komponentin pituus

**Length** `NobelNS::Vector::getY` **()** [**const** ]

Hae vektorin y-komponentin pituus.
**Return**:

y-komponentin pituus

**std::string** `NobelNS::Vector::asString` **()** [**const virtual**]

Hae vektorin tekstimuotoesitys

**Return**:

X, Y

**bool** `NobelNS::Vector::invariant` **()** [**const virtual**]

Tarkista invariantti.

**Return**:

invariantin paikkaansapitävyys

**Vector&** `NobelNS::Vector::setVector` **(const Angle& a, const Length& l)** []

Aseta vektori uudelleen.

**Parameters**:

*a* vektorin uusi kulma
*l* vektorin uusi pituus

**Return**:

viite muutettuun vektoriin

**Vector&** `NobelNS::Vector::setVector` **(const Length& x, const Length& y)** []

Aseta vektori uudelleen.

**Parameters**:

*x* uusi x-komponentin pituus
*y* uusi y-komponentin pituus

**Return**:

viite muutettuun vektoriin

**Vector&** `NobelNS::Vector::operator+=` **(const Vector& v)** []

Yhteenlaskuoperaattori sijoituksella.

**Parameters**:

*v* yhteenlaskettava

**Return**:

viite muutettuun olioon

**Vector& `NobelNS::Vector::operator-=` (const Vector& v)** []

Vähennyslaskuoperaattori sijoituksella.

**Parameters**:

| | |
|---|---|
| *v* | vähennettävä |

**Return**:

viite muutettuun olioon

**`NobelNS::Vector::Vector` (const Length& x=Length(), const Length& y=Length())** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *x* | x-komponentin pituus |
| *y* | y-komponentin pituus |

**`NobelNS::Vector::Vector` (const Angle& a, const Length& l)** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *a* | vektorin kulma |
| *l* | vektorin pituus |

**`NobelNS::Vector::Vector` (const Vector& v)** []

Kopiokonstruktori.

**Parameters**:

| | |
|---|---|
| *v* | kopioitava |

**void `NobelNS::Vector::swap` (Vector& v)** [ **virtual**]

Vaihto-operaatio.

**Parameters**:

| | |
|---|---|
| *v* | vaihdettava |

**Vector&** `NobelNS::Vector::operator=` **(const Vector& v)** []

Sijoitusoperaattori.

**Parameters**:

*v* sijoitettava.

`NobelNS::Vector::~Vector` **()** [ **virtual**]

**VectorRep*** `NobelNS::Vector::rep` [**private**]

## 85 class NobelNS::VectorRep

`#include <Nobel/VectorRep.h>`

Inherits: SelfTestingObject

*public*

()

### Public Methods


- virtual bool **invariant** () const ....130
- **VectorRep** (const Length& a, const Length& b) ....130
- virtual **~VectorRep** () ....131


### Public Members


- Length **x** ....130
- Length **y** ....130


### Detailed Description

### Member Documentation

**Length** `NobelNS::VectorRep::x`

**Length** `NobelNS::VectorRep::y`

**bool** `NobelNS::VectorRep::invariant` **()** [**const virtual**]

Invariantin tarkistus. Tarkistaa attribuuttien invariantit.

**Return**:

invariantin paikkaansapitävyys

NobelNS::VectorRep::VectorRep **(const Length& a, const Length& b)** []

NobelNS::VectorRep::~VectorRep **()** [ **virtual**]

## 86 class RGB24Image

#include <Localization/RGB24Image.h>

### Public Methods


- **RGB24Image** () ........ 132
- **RGB24Image** (int w, int h) ........ 132
- ~**RGB24Image** () ........ 132
- inline unsigned char* **rawBuffer** () ........ 132
- inline unsigned int **size** () ........ 132
- inline int **getWidth** () ........ 132
- inline int **getHeight** () ........ 132
- inline int **index** (int x, int y) ........ 132
- inline rgb **getPixel** (int x, int y) ........ 132
- inline rgb **setPixel** (int x, int y, rgb color) ........ 132
- bool **writeToPPMFile** (string filename) ........ 132
- bool **readFromPPMFile** (string filename) ........ 132


### Public Members


- rgb* **buffer** ........ 132


### Protected Members


- int **width** ........ 132
- int **height** ........ 132

### Detailed Description

### Member Documentation

**int** `RGB24Image::width` **[protected]**

**int** `RGB24Image::height` **[protected]**

**rgb*** `RGB24Image::buffer`

`RGB24Image::RGB24Image` **()** []

