

RobCodeGenerator

Erzeugt von Doxygen 1.9.7

1 Beschreibung Roboter Path Editor	1
1.1 Nutzen	1
1.2 Aufbau	1
1.2.1 Logging	1
1.2.2 Daten einlesen	1
1.2.3 Daten verarbeiten	1
1.2.4 Roboter Code erstellen	1
2 Verzeichnis der Namensbereiche	3
2.1 Liste aller Namensbereiche	3
3 Hierarchie-Verzeichnis	5
3.1 Klassenhierarchie	5
4 Klassen-Verzeichnis	7
4.1 Auflistung der Klassen	7
5 Datei-Verzeichnis	9
5.1 Auflistung der Dateien	9
6 Dokumentation der Namensbereiche	11
6.1 QT_WARNING_DISABLE_DEPRECATED-Namensbereichsreferenz	11
6.2 Ui-Namensbereichsreferenz	11
7 Klassen-Dokumentation	13
7.1 CEulerMatrix Klassenreferenz	13
7.1.1 Ausführliche Beschreibung	13
7.1.2 Beschreibung der Konstruktoren und Destruktoren	14
7.1.2.1 CEulerMatrix() [1/2]	14
7.1.2.2 CEulerMatrix() [2/2]	14
7.1.2.3 ~CEulerMatrix()	15
7.1.3 Dokumentation der Elementfunktionen	15
7.1.3.1 angels2mat()	15
7.1.3.2 calculateAngels()	15
7.1.3.3 getEulerMatrix()	16
7.1.3.4 getMatrix()	17
7.1.3.5 setMatrix()	18
7.1.4 Dokumentation der Datenelemente	18
7.1.4.1 eulerMatrix	18
7.2 CInputParameter Klassenreferenz	19
7.2.1 Ausführliche Beschreibung	20
7.2.2 Beschreibung der Konstruktoren und Destruktoren	20
7.2.2.1 CInputParameter() [1/2]	20
7.2.2.2 CInputParameter() [2/2]	21

7.2.2.3 ~CInputParameter()	22
7.2.3 Dokumentation der Elementfunktionen	22
7.2.3.1 detectJump()	22
7.2.3.2 getAngles()	23
7.2.3.3 getLoggingManual()	23
7.2.3.4 getOffset()	24
7.2.3.5 getOffsetManual()	24
7.2.3.6 getOrientationManual()	24
7.2.3.7 getPath()	25
7.2.3.8 getSpeed()	25
7.2.3.9 getSpeedManual()	25
7.2.3.10 openFile()	25
7.2.3.11 setLogging()	26
7.2.3.12 setOffset()	27
7.2.3.13 setOrientation()	27
7.2.3.14 setSpeed()	28
7.2.4 Dokumentation der Datenelemente	28
7.2.4.1 A	28
7.2.4.2 B	28
7.2.4.3 C	29
7.2.4.4 difference	29
7.2.4.5 initialPath	29
7.2.4.6 loggingManual	29
7.2.4.7 offsetManual	29
7.2.4.8 offsetX	30
7.2.4.9 offsetY	30
7.2.4.10 offsetZ	30
7.2.4.11 orientationManual	30
7.2.4.12 speed	30
7.2.4.13 speedManual	31
7.3 CInputPoint3D Klassenreferenz	31
7.3.1 Ausführliche Beschreibung	32
7.3.2 Beschreibung der Konstruktoren und Destruktoren	33
7.3.2.1 CInputPoint3D() [1/2]	33
7.3.2.2 CInputPoint3D() [2/2]	33
7.3.2.3 ~CInputPoint3D()	34
7.3.3 Dokumentation der Elementfunktionen	34
7.3.3.1 getEulerMatrix()	34
7.3.3.2 getTime()	34
7.3.3.3 setEulerMatrix()	34
7.3.3.4 setPoint()	35
7.3.3.5 setTime()	35

7.3.4 Dokumentation der Datenelemente	36
7.3.4.1 orientationMatrix	36
7.3.4.2 timestamp	36
7.4 CLine3D Klassenreferenz	36
7.4.1 Ausführliche Beschreibung	37
7.4.2 Beschreibung der Konstruktoren und Destruktoren	37
7.4.2.1 CLine3D() [1/2]	37
7.4.2.2 CLine3D() [2/2]	37
7.4.2.3 ~CLine3D()	38
7.4.3 Dokumentation der Datenelemente	38
7.4.3.1 p1	38
7.4.3.2 p2	38
7.5 CLogging Klassenreferenz	38
7.5.1 Ausführliche Beschreibung	39
7.5.2 Beschreibung der Konstruktoren und Destruktoren	39
7.5.2.1 CLogging() [1/2]	39
7.5.2.2 CLogging() [2/2]	40
7.5.2.3 ~CLogging()	40
7.5.3 Dokumentation der Elementfunktionen	40
7.5.3.1 getDetailed()	40
7.5.3.2 logData() [1/3]	40
7.5.3.3 logData() [2/3]	41
7.5.3.4 logData() [3/3]	42
7.5.3.5 setStep()	42
7.5.4 Dokumentation der Datenelemente	43
7.5.4.1 detailed	43
7.5.4.2 path	43
7.5.4.3 step	43
7.6 CMeanFilter Klassenreferenz	44
7.6.1 Ausführliche Beschreibung	44
7.6.2 Beschreibung der Konstruktoren und Destruktoren	44
7.6.2.1 CMeanFilter() [1/2]	44
7.6.2.2 CMeanFilter() [2/2]	45
7.6.2.3 ~CMeanFilter()	45
7.6.3 Dokumentation der Elementfunktionen	45
7.6.3.1 calculateMean()	45
7.6.3.2 getPath()	46
7.6.3.3 getWindowSize()	46
7.6.3.4 mean()	47
7.6.3.5 setWindowSize()	47
7.6.4 Dokumentation der Datenelemente	48
7.6.4.1 meanPath	48

7.6.4.2 windowSize	48
7.7 COutputPoint3D Klassenreferenz	48
7.7.1 Ausführliche Beschreibung	50
7.7.2 Beschreibung der Konstruktoren und Destruktoren	50
7.7.2.1 COutputPoint3D() [1/2]	50
7.7.2.2 COutputPoint3D() [2/2]	50
7.7.2.3 ~COutputPoint3D()	51
7.7.3 Dokumentation der Elementfunktionen	51
7.7.3.1 getA()	51
7.7.3.2 getB()	52
7.7.3.3 getC()	52
7.7.3.4 getSpeed()	52
7.7.3.5 setA()	52
7.7.3.6 setB()	53
7.7.3.7 setC()	53
7.7.3.8 setSpeed()	54
7.7.4 Dokumentation der Datenelemente	54
7.7.4.1 a	54
7.7.4.2 b	54
7.7.4.3 c	54
7.7.4.4 speed	55
7.8 CPathBuilder Klassenreferenz	55
7.8.1 Ausführliche Beschreibung	55
7.8.2 Beschreibung der Konstruktoren und Destruktoren	56
7.8.2.1 CPathBuilder()	56
7.8.2.2 ~CPathBuilder()	56
7.8.3 Dokumentation der Elementfunktionen	56
7.8.3.1 createPath()	56
7.8.3.2 getPath()	57
7.8.4 Dokumentation der Datenelemente	57
7.8.4.1 path	57
7.9 CPoint3D Klassenreferenz	58
7.9.1 Ausführliche Beschreibung	59
7.9.2 Beschreibung der Konstruktoren und Destruktoren	59
7.9.2.1 CPoint3D() [1/2]	59
7.9.2.2 CPoint3D() [2/2]	59
7.9.2.3 ~CPoint3D()	60
7.9.3 Dokumentation der Elementfunktionen	60
7.9.3.1 distanceTo() [1/2]	60
7.9.3.2 distanceTo() [2/2]	61
7.9.3.3 getX()	61
7.9.3.4 getY()	62

7.9.3.5 getZ()	62
7.9.3.6 set()	62
7.9.3.7 setX()	63
7.9.3.8 setY()	63
7.9.3.9 setZ()	63
7.9.4 Dokumentation der Datenelemente	65
7.9.4.1 x	65
7.9.4.2 y	65
7.9.4.3 z	65
7.10 CRobCodeGenerator Klassenreferenz	66
7.10.1 Ausführliche Beschreibung	66
7.10.2 Beschreibung der Konstruktoren und Destruktoren	66
7.10.2.1 CRobCodeGenerator() [1/2]	66
7.10.2.2 CRobCodeGenerator() [2/2]	67
7.10.2.3 ~CRobCodeGenerator()	67
7.10.3 Dokumentation der Elementfunktionen	67
7.10.3.1 calculateAngles()	67
7.10.3.2 calculateSpeed()	68
7.10.3.3 generateRobCode()	69
7.10.3.4 postProcessing()	70
7.10.4 Dokumentation der Datenelemente	70
7.10.4.1 input	70
7.10.4.2 processedPath	71
7.11 CSegmentApproximator Klassenreferenz	71
7.11.1 Ausführliche Beschreibung	72
7.11.2 Beschreibung der Konstruktoren und Destruktoren	72
7.11.2.1 CSegmentApproximator()	72
7.11.2.2 ~CSegmentApproximator()	72
7.11.3 Dokumentation der Elementfunktionen	72
7.11.3.1 approx()	72
7.11.3.2 douglasPeuckerRecursive()	73
7.11.3.3 getMaxDistance()	74
7.11.3.4 getSegmentsApproxVector()	74
7.11.3.5 setmaxDistance()	75
7.11.4 Dokumentation der Datenelemente	75
7.11.4.1 maxDistance	75
7.11.4.2 segmentsApprox	75
7.12 GUI Klassenreferenz	76
7.12.1 Ausführliche Beschreibung	77
7.12.2 Beschreibung der Konstruktoren und Destruktoren	77
7.12.2.1 GUI()	77
7.12.2.2 ~GUI()	78

7.12.3 Dokumentation der Elementfunktionen	78
7.12.3.1 activateLogging	78
7.12.3.2 activateOffset	79
7.12.3.3 activateOrientation	79
7.12.3.4 activateSpeed	80
7.12.3.5 calculate	80
7.12.3.6 setDP	81
7.12.3.7 setInputPath	81
7.12.3.8 setMean	82
7.12.3.9 setOffset	82
7.12.3.10 setOrientation	82
7.12.3.11 setOutputPath	83
7.12.3.12 setSpeed	83
7.12.4 Dokumentation der Datenelemente	83
7.12.4.1 dpTolerance	83
7.12.4.2 inputParameter	83
7.12.4.3 inputPathUI	84
7.12.4.4 meanLength	84
7.12.4.5 outputPathUI	84
7.12.4.6 ui	84
7.13 Ui::GUIClass Klassenreferenz	85
7.13.1 Ausführliche Beschreibung	86
7.14 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t Struktur- referenz	86
7.14.1 Ausführliche Beschreibung	86
7.14.2 Dokumentation der Datenelemente	86
7.14.2.1 offsetsAndSizes	86
7.14.2.2 stringdata0	86
7.14.2.3 stringdata1	87
7.14.2.4 stringdata10	87
7.14.2.5 stringdata11	87
7.14.2.6 stringdata12	87
7.14.2.7 stringdata13	87
7.14.2.8 stringdata2	87
7.14.2.9 stringdata3	87
7.14.2.10 stringdata4	87
7.14.2.11 stringdata5	88
7.14.2.12 stringdata6	88
7.14.2.13 stringdata7	88
7.14.2.14 stringdata8	88
7.14.2.15 stringdata9	88
7.15 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSwitchENDCLASS_t Struk- turereferenz	88

7.15.1 Ausführliche Beschreibung	89
7.15.2 Dokumentation der Datenelemente	89
7.15.2.1 offsetsAndSizes	89
7.15.2.2 stringdata0	89
7.15.2.3 stringdata1	89
7.15.2.4 stringdata2	89
7.16 Ui_GUIClass Klassenreferenz	89
7.16.1 Ausführliche Beschreibung	90
7.16.2 Dokumentation der Elementfunktionen	91
7.16.2.1 retranslateUi() [1/2]	91
7.16.2.2 retranslateUi() [2/2]	91
7.16.2.3 setupUi() [1/2]	92
7.16.2.4 setupUi() [2/2]	95
7.16.3 Dokumentation der Datenelemente	98
7.16.3.1 AValue	98
7.16.3.2 bLogging	98
7.16.3.3 bManOrientation	99
7.16.3.4 bOffset	99
7.16.3.5 bSpeed	99
7.16.3.6 BValue	99
7.16.3.7 centralWidget	99
7.16.3.8 CValue	99
7.16.3.9 dpToleranz	100
7.16.3.10 frame	100
7.16.3.11 label_10	100
7.16.3.12 label_11	100
7.16.3.13 label_12	100
7.16.3.14 label_13	100
7.16.3.15 label_14	101
7.16.3.16 label_15	101
7.16.3.17 label_4	101
7.16.3.18 label_5	101
7.16.3.19 label_dp	101
7.16.3.20 meanLength	101
7.16.3.21 offset	102
7.16.3.22 offsetX	102
7.16.3.23 offsetY	102
7.16.3.24 offsetZ	102
7.16.3.25 orientation	102
7.16.3.26 pathInput	102
7.16.3.27 pathOutput	103
7.16.3.28 pushInput	103

7.16.3.29 pushOutput	103
7.16.3.30 speed	103
7.16.3.31 speed_2	103
7.16.3.32 startCalculation	103
7.16.3.33 textBrowser	103
8 Datei-Dokumentation	105
8.1 header/EulerMatrix.h-Dateireferenz	105
8.1.1 Ausführliche Beschreibung	105
8.1.2 Makro-Dokumentation	105
8.1.2.1 _USE_MATH_DEFINES	105
8.2 EulerMatrix.h	106
8.3 header/GUI.h-Dateireferenz	106
8.3.1 Ausführliche Beschreibung	106
8.4 GUI.h	107
8.5 header/InputParameter.h-Dateireferenz	107
8.5.1 Ausführliche Beschreibung	108
8.6 InputParameter.h	108
8.7 header/Line3D.h-Dateireferenz	108
8.7.1 Ausführliche Beschreibung	109
8.8 Line3D.h	109
8.9 header/Logging.h-Dateireferenz	109
8.9.1 Ausführliche Beschreibung	110
8.10 Logging.h	110
8.11 header/MeanFilter.h-Dateireferenz	110
8.11.1 Ausführliche Beschreibung	110
8.12 MeanFilter.h	111
8.13 header/PathBuilder.h-Dateireferenz	111
8.13.1 Ausführliche Beschreibung	111
8.14 PathBuilder.h	112
8.15 header/Point3D.h-Dateireferenz	112
8.15.1 Ausführliche Beschreibung	112
8.16 Point3D.h	113
8.17 header/RobCodeGenerator.h-Dateireferenz	113
8.17.1 Ausführliche Beschreibung	114
8.17.2 Makro-Dokumentation	114
8.17.2.1 MAX_SPEED	114
8.18 RobCodeGenerator.h	114
8.19 header/SegmentApproximator.h-Dateireferenz	115
8.19.1 Ausführliche Beschreibung	115
8.20 SegmentApproximator.h	115
8.21 source/EulerMatrix.cpp-Dateireferenz	116

8.22 EulerMatrix.cpp	116
8.23 source/GUI.cpp-Dateireferenz	117
8.23.1 Ausführliche Beschreibung	117
8.24 GUI.cpp	118
8.25 source/InputParameter.cpp-Dateireferenz	120
8.25.1 Ausführliche Beschreibung	121
8.26 InputParameter.cpp	121
8.27 source/Line3D.cpp-Dateireferenz	123
8.27.1 Ausführliche Beschreibung	123
8.28 Line3D.cpp	123
8.29 source/Logging.cpp-Dateireferenz	123
8.29.1 Ausführliche Beschreibung	124
8.30 Logging.cpp	124
8.31 source/MeanFilter.cpp-Dateireferenz	125
8.31.1 Ausführliche Beschreibung	125
8.32 MeanFilter.cpp	126
8.33 source/PathBuilder.cpp-Dateireferenz	127
8.33.1 Ausführliche Beschreibung	127
8.34 PathBuilder.cpp	127
8.35 source/Point3D.cpp-Dateireferenz	128
8.35.1 Ausführliche Beschreibung	128
8.36 Point3D.cpp	128
8.37 source/RobCodeGenerator.cpp-Dateireferenz	131
8.37.1 Ausführliche Beschreibung	131
8.38 RobCodeGenerator.cpp	131
8.39 source/RobPathEditor.cpp-Dateireferenz	133
8.39.1 Ausführliche Beschreibung	133
8.39.2 Dokumentation der Funktionen	133
8.39.2.1 main()	133
8.40 RobPathEditor.cpp	133
8.41 source/SegmentApproximator.cpp-Dateireferenz	134
8.41.1 Ausführliche Beschreibung	134
8.42 SegmentApproximator.cpp	134
8.43 x64/Debug/moc/moc_GUI.cpp-Dateireferenz	135
8.43.1 Makro-Dokumentation	136
8.43.1.1 Q_CONSTINIT	136
8.43.1.2 QT_MOC_LITERAL	136
8.44 moc_GUI.cpp	136
8.45 x64/Release/moc/moc_GUI.cpp-Dateireferenz	139
8.45.1 Makro-Dokumentation	139
8.45.1.1 Q_CONSTINIT	139
8.45.1.2 QT_MOC_LITERAL	140

8.46 moc_GUI.cpp	140
8.47 x64/Debug/moc/moc_switch.cpp-Dateireferenz	143
8.47.1 Makro-Dokumentation	143
8.47.1.1 Q_CONSTINIT	143
8.47.1.2 QT_MOC_LITERAL	143
8.48 moc_switch.cpp	144
8.49 x64/Release/moc/moc_switch.cpp-Dateireferenz	146
8.49.1 Makro-Dokumentation	146
8.49.1.1 Q_CONSTINIT	146
8.49.1.2 QT_MOC_LITERAL	146
8.50 moc_switch.cpp	147
8.51 x64/Debug/rcc/qrc_GUI.cpp-Dateireferenz	149
8.51.1 Makro-Dokumentation	149
8.51.1.1 QT_RCC_MANGLE_NAMESPACE	149
8.51.1.2 QT_RCC_PREPEND_NAMESPACE	149
8.51.2 Dokumentation der Funktionen	149
8.51.2.1 qCleanupResources_GUI()	149
8.51.2.2 qInitResources_GUI()	149
8.52 qrc_GUI.cpp	150
8.53 x64/Release/rcc/qrc_GUI.cpp-Dateireferenz	150
8.53.1 Makro-Dokumentation	151
8.53.1.1 QT_RCC_MANGLE_NAMESPACE	151
8.53.1.2 QT_RCC_PREPEND_NAMESPACE	151
8.53.2 Dokumentation der Funktionen	151
8.53.2.1 qCleanupResources_GUI()	151
8.53.2.2 qInitResources_GUI()	151
8.54 qrc_GUI.cpp	151
8.55 x64/Debug/uic/ui_GUI.h-Dateireferenz	152
8.56 ui_GUI.h	153
8.57 x64/Release/uic/ui_GUI.h-Dateireferenz	157
8.58 ui_GUI.h	157

Kapitel 1

Beschreibung Roboter Path Editor

1.1 Nutzen

Mit diesem Programm sollen haendisch aufgenommene Pfad Daten einer Roboterbewegung zu einem fuer Kuka Roboter lesbaren File gemacht werden. Zusaetzlich soll einstellbar sein ob die Orientierung berechnet werden soll, oder eingegeben werden soll. Das selbe gilt fuer Geschwindigkeitsdaten.

1.2 Aufbau

In der Grundidee werden die eingelesenen Daten immer aus der vorhergegangenen Klasse ausgelesen und nach der Verarbeitung in der aktuellen Klasse gespeichert.

1.2.1 Logging

Zuerst wird die Loggingklasse [CLogging](#) initialisiert. In ihr wird gespeichert in welchem Schritt das Programm gerade ist. Dieser Klasse wird ein Pfad uebergeben an welchem die Daten gespeichert werden sollen.

1.2.2 Daten einlesen

Als naechstes werden die Nutzerdaten eingelesen und anschliessend die aufgenommenen Daten eingelesen. Dabei wird ueberprueft ob es sich um einen zusammenhaengenden Pfad handelt. Das passiert in der Klasse [CInputParameter](#).

1.2.3 Daten verarbeiten

In mehreren Schritten folgt eine Nachbearbeitung der Daten. Zuerst werden die Daten mit einem gleitendem Mittelwertfilter in der Klasse [CMeanFilter](#) geglaettet. Anschliessend werden Punkte mit Hilfe des Douglas-Peucker Algorithmuses in der Klasse [CSegmentApproximator](#) geloescht. Sollten es mehrere nicht zusammenhaengende Pfade sein muessen diese jetzt noch zusammengesetzt werden.

1.2.4 Roboter Code erstellen

Als letzter Schritt werden die Nutzereinstellungen in die Daten uebernommen und der Robotercode erstellt.

Kapitel 2

Verzeichnis der Namensbereiche

2.1 Liste aller Namensbereiche

Liste aller Namensbereiche mit Kurzbeschreibung:

QT_WARNING_DISABLE_DEPRECATED	11
Ui	11

Kapitel 3

Hierarchie-Verzeichnis

3.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

CEulerMatrix	13
CInputParameter	19
CLine3D	36
CLogging	38
CMeanFilter	44
CPathBuilder	55
CPoint3D	58
CInputPoint3D	31
COutputPoint3D	48
CRobCodeGenerator	66
CSegmentApproximator	71
QMainWindow	
GUI	76
QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t	86
QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSwitchENDCLASS_t	88
Ui_GUIClass	89
Ui::GUIClass	85

Kapitel 4

Klassen-Verzeichnis

4.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

CEulerMatrix	
Handling und Berechnung Euler Matrix	13
CInputParameter	
Handling Eingabedaten	19
CInputPoint3D	
Input Punkt	31
CLine3D	
Berechnung Geraden	36
CLogging	
Gleitender Mittelwertfilter	38
CMeanFilter	
Gleitender Mittelwertfilter	44
COutputPoint3D	
Output Punkt	48
CPathBuilder	
Zusammensetzen des Pfades	55
CPoint3D	
Grundklasse Punkt	58
CRobCodeGenerator	
Klasse zum erstellen des Roboter Codes	66
CSegmentApproximator	
Ausduennen des Pfades	71
GUI	
UI und Funktionen	76
Ui::GUIClass	85
QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t	86
QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSwitchENDCLASS_t	88
Ui_GUIClass	89

Kapitel 5

Datei-Verzeichnis

5.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

header/ EulerMatrix.h	
Header File handling Euler Matrix	105
header/ GUI.h	
Header File handling the User Interface	106
header/ InputParameter.h	
Header File Daten Einlesen	107
header/ Line3D.h	
Header File Daten Einlesen	108
header/ Logging.h	
Logging der Daten	109
header/ MeanFilter.h	
Berechnung des gleitenden Mittelwertfilters	110
header/ PathBuilder.h	
Setzt die einzelnen Segmente zu einem Vector zusammen	111
header/ Point3D.h	
Verarbeitung der Punkte	112
header/ RobCodeGenerator.h	
Erstellung des Roboter Codes	113
header/ SegmentApproximator.h	
Berechnung des Douglas Peuker Algorithmusses	115
source/ EulerMatrix.cpp	116
source/ GUI.cpp	
Source File User Interface	117
source/ InputParameter.cpp	
Source File Daten Einlesen	120
source/ Line3D.cpp	
Source File Line3D	123
source/ Logging.cpp	
Source File Logging	123
source/ MeanFilter.cpp	
Source File gleitender Mittelwertfilter	125
source/ PathBuilder.cpp	
Source File Segmente zu Pfad	127
source/ Point3D.cpp	
Source File Punkte	128

source/ RobCodeGenerator.cpp	
Source File Roboter Code Erstellung	131
source/ RobPathEditor.cpp	
Hier wird die main Funktion aufgerufen	133
source/ SegmentApproximator.cpp	
Source File Douglas-Peuker	134
x64/Debug/moc/ moc_GUI.cpp	135
x64/Debug/moc/ moc_switch.cpp	143
x64/Debug/rcc/ qrc_GUI.cpp	149
x64/Debug/uic/ ui_GUI.h	152
x64/Release/moc/ moc_GUI.cpp	139
x64/Release/moc/ moc_switch.cpp	146
x64/Release/rcc/ qrc_GUI.cpp	150
x64/Release/uic/ ui_GUI.h	157

Kapitel 6

Dokumentation der Namensbereiche

6.1 QT_WARNING_DISABLE_DEPRECATED-Namensbereichsreferenz

Klassen

- struct [qt_meta_stringdata_CLASSGUIENDCLASS_t](#)
- struct [qt_meta_stringdata_CLASSSwitchENDCLASS_t](#)

6.2 Ui-Namensbereichsreferenz

Klassen

- class [GUIClass](#)

Kapitel 7

Klassen-Dokumentation

7.1 CEulerMatrix Klassenreferenz

Handling und Berechnung Euler Matrix.

```
#include <EulerMatrix.h>
```

Öffentliche Methoden

- [CEulerMatrix](#) (void)
Default Konstruktor.
- [CEulerMatrix](#) (float inputMatrix[3][3])
Default Konstruktor.
- [~CEulerMatrix](#) ()
Dekonstruktor.
- void [setMatrix](#) (float inputMatrix[3][3])
Setzt eine Matrix.
- [CEulerMatrix](#) [getEulerMatrix](#) (void)
Auslesen eine Matrix.
- void [getMatrix](#) (float Matrix[][3])
Auslesen eine Matrix.
- [CEulerMatrix](#) [angels2mat](#) (double A, double B, double C)
Berechnet die neue Umdrehungsmatrix.
- tuple< double, double, double > [calculateAngels](#) (void)
Berechnet die Kuka Winkel A,B,C.

Private Attribute

- float [eulerMatrix](#) [3][3]

7.1.1 Ausführliche Beschreibung

Handling und Berechnung Euler Matrix.

Diese Klasse speichert die Euler Matrix und hat Funktionen fuer Berechnungen mit eben jener.

Definiert in Zeile [19](#) der Datei [EulerMatrix.h](#).

7.1.2 Beschreibung der Konstruktoren und Destruktoren

7.1.2.1 CEulerMatrix() [1/2]

```
CEulerMatrix::CEulerMatrix (
    void )
```

Default Konstruktor.

Initialisiert die Input Daten mit Null

Siehe auch

[CEulerMatrix\(float inputMatrix\[3\]\[3\]\)](#)

Definiert in Zeile 4 der Datei [EulerMatrix.cpp](#).

```
00005 {
00006     for (int i = 0; i < 3; i++)
00007     {
00008         for (int m = 0; m < 3; m++)
00009         {
00010             eulerMatrix[i][m] = 0;           // eulerMatrix mit 0 initialisieren
00011         }
00012     }
00013 }
```

Benutzt [eulerMatrix](#).

7.1.2.2 CEulerMatrix() [2/2]

```
CEulerMatrix::CEulerMatrix (
    float inputMatrix[3][3] )
```

Default Konstruktor.

Initialisiert die Input Daten mit Null

Parameter

<i>float</i>	inputMatrix[3][3] initialisiert die Klasse mit einer Euler Matrix
--------------	---

Siehe auch

[CEulerMatrix\(void\)](#)

Definiert in Zeile 15 der Datei [EulerMatrix.cpp](#).

```
00016 {
00017     for (int i = 0; i < 3; i++)
00018     {
00019         for (int m = 0; m < 3; m++)
00020         {
00021             eulerMatrix[i][m] = inputMatrix[i][m]; // eulerMatrix mit Startwerten initialisieren
00022         }
00023     }
00024 }
```

Benutzt [eulerMatrix](#).

7.1.2.3 ~CEulerMatrix()

```
CEulerMatrix::~CEulerMatrix ( )
```

Dekonstruktor.

Definiert in Zeile 26 der Datei [EulerMatrix.cpp](#).

```
00027 {
00028 }
```

7.1.3 Dokumentation der Elementfunktionen

7.1.3.1 angels2mat()

```
CEulerMatrix CEulerMatrix::angels2mat (
    double A,
    double B,
    double C )
```

Berechnet die neue Umdrehungsmatrix.

Parameter

<i>A</i>	double Winkel a
<i>B</i>	double Winkel b
<i>C</i>	double Winkel c

Rückgabe

: float inputMatrix[3][3] gibt die neu berechnete Matrix zurück

Definiert in Zeile 59 der Datei [EulerMatrix.cpp](#).

```
00060 {
00061     float Matrix[3][3];        // DummyMatrix erstellen
00062
00063     /* Berechnung der Matrix */
00064
00065     Matrix[0][0] = cos(A) * cos(C) - sin(A) * cos(B) * sin(C);
00066     Matrix[0][1] = -cos(A) * sin(C) - sin(A) * cos(B) * cos(C);
00067     Matrix[0][2] = sin(A) * sin(B);
00068
00069     Matrix[1][0] = sin(A) * cos(C) + cos(A) * cos(B) * sin(C);
00070     Matrix[1][1] = -sin(A) * sin(C) + cos(A) * cos(B) * cos(C);
00071     Matrix[1][2] = -cos(A) * sin(B);
00072
00073     Matrix[2][0] = sin(B) * sin(C);
00074     Matrix[2][1] = sin(B) * cos(C);
00075     Matrix[2][2] = cos(B);
00076
00077     CEulerMatrix buffer(Matrix);        // DummyMatrix in DummyEulerMatrix schreiben
00078     return buffer;                      // Matrix zurueck geben
00079 }
```

7.1.3.2 calculateAngels()

```
tuple< double, double, double > CEulerMatrix::calculateAngels (
    void )
```

Berechnet die Kuka Wunkel A,B,C.

Rückgabe

: tuple<double , double , double> gibt die berechneten Winkel A, B, C zurueck

Definiert in Zeile 82 der Datei [EulerMatrix.cpp](#).

```
00083 {
00084     double a, b, c, sin_a, cos_a, sin_b, abs_cos_b, sin_c, cos_c;
00085
00086     /*
00087     a == Winkel Alpha
00088     b == Winkel Beta
00089     c == Winkel Gamma
00090
00091     sin_a == sinus alpha
00092     cos_a == cosinus alpha
00093     sin_b == Matrix[2][0] * -1
00094     abs_cos_b == ??
00095     sin_c == sinus gamma
00096     cos_c == cosinus gamma
00097     */
00098
00099
00100     /* Berechnung von alpha*/
00101     a = atan2(eulerMatrix[1][0], eulerMatrix[0][0]);
00102
00103     /* Berechnung von beta */
00104     sin_a = sin(a);
00105     cos_a = cos(a);
00106     sin_b = eulerMatrix[2][0] * -1;
00107     abs_cos_b = cos(a) * eulerMatrix[0][0] + sin(a) * eulerMatrix[1][0];
00108
00109     b = atan2 (sin_b, abs_cos_b);
00110
00111     /* Berechnung von gamma */
00112     sin_c = sin_a * eulerMatrix[0][2] - cos_a * eulerMatrix[1][2];
00113     cos_c = -sin_a * eulerMatrix[0][1] + cos_a * eulerMatrix[1][1];
00114
00115     c = atan2(sin_c, cos_c);
00116
00117     /* Bogenmass in Gradmass umrechnen */
00118     a = a * 180 / M_PI;
00119     b = b * 180 / M_PI;
00120     c = c * 180 / M_PI;
00121
00122
00123     return make_tuple(a, b, c);    // Rueckgabe der Winkel
00124 }
```

Benutzt [eulerMatrix](#).

Wird benutzt von [CRobCodeGenerator::calculateAngles\(\)](#).

7.1.3.3 getEulerMatrix()

```
CEulerMatrix CEulerMatrix::getEulerMatrix (
    void )
```

Auslesen eine Matrix.

Rückgabe

: float inputMatrix[3][3] gibt gespeicherte Matrix zurueck

Definiert in Zeile 42 der Datei [EulerMatrix.cpp](#).

```
00043 {
00044     return eulerMatrix;    // EulerMatrix zurueck geben
00045 }
```

Benutzt [eulerMatrix](#).

Wird benutzt von [CLogging::logData\(\)](#) und [CInputParameter::openFile\(\)](#).

7.1.3.4 getMatrix()

```
void CEulerMatrix::getMatrix (
    float Matrix[][3] )
```

Auslesen eine Matrix.

Parameter

<i>float*</i>	inputMatrix[3][3] Pointer zu einer Matrix
---------------	---

Definiert in Zeile 47 der Datei [EulerMatrix.cpp](#).

```
00048 {
00049     for (int i = 0; i < 3; i++)
00050     {
00051         for (int m = 0; m < 3; m++)
00052         {
00053             Matrix[i][m] = eulerMatrix[i][m]; // eulerMatrix mit Uebergabewerten ueberschreiben
00054         }
00055     }
00056 }
```

Benutzt [eulerMatrix](#).

Wird benutzt von [CLogging::logData\(\)](#), [CLogging::logData\(\)](#) und [CLogging::logData\(\)](#).

7.1.3.5 setMatrix()

```
void CEulerMatrix::setMatrix (
    float inputMatrix[3][3] )
```

Setzt eine Matrix.

Parameter

<i>float</i>	inputMatrix[3][3] zum setzten einer Matrix
--------------	--

Definiert in Zeile 31 der Datei [EulerMatrix.cpp](#).

```
00032 {
00033     for (int i = 0; i < 3; i++)
00034     {
00035         for (int m = 0; m < 3; m++)
00036         {
00037             eulerMatrix[i][m] = inputMatrix[i][m]; // eulerMatrix mit Uebergabewerten ueberschreiben
00038         }
00039     }
00040 }
```

Benutzt [eulerMatrix](#).

Wird benutzt von [CInputParameter::openFile\(\)](#).

7.1.4 Dokumentation der Datenelemente**7.1.4.1 eulerMatrix**

```
float CEulerMatrix::eulerMatrix[3][3] [private]
```

Gespeicherte Euler Matrix

Definiert in Zeile 75 der Datei [EulerMatrix.h](#).

Wird benutzt von [calculateAngels\(\)](#), [CEulerMatrix\(\)](#), [CEulerMatrix\(\)](#), [getEulerMatrix\(\)](#), [getMatrix\(\)](#) und [setMatrix\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- header/[EulerMatrix.h](#)
- source/[EulerMatrix.cpp](#)

7.2 CInputParameter Klassenreferenz

Handling Eingabedaten.

```
#include <InputParameter.h>
```

Öffentliche Methoden

- [CInputParameter](#) (void)
Default Konstruktor.
- [CInputParameter](#) (double initSpeed, bool initSeepManual, bool initOrientationManual, double initA, double initB, double initC)
Konstruktor mit Werten.
- [~CInputParameter](#) (void)
Dekonstruktor.
- void [setOrientation](#) (bool initOrientationManual, double initA, double initB, double initC)
Setzt Orientierungs Daten.
- void [setSpeed](#) (double initSpeed, bool initSpeedManual)
Setzt Geschwindigkeits Daten.
- void [setOffset](#) (double X, double Y, double Z, bool [offsetManual](#))
Setzt den gewünschten Offset.
- void [setLogging](#) (bool initLoggingManual)
Setzt ob es ein detailliertes Logging geben soll.
- double [getSpeed](#) (void)
Gibt Geschwindigkeit zurueck.
- bool [getSpeedManual](#) (void)
Gibt zurueck ob haendische Geschwindigkeit verwendet werden soll.
- bool [getOrientationManual](#) (void)
Gibt zurueck ob haendische Orientierung verwendet werden soll.
- tuple< double, double, double > [getAngles](#) (void)
Gibt Winkel zurueck.
- bool [getOffsetManual](#) (void)
Gibt zurueck ob ein Offset eingestellt werden soll.
- bool [getLoggingManual](#) (void)
Gibt zurueck ob ein detailliertes Logging ausgegeben werden soll.
- tuple< double, double, double > [getOffset](#) (void)
Gibt Offset zurueck.
- void [openFile](#) (std::string path)
Liest die Daten aus dem Input File ein.
- bool [detectJump](#) ([CInputPoint3D](#) p, double x_prev, double y_prev, double z_prev)
Erkennt Spruenge in den Daten.
- vector< list< [CInputPoint3D](#) > > & [getPath](#) ()
Gibt Pfad zurueck.

Private Attribute

- vector< list< [CInputPoint3D](#) > > [initialPath](#)
- double [speed](#)
- bool [speedManual](#)
- bool [orientationManual](#)
- double [A](#)
- double [B](#)
- double [C](#)
- double [difference](#) = 20
- bool [offsetManual](#)
- double [offsetX](#)
- double [offsetY](#)
- double [offsetZ](#)
- bool [loggingManual](#)

7.2.1 Ausführliche Beschreibung

Handling Eingabedaten.

In dieser Klasse werden die eingelesenen einstellbaren Daten und das einlesen der Daten aus der Eingabedatei gehandelt.

Definiert in Zeile [25](#) der Datei [InputParameter.h](#).

7.2.2 Beschreibung der Konstruktoren und Destruktoren

7.2.2.1 CInputParameter() [1/2]

```
CInputParameter::CInputParameter (  
    void )
```

Default Konstruktor.

Initialisiert die Input Daten mit Null

Siehe auch

[CInputParameter\(double initSpeed, bool initSeepManual, bool initOrientationManual, double initA, double initB, double initC\)](#)

Definiert in Zeile [24](#) der Datei [InputParameter.cpp](#).

```
00025 {  
00026     speed = 0.1;  
00027     A = 0;  
00028     B = 75;  
00029     C = 0;  
00030     speedManual = false,  
00031     orientationManual = false;  
00032  
00033 }
```

Benutzt [A](#), [B](#), [C](#), [orientationManual](#), [speed](#) und [speedManual](#).

7.2.2.2 CInputParameter() [2/2]

```
CInputParameter::CInputParameter (
    double initSpeed,
    bool initSeepManual,
    bool initOrientationManual,
    double initA,
    double initB,
    double initC )
```

Konstruktor mit Werten.

Initialisiert die Input Daten

Parameter

<i>double</i>	initSpeed
<i>bool</i>	initSeepManual
<i>bool</i>	initOrientationManual
<i>double</i>	initA
<i>double</i>	initB
<i>double</i>	initC

Siehe auch[CInputParameter\(\)](#)[~CInputParameter\(\)](#)[CInputParameter\(void\);](#)

Definiert in Zeile 12 der Datei [InputParameter.cpp](#).

```

00013 {
00014     speed = initSpeed;
00015     speedManual = initSpeedManual;
00016     orientationManual = initOrientationManual;
00017     A = initA;
00018     B = initB;
00019     C = initC;
00020
00021 }
```

Benutzt [A](#), [B](#), [C](#), [orientationManual](#), [speed](#) und [speedManual](#).

7.2.2.3 ~CInputParameter()

```

CInputParameter::~~CInputParameter (
    void )
```

Dekonstruktor.

Definiert in Zeile 35 der Datei [InputParameter.cpp](#).

```

00036 {
00037
00038 }
```

7.2.3 Dokumentation der Elementfunktionen**7.2.3.1 detectJump()**

```

bool CInputParameter::detectJump (
    CInputPoint3D p,
    double x_prev,
    double y_prev,
    double z_prev )
```

Erkennt Spruenge in den Daten.

Um zu erkennen ob es mehrere Pfade sind wird nach Spruengen gesucht, bei einem Sprung wird eine neue Liste angefangen.

Parameter

CInputPoint3D	p den aktuellen Punkt
<i>double</i>	x_prev die vorherige x Position
<i>double</i>	y_prev die vorherige y Position
<i>double</i>	z_prev die vorherige z Position

Definiert in Zeile 153 der Datei [InputParameter.cpp](#).

```
00154 {
00155     if (abs(p.getX() - x_prev) > difference) // Abstand zwischen Punkten groesser max
00156         Differenz?? return true;
00157     else if (abs(p.getY() - y_prev) > difference) // Abstand zwischen Punkten groesser max
00158         Differenz?? return true;
00159     else if (abs(p.getZ() - z_prev) > difference) // Abstand zwischen Punkten groesser max
00160         Differenz?? return true;
00161     else
00162         return false;
00163 }
```

Benutzt [difference](#), [CPoint3D::getX\(\)](#), [CPoint3D::getY\(\)](#) und [CPoint3D::getZ\(\)](#).

Wird benutzt von [openFile\(\)](#).

7.2.3.2 getAngles()

```
tuple< double, double, double > CInputParameter::getAngles (
    void )
```

Gibt Winkel zurueck.

Gibt die eingegebenen Winkel als tuple zurueck

Rückgabe

: tuple <double double double> angles

Definiert in Zeile 99 der Datei [InputParameter.cpp](#).

```
00100 {
00101     return make_tuple(A, B, C); // Winkel zurueck geben
00102 }
```

Benutzt [A](#), [B](#) und [C](#).

Wird benutzt von [CRobCodeGenerator::postProcessing\(\)](#).

7.2.3.3 getLoggingManual()

```
bool CInputParameter::getLoggingManual (
    void )
```

Gibt zurueck ob ein detailliertes Logging ausgegeben werden soll.

Gibt zurueck ob ein detailliertes Logging ausgegeben werden soll

Definiert in Zeile 94 der Datei [InputParameter.cpp](#).

```
00095 {
00096     return loggingManual; // Vorgewaehte Einstellung fuer das Logging zurueck
00097 }
```

Benutzt [loggingManual](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.2.3.4 getOffset()

```
tuple< double, double, double > CInputParameter::getOffset (
    void )
```

Gibt Offset zurueck.

Gibt den Offset als tuple zurueck

Rückgabe

: tuple <double double double> offset

Definiert in Zeile 104 der Datei [InputParameter.cpp](#).

```
00105 {
00106     return make_tuple(offsetX, offsetY, offsetZ);    // Offset zurueck geben
00107 }
```

Benutzt [offsetX](#), [offsetY](#) und [offsetZ](#).

Wird benutzt von [CRobCodeGenerator::generateRobCode\(\)](#).

7.2.3.5 getOffsetManual()

```
bool CInputParameter::getOffsetManual (
    void )
```

Gibt zurueck ob ein Offset eingestellt werden soll.

Gibt zurueck ob ein Offset eingestellt werden soll

Definiert in Zeile 89 der Datei [InputParameter.cpp](#).

```
00090 {
00091     return offsetManual;    // Vorgewaehlte Einstellung fuer den Offset zurueck
00092 }
```

Benutzt [offsetManual](#).

7.2.3.6 getOrientationManual()

```
bool CInputParameter::getOrientationManual (
    void )
```

Gibt zurueck ob haendische Orientierung verwendet werden soll.

Gibt zurueck ob haendische Orientierung verwendet werden soll, sonst wird sie spaeter berechnet.

Definiert in Zeile 84 der Datei [InputParameter.cpp](#).

```
00085 {
00086     return orientationManual;    // Vorgewaehlte Einstellung fuer Orientierung zurueck geben
00087 }
```

Benutzt [orientationManual](#).

Wird benutzt von [CRobCodeGenerator::postProcessing\(\)](#).

7.2.3.7 getPath()

```
vector< list< CInputPoint3D > > & CInputParameter::getPath ( )
```

Gibt Pfad zurueck.

Rückgabe

: vector<list<CInputPoint3D>> den eingelesenen Pfad

Definiert in Zeile 69 der Datei `InputParameter.cpp`.

```
00070 {  
00071     return initialPath;           // Path zurueck geben  
00072 }
```

Benutzt `initialPath`.

Wird benutzt von `GUI::calculate()`.

7.2.3.8 getSpeed()

```
double CInputParameter::getSpeed (  
    void )
```

Gibt Geschwindigkeit zurueck.

Gibt die eingegebene Geschwindigkeit zurueck

Definiert in Zeile 74 der Datei `InputParameter.cpp`.

```
00075 {  
00076     return speed;                 // Geschwindigkeit zurueck geben  
00077 }
```

Benutzt `speed`.

Wird benutzt von `CRobCodeGenerator::generateRobCode()` und `CRobCodeGenerator::postProcessing()`.

7.2.3.9 getSpeedManual()

```
bool CInputParameter::getSpeedManual (  
    void )
```

Gibt zurueck ob haendische Geschwindigkeit verwendet werden soll.

Gibt zurueck ob haendische Geschwindigkeit verwendet werden soll, sonst wird sie spaeter berechnet.

Definiert in Zeile 79 der Datei `InputParameter.cpp`.

```
00080 {  
00081     return speedManual;           // Vorgewaehlte Einstellung fuer Geschwindigkeit zurueck geben  
00082 }
```

Benutzt `speedManual`.

Wird benutzt von `CRobCodeGenerator::generateRobCode()` und `CRobCodeGenerator::postProcessing()`.

7.2.3.10 openFile()

```
void CInputParameter::openFile (  
    std::string path )
```

Liest die Daten aus dem Input File ein.

Liest die Daten aus einen beliebigen File ein und ruft `@detectJump` auf um zu erkennen ob es mehrere Aufnahmen sind.

Parameter

File	Pfad
------	------

Definiert in Zeile 111 der Datei [InputParameter.cpp](#).

```

00112 {
00113     ifstream fin(path);
00114     CInputPoint3D tmpPoint;           // Zwischenspeicher zum konvertieren von tmpEuler in Point3D
00115     CEulerMatrix tmpEuler;           // Zwischenspeicher zum konvertieren von DummyMatrix in EulerMatrix
00116     double x, y, z;                  // Punktkoordinaten
00117     double x_prev = 0, y_prev = 0, z_prev = 0; // Zwischenspeicher fuer Punktkoordinaten
00118     double timestamp;                // Zeitstempel
00119     int segmentCount = -1;            // Segmentzaehler
00120     float dummyMatrix[3][3];         // DummyMatrix zum speichern
00121
00122
00123     if (!fin.is_open())
00124     {
00125         cerr << "Datei konnte nicht geoeffnet werden" << endl; // Fehler Datei konnte nicht
geoeffnet werden.
00126     }
00127     string line;
00128
00129     while(getline(fin, line))
00130     {
00131         std::istringstream sStream (line);
00132         sStream >> timestamp >> x >> y >> z >> dummyMatrix[0][0] >> dummyMatrix[0][1] >> dummyMatrix[0][2]
// Zeile in die einzelnen Parameter zerlegen
00133         >> dummyMatrix[1][0] >> dummyMatrix[1][1] >> dummyMatrix[1][2] >> dummyMatrix[2][0] >>
dummyMatrix[2][1] >> dummyMatrix[2][2]; // und in DummyMatrix bzw. Variablen abspeichern
00134
00135         tmpEuler.setMatrix(dummyMatrix); // DummyMatrix[3][3] in
EulerMatrix speichern
00136         tmpPoint.setPoint(timestamp, x, y, z, tmpEuler.getEulerMatrix()); // Variablen und EulerWinkel
in CPoint3D speichern
00137
00138         if (detectJump(tmpPoint, x_prev, y_prev, z_prev)) // if there is a jump in the data, start da
new segment
00139         {
00140             segmentCount++; // neues Segment anlegen
00141             initialPath.push_back(list<CInputPoint3D>()); // Punkt in Segment speichern
00142         }
00143
00144         initialPath[segmentCount].push_back(tmpPoint); // Punkt in bestehendes Segment
abspeichern
00145
00146         x_prev = x; // X-Wert zwischenspeichern
00147         y_prev = y; // Y-Wert zwischenspeichern
00148         z_prev = z; // Z-Wert zwischenspeichern
00149     }
00150     fin.close(); // Datei schliessen
00151 }

```

Benutzt [detectJump\(\)](#), [CEulerMatrix::getEulerMatrix\(\)](#), [initialPath](#), [CEulerMatrix::setMatrix\(\)](#) und [CInputPoint3D::setPoint\(\)](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.2.3.11 setLogging()

```

void CInputParameter::setLogging (
    bool initLoggingManual )

```

Setzt ob es ein detailliertes Logging geben soll.

Setzt ob es ein detailliertes Logging geben soll

Parameter

<i>bool</i>	initLoggingManual
-------------	-------------------

Definiert in Zeile 57 der Datei [InputParameter.cpp](#).

```
00058 {
00059     loggingManual = initLoggingManual;
00060 }
```

Benutzt [loggingManual](#).

Wird benutzt von [GUI::activateLogging\(\)](#).

7.2.3.12 setOffset()

```
void CInputParameter::setOffset (
    double X,
    double Y,
    double Z,
    bool offsetManual )
```

Setzt den gewuenschten Offset.

Setzt den gewuenschten Offset

Parameter

<i>double</i>	X
<i>double</i>	Y
<i>double</i>	Z
<i>bool</i>	offsetManual

Definiert in Zeile 49 der Datei [InputParameter.cpp](#).

```
00050 {
00051     offsetManual = initOffsetManual;
00052     offsetX = X;
00053     offsetY = Y;
00054     offsetZ = Z;
00055 }
```

Benutzt [offsetManual](#), [offsetX](#), [offsetY](#) und [offsetZ](#).

Wird benutzt von [GUI::activateOffset\(\)](#), [GUI::GUI\(\)](#) und [GUI::setOffset\(\)](#).

7.2.3.13 setOrientation()

```
void CInputParameter::setOrientation (
    bool initOrientationManual,
    double initA,
    double initB,
    double initC )
```

Setzt Orientierungs Daten.

Setzt ob die Orientierung Haendisch eingegeben werden soll und die drei Winkel

Parameter

<i>bool</i>	initOrientationManual
<i>double</i>	initA
<i>double</i>	initB
<i>double</i>	initC

Definiert in Zeile 41 der Datei [InputParameter.cpp](#).

```
00042 {
00043     orientationManual = initOrientationManual;
00044     A = initA;
00045     B = initB;
00046     C = initC;
00047 }
```

Benutzt [A](#), [B](#), [C](#) und [orientationManual](#).

Wird benutzt von [GUI::setOrientation\(\)](#).

7.2.3.14 setSpeed()

```
void CInputParameter::setSpeed (
    double initSpeed,
    bool initSpeedManual )
```

Setzt Geschwindigkeits Daten.

Setzt ob die Geschwindigkeit Haendisch eingegeben werden soll und die Geschwindigkeit in m/s

Parameter

<i>double</i>	initSpeed
<i>bool</i>	initSeepManual

Definiert in Zeile 63 der Datei [InputParameter.cpp](#).

```
00064 {
00065     speed = initSpeed;
00066     speedManual = initSpeedManual;
00067 }
```

Benutzt [speed](#) und [speedManual](#).

Wird benutzt von [CRobCodeGenerator::postProcessing\(\)](#) und [GUI::setSpeed\(\)](#).

7.2.4 Dokumentation der Datenelemente

7.2.4.1 A

```
double CInputParameter::A [private]
```

User eingegebener Winkel A

Definiert in Zeile 163 der Datei [InputParameter.h](#).

Wird benutzt von [CInputParameter\(\)](#), [CInputParameter\(\)](#), [getAngles\(\)](#) und [setOrientation\(\)](#).

7.2.4.2 B

```
double CInputParameter::B [private]
```

User eingegebener Winkel B

Definiert in Zeile 167 der Datei [InputParameter.h](#).

Wird benutzt von [CInputParameter\(\)](#), [CInputParameter\(\)](#), [getAngles\(\)](#) und [setOrientation\(\)](#).

7.2.4.3 C

```
double CInputParameter::C [private]
```

User eingegebener Winkel C

Definiert in Zeile 171 der Datei [InputParameter.h](#).

Wird benutzt von [CInputParameter\(\)](#), [CInputParameter\(\)](#), [getAngles\(\)](#) und [setOrientation\(\)](#).

7.2.4.4 difference

```
double CInputParameter::difference = 20 [private]
```

Sprung ab dem eine neue Liste angefangen wird

Definiert in Zeile 175 der Datei [InputParameter.h](#).

Wird benutzt von [detectJump\(\)](#).

7.2.4.5 initialPath

```
vector<list<CInputPoint3D> > CInputParameter::initialPath [private]
```

Vector mit Listen an Input Daten

Definiert in Zeile 147 der Datei [InputParameter.h](#).

Wird benutzt von [getPath\(\)](#) und [openFile\(\)](#).

7.2.4.6 loggingManual

```
bool CInputParameter::loggingManual [private]
```

Auswahl ob ein Offset eingegeben werden soll

Definiert in Zeile 195 der Datei [InputParameter.h](#).

Wird benutzt von [getLoggingManual\(\)](#) und [setLogging\(\)](#).

7.2.4.7 offsetManual

```
bool CInputParameter::offsetManual [private]
```

Auswahl ob ein Offset eingegeben werden soll

Definiert in Zeile 179 der Datei [InputParameter.h](#).

Wird benutzt von [getOffsetManual\(\)](#) und [setOffset\(\)](#).

7.2.4.8 offsetX

```
double CInputParameter::offsetX [private]
```

Offset X

Definiert in Zeile 183 der Datei [InputParameter.h](#).

Wird benutzt von [getOffset\(\)](#) und [setOffset\(\)](#).

7.2.4.9 offsetY

```
double CInputParameter::offsetY [private]
```

Offset Y

Definiert in Zeile 187 der Datei [InputParameter.h](#).

Wird benutzt von [getOffset\(\)](#) und [setOffset\(\)](#).

7.2.4.10 offsetZ

```
double CInputParameter::offsetZ [private]
```

Offset Z

Definiert in Zeile 191 der Datei [InputParameter.h](#).

Wird benutzt von [getOffset\(\)](#) und [setOffset\(\)](#).

7.2.4.11 orientationManual

```
bool CInputParameter::orientationManual [private]
```

Auswahl ob berechnete oder eingegebene Winkel verwendet werden soll

Definiert in Zeile 159 der Datei [InputParameter.h](#).

Wird benutzt von [CInputParameter\(\)](#), [CInputParameter\(\)](#), [getOrientationManual\(\)](#) und [setOrientation\(\)](#).

7.2.4.12 speed

```
double CInputParameter::speed [private]
```

User eingegebene Geschwindigkeit

Definiert in Zeile 151 der Datei [InputParameter.h](#).

Wird benutzt von [CInputParameter\(\)](#), [CInputParameter\(\)](#), [getSpeed\(\)](#) und [setSpeed\(\)](#).

7.2.4.13 speedManual

```
bool CInputParameter::speedManual [private]
```

Auswahl ob berechnete oder eingegebene Geschwindigkeit verwendet werden soll

Definiert in Zeile 155 der Datei [InputParameter.h](#).

Wird benutzt von [CInputParameter\(\)](#), [CInputParameter\(\)](#), [getSpeedManual\(\)](#) und [setSpeed\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

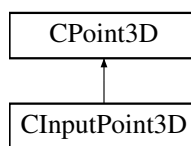
- [header/InputParameter.h](#)
- [source/InputParameter.cpp](#)

7.3 CInputPoint3D Klassenreferenz

Input Punkt.

```
#include <Point3D.h>
```

Klassendiagramm für CInputPoint3D:



Öffentliche Methoden

- [CInputPoint3D](#) (void)
Default Konstruktor.
- [CInputPoint3D](#) (double X, double Y, double Z, double Timestamp, [CEulerMatrix](#) Matrix)
Default Konstruktor.
- [~CInputPoint3D](#) (void)
Dekonstruktor.
- double [getTime](#) ()
Gibt den Zeitstempel zurueck.
- [CEulerMatrix](#) [getEulerMatrix](#) ()
Gibt die gespeicherte Eulermatrix zurueck.
- void [setTime](#) (double time)
Setzt den Zeitstempel.
- void [setEulerMatrix](#) ([CEulerMatrix](#) orientation)
Setzt die Eulermatrix.
- void [setPoint](#) (double time, double X, double Y, double Z, [CEulerMatrix](#) orientation)
Setzt einen Input Punkt.