`RGB24Image::RGB24Image` **(int w, int h)** []

`RGB24Image::~RGB24Image` **()** []

**inline unsigned char*** `RGB24Image::rawBuffer` **()** []

**inline unsigned int** `RGB24Image::size` **()** []

**inline int** `RGB24Image::getWidth` **()** []

**inline int** `RGB24Image::getHeight` **()** []

**inline int** `RGB24Image::index` **(int x, int y)** []

**inline rgb** `RGB24Image::getPixel` **(int x, int y)** []

**inline rgb** `RGB24Image::setPixel` **(int x, int y, rgb color)** []

**bool** `RGB24Image::writeToPPMFile` **(string filename)** []

**bool** `RGB24Image::readFromPPMFile` **(string filename)** []

## 87 class RobotControl

Robotin navigoinnista huolehtiva yksikkö.

`#include <RobotControl/RobotControl.h>`

Inherits: SelfTestingObject

*public*

(unknown)

### Public Methods

- bool **invariant** () const ........ 133
- bool **navigate** () ........ 133
- **RobotControl** (InputInterface &pinput, MapInterface &pmap, Movement &pmov, Localization &ploc) 133
- **~RobotControl** () ........ 133

### Private Members

- RobotControlRep * **rep** ........................................................................133

### Detailed Description

Robotin navigoinnista huolehtiva yksikkö. Kutsuus Map-oliota saadakseen seuraavan reittiaskelen. Kutsuu Movement-oliota robotin liikuttamiseksi. Kutsuu Localization-oliota sijainnin tarkistamiseksi.

**See also**: RobotControlRep

### Member Documentation

**bool** `RobotControl::invariant` **()** [**const** ]

Tarkista olion eheys.

**Return**:

olion eheys

**bool** `RobotControl::navigate` **()** []

Navigoinnin käynnistys.

`RobotControl::RobotControl` **(InputInterface &pinput, MapInterface &pmap, Movement &pmov, Localization &ploc)** []

Konstruktori.

**Parameters**:

| | |
|---|---|
| *pinput* | on abstraktin InputInterface-luokan toteuttava olio, joka lukee käyttäjän komentoja |
| *pmap* | on abstraktin MapInterface-luokan toteuttava olio, joka osaa tehdä reitinlaskennan |
| *pmov* | on abstraktin Movement-luokan toteuttava olio, joka liikuttaa robottia |
| *ploc* | on Localization-olio, joka osaa tehdä paikanmäärityksen |

`RobotControl::~RobotControl` **()** []

**RobotControlRep *** `RobotControl::rep` [**private**]

## 88 class RobotControlRep

RobotControl:n privaatti osa.

`#include <RobotControl/RobotControlRep.h>`

Inherits: SelfTestingObject

*public*

(unknown)

## Public Methods


- virtual bool **invariant** () const ....134
- bool **navigate** () ....135
- bool **goOn** () ....135
- Vector **scale** (Vector v, Length max) ....135
- bool **random** (Length l) ....135
- bool **random2** () ....135
- bool **retry** (Vector v, Length& l, Length& l2) ....135
- void **print1** (Pose odometry, Length l) ....135
- void **print2** () ....135
- string **getOdometryError** (Pose p1, Pose p2, Length l) ....135
- **RobotControlRep** (InputInterface &pinput, MapInterface &pmap,Movement &pmov, Localization &ploc) 135
- **~RobotControlRep** () ....135


## Private Members


- InputInterface& **input** ....136
- MapInterface& **map** ....136
- Movement& **mov** ....136
- Localization& **loc** ....136
- Pose **pose** ....136
- Length **etap** ....136
- int **randomVectorCount** ....136
- Length **randomVectorLength** ....136
- int **waitTime** ....136


## Detailed Description

RobotControl:n privaatti osa.

**See also**: RobotControl

## Member Documentation

**bool** `RobotControlRep::invariant` **()** [**const virtual**]

Tarkistaa olion eheyden.