Öffentliche Methoden geerbt von [CPoint3D](#)

- [CPoint3D](#) (void)
Default Konstruktor.
- [CPoint3D](#) (double X, double Y, double Z)
Default Konstruktor.
- [~CPoint3D](#) (void)
Dekonstruktor.
- double [getX](#) ()
Gibt X zurueck.
- double [getY](#) ()
Gibt Y zurueck.
- double [getZ](#) ()
Gibt Z zurueck.
- void [setX](#) (double X)
Setzt X.
- void [setY](#) (double Y)
Setzt Y.
- void [setZ](#) (double Z)
Setzt Z.
- void [set](#) (double X, double Y, double Z)
Setzt X, Y und Z.
- double [distanceTo](#) ([CPoint3D](#) point)
Berechnet die Distanz zu einem anderen Punkt.
- double [distanceTo](#) ([CLine3D](#) line)
Berechnet die Distanz zu einer Linie.

Private Attribute

- double [timestamp](#)
- [CEulerMatrix](#) [orientationMatrix](#)

Weitere Geerbte Elemente

Geschützte Attribute geerbt von [CPoint3D](#)

- double [x](#)
- double [y](#)
- double [z](#)

7.3.1 Ausführliche Beschreibung

Input Punkt.

Kind der Punkt Grundklasse, erweitert um den Timestamp und die Eulermatrix

Definiert in Zeile [106](#) der Datei [Point3D.h](#).

7.3.2 Beschreibung der Konstruktoren und Destruktoren

7.3.2.1 CInputPoint3D() [1/2]

```
CInputPoint3D::CInputPoint3D (
    void )
```

Default Konstruktor.

Initialisiert des eingelesenen Punktes mit Null

Siehe auch

[CInputPoint3D\(double X, double Y, double Z, double Timestamp, CEulerMatrix Matrix\)](#)

Definiert in Zeile 110 der Datei [Point3D.cpp](#).

```
00110                                     : CPoint3D()
00111 {
00112     timestamp = 0;          // Zeitstempel mit 0 initialisieren
00113 }
```

Benutzt [timestamp](#).

7.3.2.2 CInputPoint3D() [2/2]

```
CInputPoint3D::CInputPoint3D (
    double X,
    double Y,
    double Z,
    double Timestamp,
    CEulerMatrix Matrix )
```

Default Konstruktor.

Initialisiert des eingelesenen Punktes mit Null

Parameter

<i>double</i>	X
<i>double</i>	Y
<i>double</i>	Z
<i>double</i>	Timestamp
CEulerMatrix	Matrix

Siehe auch

[CInputPoint3D\(double X, double Y, double Z, double Timestamp, CEulerMatrix Matrix\)](#)

Definiert in Zeile 116 der Datei [Point3D.cpp](#).

```
00117 {
00118     x = X;
00119     y = Y;
00120     z = Z;
00121     timestamp = Timestamp;
00122     orientationMatrix = Matrix;
00123 }
```

Benutzt [orientationMatrix](#), [timestamp](#), [CPoint3D::x](#), [CPoint3D::y](#) und [CPoint3D::z](#).

7.3.2.3 ~CInputPoint3D()

```
CInputPoint3D::~~CInputPoint3D (
    void )
```

Dekonstruktor.

Definiert in Zeile 125 der Datei [Point3D.cpp](#).

```
00126 {
00127 }
```

7.3.3 Dokumentation der Elementfunktionen

7.3.3.1 getEulerMatrix()

```
CEulerMatrix CInputPoint3D::getEulerMatrix ( )
```

Gibt die gespeicherte Eulermatrix zurueck.

Rückgabe

[CEulerMatrix](#)

Definiert in Zeile 147 der Datei [Point3D.cpp](#).

```
00148 {
00149     return orientationMatrix; // Rueckgabe der EulerMatrix
00150 }
```

Benutzt [orientationMatrix](#).

Wird benutzt von [CRobCodeGenerator::calculateAngles\(\)](#).

7.3.3.2 getTime()

```
double CInputPoint3D::getTime ( )
```

Gibt den Zeitstempel zurueck.

Rückgabe

double Zeitstempel

Definiert in Zeile 152 der Datei [Point3D.cpp](#).

```
00153 {
00154     return timestamp; // Rueckgabe des Zeitstempel
00155 }
```

Benutzt [timestamp](#).

Wird benutzt von [CRobCodeGenerator::calculateSpeed\(\)](#).

7.3.3.3 setEulerMatrix()

```
void CInputPoint3D::setEulerMatrix (
    CEulerMatrix orientation )
```

Setzt die Eulermatrix.

Parameter

CEulerMatrix	orientation
------------------------------	-------------

Definiert in Zeile 129 der Datei [Point3D.cpp](#).

```
00130 {
00131     orientationMatrix = orientation;    // EulerMatrix setzen
00132 }
```

Benutzt [orientationMatrix](#).

Wird benutzt von [CMeanFilter::calculateMean\(\)](#), [CPathBuilder::createPath\(\)](#), [CSegmentApproximator::douglasPeuckerRecursive\(\)](#) und [setPoint\(\)](#).

7.3.3.4 setPoint()

```
void CInputPoint3D::setPoint (
    double time,
    double X,
    double Y,
    double Z,
    CEulerMatrix orientation )
```

Setzt einen Input Punkt.

Parameter

<i>double</i>	time
<i>double</i>	X
<i>double</i>	Y
<i>double</i>	Z
CEulerMatrix	orientation

Definiert in Zeile 135 der Datei [Point3D.cpp](#).

```
00136 {
00137     setTime(time);    // Zeitstempel setzen
00138     set(X, Y, Z);    // setze Punkt-Koordinaten
00139     setEulerMatrix(orientation); // EulerMatrix setzen
00140 }
```

Benutzt [CPoint3D::set\(\)](#), [setEulerMatrix\(\)](#) und [setTime\(\)](#).

Wird benutzt von [CInputParameter::openFile\(\)](#).

7.3.3.5 setTime()

```
void CInputPoint3D::setTime (
    double time )
```

Setzt den Zeitstempel.

Parameter

<i>double</i>	time
---------------	------

Definiert in Zeile 142 der Datei [Point3D.cpp](#).

```
00143 {  
00144     timestamp = time; // Zeitstempel setzen  
00145 }
```

Benutzt [timestamp](#).

Wird benutzt von [CMeanFilter::calculateMean\(\)](#), [CPathBuilder::createPath\(\)](#), [CSegmentApproximator::douglasPeuckerRecursive\(\)](#) und [setPoint\(\)](#).

7.3.4 Dokumentation der Datenelemente

7.3.4.1 orientationMatrix

```
CEulerMatrix CInputPoint3D::orientationMatrix [private]
```

Eulermatrix des Punktes

Definiert in Zeile 170 der Datei [Point3D.h](#).

Wird benutzt von [CInputPoint3D\(\)](#), [getEulerMatrix\(\)](#) und [setEulerMatrix\(\)](#).

7.3.4.2 timestamp

```
double CInputPoint3D::timestamp [private]
```

Zeitstempel

Definiert in Zeile 166 der Datei [Point3D.h](#).

Wird benutzt von [CInputPoint3D\(\)](#), [CInputPoint3D\(\)](#), [getTime\(\)](#) und [setTime\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- header/[Point3D.h](#)
- source/[Point3D.cpp](#)

7.4 CLine3D Klassenreferenz

Berechnung Geraden.

```
#include <Line3D.h>
```

Öffentliche Methoden

- [CLine3D](#) (void)
Default Konstruktor.
- [CLine3D](#) ([CPoint3D](#) P1, [CPoint3D](#) P2)
Konstruktor mit zwei Punkten.
- [~CLine3D](#) (void)
Dekonstruktor.

Öffentliche Attribute

- [CPoint3D p1](#)
- [CPoint3D p2](#)

7.4.1 Ausführliche Beschreibung

Berechnung Geraden.

In dieser Klasse werden alle Berechnungen die zwischen zwei Punkten passieren gehandhabt.

Definiert in Zeile [18](#) der Datei [Line3D.h](#).

7.4.2 Beschreibung der Konstruktoren und Destruktoren

7.4.2.1 CLine3D() [1/2]

```
CLine3D::CLine3D (  
    void )
```

Default Konstruktor.

Initialisiert die Klasse

Siehe auch

[CLine3D\(CPoint3D P1, CPoint3D P2\)](#)

Definiert in Zeile [10](#) der Datei [Line3D.cpp](#).

```
00011 {  
00012 }
```

7.4.2.2 CLine3D() [2/2]

```
CLine3D::CLine3D (  
    CPoint3D P1,  
    CPoint3D P2 )
```

Konstruktor mit zwei Punkten.

Initialisiert die Klasse

Siehe auch

[CLine3D\(void\);](#)

Definiert in Zeile [15](#) der Datei [Line3D.cpp](#).

```
00016 {  
00017     p1 = P1;  
00018     p2 = P2;  
00019 }
```

Benutzt [p1](#) und [p2](#).

7.4.2.3 ~CLine3D()

```
CLine3D::~CLine3D (
    void )
```

Dekonstruktor.

Definiert in Zeile 21 der Datei [Line3D.cpp](#).

```
00022 {
00023 }
```

7.4.3 Dokumentation der Datenelemente

7.4.3.1 p1

```
CPoint3D CLine3D::p1
```

Punkt 1

Definiert in Zeile 41 der Datei [Line3D.h](#).

Wird benutzt von [CLine3D\(\)](#) und [CPoint3D::distanceTo\(\)](#).

7.4.3.2 p2

```
CPoint3D CLine3D::p2
```

Punkt 2

Definiert in Zeile 45 der Datei [Line3D.h](#).

Wird benutzt von [CLine3D\(\)](#) und [CPoint3D::distanceTo\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [header/Line3D.h](#)
- [source/Line3D.cpp](#)

7.5 CLogging Klassenreferenz

Gleitender Mittelwertfilter.

```
#include <Logging.h>
```

Öffentliche Methoden

- `CLogging` (void)
Default Konstruktor.
- `CLogging` (string `path`, bool `detailed`)
Default Konstruktor.
- `~CLogging` (void)
Dekonstruktor.
- void `setStep` (int `Step`)
Setzt in welchem Schritt wir uns befinden.
- bool `getDetailed` (void)
Gibt es ein detailliertes Logging.
- void `logData` (vector< list< `CInputPoint3D` > > &`sourcePath`)
Ruft calculateMean fuer die einzelnen Segmente auf.
- void `logData` (vector< `CInputPoint3D` > &`sourcePath`)
Ruft calculateMean fuer die einzelnen Segmente auf.
- void `logData` (vector< `COutputPoint3D` > &`sourcePath`)
Ruft calculateMean fuer die einzelnen Segmente auf.

Private Attribute

- int `step`
- string `path`
- bool `detailed`

7.5.1 Ausführliche Beschreibung

Gleitender Mittelwertfilter.

In dieser Klasse werden die eingelesenen Daten mit einem gleitenden Mittelwertfilter mit einstellbarem Fenster gelaetet.

Definiert in Zeile 22 der Datei `Logging.h`.

7.5.2 Beschreibung der Konstruktoren und Destruktoren

7.5.2.1 CLogging() [1/2]

```
CLogging::CLogging (
    void )
```

Default Konstruktor.

Initialisiert Logging Klasse

Siehe auch

`CMeanFilter(int Window);`

Definiert in Zeile 10 der Datei `Logging.cpp`.

```
00011 {
00012     step = 0;
00013 }
```

Benutzt `step`.

7.5.2.2 CLogging() [2/2]

```
CLogging::CLogging (
    string path,
    bool detailed )
```

Default Konstruktor.

Initialisiert Logging Klasse

Siehe auch

[CMeanFilter\(int Window\);](#)

Definiert in Zeile 16 der Datei [Logging.cpp](#).

```
00017 {
00018     path = Path;
00019     detailed = Detailed;
00020 }
```

Benutzt [detailed](#) und [path](#).

7.5.2.3 ~CLogging()

```
CLogging::~CLogging (
    void )
```

Dekonstruktor.

Definiert in Zeile 22 der Datei [Logging.cpp](#).

```
00023 {
00024
00025 }
```

7.5.3 Dokumentation der Elementfunktionen

7.5.3.1 getDetailed()

```
bool CLogging::getDetailed (
    void )
```

Gibt es ein detailliertes Logging.

Rückgabe

bool detailed

Definiert in Zeile 27 der Datei [Logging.cpp](#).

```
00028 {
00029     return detailed;
00030 }
```

Benutzt [detailed](#).

Wird benutzt von [CSegmentApproximator::approx\(\)](#), [CPathBuilder::createPath\(\)](#) und [CMeanFilter::mean\(\)](#).

7.5.3.2 logData() [1/3]

```
void CLogging::logData (
    vector< CInputPoint3D > & sourcePath )
```

Ruft calculateMean fuer die einzelnen Segmente auf.

Loggingdaten werden weggespeichert

Parameter

<code>vector<CInputPoint3D>&</code>	<code>sourcePath</code>
---	-------------------------

Definiert in Zeile 73 der Datei [Logging.cpp](#).

```

00074 {
00075     string filepath;           // file Pfad
00076     float dummyMatrix[3][3];  // dummyMatrix zum Zwischenspeichern
00077     CEulerMatrix tmpEuler;     // CEulerMatrix zum Zwischenspeichern
00078
00079     filepath = path + "/" + "0" + std::to_string(step) + "_log.csv";
00080
00081     FILE* fid = fopen(filepath.c_str(), "w"); // file oeffnen
00082
00083     if (fid == NULL)
00084     {
00085         cerr << "ERROR - Can NOT write to output file!\n"; // Fehler beim file oeffnen
00086         return;
00087     }
00088
00089     /* Ausgeben der Punkte mit dummyMatrix */
00090     for (size_t s = 0; s < sourcePath.size(); s++) //for all points in the vector
00091     {
00092         tmpEuler.getMatrix(dummyMatrix);
00093
00094         fprintf(fid, "%f %f %f %f %f %f %f %f %f %f %f %f\n", (double)sourcePath[s].getTime(),
00095             (double)sourcePath[s].getX(), (double)sourcePath[s].getY(), (double)sourcePath[s].getZ(),
00096             dummyMatrix[0][0], dummyMatrix[0][1], dummyMatrix[0][2],
00097             dummyMatrix[1][0], dummyMatrix[1][1], dummyMatrix[1][2],
00098             dummyMatrix[2][0], dummyMatrix[2][1], dummyMatrix[2][2]);
00099     }
00100 }

```

Benutzt [CEulerMatrix::getMatrix\(\)](#), `path` und `step`.

7.5.3.3 logData() [2/3]

```

void CLogging::logData (
    vector< COutputPoint3D > & sourcePath )

```

Ruft `calculateMean` fuer die einzelnen Segmente auf.

Loggingdaten werden weggespeichert

Parameter

<code>vector<COutputPoint3D>&</code>	<code>sourcePath</code>
--	-------------------------

Definiert in Zeile 102 der Datei [Logging.cpp](#).

```

00103 {
00104     string filepath;           // file Pfad
00105     float dummyMatrix[3][3];  // dummyMatrix zum Zwischenspeichern
00106     CEulerMatrix tmpEuler;     // CEulerMatrix zum Zwischenspeichern
00107
00108     filepath = path + "/" + "0" + std::to_string(step) + "_path.csv";
00109
00110     FILE* fid = fopen(filepath.c_str(), "w"); // file oeffnen
00111
00112     if (fid == NULL)
00113     {
00114         cerr << "ERROR - Can NOT write to output file!\n"; // Fehler beim file oeffnen
00115         return;
00116     }
00117
00118     /* Ausgeben der Punkte mit dummyMatrix */
00119     for (size_t s = 0; s < sourcePath.size(); s++) //for all points in the vector
00120     {
00121         tmpEuler.getMatrix(dummyMatrix);
00122     }

```

```

00123         fprintf(fid, "%f %f %f %f %f %f %f %f %f %f %f %f\n", (double)sourcePath[s].getSpeed(),
00124                     (double)sourcePath[s].getX(), (double)sourcePath[s].getY(), (double)sourcePath[s].getZ(),
00125                     (double)sourcePath[s].getA(), (double)sourcePath[s].getB(), (double)sourcePath[s].getC());
00126     }
00127 }

```

Benutzt [CEulerMatrix::getMatrix\(\)](#), [path](#) und [step](#).

7.5.3.4 logData() [3/3]

```

void CLogging::logData (
    vector< list< CInputPoint3D > > & sourcePath )

```

Ruft [calculateMean](#) fuer die einzelnen Segmente auf.

Loggingdaten werden weggespeichert

Parameter

<code>vector<list<CInputPoint3D>>&</code>	<code>sourcePath</code>
---	-------------------------

Definiert in Zeile 37 der Datei [Logging.cpp](#).

```

00038 {
00039     string filepath;           // file Pfad
00040     float dummyMatrix[3][3];  // dummyMatrix zum Zwischenspeichern
00041     CEulerMatrix tmpEuler;    // CEulerMatrix zum Zwischenspeichern
00042
00043     filepath = path + "/" + "0" + std::to_string(step) + "_log.csv";
00044
00045     FILE* fid = fopen(filepath.c_str(), "w");           // file oeffnen
00046
00047     if (fid == NULL)
00048     {
00049         cerr << "ERROR - Can NOT write to output file!\n";           // Fehler beim file oeffnen
00050         return;
00051     }
00052
00053     for (size_t s = 0; s < sourcePath.size(); s++) //for all segments
00054     {
00055         list<CInputPoint3D>::iterator itr = sourcePath[s].begin();
00056
00057         tmpEuler = itr->getEulerMatrix();
00058         tmpEuler.getMatrix(dummyMatrix);
00059
00060         /* Ausgeben der Punkte mit dummyMatrix */
00061         for (; itr != sourcePath[s].end(); itr++) //for all points in the segment
00062         {
00063             fprintf(fid, "%f %f %f %f %f %f %f %f %f %f %f %f\n", (double)itr->getTime(),
00064                     (double)itr->getX(), (double)itr->getY(), (double)itr->getZ(),
00065                     dummyMatrix[0][0], dummyMatrix[0][1], dummyMatrix[0][2],
00066                     dummyMatrix[1][0], dummyMatrix[1][1], dummyMatrix[1][2],
00067                     dummyMatrix[2][0], dummyMatrix[2][1], dummyMatrix[2][2]);
00068         }
00069         itr--;
00070     }
00071 }

```

Benutzt [CEulerMatrix::getEulerMatrix\(\)](#), [CEulerMatrix::getMatrix\(\)](#), [path](#) und [step](#).

Wird benutzt von [CSegmentApproximator::approx\(\)](#), [CPathBuilder::createPath\(\)](#), [CRobCodeGenerator::generateRobCode\(\)](#) und [CMeanFilter::mean\(\)](#).

7.5.3.5 setStep()

```

void CLogging::setStep (
    int Step )

```

Setzt in welchem Schritt wir uns befinden.

Parameter

<i>int</i>	Step
------------	------

Definiert in Zeile 32 der Datei [Logging.cpp](#).

```
00033 {  
00034     step = Step;    // Step setzen  
00035 }
```

Benutzt [step](#).

Wird benutzt von [CSegmentApproximator::approx\(\)](#), [CPathBuilder::createPath\(\)](#), [CRobCodeGenerator::generateRobCode\(\)](#) und [CMeanFilter::mean\(\)](#).

7.5.4 Dokumentation der Datenelemente

7.5.4.1 detailed

```
bool CLogging::detailed [private]
```

Speicherpfad

Definiert in Zeile 81 der Datei [Logging.h](#).

Wird benutzt von [CLogging\(\)](#) und [getDetailed\(\)](#).

7.5.4.2 path

```
string CLogging::path [private]
```

Speicherpfad

Definiert in Zeile 77 der Datei [Logging.h](#).

Wird benutzt von [CLogging\(\)](#), [logData\(\)](#), [logData\(\)](#) und [logData\(\)](#).

7.5.4.3 step

```
int CLogging::step [private]
```

In welchem Schritt sind wir gerade

Definiert in Zeile 73 der Datei [Logging.h](#).

Wird benutzt von [CLogging\(\)](#), [logData\(\)](#), [logData\(\)](#), [logData\(\)](#) und [setStep\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- header/[Logging.h](#)
- source/[Logging.cpp](#)

7.6 CMeanFilter Klassenreferenz

Gleitender Mittelwertfilter.

```
#include <MeanFilter.h>
```

Öffentliche Methoden

- [CMeanFilter](#) ()
Default Konstruktor.
- [CMeanFilter](#) (int Window)
Konstruktor.
- [~CMeanFilter](#) ()
Dekonstruktor.
- void [setWindowSize](#) (int Window)
Setzt das Fenster.
- int [getWindowSize](#) ()
Gibt das gesetzte Fenster zurueck.
- vector< list< [CInputPoint3D](#) > > & [getPath](#) ()
Gibt den gegl'aetteten Pfad zurueck.
- list< [CInputPoint3D](#) > [calculateMean](#) (list< [CInputPoint3D](#) > &segment)
Berechnet gleitenden Mittelwert.
- void [mean](#) (vector< list< [CInputPoint3D](#) > > &sourcePath, [CLogging](#) log)
Ruft calculateMean fuer die einzelnen Segmente auf.

Private Attribute

- int [windowSize](#)
- vector< list< [CInputPoint3D](#) > > [meanPath](#)

7.6.1 Ausführliche Beschreibung

Gleitender Mittelwertfilter.

In dieser Klasse werden die eingelesenen Daten mit einem gleitenden Mittelwertfilter mit einstellbarem Fenster gegl'aettet.

Definiert in Zeile 21 der Datei [MeanFilter.h](#).

7.6.2 Beschreibung der Konstruktoren und Destruktoren

7.6.2.1 CMeanFilter() [1/2]

```
CMeanFilter::CMeanFilter ( )
```

Default Konstruktor.

Initialisiert des Fensters mit 3 als default Wert

Siehe auch

[CMeanFilter\(int Window\);](#)

Definiert in Zeile 11 der Datei [MeanFilter.cpp](#).

```
00012 {
00013     windowSize = 3;           // initialisieren mit Standardfenstergroesse 3
00014 }
```

Benutzt [windowSize](#).

7.6.2.2 CMeanFilter() [2/2]

```
CMeanFilter::CMeanFilter (
    int Window )
```

Konstruktor.

Initialisiert die Input Daten mit Null

Parameter

<i>int</i>	Window Konstruktor der Klasse mit Fenster @seeCMeanFilter();
------------	--

Definiert in Zeile 16 der Datei [MeanFilter.cpp](#).

```
00017 {
00018     windowSize = Window; // initialisieren der Fenstergroesse mit Uebergabewert
00019 }
```

Benutzt [windowSize](#).

7.6.2.3 ~CMeanFilter()

```
CMeanFilter::~~CMeanFilter ( )
```

Dekonstruktor.

Definiert in Zeile 21 der Datei [MeanFilter.cpp](#).

```
00022 {
00023 }
```

7.6.3 Dokumentation der Elementfunktionen

7.6.3.1 calculateMean()

```
list< CInputPoint3D > CMeanFilter::calculateMean (
    list< CInputPoint3D > & segment )
```

Berechnet gleitenden Mittelwert.

Hier wird der Mittelwert der einzelnen Segmente berechnet

Parameter

<i>list< CInputPoint3D> &</i>	segment bekommt einzelne Segmente
---	-----------------------------------

Rückgabe

: list<CInputPoint3D> gibt gelaettete Segmente zurueck

Definiert in Zeile 55 der Datei [MeanFilter.cpp](#).

```
00056 {
```

```

00057     double sumX = 0, sumY = 0, sumZ = 0;    // Variablen zum Speichern der Summe
00058     double div = 0;                        // Variable zum Speichern des Teilers
00059
00060     CInputPoint3D p;                       //Point3D zum Zwischenspeichern
00061
00062     size_t inputSize = segment.size();
00063
00064     list<CInputPoint3D>::iterator it = segment.begin();
00065     list<CInputPoint3D> newSegment;
00066
00067     for (size_t i = 0; i < inputSize - windowSize; ++i) //For each element in the Segment
00068     {
00069         sumX = 0, sumY = 0, sumZ = 0;    // Variablen zum Speichern der Summe auf 0 zurueck setzen
00070         div = 0;                        // Variable zum Speichern des Teilers auf 0 zurueck setzen
00071         p.setTime(it->getTime());
00072         p.setEulerMatrix(it->getEulerMatrix());
00073         for (size_t j = i; j < i + windowSize; ++j) // Build the sums for the three points
00074         {
00075             sumX += it->getX();
00076             sumY += it->getY();
00077             sumZ += it->getZ();
00078             div++;
00079             it++;
00080         }
00081         for (size_t index = windowSize; index > 0; index--) // Pain, the iterator has to be set back
00082         {
00083             it--;
00084         }
00085         p.set(sumX / div, sumY / div, sumZ / div); // Calculate smoothed values
00086         if(it != segment.end())
00087             it++;
00088         newSegment.push_back(p);
00089     }
00090     return newSegment;
00091 }

```

Benutzt [CPoint3D::set\(\)](#), [CInputPoint3D::setEulerMatrix\(\)](#), [CInputPoint3D::setTime\(\)](#) und [windowSize](#).

Wird benutzt von [mean\(\)](#).

7.6.3.2 getPath()

```
vector< list< CInputPoint3D > > & CMeanFilter::getPath ( )
```

Gibt den geglaetteten Pfad zurueck.

Rückgabe

: vector<list<CInputPoint3D>>

Definiert in Zeile 35 der Datei [MeanFilter.cpp](#).

```

00036 {
00037     return meanPath;           // Mittelwert zurueck geben
00038 }

```

Benutzt [meanPath](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.6.3.3 getWindowSize()

```
int CMeanFilter::getWindowSize ( )
```

Gibt das gesetzte Fenster zurueck.

Rückgabe

: Window int

Definiert in Zeile 30 der Datei [MeanFilter.cpp](#).

```

00031 {
00032     return windowSize;        // Fenstergroesse zurueck geben
00033 }

```

Benutzt [windowSize](#).

7.6.3.4 mean()

```
void CMeanFilter::mean (
    vector< list< CInputPoint3D > > & sourcePath,
    CLogging log )
```

Ruft calculateMean fuer die einzelnen Segmente auf.

Hier wird durch die Segmente iteriert und fuer jedes die calculate Mean Funktion aufgerufen. Anschliessend werden sie in meanPath abgespeichert.

Parameter

<i>list< CInputPoint3D>&</i>	segment bekommt einzelne Segmente
<i>CLogging</i>	log fuer das Logging

Definiert in Zeile 40 der Datei [MeanFilter.cpp](#).

```
00041 {
00042     list<CInputPoint3D> dummyList;
00043     for (size_t s = 0; s < sourcePath.size(); s++)
00044     {
00045         dummyList = calculateMean(sourcePath[s]);
00046         meanPath.push_back(dummyList);
00047     }
00048     if (log.getDetailed())
00049     {
00050         log.setStep(1);
00051         log.logData(meanPath);
00052     }
00053 }
```

Benutzt [calculateMean\(\)](#), [CLogging::getDetailed\(\)](#), [CLogging::logData\(\)](#), [meanPath](#) und [CLogging::setStep\(\)](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.6.3.5 setWindowSize()

```
void CMeanFilter::setWindowSize (
    int Window )
```

Setzt das Fenster.

Parameter

<i>Window</i>	int
---------------	-----

Definiert in Zeile 25 der Datei [MeanFilter.cpp](#).

```
00026 {
00027     windowSize = Window; // setzen der Fenstergroesse mit Uebergabewert
00028 }
```

Benutzt [windowSize](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.6.4 Dokumentation der Datenelemente

7.6.4.1 meanPath

```
vector<list<CInputPoint3D> > CMeanFilter::meanPath [private]
```

Hier werden die gelaetzten Daten gespeichert

Definiert in Zeile 83 der Datei [MeanFilter.h](#).

Wird benutzt von [getPath\(\)](#) und [mean\(\)](#).

7.6.4.2 windowSize

```
int CMeanFilter::windowSize [private]
```

Groesse des Fensters des gleitenden Mittelwerts

Definiert in Zeile 79 der Datei [MeanFilter.h](#).

Wird benutzt von [calculateMean\(\)](#), [CMeanFilter\(\)](#), [CMeanFilter\(\)](#), [getWindowSize\(\)](#) und [setWindowSize\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

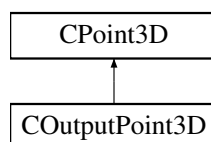
- header/[MeanFilter.h](#)
- source/[MeanFilter.cpp](#)

7.7 COutputPoint3D Klassenreferenz

Output Punkt.

```
#include <Point3D.h>
```

Klassendiagramm für COutputPoint3D:



Öffentliche Methoden

- **COutputPoint3D** (void)
Default Konstruktor.
- **COutputPoint3D** (double Speed, double X, double Y, double Z, double A, double B, double C)
Default Konstruktor.
- **~COutputPoint3D** (void)
Dekonstruktor.
- double **getSpeed** ()
Gibt die Geschwindigkeit zurueck.
- double **getA** ()
Gibt A zurueck.
- double **getB** ()
Gibt B zurueck.
- double **getC** ()
Gibt C zurueck.
- void **setSpeed** (double speed)
Setzt die Geschwindigkeit.
- void **setA** (double A)
Setzt A.
- void **setB** (double B)
Setzt B.
- void **setC** (double C)
Setzt C.

Öffentliche Methoden geerbt von **CPoint3D**

- **CPoint3D** (void)
Default Konstruktor.
- **CPoint3D** (double X, double Y, double Z)
Default Konstruktor.
- **~CPoint3D** (void)
Dekonstruktor.
- double **getX** ()
Gibt X zurueck.
- double **getY** ()
Gibt Y zurueck.
- double **getZ** ()
Gibt Z zurueck.
- void **setX** (double X)
Setzt X.
- void **setY** (double Y)
Setzt Y.
- void **setZ** (double Z)
Setzt Z.
- void **set** (double X, double Y, double Z)
Setzt X, Y und Z.
- double **distanceTo** (**CPoint3D** point)
Berechnet die Distanz zu einem anderen Punkt.
- double **distanceTo** (**CLine3D** line)
Berechnet die Distanz zu einer Linie.

Private Attribute

- double [a](#)
- double [b](#)
- double [c](#)
- double [speed](#)

Weitere Geerbte Elemente

Geschützte Attribute geerbt von [CPoint3D](#)

- double [x](#)
- double [y](#)
- double [z](#)

7.7.1 Ausführliche Beschreibung

Output Punkt.

Kind der Punkt Grundklasse, erweitert um die Geschwindigkeit und die Drehwinkel

Definiert in Zeile [177](#) der Datei [Point3D.h](#).

7.7.2 Beschreibung der Konstruktoren und Destruktoren

7.7.2.1 COutputPoint3D() [1/2]

```
COutputPoint3D::COutputPoint3D (  
    void )
```

Default Konstruktor.

Initialisiert des fertig bearbeiteten Punktes mit Null

Siehe auch

[COutputPoint3D\(double Speed, double X, double Y, double Z, double A, double B, double C\);](#)

Definiert in Zeile [159](#) der Datei [Point3D.cpp](#).

```
00159                                     : CPoint3D()  
00160 {  
00161     speed = 0;  
00162     a = 0;  
00163     b = 0;  
00164     c = 0;  
00165 }
```

Benutzt [a](#), [b](#), [c](#) und [speed](#).

7.7.2.2 COutputPoint3D() [2/2]

```
COutputPoint3D::COutputPoint3D (  
    double Speed,  
    double X,  
    double Y,  
    double Z,  
    double A,  
    double B,  
    double C )
```

Default Konstruktor.

Initialisiert des eingelesenen Punktes mit Null

Parameter

<i>double</i>	Speed
<i>double</i>	X
<i>double</i>	Y
<i>double</i>	Z
<i>double</i>	A
<i>double</i>	B
<i>double</i>	C

Siehe auch

[CInputPoint3D\(double X, double Y, double Z, double Timestamp, CEulerMatrix Matrix\)](#)

Definiert in Zeile 168 der Datei [Point3D.cpp](#).

```
00169 {
00170     speed = Speed;
00171     a = A;
00172     b = B;
00173     c = C;
00174     x = X;
00175     y = Y;
00176     z = Z;
00177 }
```

Benutzt [a](#), [b](#), [c](#), [speed](#), [CPoint3D::x](#), [CPoint3D::y](#) und [CPoint3D::z](#).

7.7.2.3 ~COutputPoint3D()

```
COutputPoint3D::~COutputPoint3D (
    void )
```

Dekonstruktor.

Definiert in Zeile 179 der Datei [Point3D.cpp](#).

```
00180 {
00181
00182 }
```

7.7.3 Dokumentation der Elementfunktionen

7.7.3.1 getA()

```
double COutputPoint3D::getA (
    void )
```

Gibt A zurueck.

Rückgabe

: double A

Definiert in Zeile 184 der Datei [Point3D.cpp](#).

```
00185 {
00186     return a; // Rueckgabe Winkel alpha
00187 }
```

Benutzt [a](#).

7.7.3.2 getB()

```
double COutputPoint3D::getB (
    void )
```

Gibt B zurueck.

Rückgabe

: double B

Definiert in Zeile 189 der Datei [Point3D.cpp](#).

```
00190 {
00191     return b; // Rueckgabe Winkel beta
00192 }
```

Benutzt [b](#).

7.7.3.3 getC()

```
double COutputPoint3D::getC (
    void )
```

Gibt C zurueck.

Rückgabe

: double C

Definiert in Zeile 194 der Datei [Point3D.cpp](#).

```
00195 {
00196     return c; // Rueckgabe Winkel gamma
00197 }
```

Benutzt [c](#).

7.7.3.4 getSpeed()

```
double COutputPoint3D::getSpeed (
    void )
```

Gibt die Geschwindigkeit zurueck.

Rückgabe

: double Geschwindigkeit

Definiert in Zeile 199 der Datei [Point3D.cpp](#).

```
00200 {
00201     return speed; // Rueckgabe Geschwindigkeit
00202 }
```

Benutzt [speed](#).

7.7.3.5 setA()

```
void COutputPoint3D::setA (
    double A )
```

Setzt A.

Parameter

<i>double</i>	A
---------------	---

Definiert in Zeile 204 der Datei [Point3D.cpp](#).

```
00205 {  
00206     a = A;      // setze Winkel alpha  
00207 }
```

Benutzt [a](#).

Wird benutzt von [CRobCodeGenerator::calculateAngles\(\)](#) und [CRobCodeGenerator::postProcessing\(\)](#).

7.7.3.6 setB()

```
void COutputPoint3D::setB (  
    double B )
```

Setzt B.

Parameter

<i>double</i>	B
---------------	---

Definiert in Zeile 209 der Datei [Point3D.cpp](#).

```
00210 {  
00211     b = B;      // setze Winkel beta  
00212 }
```

Benutzt [b](#).

Wird benutzt von [CRobCodeGenerator::calculateAngles\(\)](#) und [CRobCodeGenerator::postProcessing\(\)](#).

7.7.3.7 setC()

```
void COutputPoint3D::setC (  
    double C )
```

Setzt C.

Parameter

<i>double</i>	C
---------------	---

Definiert in Zeile 214 der Datei [Point3D.cpp](#).

```
00215 {  
00216     c = C;      // setze Winkel gamma  
00217 }
```

Benutzt [c](#).

Wird benutzt von [CRobCodeGenerator::calculateAngles\(\)](#) und [CRobCodeGenerator::postProcessing\(\)](#).

7.7.3.8 setSpeed()

```
void COutputPoint3D::setSpeed (
    double speed )
```

Setzt die Geschwindigkeit.

Parameter

<i>double</i>	speed
---------------	-------

Definiert in Zeile 219 der Datei [Point3D.cpp](#).

```
00220 {
00221     speed = Speed;      // setze Geschwindigkeit
00222 }
```

Benutzt [speed](#).

Wird benutzt von [CRobCodeGenerator::postProcessing\(\)](#).

7.7.4 Dokumentation der Datenelemente

7.7.4.1 a

```
double COutputPoint3D::a [private]
```

Drehwinkel des Punktes

Definiert in Zeile 249 der Datei [Point3D.h](#).

Wird benutzt von [COutputPoint3D\(\)](#), [COutputPoint3D\(\)](#), [getA\(\)](#) und [setA\(\)](#).

7.7.4.2 b

```
double COutputPoint3D::b [private]
```

Definiert in Zeile 249 der Datei [Point3D.h](#).

Wird benutzt von [COutputPoint3D\(\)](#), [COutputPoint3D\(\)](#), [getB\(\)](#) und [setB\(\)](#).

7.7.4.3 c

```
double COutputPoint3D::c [private]
```

Definiert in Zeile 249 der Datei [Point3D.h](#).

Wird benutzt von [COutputPoint3D\(\)](#), [COutputPoint3D\(\)](#), [getC\(\)](#) und [setC\(\)](#).

7.7.4.4 speed

```
double COutputPoint3D::speed [private]
```

Geschwindigkeit zum Punkt hin? TODO: ueberpruefen

Definiert in Zeile 253 der Datei [Point3D.h](#).

Wird benutzt von [COutputPoint3D\(\)](#), [COutputPoint3D\(\)](#), [getSpeed\(\)](#) und [setSpeed\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- header/[Point3D.h](#)
- source/[Point3D.cpp](#)

7.8 CPathBuilder Klassenreferenz

Zusammensetzten des Pfades.

```
#include <PathBuilder.h>
```

Öffentliche Methoden

- [CPathBuilder](#) (void)
Default Konstruktor.
- [~CPathBuilder](#) (void)
Dekonstruktor.
- [vector< CInputPoint3D > & getPath \(\)](#)
Gibt Pfad zurueck.
- [void createPath](#) ([vector< list< CInputPoint3D > > &segments](#), [CLogging](#) log)
Gibt Pfad zurueck.

Private Attribute

- [vector< CInputPoint3D > path](#)

7.8.1 Ausführliche Beschreibung

Zusammensetzten des Pfades.

In dieser Klasse wird aus den Segmenten ein zusammenhaengender Pfad erstellt

Definiert in Zeile 21 der Datei [PathBuilder.h](#).

7.8.2 Beschreibung der Konstruktoren und Destruktoren

7.8.2.1 CPathBuilder()

```
CPathBuilder::CPathBuilder (
    void )
```

Default Konstruktor.

Initialisiert der Klasse

Definiert in Zeile 9 der Datei [PathBuilder.cpp](#).

```
00010 {
00011 }
```

7.8.2.2 ~CPathBuilder()

```
CPathBuilder::~CPathBuilder (
    void )
```

Dekonstruktor.

Definiert in Zeile 14 der Datei [PathBuilder.cpp](#).

```
00015 {
00016 }
```

7.8.3 Dokumentation der Elementfunktionen

7.8.3.1 createPath()

```
void CPathBuilder::createPath (
    vector< list< CInputPoint3D > > & segments,
    CLogging log )
```

Gibt Pfad zurueck.

Parameter

<i>segments</i>	vector<list<CInputPoint3D>>& Pfad aus Segmenten
<i>CLogging</i>	log fuer das Logging

Definiert in Zeile 23 der Datei [PathBuilder.cpp](#).

```
00024 {
00025     CInputPoint3D point; //startpoint
00026
00027     for (size_t s = 0; s < segments.size(); s++) //for all segments
00028     {
00029         list<CInputPoint3D>::iterator itr = segments[s].begin();
00030
00031         for (; itr != segments[s].end(); itr++) //for all points in the segment
00032         {
00033             point.set((double)itr->getX(), (double)itr->getY(), (double)itr->getZ());
00034             point.setTime(itr->getTime());
00035             point.setEulerMatrix(itr->getEulerMatrix());
00036             path.push_back(point);
00037         }
00038 }
```

```
00039         itr--;
00040     }
00041     if (log.getDetailed())
00042     {
00043         log.setStep(3);
00044         log.logData(path);
00045     }
00046 }
```

Benutzt [CLogging::getDetailed\(\)](#), [CLogging::logData\(\)](#), [path](#), [CPoint3D::set\(\)](#), [CInputPoint3D::setEulerMatrix\(\)](#), [CLogging::setStep\(\)](#) und [CInputPoint3D::setTime\(\)](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.8.3.2 getPath()

```
vector< CInputPoint3D > & CPathBuilder::getPath ( )
```

Gibt Pfad zurueck.

Rückgabe

: vector<CInputPoint3D> zusammengesetzter Pfad

Definiert in Zeile 18 der Datei [PathBuilder.cpp](#).

```
00019 {
00020     return path;
00021 }
```

Benutzt [path](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.8.4 Dokumentation der Datenelemente

7.8.4.1 path

```
vector<CInputPoint3D> CPathBuilder::path [private]
```

Vector mit den zusammengesetzten Pfad Daten

Definiert in Zeile 50 der Datei [PathBuilder.h](#).

Wird benutzt von [createPath\(\)](#) und [getPath\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

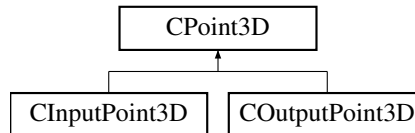
- header/[PathBuilder.h](#)
- source/[PathBuilder.cpp](#)

7.9 CPoint3D Klassenreferenz

Grundklasse Punkt.

```
#include <Point3D.h>
```

Klassendiagramm für CPoint3D:



Öffentliche Methoden

- **CPoint3D** (void)
Default Konstruktor.
- **CPoint3D** (double X, double Y, double Z)
Default Konstruktor.
- **~CPoint3D** (void)
Dekonstruktor.
- double **getX** ()
Gibt X zurueck.
- double **getY** ()
Gibt Y zurueck.
- double **getZ** ()
Gibt Z zurueck.
- void **setX** (double X)
Setzt X.
- void **setY** (double Y)
Setzt Y.
- void **setZ** (double Z)
Setzt Z.
- void **set** (double X, double Y, double Z)
Setzt X, Y und Z.
- double **distanceTo** (**CPoint3D** point)
Berechnet die Distanz zu einem anderen Punkt.
- double **distanceTo** (**CLine3D** line)
Berechnet die Distanz zu einer Linie.

Geschützte Attribute

- double **x**
- double **y**
- double **z**

7.9.1 Ausführliche Beschreibung

Grundklasse Punkt.

Das ist die Grundklasse eines Punktes, hier lassen sich die Basiswerte setzen und Abstände zwischen Punkten berechnen.

Definiert in Zeile 20 der Datei [Point3D.h](#).

7.9.2 Beschreibung der Konstruktoren und Destruktoren

7.9.2.1 CPoint3D() [1/2]

```
CPoint3D::CPoint3D (
    void )
```

Default Konstruktor.

Initialisiert des Punktes mit Null

Siehe auch

[CPoint3D\(double X, double Y, double Z\)](#)

Definiert in Zeile 13 der Datei [Point3D.cpp](#).

```
00014 {
00015     x = 0;
00016     y = 0;
00017     z = 0;
00018 }
```

Benutzt [x](#), [y](#) und [z](#).

7.9.2.2 CPoint3D() [2/2]

```
CPoint3D::CPoint3D (
    double X,
    double Y,
    double Z )
```

Default Konstruktor.

Initialisiert des Punktes mit Null

Parameter

<i>double</i>	X
<i>double</i>	Y
<i>double</i>	Z

Siehe auch

[CPoint3D\(void\)](#)

Definiert in Zeile 21 der Datei [Point3D.cpp](#).

```
00022 {
00023     x = X;
00024     y = Y;
00025     z = Z;
00026 }
```

Benutzt [x](#), [y](#) und [z](#).

7.9.2.3 ~CPoint3D()

```
CPoint3D::~~CPoint3D (
    void )
```

Dekonstruktor.

Definiert in Zeile 28 der Datei [Point3D.cpp](#).

```
00029 {
00030 }
```

7.9.3 Dokumentation der Elementfunktionen

7.9.3.1 distanceTo() [1/2]

```
double CPoint3D::distanceTo (
    CLine3D line )
```

Berechnet die Distanz zu einer Linie.

Parameter

CLine3D	line
-------------------------	------

Rückgabe

double Distanz

Definiert in Zeile 76 der Datei [Point3D.cpp](#).

```
00077 {
00078     double bx, by, bz, rv_sq, dist, vp1, vp2, vp3;           // Variablen Anlegen
00079
00080     /*
00081     Vermessen wird der Punkt selbst
00082
00083     bx, by, bz      == Vektordifferenz
00084     rv_sq           == Betrag des Linienvektors
00085     dist            == Distanz von Punkt zu Linie
00086     vp1, vp2, vp3   == Vektorprodukte
00087     */
00088
00089     int rvx = line.p1.x - line.p2.x;    // Parameter X des Linienvektor berechnen
00090     int rvy = line.p1.y - line.p2.y;    // Parameter Y des Linienvektor berechnen
00091     int rvz = line.p1.z - line.p2.z;    // Parameter Z des Linienvektor berechnen
00092
00093     rv_sq = sqrt(((double)rvx * (double)rvx) + ((double)rvy * (double)rvy) + ((double)rvz *
(double)rvz));    // Betrag des Linienvektor berechnen
```



```

00094
00095     bx = x - (double)line.p1.x;           // X(Punkt) - X(Aufpunkt)
00096     by = y - (double)line.p1.y;           // Y(Punkt) - Y(Aufpunkt)
00097     bz = z - (double)line.p1.z;           // Z(Punkt) - Z(Aufpunkt)
00098
00099     vp1 = by * rvz - bz * rvy;             // Parameter X Vektorprodukt
00100     vp2 = bz * rvx - bx * rvz;             // Parameter Y Vektorprodukt
00101     vp3 = bx * rvy - by * rvx;             // Parameter Z Vektorprodukt
00102
00103     dist = sqrt(vp1 * vp1 + vp2 * vp2 + vp3 * vp3) / rv_sq; // Betrag des Vektors berechnen
00104
00105     return dist;
00106 }

```

Benutzt [CLine3D::p1](#), [CLine3D::p2](#), [x](#), [y](#) und [z](#).

7.9.3.2 distanceTo() [2/2]

```

double CPoint3D::distanceTo (
    CPoint3D point )

```

Berechnet die Distanz zu einem anderen Punkt.

Parameter

CPoint3D	point
--------------------------	-------

Rückgabe

double Distanz

Definiert in Zeile [71](#) der Datei [Point3D.cpp](#).

```

00072 {
00073     return sqrt(pow((double)(x - (double)point.getX()), 2) + pow((double)(y - (double)point.getY()),
00074     2) + pow((double)(z - (double)point.getZ()), 2)); // Pythagoras 3D
00074 }

```

Benutzt [getX\(\)](#), [getY\(\)](#), [getZ\(\)](#), [x](#), [y](#) und [z](#).

7.9.3.3 getX()

```

double CPoint3D::getX (
    void )

```

Gibt X zurueck.

Rückgabe

double

Definiert in Zeile [32](#) der Datei [Point3D.cpp](#).

```

00033 {
00034     return x; // X-Koordinate zurueck geben
00035 }

```

Benutzt [x](#).

Wird benutzt von [CInputParameter::detectJump\(\)](#) und [distanceTo\(\)](#).

7.9.3.4 getY()

```
double CPoint3D::getY (
    void )
```

Gibt Y zurueck.

Rückgabe

double

Definiert in Zeile 37 der Datei [Point3D.cpp](#).

```
00038 {
00039     return y; // Y-Koordinate zurueck geben
00040 }
```

Benutzt [y](#).

Wird benutzt von [CInputParameter::detectJump\(\)](#) und [distanceTo\(\)](#).

7.9.3.5 getZ()

```
double CPoint3D::getZ (
    void )
```

Gibt Z zurueck.

Rückgabe

double

Definiert in Zeile 42 der Datei [Point3D.cpp](#).

```
00043 {
00044     return z; // Z-Koordinate zurueck geben
00045 }
```

Benutzt [z](#).

Wird benutzt von [CInputParameter::detectJump\(\)](#) und [distanceTo\(\)](#).

7.9.3.6 set()

```
void CPoint3D::set (
    double X,
    double Y,
    double Z )
```

Setzt X, Y und Z.

Parameter

<i>double</i>	X
<i>double</i>	Y
<i>double</i>	Z

Definiert in Zeile 63 der Datei [Point3D.cpp](#).

```
00064 {  
00065     x = X; // X-Koordinate setzen  
00066     y = Y; // Y-Koordinate setzen  
00067     z = Z; // Z-Koordinate setzen  
00068 }
```

Benutzt [x](#), [y](#) und [z](#).

Wird benutzt von [CMeanFilter::calculateMean\(\)](#), [CPathBuilder::createPath\(\)](#), [CRobCodeGenerator::postProcessing\(\)](#) und [CInputPoint3D::setPoint\(\)](#).

7.9.3.7 setX()

```
void CPoint3D::setX (  
    double X )
```

Setzt X.

Parameter

<i>double</i>	X
---------------	---

Definiert in Zeile 47 der Datei [Point3D.cpp](#).

```
00048 {  
00049     x = X; // X-Koordinate setzen  
00050 }
```

Benutzt [x](#).

Wird benutzt von [CSegmentApproximator::douglasPeuckerRecursive\(\)](#).

7.9.3.8 setY()

```
void CPoint3D::setY (  
    double Y )
```

Setzt Y.

Parameter

<i>double</i>	Y
---------------	---

Definiert in Zeile 52 der Datei [Point3D.cpp](#).

```
00053 {  
00054     y = Y; // Y-Koordinate setzen  
00055 }
```

Benutzt [y](#).

Wird benutzt von [CSegmentApproximator::douglasPeuckerRecursive\(\)](#).

7.9.3.9 setZ()

```
void CPoint3D::setZ (  
    double Z )
```

Setzt Z.

Parameter

<i>double</i>	<i>Z</i>
---------------	----------

Definiert in Zeile 57 der Datei [Point3D.cpp](#).

```
00058 {  
00059     z = Z; // Z-Koordinate setzen  
00060 }
```

Benutzt [z](#).

Wird benutzt von [CSegmentApproximator::douglasPeuckerRecursive\(\)](#).

7.9.4 Dokumentation der Datenelemente

7.9.4.1 x

```
double CPoint3D::x [protected]
```

Koordinaten des Punkts

Definiert in Zeile 99 der Datei [Point3D.h](#).

Wird benutzt von [CInputPoint3D::CInputPoint3D\(\)](#), [COutputPoint3D::COutputPoint3D\(\)](#), [CPoint3D\(\)](#), [CPoint3D\(\)](#), [distanceTo\(\)](#), [distanceTo\(\)](#), [getX\(\)](#), [set\(\)](#) und [setX\(\)](#).

7.9.4.2 y

```
double CPoint3D::y [protected]
```

Definiert in Zeile 99 der Datei [Point3D.h](#).

Wird benutzt von [CInputPoint3D::CInputPoint3D\(\)](#), [COutputPoint3D::COutputPoint3D\(\)](#), [CPoint3D\(\)](#), [CPoint3D\(\)](#), [distanceTo\(\)](#), [distanceTo\(\)](#), [getY\(\)](#), [set\(\)](#) und [setY\(\)](#).

7.9.4.3 z

```
double CPoint3D::z [protected]
```

Definiert in Zeile 99 der Datei [Point3D.h](#).

Wird benutzt von [CInputPoint3D::CInputPoint3D\(\)](#), [COutputPoint3D::COutputPoint3D\(\)](#), [CPoint3D\(\)](#), [CPoint3D\(\)](#), [distanceTo\(\)](#), [distanceTo\(\)](#), [getZ\(\)](#), [set\(\)](#) und [setZ\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- header/[Point3D.h](#)
- source/[Point3D.cpp](#)

7.10 CRobCodeGenerator Klassenreferenz

Klasse zum erstellen des Roboter Codes.

```
#include <RobCodeGenerator.h>
```

Öffentliche Methoden

- [CRobCodeGenerator](#) (void)
Default Konstruktor.
- [CRobCodeGenerator](#) ([CInputParameter](#) inputParam)
Konstruktor.
- [~CRobCodeGenerator](#) (void)
Dekonstruktor.
- void [generateRobCode](#) (vector< [CInputPoint3D](#) > &path, string filepath, string filename, [CLogging](#) log)
Erstellt Roboter Code File.
- void [postProcessing](#) (vector< [CInputPoint3D](#) > &path)
Nachbearbeitung der Daten.
- double [calculateSpeed](#) ([CInputPoint3D](#) &p, size_t i, double timePrev)
Berechnet die Geschwindigkeit zwischen zwei Punkten.
- void [calculateAngles](#) ([COutputPoint3D](#) &p, [CInputPoint3D](#) &pIn)
Berechnet die Geschwindigkeit zwischen zwei Punkten.

Private Attribute

- vector< [COutputPoint3D](#) > [processedPath](#)
- [CInputParameter](#) [input](#)

7.10.1 Ausführliche Beschreibung

Klasse zum erstellen des Roboter Codes.

In dieser Klasse wird die Nachbearbeitung der Daten mit den einstellbaren Daten durchgeführt.

Definiert in Zeile 25 der Datei [RobCodeGenerator.h](#).

7.10.2 Beschreibung der Konstruktoren und Destruktoren

7.10.2.1 CRobCodeGenerator() [1/2]

```
CRobCodeGenerator::CRobCodeGenerator (
    void )
```

Default Konstruktor.

Initialisiert der Daten

Siehe auch

[CRobCodeGenerator\(double speedIn, bool speedManualIn, bool orientationManualIn, tuple<double, double, double> angles\)](#)

Definiert in Zeile 10 der Datei [RobCodeGenerator.cpp](#).

```
00011 {
00012 }
```

7.10.2.2 CRobCodeGenerator() [2/2]

```
CRobCodeGenerator::CRobCodeGenerator (
    CInputParameter inputParam )
```

Konstruktor.

Initialisiert der Daten

Parameter

<i>initSpeed</i>	double
<i>initSpeedManual</i>	bool
<i>initOrientationManual</i>	bool
<i>initA</i>	double
<i>initB</i>	double
<i>initC</i>	double

Siehe auch

[CRobCodeGenerator\(void\)](#)

Definiert in Zeile 15 der Datei [RobCodeGenerator.cpp](#).

```
00016 {
00017     input = inputParam;
00018 }
```

Benutzt [input](#).

7.10.2.3 ~CRobCodeGenerator()

```
CRobCodeGenerator::~~CRobCodeGenerator (
    void )
```

Dekonstruktor.

Definiert in Zeile 20 der Datei [RobCodeGenerator.cpp](#).

```
00021 {
00022 }
```

7.10.3 Dokumentation der Elementfunktionen

7.10.3.1 calculateAngles()

```
void CRobCodeGenerator::calculateAngles (
    COutputPoint3D & p,
    CInputPoint3D & pIn )
```

Berechnet die Geschwindigkeit zwischen zwei Punkten.