**Return**:

olion eheys

**bool** `RobotControlRep::navigate` **()** []

Navigointi

**Return**:

aloitettiinko navigointi

**bool** `RobotControlRep::goOn` **()** []

**Vector** `RobotControlRep::scale` **(Vector v, Length max)** []

**bool** `RobotControlRep::random` **(Length l)** []

**bool** `RobotControlRep::random2` **()** []

**bool** `RobotControlRep::retry` **(Vector v, Length& l, Length& l2)** []

**void** `RobotControlRep::print1` **(Pose odometry, Length l)** []

**void** `RobotControlRep::print2` **()** []

**string** `RobotControlRep::getOdometryError` **(Pose p1, Pose p2, Length l)** []

Palauttaa ilmoituksen odometriavirheestä

`RobotControlRep::RobotControlRep` **(InputInterface &pinput, MapInterface &pmap,Movement &pmov, Localization &ploc)** []

Konstruktori

**Parameters**:

| | |
|---|---|
| *pinput* | on abstraktin InputInterface-luokan toteuttava olio, joka lukee käyttäjän komentoja |
| *pmap* | on abstraktin MapInterface-luokan toteuttava olio, joka osaa tehdä reitinlaskennan |
| *pmov* | on abstraktin Movement-luokan toteuttava olio, joka liikuttaa robottia |
| *ploc* | on Localization-olio, joka osaa tehdä paikanmäärityksen |

RobotControlRep::~RobotControlRep **()** []

**InputInterface&** RobotControlRep::input [**private**]

**MapInterface&** RobotControlRep::map [**private**]

**Movement&** RobotControlRep::mov [**private**]

**Localization&** RobotControlRep::loc [**private**]

**Pose** RobotControlRep::pose [**private**]

**Length** RobotControlRep::etap [**private**]

**int** RobotControlRep::randomVectorCount [**private**]

**Length** RobotControlRep::randomVectorLength [**private**]

**int** RobotControlRep::waitTime [**private**]

# 89 class STC

Yllättävien poikkeusten käsittelijän alustaja.

`#include <Main/STC.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

## Public Methods


- virtual bool **invariant** () const ....137
- **STC** (unexpected_handler f, std::string message) ....137
- **STC** (const STC& r) ....137
- void **swap** (STC& a) ....137
- STC& **operator=** (const STC& a) ....137
- virtual **~STC** () ....137


## Private Members


- STCRep* **rep** ....137


## Detailed Description

Yllättävien poikkeusten käsittelijän alustaja. Toteuttaa "resurssien varaus on alustamista" menetelmän yllättävän poikkeuksen käsittelijän ja sen tulostaman virheilmoituksen asentamiselle.

### Member Documentation

**bool** `STC::invariant` **() [const virtual]**

Invariantin tarkistus.

**Return**:

true, jos olio on eheässä tilassa, false muuten

`STC::STC` **(unexpected_handler f, std::string message) []**

Konstruktori. Asentaa uuden käsittelijän ja virheilmoituksen, ja tallettaa vanhat.

**Parameters**:

| | |
|---|---|
| *f* | uusi käsittelijä |
| *message* | uusi viesti |

`STC::STC` **(const STC& r) []**

Kopiokonstruktori.

**Parameters**:

| | |
|---|---|
| *r* | kopioitava olio |

**void** `STC::swap` **(STC& a) []**

Vaihto-operaatio.

**Parameters**:

| | |
|---|---|
| *a* | vaihdettava |

**STC&** `STC::operator=` **(const STC& a) []**

Sijoitusoperaattori.

**Parameters**:

| | |
|---|---|
| *a* | sijoitettava |

`STC::~STC` **() [ virtual]**

Destruktori. Palauttaa vanhan käsittelijän ja virheilmoituksen, jotka konstruktori talletti.

**STCRep*** `STC::rep` [**private**]

## 90 class STCRep

Sisäinen esitys poikkeuskäsittelijän alustajalle.

`#include <Main/STCRep.h>`

Inherits: NobelNS::SelfTestingObject

*public*

()

### Public Methods


- bool **invariant** () const ........ 138
- virtual ~**STCRep** () ........ 138


### Public Members


- unexpected_handler **oldHandler** ........ 138
- string **oldMessage** ........ 138


### Detailed Description

Sisäinen esitys poikkeuskäsittelijän alustajalle. Sisältää kentät vanhan käsittelijän ja virheilmoituksen tallentamiseen.

### Member Documentation

**unexpected_handler** `STCRep::oldHandler`

**string** `STCRep::oldMessage`

**bool** `STCRep::invariant` **()** [**const** ]

Invariantin tarkistus.

**Return**:

true.