Parameter

<i>COutputPoint3D</i> &	p
<i>CInputPoint3D</i> &	p↔ In

Definiert in Zeile 131 der Datei [RobCodeGenerator.cpp](#).

```
00132 {
00133     // Funktion in Eulermatrix aufrufen die a/b/c neu berechnet
00134
00135     CEulerMatrix matrix = pIn.getEulerMatrix();
00136     tuple<double, double, double> abc;
00137
00138     abc = matrix.calculateAngels();
00139
00140     p.setA(get<0>(abc));
00141     p.setB(get<1>(abc));
00142     p.setC(get<2>(abc));
00143 }
```

Benutzt [CEulerMatrix::calculateAngels\(\)](#), [CInputPoint3D::getEulerMatrix\(\)](#), [COutputPoint3D::setA\(\)](#), [COutputPoint3D::setB\(\)](#) und [COutputPoint3D::setC\(\)](#).

Wird benutzt von [postProcessing\(\)](#).

7.10.3.2 calculateSpeed()

```
double CRobCodeGenerator::calculateSpeed (
    CInputPoint3D & p,
    size_t i,
    double timePrev )
```

Berechnet die Geschwindigkeit zwischen zwei Punkten.

Parameter

<i>CInputPoint3D</i> &	p aktueller Punkt
<i>size_t</i>	i Position im processedPath
<i>double</i>	timePrev Zeitstempel des vorherigen Punkts

Rückgabe

double

Definiert in Zeile 114 der Datei [RobCodeGenerator.cpp](#).

```
00115 {
00116     double distance = 0;
00117     double time = 0;
00118     double speed;
00119
00120     distance = processedPath[s - 1].distanceTo(p); //Strecke zwischen p und dem Punkt zuvor
00121     time = p.getTime() - timePrev; //Zeit zwischen p-1 und p
00122
00123     speed = distance / time; // Berechnung Geschwindigkeit zwischen zwei Punkten
00124
00125     if (speed > MAX_SPEED) //Begrenzung auf maximale Geschwindigkeit, falls Trackerdaten höheren Wert
aufweisen
00126         speed = MAX_SPEED;
00127
00128     return speed; //Zuweisung der Geschwindigkeit
00129 }
```

Benutzt [CInputPoint3D::getTime\(\)](#), [MAX_SPEED](#) und [processedPath](#).

Wird benutzt von [postProcessing\(\)](#).

7.10.3.3 generateRobCode()

```
void CRobCodeGenerator::generateRobCode (
    vector< CInputPoint3D > & path,
    string filepath,
    string filename,
    CLogging log )
```

Erstellt Roboter Code File.

Ruft das Postprocessing auf und speichert die bearbeiteten Daten als .src File ab

Parameter

<i>vector<CInputPoint3D>&</i>	path
<i>string</i>	filename

Definiert in Zeile 24 der Datei [RobCodeGenerator.cpp](#).

```
00025 {
00026     string filename;
00027
00028     postProcessing(points); // Calculates all the necessary values
00029
00030     log.setStep(4);
00031     log.logData(processedPath);
00032
00033     errno_t err;
00034
00035     FILE* fid;
00036
00037     filename = inputPath.substr(inputPath.find_last_of("/") + 1);
00038     filename.erase(filename.end() - 4, filename.end());
00039     filename = filename + ".src";
00040
00041     string fullPath = filepath + "/" + filename;
00042
00043     if ((err = fopen_s(&fid, fullPath.c_str(), "w")) != 0) // Errorhandling for File opening
00044     {
00045         string msg = "Open file: ";
00046         msg += filename;
00047         msg += " failed!";
00048
00049         throw exception(msg.c_str());
00050     }
00051
00052     filename.erase(filename.end()-4,filename.end()); // loescht .src
00053     fprintf(fid, "DEF %s \n", filename.c_str()); // DEF in file schreiben
00054
00055     fputs("PTP $POS_ACT\n", fid); // PTP zur aktuellen Position in file
00056     schreiben
00057     if (input.getSpeedManual()) // If the speed is set to manual, it will be defined once at the
00058     beginning of the file
00059     {
00059         fprintf(fid, "$VEL.CP = %f\n", input.getSpeed()); // Geschwindigkeit ein file schreiben
00060     }
00061
00062     tuple <double, double, double> offset = input.getOffset();
00063
00064     for (size_t s = 0; s < points.size(); s++)
00065     {
00066         if (!input.getSpeedManual()) // If the speed is calculated it needs to be before every LIN
00067         command
00068             fprintf(fid, "$VEL.CP = %f\n", (float)processedPath[s].getSpeed());
00069             fprintf(fid, "LIN {X %f, Y %f, Z %f, A %f, B %f, C %f}\n", processedPath[s].getX() +
00070             get<0>(offset), processedPath[s].getY() + get<1>(offset),
00071             processedPath[s].getZ() + get<2>(offset), processedPath[s].getA(),
00072             processedPath[s].getB(),
00073             processedPath[s].getC());
00074     }
00075     fputs("END", fid);
00076 }
```

Benutzt [CInputParameter::getOffset\(\)](#), [CInputParameter::getSpeed\(\)](#), [CInputParameter::getSpeedManual\(\)](#), [input](#), [CLogging::logData\(\)](#), [postProcessing\(\)](#), [processedPath](#) und [CLogging::setStep\(\)](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.10.3.4 postProcessing()

```
void CRobCodeGenerator::postProcessing (
    vector< CInputPoint3D > & path )
```

Nachbearbeitung der Daten.

Integration der Manuellen Eingabedaten in die eingelesenen und bearbeiteten Daten Hier werden calculateSpeed und calculateAngles aufgerufen.

Parameter

<code>vector<CInputPoint3D>&</code>	<code>path</code>
---	-------------------

Definiert in Zeile 76 der Datei [RobCodeGenerator.cpp](#).

```
00077 {
00078     COutputPoint3D p;
00079     double timePrev = 1;
00080
00081     for (size_t s = 0; s < path.size(); s++) // Fuer jeden Punkt in dem Vector
00082     {
00083         p.set(path[s].getX(), path[s].getY(), path[s].getZ());
00084         if (input.getSpeedManual())
00085         {
00086             if (input.getSpeed() > MAX_SPEED) //Wenn maximale Geschwindigkeit ueberschritten wird,
00087                 Geschwindigkeit begrenzen
00088                 input.setSpeed(MAX_SPEED, true);
00089             else
00090             {
00091                 if (s == 0)
00092                     p.setSpeed(1); //Der erste Punkt(0) wird mit Standardgeschwindigkeit 1m/s angefahren.
00093                 else
00094                     p.setSpeed(calculateSpeed(path[s], s, timePrev)); //Die Geschwindigkeit zwischen den
00095                     weiteren Punkten wird berechnet.
00096             }
00097
00098             if (input.getOrientationManual()) // Wenn der Winkel vorgegeben ist diesen setzten
00099             {
00100                 tuple <double, double, double> angles;
00101                 angles = input.getAngles();
00102                 p.setA(get<0>(angles));
00103                 p.setB(get<1>(angles));
00104                 p.setC(get<2>(angles));
00105             }
00106             else // Sonst den Winkel berechnen
00107                 calculateAngles(p, path[s]);
00108             timePrev = path[s].getTime();
00109             processedPath.push_back(p);
00110         }
00111     }
00112 }
```

Benutzt [calculateAngles\(\)](#), [calculateSpeed\(\)](#), [CInputParameter::getAngles\(\)](#), [CInputParameter::getOrientationManual\(\)](#), [CInputParameter::getSpeed\(\)](#), [CInputParameter::getSpeedManual\(\)](#), [input](#), [MAX_SPEED](#), [processedPath](#), [CPoint3D::set\(\)](#), [COutputPoint3D::setA\(\)](#), [COutputPoint3D::setB\(\)](#), [COutputPoint3D::setC\(\)](#), [CInputParameter::setSpeed\(\)](#) und [COutputPoint3D::setSpeed\(\)](#).

Wird benutzt von [generateRobCode\(\)](#).

7.10.4 Dokumentation der Datenelemente

7.10.4.1 input

```
CInputParameter CRobCodeGenerator::input [private]
```

User eingegebene Geschwindigkeit

Definiert in Zeile 88 der Datei [RobCodeGenerator.h](#).

Wird benutzt von [CRobCodeGenerator\(\)](#), [generateRobCode\(\)](#) und [postProcessing\(\)](#).

7.10.4.2 processedPath

```
vector<COutputPoint3D> CRobCodeGenerator::processedPath [private]
```

Fertig bearbeiteter Pfad

Definiert in Zeile 84 der Datei [RobCodeGenerator.h](#).

Wird benutzt von [calculateSpeed\(\)](#), [generateRobCode\(\)](#) und [postProcessing\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- header/[RobCodeGenerator.h](#)
- source/[RobCodeGenerator.cpp](#)

7.11 CSegmentApproximator Klassenreferenz

Ausduennen des Pfades.

```
#include <SegmentApproximator.h>
```

Öffentliche Methoden

- [CSegmentApproximator](#) (void)
Default Konstruktor.
- [~CSegmentApproximator](#) (void)
Dekonstruktor.
- void [approx](#) (const vector< list< [CInputPoint3D](#) > > &Segments, [CLogging](#) log)
Ruft die Douglas-Peuker Funktion auf.
- void [setmaxDistance](#) (double maxDistanceSource)
Setzt die maximale Abweichung.
- double [getmaxDistance](#) ()
Gibt die maximale Abweichung zurueck.
- vector< list< [CInputPoint3D](#) > > & [getSegmentsApproxVector](#) ()
Gibt den bereinigten Pfad zurueck.

Private Methoden

- void [douglasPeuckerRecursive](#) (list< [CInputPoint3D](#) > &segment, std::list< [CInputPoint3D](#) >::iterator startItr, std::list< [CInputPoint3D](#) >::iterator endItr)
Rekursive Douglas Peuker Funktion.

Private Attribute

- `vector< list< CInputPoint3D > > segmentsApprox`
- `double maxDistance`

7.11.1 Ausführliche Beschreibung

Ausduennen des Pfades.

In dieser Klasse wird der Pfad mit hilfe des Douglas-Peuker Algorithmusses ausgeduennt

Definiert in Zeile [22](#) der Datei [SegmentApproximator.h](#).

7.11.2 Beschreibung der Konstruktoren und Destruktoren

7.11.2.1 CSegmentApproximator()

```
CSegmentApproximator::CSegmentApproximator (  
    void )
```

Default Konstruktor.

Initialisiert die Klasse

Definiert in Zeile [11](#) der Datei [SegmentApproximator.cpp](#).

```
00012 {  
00013 }
```

7.11.2.2 ~CSegmentApproximator()

```
CSegmentApproximator::~~CSegmentApproximator (  
    void )
```

Dekonstruktor.

Definiert in Zeile [15](#) der Datei [SegmentApproximator.cpp](#).

```
00016 {  
00017 }
```

7.11.3 Dokumentation der Elementfunktionen

7.11.3.1 approx()

```
void CSegmentApproximator::approx (  
    const vector< list< CInputPoint3D > > & segments,  
    CLogging log )
```

Ruft die Douglas-Peuker Funktion auf.

Iteriert durch die Listen im Vektor und ruft die Douglas-Peuker-Funktion auf

Parameter

<i>Segments</i>	const vector<list<CInputPoint3D>>&
<i>CLogging</i>	log fuer das Logging

Definiert in Zeile 19 der Datei [SegmentApproximator.cpp](#).

```

00020 {
00021     CInputPoint3D p;
00022
00023     segmentsApprox = segments;
00024
00025     /* Douglas Peucker fuer Segmente aufrufen*/
00026     for (size_t s = 0; s < segments.size(); s++)
00027     {
00028         douglasPeuckerRecursive(segmentsApprox[s], segmentsApprox[s].begin(),
--(segmentsApprox[s].end()));
00029     }
00030
00031     /* Logging der Daten*/
00032     if (log.getDetailed())
00033     {
00034         log.setStep(2);
00035         log.logData(segmentsApprox);
00036     }
00037 }

```

Benutzt [douglasPeuckerRecursive\(\)](#), [CLogging::getDetailed\(\)](#), [CLogging::logData\(\)](#), [segmentsApprox](#) und [CLogging::setStep\(\)](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.11.3.2 douglasPeuckerRecursive()

```

void CSegmentApproximator::douglasPeuckerRecursive (
    list< CInputPoint3D > & segment,
    std::list< CInputPoint3D >::iterator startItr,
    std::list< CInputPoint3D >::iterator endItr ) [private]

```

Rekursive Douglas Peuker Funktion.

Rekursive Funktion die durch das Segment geht und Punkte aus dem Pfad loescht wenn ihr Abstand zu gross wird.

Parameter

<i>list< CInputPoint3D> &</i>	segment
<i>std::list< CInputPoint3D>::iterator</i>	startItr
<i>std::list< CInputPoint3D>::iterator</i>	endItr

Definiert in Zeile 54 der Datei [SegmentApproximator.cpp](#).

```

00055 {
00056     if (segment.size() < 3) return; // min Groesse pro Seg 3
00057     if (distance(startItr, endItr) == 2) return; // Zeigerabstand == 2
00058     CInputPoint3D pStart; CInputPoint3D pEnd; // Variablen deklarieren
00059
00060
00061     /* Startpunkt setzen */
00062     pStart.setX(startItr->getX()); pStart.setY(startItr->getY()); pStart.setZ(startItr->getZ());
00063     pStart.setTime(startItr->getTime());
00064     pStart.setEulerMatrix(startItr->getEulerMatrix());
00065
00066     /* Endpunkt setzen */
00067     pEnd.setX(endItr->getX()); pEnd.setY(endItr->getY()); pEnd.setZ(endItr->getZ());
00068     pEnd.setTime(endItr->getTime());
00069     pEnd.setEulerMatrix(endItr->getEulerMatrix());

```

```

00070
00071     double dist = 0.0, maxDist = 0.0;           // dist und maxDist initialisieren
00072     std::list<CInputPoint3D>::iterator maxItr, itr; // Zeiger bilden
00073
00074
00075     /* am weitesten Entfernten Punkt suchen */
00076     for (itr = startItr; itr != endItr; itr++)
00077     {
00078         CLine3D line = CLine3D(pStart, pEnd);
00079         // calc distance
00080         dist = itr->distanceTo(line);
00081         if (dist > maxDist) {
00082             maxDist = dist;
00083             maxItr = itr;
00084         }
00085     }
00086
00087     if (maxDist <= maxDistance) {
00088
00089         segment.erase(++startItr, endItr); // Punkt loeschen
00090         return;
00091     }
00092
00093     /* Douglas Peucker erneut aufrufen */
00094     douglasPeuckerRecursive(segment, startItr, maxItr);
00095     douglasPeuckerRecursive(segment, maxItr, endItr);
00096 }

```

Benutzt [douglasPeuckerRecursive\(\)](#), [maxDistance](#), [CInputPoint3D::setEulerMatrix\(\)](#), [CInputPoint3D::setTime\(\)](#), [CPoint3D::setX\(\)](#), [CPoint3D::setY\(\)](#) und [CPoint3D::setZ\(\)](#).

Wird benutzt von [approx\(\)](#) und [douglasPeuckerRecursive\(\)](#).

7.11.3.3 getMaxDistance()

```
double CSegmentApproximator::getMaxDistance ( )
```

Gibt die maximale Abweichung zurueck.

Rückgabe

maxDistanceSource double

Definiert in Zeile 44 der Datei [SegmentApproximator.cpp](#).

```

00045 {
00046     return maxDistance; // Rueckgabe von maxDistance
00047 }

```

Benutzt [maxDistance](#).

7.11.3.4 getSegmentsApproxVector()

```
vector< list< CInputPoint3D > > & CSegmentApproximator::getSegmentsApproxVector ( )
```

Gibt den bereinigten Pfad zurueck.

Rückgabe

vector<list<CInputPoint3D>>&

Definiert in Zeile 49 der Datei [SegmentApproximator.cpp](#).

```

00050 {
00051     return segmentsApprox; // Rueckgabe der Segmente
00052 }

```

Benutzt [segmentsApprox](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.11.3.5 setmaxDistance()

```
void CSegmentApproximator::setmaxDistance (
    double maxDistanceSource )
```

Setzt die maximale Abweichung.

Parameter

<i>maxDistanceSource</i>	double
--------------------------	--------

Definiert in Zeile 39 der Datei [SegmentApproximator.cpp](#).

```
00040 {
00041     maxDistance = maxDistanceSource;    // setze maxDistance
00042 }
```

Benutzt [maxDistance](#).

Wird benutzt von [GUI::calculate\(\)](#).

7.11.4 Dokumentation der Datenelemente

7.11.4.1 maxDistance

```
double CSegmentApproximator::maxDistance [private]
```

Einstellbare Distanz fuer den Douglas-Peucker-Algorithmus

Definiert in Zeile 67 der Datei [SegmentApproximator.h](#).

Wird benutzt von [douglasPeuckerRecursive\(\)](#), [getmaxDistance\(\)](#) und [setmaxDistance\(\)](#).

7.11.4.2 segmentsApprox

```
vector<list<CInputPoint3D> > CSegmentApproximator::segmentsApprox [private]
```

Bereinigten Pfad

Definiert in Zeile 63 der Datei [SegmentApproximator.h](#).

Wird benutzt von [approx\(\)](#) und [getSegmentsApproxVector\(\)](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

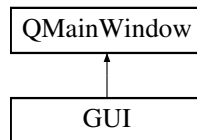
- [header/SegmentApproximator.h](#)
- [source/SegmentApproximator.cpp](#)

7.12 GUI Klassenreferenz

UI und Funktionen.

```
#include <GUI.h>
```

Klassendiagramm für GUI:



Öffentliche Methoden

- [GUI](#) (QWidget *parent=nullptr)
Default Konstruktor.
- [~GUI](#) ()
Dekonstruktor.

Private Slots

- void [calculate](#) (void)
Ruft nacheinander die Klassen fuer die Berechnung auf und erstellt alle notwendigen Files.
- void [setInputPath](#) (void)
Liest den Pfad der Input Datei ein.
- void [setOutputPath](#) (void)
Liest den Pfad des Output Ordners aus.
- void [setDP](#) (void)
Liest die DP Toleranz ein.
- void [setMean](#) (void)
Liest das Fenster fuer den gleitenden Mittelwertsfilter ein.
- void [activateSpeed](#) (void)
Aktiviert die das manuelle Einstellen der Geschwindigkeit.
- void [setSpeed](#) (void)
Setzt eine gewaehlte Geschwindigkeit.
- void [activateOrientation](#) (void)
Aktiviert die das manuelle Einstellen der Orientierung.
- void [setOrientation](#) (void)
Setzt eine gewaehlte Orientierung.
- void [activateOffset](#) (void)
Aktiviert die das manuelle Einstellen des Offsets.
- void [setOffset](#) (void)
Setzt eine gewaehlten Offset.
- void [activateLogging](#) (void)
Das detaillierte Logging.

Private Attribute

- [Ui::GUIClass ui](#)
- [QString inputPathUI](#)
- [QString outputPathUI](#)
- [double dpTolerance](#)
- [double meanLength](#)
- [CInputParameter inputParameter](#)

7.12.1 Ausführliche Beschreibung

UI und Funktionen.

Diese Klasse ruft das User Interface auf und stellt alle Funktionalitäten bereit

Definiert in Zeile [21](#) der Datei [GUI.h](#).

7.12.2 Beschreibung der Konstruktoren und Destruktoren

7.12.2.1 GUI()

```
GUI::GUI (
    QWidget * parent = nullptr )
```

Default Konstruktor.

Initialisiert das UI Fenster

Definiert in Zeile [15](#) der Datei [GUI.cpp](#).

```
00016     : QMainWindow(parent)
00017 {
00018     ui.setupUi(this);
00019
00020     //Dateioperationen und Anzeige
00021     inputPathUI = "";
00022     ui.pathInput->setText(inputPathUI);
00023     outputPathUI = "";
00024     ui.pathOutput->setText(outputPathUI);
00025
00026     connect(ui.pushOutput, &QPushButton::clicked, this, &GUI::setOutputPath);
00027     connect(ui.pushInput, &QPushButton::clicked, this, &GUI::setInputPath);
00028
00029     //Zwingende Einstellwerte
00030     //Douglas-Peuker-Toleranz
00031     ui.dpToleranz->setRange(1, 100);
00032     ui.dpToleranz->setSingleStep(1);
00033     ui.dpToleranz->setValue(40);
00034     connect(ui.dpToleranz, &QSpinBox::valueChanged, this, &GUI::setDP);
00035
00036     //Fenster fuer gleitenden Mittelwert
00037     ui.meanLength->setRange(3, 500);
00038     ui.meanLength->setSingleStep(1);
00039     ui.meanLength->setValue(50);
00040     connect(ui.meanLength, &QSpinBox::valueChanged, this, &GUI::setMean);
00041
00042     //Geschwindigkeit
00043     connect(ui.bSpeed, &QCheckBox::clicked, this, &GUI::activateSpeed);
00044     ui.speed->setRange(0.01, 2);
00045     ui.speed->setSingleStep(0.01);
00046     ui.speed->setValue(1);
00047     connect(ui.speed, &QDoubleSpinBox::valueChanged, this, &GUI::setSpeed);
00048
00049     //Ausrichtung
00050     connect(ui.bManOrientation, &QCheckBox::clicked, this, &GUI::activateOrientation);
00051     ui.AValue->setRange(-180, 180);
00052     ui.AValue->setSingleStep(5);
```

```

00053     ui.AValue->setValue(0);
00054     connect(ui.AValue, &QDoubleSpinBox::valueChanged, this, &GUI::setOrientation);
00055
00056     ui.BValue->setRange(-180, 180);
00057     ui.BValue->setSingleStep(5);
00058     ui.BValue->setValue(90);
00059     connect(ui.BValue, &QDoubleSpinBox::valueChanged, this, &GUI::setOrientation);
00060
00061     ui.CValue->setRange(-180, 180);
00062     ui.CValue->setSingleStep(5);
00063     ui.CValue->setValue(0);
00064     connect(ui.CValue, &QDoubleSpinBox::valueChanged, this, &GUI::setOrientation);
00065
00066     //Offset
00067     connect(ui.bOffset, &QCheckBox::clicked, this, &GUI::activateOffset);
00068     ui.offsetX->setRange(-400, 400);
00069     ui.offsetX->setSingleStep(10);
00070     ui.offsetX->setValue(0);
00071     connect(ui.offsetX, &QDoubleSpinBox::valueChanged, this, &GUI::setOffset);
00072
00073     ui.offsetY->setRange(-400, 400);
00074     ui.offsetY->setSingleStep(10);
00075     ui.offsetY->setValue(0);
00076     connect(ui.offsetY, &QDoubleSpinBox::valueChanged, this, &GUI::setOffset);
00077
00078     ui.offsetZ->setRange(-400, 400);
00079     ui.offsetZ->setSingleStep(10);
00080     ui.offsetZ->setValue(0);
00081     connect(ui.offsetZ, &QDoubleSpinBox::valueChanged, this, &GUI::setOffset);
00082
00083     //Logging
00084     ui.bLogging->setChecked(true);
00085     inputParameter.setOffset(0, 0, 0, false);
00086     connect(ui.bLogging, &QCheckBox::clicked, this, &GUI::activateLogging);
00087
00088     connect(ui.startCalculation, &QPushButton::clicked, this, &GUI::calculate);
00089 }

```

Benutzt [activateLogging\(\)](#), [activateOffset\(\)](#), [activateOrientation\(\)](#), [activateSpeed\(\)](#), [Ui_GUIClass::AValue](#), [Ui_GUIClass::bLogging](#), [Ui_GUIClass::bManOrientation](#), [Ui_GUIClass::bOffset](#), [Ui_GUIClass::bSpeed](#), [Ui_GUIClass::BValue](#), [calculate\(\)](#), [Ui_GUIClass::CValue](#), [Ui_GUIClass::dpToleranz](#), [inputParameter](#), [inputPathUI](#), [Ui_GUIClass::meanLength](#), [Ui_GUIClass::offsetX](#), [Ui_GUIClass::offsetY](#), [Ui_GUIClass::offsetZ](#), [outputPathUI](#), [Ui_GUIClass::pathInput](#), [Ui_GUIClass::pathOutput](#), [Ui_GUIClass::pushInput](#), [Ui_GUIClass::pushOutput](#), [setDP\(\)](#), [setInputPath\(\)](#), [setMean\(\)](#), [CInputParameter::setOffset\(\)](#), [setOffset\(\)](#), [setOrientation\(\)](#), [setOutputPath\(\)](#), [setSpeed\(\)](#), [Ui_GUIClass::setupUi\(\)](#), [Ui_GUIClass::speed](#), [Ui_GUIClass::startCalculation](#) und [ui](#).

7.12.2.2 ~GUI()

GUI::~GUI ()

Dekonstruktor.

Definiert in Zeile 91 der Datei [GUI.cpp](#).

```
00092 {}
```

7.12.3 Dokumentation der Elementfunktionen

7.12.3.1 activateLogging

```

void GUI::activateLogging (
    void ) [private], [slot]

```

Das detaillierte Logging.

Definiert in Zeile 94 der Datei [GUI.cpp](#).

```

00095 {
00096     inputParameter.setLogging(ui.bLogging->isChecked());
00097 }

```

Benutzt [Ui_GUIClass::bLogging](#), [inputParameter](#), [CInputParameter::setLogging\(\)](#) und [ui](#).

Wird benutzt von [GUI\(\)](#).

7.12.3.2 activateOffset

```
void GUI::activateOffset (
    void ) [private], [slot]
```

Aktiviert die das manuelle Einstellen des Offsets.

Definiert in Zeile 105 der Datei [GUI.cpp](#).

```
00106 {
00107     if (ui.bOffset->isChecked())
00108     {
00109         ui.offset->setEnabled(true);
00110         ui.offset->setStyleSheet("background-color: rgb(67, 72, 91); color: rgb(3, 8, 14); border:
        1px solid black;");
00111     }
00112     else
00113     {
00114         ui.offset->setEnabled(false);
00115         ui.offset->setStyleSheet("background-color: rgb(210,211,218); color: rgb(117,125,149)");
00116         ui.offsetX->setValue(0);
00117         ui.offsetY->setValue(0);
00118         ui.offsetZ->setValue(0);
00119         inputParameter.setOffset(0, 0, 0, false);
00120     }
00121 }
```

Benutzt [Ui_GUIClass::bOffset](#), [inputParameter](#), [Ui_GUIClass::offset](#), [Ui_GUIClass::offsetX](#), [Ui_GUIClass::offsetY](#), [Ui_GUIClass::offsetZ](#), [CInputParameter::setOffset\(\)](#) und [ui](#).

Wird benutzt von [GUI\(\)](#).

7.12.3.3 activateOrientation

```
void GUI::activateOrientation (
    void ) [private], [slot]
```

Aktiviert die das manuelle Einstellen der Orientierung.

Definiert in Zeile 129 der Datei [GUI.cpp](#).

```
00130 {
00131     if (ui.bManOrientation->isChecked())
00132     {
00133         ui.orientation->setEnabled(true);
00134         ui.orientation->setStyleSheet("background-color: rgb(67, 72, 91); color: rgb(3, 8, 14);
        border: 1px solid black;");
00135     }
00136     else
00137     {
00138         ui.orientation->setEnabled(false);
00139         ui.orientation->setStyleSheet("background-color: rgb(210,211,218); color: rgb(117,125,149)");
00140     }
00141 }
```

Benutzt [Ui_GUIClass::bManOrientation](#), [Ui_GUIClass::orientation](#) und [ui](#).

Wird benutzt von [GUI\(\)](#).

7.12.3.4 activateSpeed

```
void GUI::activateSpeed (
    void ) [private], [slot]
```

Aktiviert die das manuelle Einstellen der Geschwindigkeit.

Definiert in Zeile 148 der Datei [GUI.cpp](#).

```
00149 {
00150     if (ui.bSpeed->isChecked())
00151     {
00152         ui.speed_2->setEnabled(true);
00153         ui.speed_2->setStyleSheet("background-color:  rgb(67, 72, 91); color: rgb(3, 8, 14); border:
1px solid black; ");
00154     }
00155     else
00156     {
00157         ui.speed_2->setEnabled(false);
00158         ui.speed_2->setStyleSheet("background-color:  rgb(210,211,218); color:  rgb(117,125,149)");
00159     }
00160 }
```

Benutzt [Ui_GUIClass::bSpeed](#), [Ui_GUIClass::speed_2](#) und [ui](#).

Wird benutzt von [GUI\(\)](#).

7.12.3.5 calculate

```
void GUI::calculate (
    void ) [private], [slot]
```

Ruft nacheinander die Klassen fuer die Berechnung auf und erstellt alle notwendigen Files.

Definiert in Zeile 185 der Datei [GUI.cpp](#).

```
00186 {
00187
00188     if (inputPathUI.isEmpty() || outputPathUI.isEmpty())
00189     {
00190         QMessageBox messageBox;
00191         messageBox.critical(0, "Error", "Keine Datei ausgewaehlt!");
00192         messageBox.setFixedSize(500, 200);
00193         return;
00194     }
00195
00196     try
00197     {
00198         string outputPath = outputPathUI.toUtf8().constData();
00199         string inputPath = inputPathUI.toUtf8().constData();
00200         ui.textBrowser->clear();
00201
00202         //logging Initialisieren
00203         CLogging logging(outputPath, inputParameter.getLoggingManual());
00204
00205         //read Data
00206         inputParameter.openFile(inputPath);
00207         ui.textBrowser->insertPlainText("Datei eingelesen\n");
00208
00209         //moving Average
00210
00211         CMeanFilter meanFilter;
00212         meanFilter.setWindowSize(meanLength);
00213         meanFilter.mean(inputParameter.getPath(), logging);
00214         ui.textBrowser->insertPlainText("Gleitender Mittelwert berechnet\n");
00215
00216         // Douglas-Peuker Algorithm
00217
00218         CSegmentApproximator segmentApproximator;
00219         segmentApproximator.setmaxDistance(dpTolerance);
00220         segmentApproximator.approx(meanFilter.getPath(), logging);
00221         ui.textBrowser->insertPlainText("Douglas-Peuker-Algorithmus berechnet\n");
00222
00223         // Puts the Segments together to one path
00224
00225         CPathBuilder pathBuilder;
```

```

00226         pathBuilder.createPath(segmentApproximator.getSegmentsApproxVector(), logging);
00227         ui.textBrowser->insertPlainText("Pfad zusammengesetzt\n");
00228
00229         // Calculates Speed, Angle and generates the Output Data
00230
00231         CRobCodeGenerator codeGenerator(inputParameter);
00232         codeGenerator.generateRobCode(pathBuilder.getPath(), outputPath, inputPath, logging);
00233         ui.textBrowser->insertPlainText("Datei erstellt\n");
00234     }
00235
00236     catch (exception& e)
00237     {
00238         cerr << e.what() << "\n";
00239     }
00240 }

```

Benutzt [CSegmentApproximator::approx\(\)](#), [CPathBuilder::createPath\(\)](#), [dpTolerance](#), [CRobCodeGenerator::generateRobCode\(\)](#), [CInputParameter::getLoggingManual\(\)](#), [CInputParameter::getPath\(\)](#), [CMeanFilter::getPath\(\)](#), [CPathBuilder::getPath\(\)](#), [CSegmentApproximator::getSegmentsApproxVector\(\)](#), [inputParameter](#), [inputPathUI](#), [CMeanFilter::mean\(\)](#), [meanLength](#), [CInputParameter::openFile\(\)](#), [outputPathUI](#), [CSegmentApproximator::setMaxDistance\(\)](#), [CMeanFilter::setWindowSize\(\)](#), [Ui_GUIClass::textBrowser](#) und [ui](#).

Wird benutzt von [GUI\(\)](#).

7.12.3.6 setDP

```

void GUI::setDP (
    void ) [private], [slot]

```

Liest die DP Toleranz ein.

Definiert in Zeile 168 der Datei [GUI.cpp](#).

```

00169 {
00170     dpTolerance = ui.dpToleranz->value();
00171 }

```

Benutzt [dpTolerance](#), [Ui_GUIClass::dpToleranz](#) und [ui](#).

Wird benutzt von [GUI\(\)](#).

7.12.3.7 setInputPath

```

void GUI::setInputPath (
    void ) [private], [slot]

```

Liest den Pfad der Input Datei ein.

Definiert in Zeile 173 der Datei [GUI.cpp](#).

```

00174 {
00175     inputPathUI = QFileDialog::getOpenFileName(this);
00176     ui.pathInput->setText(inputPathUI);
00177 }

```

Benutzt [inputPathUI](#), [Ui_GUIClass::pathInput](#) und [ui](#).

Wird benutzt von [GUI\(\)](#).

7.12.3.8 setMean

```
void GUI::setMean (
    void ) [private], [slot]
```

Liest das Fenster fuer den gleitenden Mittelwertsfilter ein.

Definiert in Zeile 162 der Datei [GUI.cpp](#).

```
00163 {
00164     meanLength = ui.meanLength->value();
00165 }
```

Benutzt [meanLength](#), [Ui_GUIClass::meanLength](#) und [ui](#).

Wird benutzt von [GUI\(\)](#).

7.12.3.9 setOffset

```
void GUI::setOffset (
    void ) [private], [slot]
```

Setzt eine gewaehlten Offset.

Definiert in Zeile 99 der Datei [GUI.cpp](#).

```
00100 {
00101     inputParameter.setOffset(ui.offsetX->value(), ui.offsetY->value(), ui.offsetZ->value(),
00102         ui.bOffset->isChecked());
00103 }
```

Benutzt [Ui_GUIClass::bOffset](#), [inputParameter](#), [Ui_GUIClass::offsetX](#), [Ui_GUIClass::offsetY](#), [Ui_GUIClass::offsetZ](#), [CInputParameter::setOffset\(\)](#) und [ui](#).

Wird benutzt von [GUI\(\)](#).

7.12.3.10 setOrientation

```
void GUI::setOrientation (
    void ) [private], [slot]
```

Setzt eine gewaahlte Orientierung.

Definiert in Zeile 123 der Datei [GUI.cpp](#).

```
00124 {
00125     inputParameter.setOrientation(ui.AValue->value(), ui.BValue->value(), ui.CValue->value(),
00126         ui.bManOrientation->isChecked());
00127 }
```

Benutzt [Ui_GUIClass::AValue](#), [Ui_GUIClass::bManOrientation](#), [Ui_GUIClass::BValue](#), [Ui_GUIClass::CValue](#), [inputParameter](#), [CInputParameter::setOrientation\(\)](#) und [ui](#).

Wird benutzt von [GUI\(\)](#).

7.12.3.11 setOutputPath

```
void GUI::setOutputPath (
    void ) [private], [slot]
```

Liest den Pfad des Output Ordners aus.

Definiert in Zeile 179 der Datei GUI.cpp.

```
00180 {
00181     outputPathUI = QFileDialog::getExistingDirectory(this);
00182     ui.pathOutput->setText(outputPathUI);
00183 }
```

Benutzt outputPathUI, Ui_GUIClass::pathOutput und ui.

Wird benutzt von GUI().

7.12.3.12 setSpeed

```
void GUI::setSpeed (
    void ) [private], [slot]
```

Setzt eine gewaehlte Geschwindigkeit.

Definiert in Zeile 143 der Datei GUI.cpp.

```
00144 {
00145     inputParameter.setSpeed(ui.speed->value(), ui.bSpeed->isChecked());
00146 }
```

Benutzt Ui_GUIClass::bSpeed, inputParameter, CInputParameter::setSpeed(), Ui_GUIClass::speed und ui.

Wird benutzt von GUI().

7.12.4 Dokumentation der Datenelemente

7.12.4.1 dpTolerance

```
double GUI::dpTolerance [private]
```

Definiert in Zeile 99 der Datei GUI.h.

Wird benutzt von calculate() und setDP().

7.12.4.2 inputParameter

```
CInputParameter GUI::inputParameter [private]
```

Definiert in Zeile 101 der Datei GUI.h.

Wird benutzt von activateLogging(), activateOffset(), calculate(), GUI(), setOffset(), setOrientation() und setSpeed().

7.12.4.3 inputPathUI

```
QString GUI::inputPathUI [private]
```

Pfad des Eingabefiles

Definiert in Zeile 94 der Datei [GUI.h](#).

Wird benutzt von [calculate\(\)](#), [GUI\(\)](#) und [setInputPath\(\)](#).

7.12.4.4 meanLength

```
double GUI::meanLength [private]
```

Definiert in Zeile 100 der Datei [GUI.h](#).

Wird benutzt von [calculate\(\)](#) und [setMean\(\)](#).

7.12.4.5 outputPathUI

```
QString GUI::outputPathUI [private]
```

Pfad fuer Ausgabedateien

Definiert in Zeile 98 der Datei [GUI.h](#).

Wird benutzt von [calculate\(\)](#), [GUI\(\)](#) und [setOutputPath\(\)](#).

7.12.4.6 ui

```
Ui::GUIClass GUI::ui [private]
```

Verbindung mit der im ui file graphisch erstellten Oberflaeche

Definiert in Zeile 90 der Datei [GUI.h](#).

Wird benutzt von [activateLogging\(\)](#), [activateOffset\(\)](#), [activateOrientation\(\)](#), [activateSpeed\(\)](#), [calculate\(\)](#), [GUI\(\)](#), [setDP\(\)](#), [setInputPath\(\)](#), [setMean\(\)](#), [setOffset\(\)](#), [setOrientation\(\)](#), [setOutputPath\(\)](#) und [setSpeed\(\)](#).

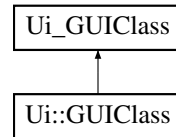
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [header/GUI.h](#)
- [source/GUI.cpp](#)

7.13 Ui::GUIClass Klassenreferenz

```
#include <ui_GUI.h>
```

Klassendiagramm für Ui::GUIClass:



Weitere Geerbte Elemente

Öffentliche Methoden geerbt von [Ui_GUIClass](#)

- void [setupUi](#) (QMainWindow *GUIClass)
- void [retranslateUi](#) (QMainWindow *GUIClass)
- void [setupUi](#) (QMainWindow *GUIClass)
- void [retranslateUi](#) (QMainWindow *GUIClass)

Öffentliche Attribute geerbt von [Ui_GUIClass](#)

- QWidget * [centralWidget](#)
- QLabel * [pathInput](#)
- QFrame * [frame](#)
- QWidget * [speed_2](#)
- QDoubleSpinBox * [speed](#)
- QLabel * [label_5](#)
- QCheckBox * [bSpeed](#)
- QWidget * [orientation](#)
- QLabel * [label_10](#)
- QLabel * [label_11](#)
- QLabel * [label_12](#)
- QDoubleSpinBox * [AValue](#)
- QDoubleSpinBox * [BValue](#)
- QDoubleSpinBox * [CValue](#)
- QCheckBox * [bManOrientation](#)
- QSpinBox * [meanLength](#)
- QLabel * [label_dp](#)
- QLabel * [label_4](#)
- QSpinBox * [dpToleranz](#)
- QCheckBox * [bLogging](#)
- QCheckBox * [bOffset](#)
- QWidget * [offset](#)
- QLabel * [label_13](#)
- QLabel * [label_14](#)
- QLabel * [label_15](#)
- QDoubleSpinBox * [offsetX](#)
- QDoubleSpinBox * [offsetY](#)
- QDoubleSpinBox * [offsetZ](#)
- QPushButton * [pushInput](#)
- QPushButton * [pushOutput](#)
- QTextBrowser * [textBrowser](#)
- QPushButton * [startCalculation](#)
- QLabel * [pathOutput](#)

7.13.1 Ausführliche Beschreibung

Definiert in Zeile [358](#) der Datei [ui_GUI.h](#).

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- [x64/Debug/uic/ui_GUI.h](#)

7.14 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_↔ CLASSGUIENDCLASS_t Strukturreferenz

Öffentliche Attribute

- uint [offsetsAndSizes](#) [28]
- char [stringdata0](#) [4]
- char [stringdata1](#) [10]
- char [stringdata2](#) [1]
- char [stringdata3](#) [13]
- char [stringdata4](#) [14]
- char [stringdata5](#) [6]
- char [stringdata6](#) [8]
- char [stringdata7](#) [14]
- char [stringdata8](#) [9]
- char [stringdata9](#) [20]
- char [stringdata10](#) [15]
- char [stringdata11](#) [15]
- char [stringdata12](#) [10]
- char [stringdata13](#) [16]

7.14.1 Ausführliche Beschreibung

Definiert in Zeile [58](#) der Datei [moc_GUI.cpp](#).

7.14.2 Dokumentation der Datenelemente

7.14.2.1 offsetsAndSizes

```
uint QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::offsetsAndSizes
```

Definiert in Zeile [59](#) der Datei [moc_GUI.cpp](#).

7.14.2.2 stringdata0

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata0
```

Definiert in Zeile [60](#) der Datei [moc_GUI.cpp](#).

7.14.2.3 stringdata1

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata1
```

Definiert in Zeile 61 der Datei [moc_GUI.cpp](#).

7.14.2.4 stringdata10

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata10
```

Definiert in Zeile 70 der Datei [moc_GUI.cpp](#).

7.14.2.5 stringdata11

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata11
```

Definiert in Zeile 71 der Datei [moc_GUI.cpp](#).

7.14.2.6 stringdata12

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata12
```

Definiert in Zeile 72 der Datei [moc_GUI.cpp](#).

7.14.2.7 stringdata13

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata13
```

Definiert in Zeile 73 der Datei [moc_GUI.cpp](#).

7.14.2.8 stringdata2

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata2
```

Definiert in Zeile 62 der Datei [moc_GUI.cpp](#).

7.14.2.9 stringdata3

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata3
```

Definiert in Zeile 63 der Datei [moc_GUI.cpp](#).

7.14.2.10 stringdata4

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata4
```

Definiert in Zeile 64 der Datei [moc_GUI.cpp](#).

7.14.2.11 stringdata5

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata5
```

Definiert in Zeile 65 der Datei [moc_GUI.cpp](#).

7.14.2.12 stringdata6

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata6
```

Definiert in Zeile 66 der Datei [moc_GUI.cpp](#).

7.14.2.13 stringdata7

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata7
```

Definiert in Zeile 67 der Datei [moc_GUI.cpp](#).

7.14.2.14 stringdata8

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata8
```

Definiert in Zeile 68 der Datei [moc_GUI.cpp](#).

7.14.2.15 stringdata9

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t::stringdata9
```

Definiert in Zeile 69 der Datei [moc_GUI.cpp](#).

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Dateien:

- x64/Debug/moc/[moc_GUI.cpp](#)
- x64/Release/moc/[moc_GUI.cpp](#)

7.15 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_↵ CLASSSwitchENDCLASS_t Strukturreferenz

Öffentliche Attribute

- uint [offsetsAndSizes](#) [6]
- char [stringdata0](#) [7]
- char [stringdata1](#) [7]
- char [stringdata2](#) [6]

7.15.1 Ausführliche Beschreibung

Definiert in Zeile 48 der Datei [moc_switch.cpp](#).

7.15.2 Dokumentation der Datenelemente

7.15.2.1 offsetsAndSizes

```
uint QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSwitchENDCLASS_t::offsetsAndSizes
```

Definiert in Zeile 49 der Datei [moc_switch.cpp](#).

7.15.2.2 stringdata0

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSwitchENDCLASS_t::stringdata0
```

Definiert in Zeile 50 der Datei [moc_switch.cpp](#).

7.15.2.3 stringdata1

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSwitchENDCLASS_t::stringdata1
```

Definiert in Zeile 51 der Datei [moc_switch.cpp](#).

7.15.2.4 stringdata2

```
char QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSwitchENDCLASS_t::stringdata2
```

Definiert in Zeile 52 der Datei [moc_switch.cpp](#).

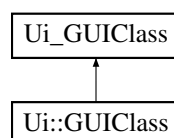
Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Dateien:

- [x64/Debug/moc/moc_switch.cpp](#)
- [x64/Release/moc/moc_switch.cpp](#)

7.16 Ui_GUIClass Klassenreferenz

```
#include <ui_GUI.h>
```

Klassendiagramm für Ui_GUIClass:



Öffentliche Methoden

- void [setupUi](#) (QMainWindow *GUIClass)
- void [retranslateUi](#) (QMainWindow *GUIClass)
- void [setupUi](#) (QMainWindow *GUIClass)
- void [retranslateUi](#) (QMainWindow *GUIClass)

Öffentliche Attribute

- QWidget * [centralWidget](#)
- QLabel * [pathInput](#)
- QFrame * [frame](#)
- QWidget * [speed_2](#)
- QDoubleSpinBox * [speed](#)
- QLabel * [label_5](#)
- QCheckBox * [bSpeed](#)
- QWidget * [orientation](#)
- QLabel * [label_10](#)
- QLabel * [label_11](#)
- QLabel * [label_12](#)
- QDoubleSpinBox * [AValue](#)
- QDoubleSpinBox * [BValue](#)
- QDoubleSpinBox * [CValue](#)
- QCheckBox * [bManOrientation](#)
- QSpinBox * [meanLength](#)
- QLabel * [label_dp](#)
- QLabel * [label_4](#)
- QSpinBox * [dpToleranz](#)
- QCheckBox * [bLogging](#)
- QCheckBox * [bOffset](#)
- QWidget * [offset](#)
- QLabel * [label_13](#)
- QLabel * [label_14](#)
- QLabel * [label_15](#)
- QDoubleSpinBox * [offsetX](#)
- QDoubleSpinBox * [offsetY](#)
- QDoubleSpinBox * [offsetZ](#)
- QPushButton * [pushInput](#)
- QPushButton * [pushOutput](#)
- QTextBrowser * [textBrowser](#)
- QPushButton * [startCalculation](#)
- QLabel * [pathOutput](#)

7.16.1 Ausführliche Beschreibung

Definiert in Zeile [26](#) der Datei [ui_GUI.h](#).

7.16.2 Dokumentation der Elementfunktionen

7.16.2.1 retranslateUi() [1/2]

```
void Ui_GUIClass::retranslateUi (
    QMainWindow * GUIClass ) [inline]
```

Definiert in Zeile 329 der Datei `ui_GUI.h`.

```
00330 {
00331     GUIClass->setWindowTitle(QCoreApplication::translate("GUIClass", "Roboter Pfad Editor",
00332         nullptr));
00333     pathInput->setText(QCoreApplication::translate("GUIClass", "Eingabedatei", nullptr));
00334     speed->setSuffix(QCoreApplication::translate("GUIClass", " m/s", nullptr));
00335     label_5->setText(QCoreApplication::translate("GUIClass", "Geschwindigkeit", nullptr));
00336     bSpeed->setText(QCoreApplication::translate("GUIClass", "Manuelle Geschwindigkeit", nullptr));
00337     label_10->setText(QCoreApplication::translate("GUIClass", "Phi", nullptr));
00338     label_11->setText(QCoreApplication::translate("GUIClass", "Theta", nullptr));
00339     label_12->setText(QCoreApplication::translate("GUIClass", "Psi", nullptr));
00340     bManOrientation->setText(QCoreApplication::translate("GUIClass", "Manuelle Ausrichtung",
00341         nullptr));
00342     label_dp->setText(QCoreApplication::translate("GUIClass", "Douglas-Peuker-Toleranz",
00343         nullptr));
00344     label_4->setText(QCoreApplication::translate("GUIClass", "Filterlaenge gleitender Mittelwert",
00345         nullptr));
00346     dpToleranz->setSpecialValueText(QString());
00347     dpToleranz->setSuffix(QString());
00348     bLogging->setText(QCoreApplication::translate("GUIClass", "Detailliertes Logging", nullptr));
00349     bOffset->setText(QCoreApplication::translate("GUIClass", "Einstellung Offset", nullptr));
00350     label_13->setText(QCoreApplication::translate("GUIClass", "X", nullptr));
00351     label_14->setText(QCoreApplication::translate("GUIClass", "Y", nullptr));
00352     label_15->setText(QCoreApplication::translate("GUIClass", "Z", nullptr));
00353     pushInput->setText(QCoreApplication::translate("GUIClass", "Datei waehlen", nullptr));
00354     pushOutput->setText(QCoreApplication::translate("GUIClass", "Pfad waehlen", nullptr));
00355     startCalculation->setText(QCoreApplication::translate("GUIClass", "Datei erstellen",
00356         nullptr));
00357     pathOutput->setText(QCoreApplication::translate("GUIClass", "Ausgabeordner", nullptr));
00358 } // retranslateUi
```

Benutzt `bLogging`, `bManOrientation`, `bOffset`, `bSpeed`, `dpToleranz`, `label_10`, `label_11`, `label_12`, `label_13`, `label_14`, `label_15`, `label_4`, `label_5`, `label_dp`, `pathInput`, `pathOutput`, `pushInput`, `pushOutput`, `speed` und `startCalculation`.

Wird benutzt von `setupUi()`.

7.16.2.2 retranslateUi() [2/2]

```
void Ui_GUIClass::retranslateUi (
    QMainWindow * GUIClass ) [inline]
```

Definiert in Zeile 329 der Datei `ui_GUI.h`.

```
00330 {
00331     GUIClass->setWindowTitle(QCoreApplication::translate("GUIClass", "Roboter Pfad Editor",
00332         nullptr));
00333     pathInput->setText(QCoreApplication::translate("GUIClass", "Eingabedatei", nullptr));
00334     speed->setSuffix(QCoreApplication::translate("GUIClass", " m/s", nullptr));
00335     label_5->setText(QCoreApplication::translate("GUIClass", "Geschwindigkeit", nullptr));
00336     bSpeed->setText(QCoreApplication::translate("GUIClass", "Manuelle Geschwindigkeit", nullptr));
00337     label_10->setText(QCoreApplication::translate("GUIClass", "A", nullptr));
00338     label_11->setText(QCoreApplication::translate("GUIClass", "B", nullptr));
00339     label_12->setText(QCoreApplication::translate("GUIClass", "C", nullptr));
00340     bManOrientation->setText(QCoreApplication::translate("GUIClass", "Manuelle Ausrichtung",
00341         nullptr));
00342     label_dp->setText(QCoreApplication::translate("GUIClass", "Douglas-Peuker-Toleranz",
00343         nullptr));
00344     label_4->setText(QCoreApplication::translate("GUIClass", "Filterlaenge gleitender Mittelwert",
00345         nullptr));
00346     dpToleranz->setSpecialValueText(QString());
00347     dpToleranz->setSuffix(QString());
00348     bLogging->setText(QCoreApplication::translate("GUIClass", "Detailliertes Logging", nullptr));
00349     bOffset->setText(QCoreApplication::translate("GUIClass", "Einstellung Offset", nullptr));
00350     label_13->setText(QCoreApplication::translate("GUIClass", "X", nullptr));
00351     label_14->setText(QCoreApplication::translate("GUIClass", "Y", nullptr));
00352     label_15->setText(QCoreApplication::translate("GUIClass", "Z", nullptr));
00353     pushInput->setText(QCoreApplication::translate("GUIClass", "Datei waehlen", nullptr));
00354     pushOutput->setText(QCoreApplication::translate("GUIClass", "Pfad waehlen", nullptr));
```

```

00351         startCalculation->setText(QCoreApplication::translate("GUIClass", "Datei erstellen",
    nullptr));
00352         pathOutput->setText(QCoreApplication::translate("GUIClass", "Ausgabeordner", nullptr));
00353     } // retranslateUi

```

Benutzt `bLogging`, `bManOrientation`, `bOffset`, `bSpeed`, `dpToleranz`, `label_10`, `label_11`, `label_12`, `label_13`, `label_14`, `label_15`, `label_4`, `label_5`, `label_dp`, `pathInput`, `pathOutput`, `pushInput`, `pushOutput`, `speed` und `startCalculation`.