`STCRep::~STCRep` **()** [ **virtual**]

## 91 class YUV422Image

`#include <Localization/YUV422Image.h>`

### Public Methods


- **YUV422Image** () ........ 140

- **YUV422Image** (int w, int h) .......... 140
- **YUV422Image** (const YUV422Image& y) .......... 140
- **~YUV422Image** () .......... 140
- inline unsigned char* **rawBuffer** () .......... 140
- inline unsigned int **size** () .......... 140
- inline unsigned int **getWidth** () .......... 140
- inline unsigned int **getHeight** () .......... 140
- inline int **index** (int x, int y) .......... 140
- inline yuv **getYUVPixel** (int x, int y) .......... 140
- bool **writeToUYVYFile** (string prefix) .......... 140
- bool **readFromUYVYFile** (string filename, int w, int h) .......... 140

### Public Members

- yuv422* **buffer** .......... 140

### Protected Members

- int **width** .......... 140
- int **height** .......... 140

### Detailed Description

### Member Documentation

**int** `YUV422Image::width` [protected]

**int** `YUV422Image::height` [protected]

**yuv422*** `YUV422Image::buffer`

`YUV422Image::YUV422Image` **()** []

`YUV422Image::YUV422Image` **(int w, int h)** []

`YUV422Image::YUV422Image` **(const YUV422Image& y)** []

`YUV422Image::~YUV422Image` **()** []

**inline unsigned char*** `YUV422Image::rawBuffer` **()** []

**inline unsigned int** `YUV422Image::size` **()** []

**inline unsigned int** `YUV422Image::getWidth` **()** []

**inline unsigned int** `YUV422Image::getHeight` **()** []

**inline int** `YUV422Image::index` **(int x, int y)** []

**inline yuv** `YUV422Image::getYUVPixel` **(int x, int y)** []

**bool** `YUV422Image::writeToUYVYFile` **(string prefix)** []

**bool** `YUV422Image::readFromUYVYFile` **(string filename, int w, int h)** []

## 92 struct poly_vertex

`#include <rmap/poly.h>`

### Public Members


- int **x** ........ 141
- int **y** ........ 141
- struct poly_vertex * **next** ........ 141

### Detailed Description

### Member Documentation

**int** `poly_vertex::x`

**int** `poly_vertex::y`

**struct poly_vertex *** `poly_vertex::next`

## 93 struct polygon

`#include <rmap/poly.h>`

### Public Members


- int **num_vertices** ........ 141
- struct poly_vertex * **vertex1** ........ 141
- struct polygon * **next** ........ 141


### Detailed Description

### Member Documentation

**int** `polygon::num_vertices`

**struct poly_vertex *** `polygon::vertex1`

**struct polygon *** `polygon::next`

## 94 struct rgb

`#include <Localization/helvision.h>`

### Public Members


- unsigned char **red** ........ 142
- unsigned char **green** ........ 142
- unsigned char **blue** ........ 142

### Detailed Description

### Member Documentation

**unsigned char** `rgb::red`

**unsigned char** `rgb::green`

**unsigned char** `rgb::blue`

## 95 struct rgb24

`#include <Localization/RGB24Image.h>`

### Public Members

- unsigned char **red** ........ 142
- unsigned char **green** ........ 142
- unsigned char **blue** ........ 142

### Detailed Description

### Member Documentation

**unsigned char** `rgb24::red`

**unsigned char** `rgb24::green`

**unsigned char** `rgb24::blue`

## 96 struct yuv

`#include <Localization/YUV422Image.h>`

### Public Members

- unsigned char **y** ........ 143
- unsigned char **u** ........ 143
- unsigned char **v** ........ 143

### Detailed Description

### Member Documentation

**unsigned char** `yuv::y`

**unsigned char** `yuv::u`

**unsigned char** `yuv::v`

## 97 struct yuv422

`#include <Localization/YUV422Image.h>`

### Public Members


- unsigned char **y1** ........ 143
- unsigned char **u** ........ 143
- unsigned char **y2** ........ 143
- unsigned char **v** ........ 143
- unsigned char **y1** ........ 143
- unsigned char **u** ........ 143
- unsigned char **y2** ........ 143
- unsigned char **v** ........ 143


### Detailed Description

### Member Documentation

**unsigned char** `yuv422::y1`

**unsigned char** `yuv422::u`

**unsigned char** `yuv422::y2`

**unsigned char** `yuv422::v`

**unsigned char** `yuv422::y1`

**unsigned char** `yuv422::u`

**unsigned char** `yuv422::y2`

**unsigned char** `yuv422::v`