7.16.2.3 setupUi() [1/2]

```

void Ui_GUIClass::setupUi (
    QMainWindow * GUIClass ) [inline]

```

Definiert in Zeile 63 der Datei `ui_GUI.h`.

```

00064     {
00065         if (GUIClass->objectName().isEmpty())
00066             GUIClass->setObjectName("GUIClass");
00067         GUIClass->resize(355, 700);
00068         QSizePolicy sizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);
00069         sizePolicy.setHorizontalStretch(0);
00070         sizePolicy.setVerticalStretch(0);
00071         sizePolicy.setHeightForWidth(GUIClass->sizePolicy().hasHeightForWidth());
00072         GUIClass->setSizePolicy(sizePolicy);
00073         GUIClass->setMinimumSize(QSize(355, 530));
00074         GUIClass->setMaximumSize(QSize(355, 700));
00075         QFont font;
00076         font.setFamilies({QString::fromUtf8("Rubik")});
00077         GUIClass->setFont(font);
00078         GUIClass->setAnimated(true);
00079         GUIClass->setTabShape(QTabWidget::Rounded);
00080         GUIClass->setUnifiedTitleAndToolBarOnMac(false);
00081         centralWidget = new QWidget(GUIClass);
00082         centralWidget->setObjectName("centralWidget");
00083         QFont font1;
00084         font1.setFamilies({QString::fromUtf8("Rubik")});
00085         font1.setBold(false);
00086         font1.setItalic(false);
00087         centralWidget->setFont(font1);
00088         centralWidget->setStyleSheet(QString::fromUtf8("background-color: rgb(3, 8, 14);\n"
00089 "color: rgb(3, 8, 14);\n"
00090 "));
00091         pathInput = new QLabel(centralWidget);
00092         pathInput->setObjectName("pathInput");
00093         pathInput->setGeometry(QRect(10, 10, 331, 31));
00094         pathInput->setFont(font1);
00095         pathInput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00096 "border-radius: 10px;\n"
00097 "background-color: rgb(210, 211, 218);\n"
00098 "color: rgb(3, 8, 14);"));
00099         frame = new QFrame(centralWidget);
00100         frame->setObjectName("frame");
00101         frame->setGeometry(QRect(10, 90, 331, 491));
00102         frame->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00103 "border-radius: 10px;\n"
00104 "background-color: rgb(117, 125, 149);"));
00105         frame->setFrameShape(QFrame::Box);
00106         frame->setFrameShadow(QFrame::Raised);
00107         frame->setLineWidth(1);
00108         speed_2 = new QWidget(frame);
00109         speed_2->setObjectName("speed_2");
00110         speed_2->setEnabled(false);
00111         speed_2->setGeometry(QRect(15, 110, 301, 41));
00112         QFont font2;
00113         font2.setFamilies({QString::fromUtf8("Rubik")});
00114         font2.setBold(true);
00115         speed_2->setFont(font2);
00116         speed_2->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00117 "color: rgb(117, 125, 149);\n"
00118 "border: 1px solid rgb(67, 72, 91); "));
00119         speed = new QDoubleSpinBox(speed_2);
00120         speed->setObjectName("speed");
00121         speed->setGeometry(QRect(159, 12, 136, 20));
00122         speed->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00123 "\n"
00124 "border-radius: 6px;"));
00125         speed->setAlignment(Qt::AlignCenter);
00126         label_5 = new QLabel(speed_2);
00127         label_5->setObjectName("label_5");
00128         label_5->setGeometry(QRect(11, 12, 93, 16));
00129         label_5->setFont(font1);

```



```

00130         label_5->setStyleSheet(QString::fromUtf8("border: 0px"));
00131         bSpeed = new QCheckBox(frame);
00132         bSpeed->setObjectName("bSpeed");
00133         bSpeed->setGeometry(QRect(15, 80, 171, 20));
00134         bSpeed->setFont(font1);
00135         bSpeed->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00136 "border: 0px"));
00137         orientation = new QWidget(frame);
00138         orientation->setObjectName("orientation");
00139         orientation->setEnabled(false);
00140         orientation->setGeometry(QRect(15, 190, 301, 111));
00141         orientation->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00142 "color: rgb(117, 125, 149);\n"
00143 "border: 1px solid rgb(67, 72, 91);"));
00144         label_10 = new QLabel(orientation);
00145         label_10->setObjectName("label_10");
00146         label_10->setGeometry(QRect(10, 15, 18, 16));
00147         label_10->setFont(font1);
00148         label_10->setStyleSheet(QString::fromUtf8("border: 0px"));
00149         label_11 = new QLabel(orientation);
00150         label_11->setObjectName("label_11");
00151         label_11->setGeometry(QRect(11, 47, 33, 16));
00152         label_11->setFont(font1);
00153         label_11->setStyleSheet(QString::fromUtf8("\n"
00154 "border: 0px"));
00155         label_12 = new QLabel(orientation);
00156         label_12->setObjectName("label_12");
00157         label_12->setGeometry(QRect(11, 78, 17, 16));
00158         label_12->setFont(font1);
00159         label_12->setStyleSheet(QString::fromUtf8("border: 0px"));
00160         AValue = new QDoubleSpinBox(orientation);
00161         AValue->setObjectName("AValue");
00162         AValue->setGeometry(QRect(153, 15, 141, 20));
00163         AValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00164 "\n"
00165 "border-radius: 6px;"));
00166         BValue = new QDoubleSpinBox(orientation);
00167         BValue->setObjectName("BValue");
00168         BValue->setGeometry(QRect(154, 47, 141, 20));
00169         BValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00170 "\n"
00171 "border-radius: 6px;"));
00172         CValue = new QDoubleSpinBox(orientation);
00173         CValue->setObjectName("CValue");
00174         CValue->setGeometry(QRect(154, 78, 141, 20));
00175         CValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00176 "\n"
00177 "border-radius: 6px;"));
00178         label_12->raise();
00179         label_10->raise();
00180         AValue->raise();
00181         BValue->raise();
00182         CValue->raise();
00183         label_11->raise();
00184         bManOrientation = new QCheckBox(frame);
00185         bManOrientation->setObjectName("bManOrientation");
00186         bManOrientation->setGeometry(QRect(15, 160, 151, 20));
00187         bManOrientation->setFont(font1);
00188         bManOrientation->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00189 "border: 0px"));
00190         meanLength = new QSpinBox(frame);
00191         meanLength->setObjectName("meanLength");
00192         meanLength->setGeometry(QRect(200, 48, 116, 20));
00193         meanLength->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00194 "\n"
00195 "border-radius: 6px;"));
00196         label_dp = new QLabel(frame);
00197         label_dp->setObjectName("label_dp");
00198         label_dp->setGeometry(QRect(14, 22, 144, 16));
00199         label_dp->setFont(font1);
00200         label_dp->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00201 "border: 0px"));
00202         label_4 = new QLabel(frame);
00203         label_4->setObjectName("label_4");
00204         label_4->setGeometry(QRect(14, 48, 180, 16));
00205         label_4->setFont(font1);
00206         label_4->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00207 "border: 0px"));
00208         dpToleranz = new QSpinBox(frame);
00209         dpToleranz->setObjectName("dpToleranz");
00210         dpToleranz->setGeometry(QRect(200, 22, 116, 20));
00211         dpToleranz->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00212 "\n"
00213 "border-radius: 6px;"));
00214         dpToleranz->setFrame(true);
00215         dpToleranz->setButtonSymbols(QAbstractSpinBox::UpDownArrows);
00216         dpToleranz->setAccelerated(false);

```

```

00217         bLogging = new QCheckBox(frame);
00218         bLogging->setObjectName("bLogging");
00219         bLogging->setGeometry(QRect(15, 460, 151, 20));
00220         bLogging->setFont(font1);
00221         bLogging->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00222 "border: 0px"));
00223         bOffset = new QCheckBox(frame);
00224         bOffset->setObjectName("bOffset");
00225         bOffset->setGeometry(QRect(15, 310, 151, 20));
00226         bOffset->setFont(font1);
00227         bOffset->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00228 "border: 0px"));
00229         offset = new QWidget(frame);
00230         offset->setObjectName("offset");
00231         offset->setEnabled(false);
00232         offset->setGeometry(QRect(15, 340, 301, 111));
00233         offset->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00234 "color: rgb(117, 125, 149);\n"
00235 "border: 1px solid rgb(67, 72, 91);"));
00236         label_13 = new QLabel(offset);
00237         label_13->setObjectName("label_13");
00238         label_13->setGeometry(QRect(10, 15, 18, 16));
00239         label_13->setFont(font1);
00240         label_13->setStyleSheet(QString::fromUtf8("border: 0px"));
00241         label_14 = new QLabel(offset);
00242         label_14->setObjectName("label_14");
00243         label_14->setGeometry(QRect(11, 47, 33, 16));
00244         label_14->setFont(font1);
00245         label_14->setStyleSheet(QString::fromUtf8("\n"
00246 "border: 0px"));
00247         label_15 = new QLabel(offset);
00248         label_15->setObjectName("label_15");
00249         label_15->setGeometry(QRect(11, 78, 17, 16));
00250         label_15->setFont(font1);
00251         label_15->setStyleSheet(QString::fromUtf8("border: 0px"));
00252         offsetX = new QDoubleSpinBox(offset);
00253         offsetX->setObjectName("offsetX");
00254         offsetX->setGeometry(QRect(153, 15, 141, 20));
00255         offsetX->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00256 "\n"
00257 "border-radius: 6px;"));
00258         offsetY = new QDoubleSpinBox(offset);
00259         offsetY->setObjectName("offsetY");
00260         offsetY->setGeometry(QRect(154, 47, 141, 20));
00261         offsetY->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00262 "\n"
00263 "border-radius: 6px;"));
00264         offsetZ = new QDoubleSpinBox(offset);
00265         offsetZ->setObjectName("offsetZ");
00266         offsetZ->setGeometry(QRect(154, 78, 141, 20));
00267         offsetZ->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00268 "\n"
00269 "border-radius: 6px;"));
00270         pushInput = new QPushButton(centralWidget);
00271         pushInput->setObjectName("pushInput");
00272         pushInput->setGeometry(QRect(240, 10, 101, 31));
00273         pushInput->setFont(font1);
00274         pushInput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00275 "border-radius: 10px;\n"
00276 "background-color: rgb(117, 125, 149);\n"
00277 "color: rgb(3, 8, 14);"));
00278         pushOutput = new QPushButton(centralWidget);
00279         pushOutput->setObjectName("pushOutput");
00280         pushOutput->setGeometry(QRect(240, 50, 101, 31));
00281         pushOutput->setFont(font1);
00282         pushOutput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00283 "border-radius: 10px;\n"
00284 "background-color: rgb(117, 125, 149);\n"
00285 "color: rgb(3, 8, 14);"));
00286         textBrowser = new QTextBrowser(centralWidget);
00287         textBrowser->setObjectName("textBrowser");
00288         textBrowser->setGeometry(QRect(0, 630, 355, 71));
00289         textBrowser->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00290 "border-radius: 0px"));
00291         startCalculation = new QPushButton(centralWidget);
00292         startCalculation->setObjectName("startCalculation");
00293         startCalculation->setGeometry(QRect(10, 590, 331, 31));
00294         sizePolicy.setHeightForWidth(startCalculation->sizePolicy().hasHeightForWidth());
00295         startCalculation->setSizePolicy(sizePolicy);
00296         QFont font3;
00297         font3.setFamilies({QString::fromUtf8("Rubik")});
00298         font3.setPointSize(10);
00299         font3.setBold(false);
00300         font3.setItalic(false);
00301         startCalculation->setFont(font3);
00302         startCalculation->setStyleSheet(QString::fromUtf8("border: 1px solid black;\n"
00303 "border-radius: 10px;\n"

```

```

00304 "background-color: rgb(67, 72, 91);\n"
00305 "color: rgb(210, 211, 218)");
00306     pathOutput = new QLabel(centralWidget);
00307     pathOutput->setObjectName("pathOutput");
00308     pathOutput->setGeometry(QRect(10, 50, 331, 31));
00309     pathOutput->setFont(font1);
00310     pathOutput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00311 "border-radius: 10px;\n"
00312 "background-color: rgb(210, 211, 218);\n"
00313 "color: rgb(3, 8, 14);"));
00314     pathOutput->setTextFormat(Qt::MarkdownText);
00315     GUIClass->setCentralWidget(centralWidget);
00316     pathInput->raise();
00317     frame->raise();
00318     textBrowser->raise();
00319     startCalculation->raise();
00320     pathOutput->raise();
00321     pushOutput->raise();
00322     pushInput->raise();
00323
00324     retranslateUi(GUIClass);
00325
00326     QMetaObject::connectSlotsByName(GUIClass);
00327 } // setupUi

```

Benutzt [AValue](#), [bLogging](#), [bManOrientation](#), [bOffset](#), [bSpeed](#), [BValue](#), [centralWidget](#), [CValue](#), [dpToleranz](#), [frame](#), [label_10](#), [label_11](#), [label_12](#), [label_13](#), [label_14](#), [label_15](#), [label_4](#), [label_5](#), [label_dp](#), [meanLength](#), [offset](#), [offsetX](#), [offsetY](#), [offsetZ](#), [orientation](#), [pathInput](#), [pathOutput](#), [pushInput](#), [pushOutput](#), [retranslateUi\(\)](#), [speed](#), [speed_2](#), [startCalculation](#) und [textBrowser](#).

Wird benutzt von [GUI::GUI\(\)](#).

7.16.2.4 setupUi() [2/2]

```

void Ui_GUIClass::setupUi (
    QMainWindow * GUIClass ) [inline]

```

Definiert in Zeile 63 der Datei [ui_GUI.h](#).

```

00064 {
00065     if (GUIClass->objectName().isEmpty())
00066         GUIClass->setObjectName("GUIClass");
00067     GUIClass->resize(355, 700);
00068     QSizePolicy sizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);
00069     sizePolicy.setHorizontalStretch(0);
00070     sizePolicy.setVerticalStretch(0);
00071     sizePolicy.setHeightForWidth(GUIClass->sizePolicy().hasHeightForWidth());
00072     GUIClass->setSizePolicy(sizePolicy);
00073     GUIClass->setMinimumSize(QSize(355, 530));
00074     GUIClass->setMaximumSize(QSize(355, 700));
00075     QFont font;
00076     font.setFamilies({QString::fromUtf8("Rubik")});
00077     GUIClass->setFont(font);
00078     GUIClass->setAnimated(true);
00079     GUIClass->setTabShape(QTabWidget::Rounded);
00080     GUIClass->setUnifiedTitleAndToolBarOnMac(false);
00081     centralWidget = new QWidget(GUIClass);
00082     centralWidget->setObjectName("centralWidget");
00083     QFont font1;
00084     font1.setFamilies({QString::fromUtf8("Rubik")});
00085     font1.setBold(false);
00086     font1.setItalic(false);
00087     centralWidget->setFont(font1);
00088     centralWidget->setStyleSheet(QString::fromUtf8("background-color: rgb(3, 8, 14);\n"
00089 "color: rgb(3, 8, 14);\n"
00090 "));
00091     pathInput = new QLabel(centralWidget);
00092     pathInput->setObjectName("pathInput");
00093     pathInput->setGeometry(QRect(10, 10, 331, 31));
00094     pathInput->setFont(font1);
00095     pathInput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00096 "border-radius: 10px;\n"
00097 "background-color: rgb(210, 211, 218);\n"
00098 "color: rgb(3, 8, 14);"));
00099     frame = new QFrame(centralWidget);
00100     frame->setObjectName("frame");
00101     frame->setGeometry(QRect(10, 90, 331, 491));
00102     frame->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00103 "border-radius: 10px;\n"

```

```

00104 "background-color: rgb(117, 125, 149)");
00105     frame->setFrameShape(QFrame::Box);
00106     frame->setFrameShadow(QFrame::Raised);
00107     frame->setLineWidth(1);
00108     speed_2 = new QWidget(frame);
00109     speed_2->setObjectName("speed_2");
00110     speed_2->setEnabled(false);
00111     speed_2->setGeometry(QRect(15, 110, 301, 41));
00112     QFont font2;
00113     font2.setFamilies({QString::fromUtf8("Rubik")});
00114     font2.setBold(true);
00115     speed_2->setFont(font2);
00116     speed_2->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00117 "color: rgb(117, 125, 149);\n"
00118 "border: 1px solid rgb(67, 72, 91); ");
00119     speed = new QDoubleSpinBox(speed_2);
00120     speed->setObjectName("speed");
00121     speed->setGeometry(QRect(159, 12, 136, 20));
00122     speed->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00123 "\n"
00124 "border-radius: 6px;"));
00125     speed->setAlignment(Qt::AlignCenter);
00126     label_5 = new QLabel(speed_2);
00127     label_5->setObjectName("label_5");
00128     label_5->setGeometry(QRect(11, 12, 93, 16));
00129     label_5->setFont(font1);
00130     label_5->setStyleSheet(QString::fromUtf8("border: 0px"));
00131     bSpeed = new QCheckBox(frame);
00132     bSpeed->setObjectName("bSpeed");
00133     bSpeed->setGeometry(QRect(15, 80, 171, 20));
00134     bSpeed->setFont(font1);
00135     bSpeed->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00136 "border: 0px"));
00137     orientation = new QWidget(frame);
00138     orientation->setObjectName("orientation");
00139     orientation->setEnabled(false);
00140     orientation->setGeometry(QRect(15, 190, 301, 111));
00141     orientation->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00142 "color: rgb(117, 125, 149);\n"
00143 "border: 1px solid rgb(67, 72, 91); ");
00144     label_10 = new QLabel(orientation);
00145     label_10->setObjectName("label_10");
00146     label_10->setGeometry(QRect(10, 15, 18, 16));
00147     label_10->setFont(font1);
00148     label_10->setStyleSheet(QString::fromUtf8("border: 0px"));
00149     label_11 = new QLabel(orientation);
00150     label_11->setObjectName("label_11");
00151     label_11->setGeometry(QRect(11, 47, 33, 16));
00152     label_11->setFont(font1);
00153     label_11->setStyleSheet(QString::fromUtf8("\n"
00154 "border: 0px"));
00155     label_12 = new QLabel(orientation);
00156     label_12->setObjectName("label_12");
00157     label_12->setGeometry(QRect(11, 78, 17, 16));
00158     label_12->setFont(font1);
00159     label_12->setStyleSheet(QString::fromUtf8("border: 0px"));
00160     AValue = new QDoubleSpinBox(orientation);
00161     AValue->setObjectName("AValue");
00162     AValue->setGeometry(QRect(153, 15, 141, 20));
00163     AValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00164 "\n"
00165 "border-radius: 6px;"));
00166     BValue = new QDoubleSpinBox(orientation);
00167     BValue->setObjectName("BValue");
00168     BValue->setGeometry(QRect(154, 47, 141, 20));
00169     BValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00170 "\n"
00171 "border-radius: 6px;"));
00172     CValue = new QDoubleSpinBox(orientation);
00173     CValue->setObjectName("CValue");
00174     CValue->setGeometry(QRect(154, 78, 141, 20));
00175     CValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00176 "\n"
00177 "border-radius: 6px;"));
00178     label_12->raise();
00179     label_10->raise();
00180     AValue->raise();
00181     BValue->raise();
00182     CValue->raise();
00183     label_11->raise();
00184     bManOrientation = new QCheckBox(frame);
00185     bManOrientation->setObjectName("bManOrientation");
00186     bManOrientation->setGeometry(QRect(15, 160, 151, 20));
00187     bManOrientation->setFont(font1);
00188     bManOrientation->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00189 "border: 0px"));
00190     meanLength = new QSpinBox(frame);

```

```

00191         meanLength->setObjectName("meanLength");
00192         meanLength->setGeometry(QRect(200, 48, 116, 20));
00193         meanLength->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00194 "\n"
00195 "border-radius: 6px;"));
00196         label_dp = new QLabel(frame);
00197         label_dp->setObjectName("label_dp");
00198         label_dp->setGeometry(QRect(14, 22, 144, 16));
00199         label_dp->setFont(font1);
00200         label_dp->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00201 "border: 0px;"));
00202         label_4 = new QLabel(frame);
00203         label_4->setObjectName("label_4");
00204         label_4->setGeometry(QRect(14, 48, 180, 16));
00205         label_4->setFont(font1);
00206         label_4->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00207 "border: 0px;"));
00208         dpToleranz = new QSpinBox(frame);
00209         dpToleranz->setObjectName("dpToleranz");
00210         dpToleranz->setGeometry(QRect(200, 22, 116, 20));
00211         dpToleranz->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00212 "\n"
00213 "border-radius: 6px;"));
00214         dpToleranz->setFrame(true);
00215         dpToleranz->setButtonSymbols(QAbstractSpinBox::UpDownArrows);
00216         dpToleranz->setAccelerated(false);
00217         bLogging = new QCheckBox(frame);
00218         bLogging->setObjectName("bLogging");
00219         bLogging->setGeometry(QRect(15, 460, 151, 20));
00220         bLogging->setFont(font1);
00221         bLogging->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00222 "border: 0px;"));
00223         bOffset = new QCheckBox(frame);
00224         bOffset->setObjectName("bOffset");
00225         bOffset->setGeometry(QRect(15, 310, 151, 20));
00226         bOffset->setFont(font1);
00227         bOffset->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00228 "border: 0px;"));
00229         offset = new QWidget(frame);
00230         offset->setObjectName("offset");
00231         offset->setEnabled(false);
00232         offset->setGeometry(QRect(15, 340, 301, 111));
00233         offset->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00234 "color: rgb(117, 125, 149);\n"
00235 "border: 1px solid rgb(67, 72, 91); "));
00236         label_13 = new QLabel(offset);
00237         label_13->setObjectName("label_13");
00238         label_13->setGeometry(QRect(10, 15, 18, 16));
00239         label_13->setFont(font1);
00240         label_13->setStyleSheet(QString::fromUtf8("border: 0px;"));
00241         label_14 = new QLabel(offset);
00242         label_14->setObjectName("label_14");
00243         label_14->setGeometry(QRect(11, 47, 33, 16));
00244         label_14->setFont(font1);
00245         label_14->setStyleSheet(QString::fromUtf8("\n"
00246 "border: 0px;"));
00247         label_15 = new QLabel(offset);
00248         label_15->setObjectName("label_15");
00249         label_15->setGeometry(QRect(11, 78, 17, 16));
00250         label_15->setFont(font1);
00251         label_15->setStyleSheet(QString::fromUtf8("border: 0px;"));
00252         offsetX = new QDoubleSpinBox(offset);
00253         offsetX->setObjectName("offsetX");
00254         offsetX->setGeometry(QRect(153, 15, 141, 20));
00255         offsetX->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00256 "\n"
00257 "border-radius: 6px;"));
00258         offsetY = new QDoubleSpinBox(offset);
00259         offsetY->setObjectName("offsetY");
00260         offsetY->setGeometry(QRect(154, 47, 141, 20));
00261         offsetY->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00262 "\n"
00263 "border-radius: 6px;"));
00264         offsetZ = new QDoubleSpinBox(offset);
00265         offsetZ->setObjectName("offsetZ");
00266         offsetZ->setGeometry(QRect(154, 78, 141, 20));
00267         offsetZ->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00268 "\n"
00269 "border-radius: 6px;"));
00270         pushInput = new QPushButton(centralWidget);
00271         pushInput->setObjectName("pushInput");
00272         pushInput->setGeometry(QRect(240, 10, 101, 31));
00273         pushInput->setFont(font1);
00274         pushInput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00275 "border-radius: 10px;\n"
00276 "background-color: rgb(117, 125, 149);\n"
00277 "color: rgb(3, 8, 14);"));

```

```

00278         pushOutput = new QPushButton(centralWidget);
00279         pushOutput->setObjectName("pushOutput");
00280         pushOutput->setGeometry(QRect(240, 50, 101, 31));
00281         pushOutput->setFont(font1);
00282         pushOutput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00283 "border-radius: 10px;\n"
00284 "background-color: rgb(117, 125, 149);\n"
00285 "color: rgb(3, 8, 14);"));
00286         textBrowser = new QTextBrowser(centralWidget);
00287         textBrowser->setObjectName("textBrowser");
00288         textBrowser->setGeometry(QRect(0, 630, 355, 71));
00289         textBrowser->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00290 "border-radius: 0px;"));
00291         startCalculation = new QPushButton(centralWidget);
00292         startCalculation->setObjectName("startCalculation");
00293         startCalculation->setGeometry(QRect(10, 590, 331, 31));
00294         sizePolicy.setHeightForWidth(startCalculation->sizePolicy().hasHeightForWidth());
00295         startCalculation->setSizePolicy(sizePolicy);
00296         QFont font3;
00297         font3.setFamilies({QString::fromUtf8("Rubik")});
00298         font3.setPointSize(10);
00299         font3.setBold(false);
00300         font3.setItalic(false);
00301         startCalculation->setFont(font3);
00302         startCalculation->setStyleSheet(QString::fromUtf8("border: 1px solid black;\n"
00303 "border-radius: 10px;\n"
00304 "background-color: rgb(67, 72, 91);\n"
00305 "color: rgb(210, 211, 218);"));
00306         pathOutput = new QLabel(centralWidget);
00307         pathOutput->setObjectName("pathOutput");
00308         pathOutput->setGeometry(QRect(10, 50, 331, 31));
00309         pathOutput->setFont(font1);
00310         pathOutput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00311 "border-radius: 10px;\n"
00312 "background-color: rgb(210, 211, 218);\n"
00313 "color: rgb(3, 8, 14);"));
00314         pathOutput->setTextFormat(Qt::MarkdownText);
00315         GUIClass->setCentralWidget(centralWidget);
00316         pathInput->raise();
00317         frame->raise();
00318         textBrowser->raise();
00319         startCalculation->raise();
00320         pathOutput->raise();
00321         pushOutput->raise();
00322         pushInput->raise();
00323
00324         retranslateUi(GUIClass);
00325
00326         QMetaObject::connectSlotsByName(GUIClass);
00327     } // setupUi

```

Benutzt [AValue](#), [bLogging](#), [bManOrientation](#), [bOffset](#), [bSpeed](#), [BValue](#), [centralWidget](#), [CValue](#), [dpToleranz](#), [frame](#), [label_10](#), [label_11](#), [label_12](#), [label_13](#), [label_14](#), [label_15](#), [label_4](#), [label_5](#), [label_dp](#), [meanLength](#), [offset](#), [offsetX](#), [offsetY](#), [offsetZ](#), [orientation](#), [pathInput](#), [pathOutput](#), [pushInput](#), [pushOutput](#), [retranslateUi\(\)](#), [speed](#), [speed_2](#), [startCalculation](#) und [textBrowser](#).

7.16.3 Dokumentation der Datenelemente

7.16.3.1 AValue

`QDoubleSpinBox * Ui_GUIClass::AValue`

Definiert in Zeile 40 der Datei [ui_GUI.h](#).

Wird benutzt von [GUI::GUI\(\)](#), [GUI::setOrientation\(\)](#) und [setupUi\(\)](#).

7.16.3.2 bLogging

`QCheckBox * Ui_GUIClass::bLogging`

Definiert in Zeile 48 der Datei [ui_GUI.h](#).

Wird benutzt von [GUI::activateLogging\(\)](#), [GUI::GUI\(\)](#), [retranslateUi\(\)](#) und [setupUi\(\)](#).

7.16.3.3 bManOrientation

`QCheckBox * Ui_GUIClass::bManOrientation`

Definiert in Zeile 43 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::activateOrientation()`, `GUI::GUI()`, `retranslateUi()`, `GUI::setOrientation()` und `setupUi()`.

7.16.3.4 bOffset

`QCheckBox * Ui_GUIClass::bOffset`

Definiert in Zeile 49 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::activateOffset()`, `GUI::GUI()`, `retranslateUi()`, `GUI::setOffset()` und `setupUi()`.

7.16.3.5 bSpeed

`QCheckBox * Ui_GUIClass::bSpeed`

Definiert in Zeile 35 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::activateSpeed()`, `GUI::GUI()`, `retranslateUi()`, `GUI::setSpeed()` und `setupUi()`.

7.16.3.6 BValue

`QDoubleSpinBox * Ui_GUIClass::BValue`

Definiert in Zeile 41 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::GUI()`, `GUI::setOrientation()` und `setupUi()`.

7.16.3.7 centralWidget

`QWidget * Ui_GUIClass::centralWidget`

Definiert in Zeile 29 der Datei `ui_GUI.h`.

Wird benutzt von `setupUi()`.

7.16.3.8 CValue

`QDoubleSpinBox * Ui_GUIClass::CValue`

Definiert in Zeile 42 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::GUI()`, `GUI::setOrientation()` und `setupUi()`.

7.16.3.9 dpToleranz

```
QSpinBox * Ui_GUIClass::dpToleranz
```

Definiert in Zeile 47 der Datei [ui_GUI.h](#).

Wird benutzt von [GUI::GUI\(\)](#), [retranslateUi\(\)](#), [GUI::setDP\(\)](#) und [setupUi\(\)](#).

7.16.3.10 frame

```
QFrame * Ui_GUIClass::frame
```

Definiert in Zeile 31 der Datei [ui_GUI.h](#).

Wird benutzt von [setupUi\(\)](#).

7.16.3.11 label_10

```
QLabel * Ui_GUIClass::label_10
```

Definiert in Zeile 37 der Datei [ui_GUI.h](#).

Wird benutzt von [retranslateUi\(\)](#) und [setupUi\(\)](#).

7.16.3.12 label_11

```
QLabel * Ui_GUIClass::label_11
```

Definiert in Zeile 38 der Datei [ui_GUI.h](#).

Wird benutzt von [retranslateUi\(\)](#) und [setupUi\(\)](#).

7.16.3.13 label_12

```
QLabel * Ui_GUIClass::label_12
```

Definiert in Zeile 39 der Datei [ui_GUI.h](#).

Wird benutzt von [retranslateUi\(\)](#) und [setupUi\(\)](#).

7.16.3.14 label_13

```
QLabel * Ui_GUIClass::label_13
```

Definiert in Zeile 51 der Datei [ui_GUI.h](#).

Wird benutzt von [retranslateUi\(\)](#) und [setupUi\(\)](#).

7.16.3.15 label_14

```
QLabel * Ui_GUIClass::label_14
```

Definiert in Zeile 52 der Datei [ui_GUI.h](#).

Wird benutzt von [retranslateUi\(\)](#) und [setupUi\(\)](#).

7.16.3.16 label_15

```
QLabel * Ui_GUIClass::label_15
```

Definiert in Zeile 53 der Datei [ui_GUI.h](#).

Wird benutzt von [retranslateUi\(\)](#) und [setupUi\(\)](#).

7.16.3.17 label_4

```
QLabel * Ui_GUIClass::label_4
```

Definiert in Zeile 46 der Datei [ui_GUI.h](#).

Wird benutzt von [retranslateUi\(\)](#) und [setupUi\(\)](#).

7.16.3.18 label_5

```
QLabel * Ui_GUIClass::label_5
```

Definiert in Zeile 34 der Datei [ui_GUI.h](#).

Wird benutzt von [retranslateUi\(\)](#) und [setupUi\(\)](#).

7.16.3.19 label_dp

```
QLabel * Ui_GUIClass::label_dp
```

Definiert in Zeile 45 der Datei [ui_GUI.h](#).

Wird benutzt von [retranslateUi\(\)](#) und [setupUi\(\)](#).

7.16.3.20 meanLength

```
QSpinBox * Ui_GUIClass::meanLength
```

Definiert in Zeile 44 der Datei [ui_GUI.h](#).

Wird benutzt von [GUI::GUI\(\)](#), [GUI::setMean\(\)](#) und [setupUi\(\)](#).

7.16.3.21 offset

```
QWidget * Ui_GUIClass::offset
```

Definiert in Zeile 50 der Datei [ui_GUI.h](#).

Wird benutzt von [GUI::activateOffset\(\)](#) und [setupUi\(\)](#).

7.16.3.22 offsetX

```
QDoubleSpinBox * Ui_GUIClass::offsetX
```

Definiert in Zeile 54 der Datei [ui_GUI.h](#).

Wird benutzt von [GUI::activateOffset\(\)](#), [GUI::GUI\(\)](#), [GUI::setOffset\(\)](#) und [setupUi\(\)](#).

7.16.3.23 offsetY

```
QDoubleSpinBox * Ui_GUIClass::offsetY
```

Definiert in Zeile 55 der Datei [ui_GUI.h](#).

Wird benutzt von [GUI::activateOffset\(\)](#), [GUI::GUI\(\)](#), [GUI::setOffset\(\)](#) und [setupUi\(\)](#).

7.16.3.24 offsetZ

```
QDoubleSpinBox * Ui_GUIClass::offsetZ
```

Definiert in Zeile 56 der Datei [ui_GUI.h](#).

Wird benutzt von [GUI::activateOffset\(\)](#), [GUI::GUI\(\)](#), [GUI::setOffset\(\)](#) und [setupUi\(\)](#).

7.16.3.25 orientation

```
QWidget * Ui_GUIClass::orientation
```

Definiert in Zeile 36 der Datei [ui_GUI.h](#).

Wird benutzt von [GUI::activateOrientation\(\)](#) und [setupUi\(\)](#).

7.16.3.26 pathInput

```
QLabel * Ui_GUIClass::pathInput
```

Definiert in Zeile 30 der Datei [ui_GUI.h](#).

Wird benutzt von [GUI::GUI\(\)](#), [retranslateUi\(\)](#), [GUI::setInputPath\(\)](#) und [setupUi\(\)](#).

7.16.3.27 pathOutput

`QLabel * Ui_GUIClass::pathOutput`

Definiert in Zeile 61 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::GUI()`, `retranslateUi()`, `GUI::setOutputPath()` und `setupUi()`.

7.16.3.28 pushInput

`QPushButton * Ui_GUIClass::pushInput`

Definiert in Zeile 57 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::GUI()`, `retranslateUi()` und `setupUi()`.

7.16.3.29 pushOutput

`QPushButton * Ui_GUIClass::pushOutput`

Definiert in Zeile 58 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::GUI()`, `retranslateUi()` und `setupUi()`.

7.16.3.30 speed

`QDoubleSpinBox * Ui_GUIClass::speed`

Definiert in Zeile 33 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::GUI()`, `retranslateUi()`, `GUI::setSpeed()` und `setupUi()`.

7.16.3.31 speed_2

`QWidget * Ui_GUIClass::speed_2`

Definiert in Zeile 32 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::activateSpeed()` und `setupUi()`.

7.16.3.32 startCalculation

`QPushButton * Ui_GUIClass::startCalculation`

Definiert in Zeile 60 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::GUI()`, `retranslateUi()` und `setupUi()`.

7.16.3.33 textBrowser

`QTextBrowser * Ui_GUIClass::textBrowser`

Definiert in Zeile 59 der Datei `ui_GUI.h`.

Wird benutzt von `GUI::calculate()` und `setupUi()`.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- `x64/Debug/uic/ui_GUI.h`
- `x64/Release/uic/ui_GUI.h`

Kapitel 8

Datei-Dokumentation

8.1 header/EulerMatrix.h-Dateireferenz

Header File handling Euler Matrix.

```
#include <tuple>
#include <cmath>
```

Klassen

- class [CEulerMatrix](#)
Handling und Berechnung Euler Matrix.

Makrodefinitionen

- #define [_USE_MATH_DEFINES](#)

8.1.1 Ausführliche Beschreibung

Header File handling Euler Matrix.

Definiert in Datei [EulerMatrix.h](#).

8.1.2 Makro-Dokumentation

8.1.2.1 _USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

Definiert in Zeile [11](#) der Datei [EulerMatrix.h](#).

8.2 EulerMatrix.h

[gehe zur Dokumentation dieser Datei](#)

```

00001
00006 #pragma once
00007
00008 #include <tuple>
00009 #include <cmath>
00010
00011 #define _USE_MATH_DEFINES
00012
00013 using namespace std;
00014
00019 class CEulerMatrix
00020 {
00021 public:
00027     CEulerMatrix(void);
00034     CEulerMatrix(float inputMatrix[3][3]);
00038     ~CEulerMatrix();
00039
00044     void setMatrix(float inputMatrix[3][3]);
00049     CEulerMatrix getEulerMatrix(void);
00050
00055     void getMatrix(float Matrix[][3]);
00063     CEulerMatrix angels2mat(double A, double B, double C);
00064
00069     tuple<double , double , double> calculateAngels(void);
00070
00071 private:
00075     float eulerMatrix[3][3];
00076
00077 };
00078

```

8.3 header/GUI.h-Dateireferenz

Header File handling the User Interface.

```

#include <QtWidgets/QMainWindow>
#include <QtWidgets/qfiledialog.h>
#include <QtWidgets/qmessagebox.h>
#include "InputParameter.h"
#include "ui_GUI.h"
#include <string>

```

Klassen

- class [GUI](#)

UI und Funktionen.

8.3.1 Ausführliche Beschreibung

Header File handling the User Interface.

Definiert in Datei [GUI.h](#).

8.4 GUI.h

[gehe zur Dokumentation dieser Datei](#)

```

00001
00007 #pragma once
00008
00009 #include <QtWidgets/QMainWindow>
00010 #include <QtWidgets/qfiledialog.h>
00011 #include <QtWidgets/qmessagebox.h>
00012
00013 #include "InputParameter.h"
00014 #include "ui_GUI.h"
00015 #include <string>
00016
00021 class GUI : public QMainWindow
00022 {
00023     Q_OBJECT
00024
00025 public:
00030     GUI(QWidget *parent = nullptr);
00034     ~GUI();
00035
00036 private slots:
00040     void calculate(void);
00044     void setInputPath(void);
00048     void setOutputPath(void);
00052     void setDP(void);
00056     void setMean(void);
00060     void activateSpeed(void);
00064     void setSpeed(void);
00068     void activateOrientation(void);
00072     void setOrientation(void);
00076     void activateOffset(void);
00080     void setOffset(void);
00084     void activateLogging(void);
00085
00086 private:
00090     Ui::GUIClass ui;
00094     QString inputPathUI;
00098     QString outputPathUI;
00099     double dpTolerance;
00100     double meanLength;
00101     CInputParameter inputParameter;
00102
00103 };

```

8.5 header/InputParameter.h-Dateireferenz

Header File Daten Einlesen.

```

#include "EulerMatrix.h"
#include "Point3D.h"
#include <string>
#include <vector>
#include <list>
#include <iostream>
#include <fstream>
#include <sstream>
#include <tuple>

```

Klassen

- class [CInputParameter](#)
Handling Eingabedaten.

8.5.1 Ausführliche Beschreibung

Header File Daten Einlesen.

Definiert in Datei [InputParameter.h](#).

8.6 InputParameter.h

[gehe zur Dokumentation dieser Datei](#)

```

00001
00007 #include "EulerMatrix.h"
00008 #include "Point3D.h"
00009 #include <string>
00010 #include <vector>
00011 #include <list>
00012 #include <iostream>
00013 #include <fstream>
00014 #include <sstream>
00015 #include <tuple>
00016
00017 using namespace std;
00018
00019 #pragma once
00020
00025 class CInputParameter
00026 {
00027 public:
00033     CInputParameter(void);
00047     CInputParameter(double initSpeed, bool initSeepManual, bool initOrientationManual, double initA,
double initB, double initC);
00051     ~CInputParameter(void);
00052
00061     void setOrientation(bool initOrientationManual, double initA, double initB, double initC);
00068     void setSpeed(double initSpeed, bool initSpeedManual);
00077     void setOffset(double X, double Y, double Z, bool offsetManual);
00083     void setLogging(bool initLoggingManual);
00088     double getSpeed(void);
00093     bool getSpeedManual(void);
00098     bool getOrientationManual(void);
00104     tuple <double, double, double> getAngles(void);
00109     bool getOffsetManual(void);
00114     bool getLoggingManual(void);
00120     tuple <double, double, double> getOffset(void);
00121
00127     void openFile(std::string path);
00136     bool detectJump(CInputPoint3D p, double x_prev, double y_prev, double z_prev);
00141     vector<list<CInputPoint3D>& getPath();
00142
00143 private:
00147     vector<list<CInputPoint3D>& initialPath;
00151     double speed;
00155     bool speedManual;
00159     bool orientationManual;
00163     double A;
00167     double B;
00171     double C;
00175     double difference = 20;
00179     bool offsetManual;
00183     double offsetX;
00187     double offsetY;
00191     double offsetZ;
00195     bool loggingManual;
00196 };
00197

```

8.7 header/Line3D.h-Dateireferenz

Header File Daten Einlesen.

```

#include "Point3D.h"
#include <math.h>

```


Klassen

- class [CLine3D](#)
Berechnung Geraden.

8.7.1 Ausführliche Beschreibung

Header File Daten Einlesen.

Definiert in Datei [Line3D.h](#).

8.8 Line3D.h

[gehe zur Dokumentation dieser Datei](#)

```
00001
00007 #include "Point3D.h"
00008 #include <math.h>
00009
00010 using namespace std;
00011
00012 #pragma once
00013
00018 class CLine3D
00019 {
00020 public:
00026     CLine3D(void);
00032     CLine3D(CPoint3D P1, CPoint3D P2);
00036     ~CLine3D(void);
00037
00041     CPoint3D p1;
00045     CPoint3D p2;
00046 };
00047
```

8.9 header/Logging.h-Dateireferenz

Logging der Daten.

```
#include "EulerMatrix.h"
#include "Point3D.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <list>
```

Klassen

- class [CLogging](#)
Gleitender Mittelwertfilter.

8.9.1 Ausführliche Beschreibung

Logging der Daten.

Definiert in Datei [Logging.h](#).

8.10 Logging.h

[gehe zur Dokumentation dieser Datei](#)

```
00001
00007 #include "EulerMatrix.h"
00008 #include "Point3D.h"
00009
00010 #include <iostream>
00011 #include <fstream>
00012 #include <sstream>
00013 #include <string>
00014 #include <vector>
00015 #include <list>
00016
00017 #pragma once
00022 class CLogging
00023 {
00024 public:
00030     CLogging(void);
00036     CLogging(string path, bool detailed);
00040     ~CLogging(void);
00045     void setStep(int Step);
00050     bool getDetailed(void);
00056     void logData(vector<list<CInputPoint3D>& sourcePath);
00062     void logData(vector<CInputPoint3D>& sourcePath);
00068     void logData(vector<COutputPoint3D>& sourcePath);
00069 private:
00073     int step;
00077     string path;
00081     bool detailed;
00082
00083
00084 };
00085
00086
```

8.11 header/MeanFilter.h-Dateireferenz

Berechnung des gleitenden Mittelwertfilters.

```
#include <vector>
#include <list>
#include <string>
#include "Point3D.h"
#include "Logging.h"
```

Klassen

- class [CMeanFilter](#)
Gleitender Mittelwertfilter.

8.11.1 Ausführliche Beschreibung

Berechnung des gleitenden Mittelwertfilters.

Definiert in Datei [MeanFilter.h](#).

8.12 MeanFilter.h

[gehe zur Dokumentation dieser Datei](#)

```

00001
00007 #include <vector>
00008 #include <list>
00009 #include <string>
00010 #include "Point3D.h"
00011 #include "Logging.h"
00012
00013 #pragma once
00014
00015 using namespace std;
00016
00021 class CMeanFilter
00022 {
00023 public:
00029     CMeanFilter();
00036     CMeanFilter(int Window);
00040     ~CMeanFilter();
00041
00046     void setWindowSize(int Window);
00047
00052     int getWindowSize();
00057     vector<list<CInputPoint3D>&& getPath();
00058
00065     list<CInputPoint3D> calculateMean(list<CInputPoint3D>& segment);
00073     void mean(vector<list<CInputPoint3D>&& sourcePath, CLogging log);
00074
00075 private:
00079     int windowSize;
00083     vector<list<CInputPoint3D>&& meanPath;
00084 };
00085

```

8.13 header/PathBuilder.h-Dateireferenz

Setzt die einzelnen Segmente zu einem Vector zusammen.

```

#include <vector>
#include <list>
#include <iostream>
#include "Point3D.h"
#include "Logging.h"

```

Klassen

- class [CPathBuilder](#)

Zusammensetzen des Pfades.

8.13.1 Ausführliche Beschreibung

Setzt die einzelnen Segmente zu einem Vector zusammen.

Definiert in Datei [PathBuilder.h](#).

8.14 PathBuilder.h

[gehe zur Dokumentation dieser Datei](#)

```
00001
00007 #include <vector>
00008 #include <list>
00009 #include <iostream>
00010 #include "Point3D.h"
00011 #include "Logging.h"
00012
00013 using namespace std;
00014
00015 #pragma once
00016
00021 class CPathBuilder
00022 {
00023 public:
00028     CPathBuilder(void);
00032     ~CPathBuilder(void);
00033
00038     vector<CInputPoint3D>& getPath();
00044     void createPath(vector<list<CInputPoint3D>& segments, CLogging log);
00045
00046 private:
00050     vector<CInputPoint3D> path;
00051 };
00052
```

8.15 header/Point3D.h-Dateireferenz

Verarbeitung der Punkte.

```
#include "EulerMatrix.h"
```

Klassen

- class [CPoint3D](#)
Grundklasse Punkt.
- class [CInputPoint3D](#)
Input Punkt.
- class [COutputPoint3D](#)
Output Punkt.

8.15.1 Ausführliche Beschreibung

Verarbeitung der Punkte.

Definiert in Datei [Point3D.h](#).

8.16 Point3D.h

[gehe zur Dokumentation dieser Datei](#)

```

00001
00007 #include "EulerMatrix.h"
00008
00009 class CLine3D;
00010
00011 using namespace std;
00012
00013 #pragma once
00014
00020 class CPoint3D
00021 {
00022 public:
00028     CPoint3D(void);
00037     CPoint3D(double X, double Y, double Z);
00041     ~CPoint3D(void);
00042
00047     double getX();
00052     double getY();
00057     double getZ();
00058
00063     void setX(double X);
00068     void setY(double Y);
00073     void setZ(double Z);
00074
00081     void set(double X, double Y, double Z);
00087     double distanceTo(CPoint3D point);
00093     double distanceTo(CLine3D line);
00094
00095 protected:
00099     double x, y, z;
00100 };
00101
00106 class CInputPoint3D : public CPoint3D
00107 {
00108 public:
00114     CInputPoint3D(void);
00125     CInputPoint3D(double X, double Y, double Z, double Timestamp, CEulerMatrix Matrix);
00129     ~CInputPoint3D(void);
00130
00135     double getTime();
00140     CEulerMatrix getEulerMatrix();
00141
00146     void setTime(double time);
00151     void setEulerMatrix(CEulerMatrix orientation);
00160     void setPoint(double time, double X, double Y, double Z, CEulerMatrix orientation);
00161
00162 private:
00166     double timestamp;
00170     CEulerMatrix orientationMatrix;
00171 };
00172
00177 class COutputPoint3D : public CPoint3D
00178 {
00179 public:
00185     COutputPoint3D(void);
00198     COutputPoint3D(double Speed, double X, double Y, double Z, double A, double B, double C);
00202     ~COutputPoint3D(void);
00203
00208     double getSpeed();
00213     double getA();
00218     double getB();
00223     double getC();
00224
00229     void setSpeed(double speed);
00234     void setA(double A);
00239     void setB(double B);
00244     void setC(double C);
00245 private:
00249     double a, b, c;
00253     double speed;
00254 };

```

8.17 header/RobCodeGenerator.h-Dateireferenz

Erstellung des Roboter Codes.

```
#include <vector>
#include <iostream>
#include "Point3D.h"
#include "Logging.h"
#include "InputParameter.h"
#include <tuple>
```

Klassen

- class [CRobCodeGenerator](#)
Klasse zum erstellen des Roboter Codes.

Makrodefinitionen

- #define [MAX_SPEED](#) 2.0

8.17.1 Ausführliche Beschreibung

Erstellung des Roboter Codes.

Definiert in Datei [RobCodeGenerator.h](#).

8.17.2 Makro-Dokumentation

8.17.2.1 MAX_SPEED

```
#define MAX_SPEED 2.0
```

Definiert in Zeile 18 der Datei [RobCodeGenerator.h](#).

8.18 RobCodeGenerator.h

[gehe zur Dokumentation dieser Datei](#)

```
00001
00007 #include <vector>
00008 #include <iostream>
00009 #include "Point3D.h"
00010 #include "Logging.h"
00011 #include "InputParameter.h"
00012 #include <tuple>
00013
00014 using namespace std;
00015
00016 #pragma once
00017
00018 #define MAX_SPEED 2.0
00019
00025 class CRobCodeGenerator
00026 {
00027 public:
00033     CRobCodeGenerator(void);
00045     CRobCodeGenerator(CInputParameter inputParam);
00049     ~CRobCodeGenerator(void);
00050
00057     void generateRobCode(vector<CInputPoint3D>& path, string filepath, string filename, CLogging log);
00064     void postProcessing(vector<CInputPoint3D>& path);
00072     double calculateSpeed(CInputPoint3D& p, size_t i, double timePrev);
00078     void calculateAngles(COutputPoint3D& p, CInputPoint3D& pIn);
00079
00080 private:
00084     vector<COutputPoint3D> processedPath;
00088     CInputParameter input;
00089
00090 };
00091
```

8.19 header/SegmentApproximator.h-Dateireferenz

Berechnung des Douglas Peuker Algorithmusses.

```
#include <vector>
#include <list>
#include <iostream>
#include <math.h>
#include "Point3D.h"
#include "Logging.h"
```

Klassen

- class [CSegmentApproximator](#)

Ausduennen des Pfades.

8.19.1 Ausführliche Beschreibung

Berechnung des Douglas Peuker Algorithmusses.

Definiert in Datei [SegmentApproximator.h](#).

8.20 SegmentApproximator.h

[gehe zur Dokumentation dieser Datei](#)

```
00001
00007 #include <vector>
00008 #include <list>
00009 #include <iostream>
00010 #include <math.h>
00011 #include "Point3D.h"
00012 #include "Logging.h"
00013
00014 using namespace std;
00015
00016 #pragma once
00017
00022 class CSegmentApproximator
00023 {
00024 public:
00029     CSegmentApproximator(void);
00033     ~CSegmentApproximator(void);
00034
00041     void approx(const vector<list<CInputPoint3D>& Segments, CLogging log);
00046     void setmaxDistance(double maxDistanceSource);
00051     double getmaxDistance();
00052
00057     vector<list<CInputPoint3D>& getSegmentsApproxVector();
00058
00059 private:
00063     vector<list<CInputPoint3D> segmentsApprox;
00067     double maxDistance;
00068
00077     void douglasPeuckerRecursive(list<CInputPoint3D>& segment, std::list<CInputPoint3D>::iterator
startItr, std::list<CInputPoint3D>::iterator endItr);
00078 };
```

8.21 source/EulerMatrix.cpp-Dateireferenz

```
#include "../header/EulerMatrix.h"
#include <math.h>
```

8.22 EulerMatrix.cpp

[gehe zur Dokumentation dieser Datei](#)

```
00001 #include "../header/EulerMatrix.h"
00002 #include <math.h>
00003
00004 CEulerMatrix::CEulerMatrix(void)
00005 {
00006     for (int i = 0; i < 3; i++)
00007     {
00008         for (int m = 0; m < 3; m++)
00009         {
00010             eulerMatrix[i][m] = 0; // eulerMatrix mit 0 initialisieren
00011         }
00012     }
00013 }
00014
00015 CEulerMatrix::CEulerMatrix(float inputMatrix[3][3])
00016 {
00017     for (int i = 0; i < 3; i++)
00018     {
00019         for (int m = 0; m < 3; m++)
00020         {
00021             eulerMatrix[i][m] = inputMatrix[i][m]; // eulerMatrix mit Startwerten initialisieren
00022         }
00023     }
00024 }
00025
00026 CEulerMatrix::~CEulerMatrix()
00027 {
00028 }
00029
00030
00031 void CEulerMatrix::setMatrix(float inputMatrix[3][3])
00032 {
00033     for (int i = 0; i < 3; i++)
00034     {
00035         for (int m = 0; m < 3; m++)
00036         {
00037             eulerMatrix[i][m] = inputMatrix[i][m]; // eulerMatrix mit Uebergabewerten ueberschreiben
00038         }
00039     }
00040 }
00041
00042 CEulerMatrix CEulerMatrix::getEulerMatrix()
00043 {
00044     return eulerMatrix; // EulerMatrix zurueck geben
00045 }
00046
00047 void CEulerMatrix::getMatrix(float Matrix[][3])
00048 {
00049     for (int i = 0; i < 3; i++)
00050     {
00051         for (int m = 0; m < 3; m++)
00052         {
00053             Matrix[i][m] = eulerMatrix[i][m]; // eulerMatrix mit Uebergabewerten ueberschreiben
00054         }
00055     }
00056 }
00057
00058 //TODO: Kommentar
00059 CEulerMatrix CEulerMatrix::angels2mat(double A, double B, double C)
00060 {
00061     float Matrix[3][3]; // DummyMatrix erstellen
00062
00063     /* Berechnung der Matrix */
00064
00065     Matrix[0][0] = cos(A) * cos(C) - sin(A) * cos(B) * sin(C);
00066     Matrix[0][1] = -cos(A) * sin(C) - sin(A) * cos(B) * cos(C);
00067     Matrix[0][2] = sin(A) * sin(B);
00068
00069     Matrix[1][0] = sin(A) * cos(C) + cos(A) * cos(B) * sin(C);
```



```

00070     Matrix[1][1] = -sin(A) * sin(C) + cos(A) * cos(B) * cos(C);
00071     Matrix[1][2] = -cos(A) * sin(B);
00072
00073     Matrix[2][0] = sin(B) * sin(C);
00074     Matrix[2][1] = sin(B) * cos(C);
00075     Matrix[2][2] = cos(B);
00076
00077     CEulerMatrix buffer(Matrix);          // DummyMatrix in DummyEulerMatrix schreiben
00078     return buffer;                        // Matrix zurueck geben
00079 }
00080
00081 //TODO: Kommentar
00082 tuple<double , double , double> CEulerMatrix::calculateAngels(void)
00083 {
00084     double a, b, c, sin_a, cos_a, sin_b, abs_cos_b, sin_c, cos_c;
00085
00086     /*
00087     a == Winkel Alpha
00088     b == Winkel Beta
00089     c == Winkel Gamma
00090
00091     sin_a == sinus alpha
00092     cos_a == cosinus alpha
00093     sin_b == Matrix[2][0] * -1
00094     abs_cos_b == ??
00095     sin_c == sinus gamma
00096     cos_c == cosinus gamma
00097     */
00098
00099     /* Berechnung von alpha */
00100     a = atan2(eulerMatrix[1][0], eulerMatrix[0][0]);
00101
00102     /* Berechnung von beta */
00103     sin_a = sin(a);
00104     cos_a = cos(a);
00105     sin_b = eulerMatrix[2][0] * -1;
00106     abs_cos_b = cos(a) * eulerMatrix[0][0] + sin(a) * eulerMatrix[1][0];
00107
00108     b = atan2 (sin_b, abs_cos_b);
00109
00110     /* Berechnung von gamma */
00111     sin_c = sin_a * eulerMatrix[0][2] - cos_a * eulerMatrix[1][2];
00112     cos_c = -sin_a * eulerMatrix[0][1] + cos_a * eulerMatrix[1][1];
00113
00114     c = atan2(sin_c, cos_c);
00115
00116     /* Bogenmass in Gradmass umrechnen */
00117     a = a * 180 / M_PI;
00118     b = b * 180 / M_PI;
00119     c = c * 180 / M_PI;
00120
00121
00122     return make_tuple(a, b, c);          // Rueckgabe der Winkel
00123 }
00124
00125

```

8.23 source/GUI.cpp-Dateireferenz

Source File User Interface.

```

#include "../header/GUI.h"
#include "../header/SegmentApproximator.h"
#include "../header/PathBuilder.h"
#include "../header/RobCodeGenerator.h"
#include "../header/MeanFilter.h"
#include "../header/Logging.h"

```

8.23.1 Ausführliche Beschreibung

Source File User Interface.

Definiert in Datei [GUI.cpp](#).

8.24 GUI.cpp

[gehe zur Dokumentation dieser Datei](#)

```

00001
00007 #include "../header/GUI.h"
00008
00009 #include "../header/SegmentApproximator.h"
00010 #include "../header/PathBuilder.h"
00011 #include "../header/RobCodeGenerator.h"
00012 #include "../header/MeanFilter.h"
00013 #include "../header/Logging.h"
00014
00015 GUI::GUI(QWidget *parent)
00016     : QMainWindow(parent)
00017 {
00018     ui.setupUi(this);
00019
00020     //Dateioperationen und Anzeige
00021     inputPathUI = "";
00022     ui.pathInput->setText(inputPathUI);
00023     outputPathUI = "";
00024     ui.pathOutput->setText(outputPathUI);
00025
00026     connect(ui.pushOutput, &QPushButton::clicked, this, &GUI::setOutputPath);
00027     connect(ui.pushInput, &QPushButton::clicked, this, &GUI::setInputPath);
00028
00029     //Zwingende Einstellwerte
00030     //Douglas-Peucker-Toleranz
00031     ui.dpToleranz->setRange(1, 100);
00032     ui.dpToleranz->setSingleStep(1);
00033     ui.dpToleranz->setValue(40);
00034     connect(ui.dpToleranz, &QSpinBox::valueChanged, this, &GUI::setDP);
00035
00036     //Fenster fuer gleitenden Mittelwert
00037     ui.meanLength->setRange(3, 500);
00038     ui.meanLength->setSingleStep(1);
00039     ui.meanLength->setValue(50);
00040     connect(ui.meanLength, &QSpinBox::valueChanged, this, &GUI::setMean);
00041
00042     //Geschwindigkeit
00043     connect(ui.bSpeed, &QCheckBox::clicked, this, &GUI::activateSpeed);
00044     ui.speed->setRange(0.01, 2);
00045     ui.speed->setSingleStep(0.01);
00046     ui.speed->setValue(1);
00047     connect(ui.speed, &QDoubleSpinBox::valueChanged, this, &GUI::setSpeed);
00048
00049     //Ausrichtung
00050     connect(ui.bManOrientation, &QCheckBox::clicked, this, &GUI::activateOrientation);
00051     ui.AValue->setRange(-180, 180);
00052     ui.AValue->setSingleStep(5);
00053     ui.AValue->setValue(0);
00054     connect(ui.AValue, &QDoubleSpinBox::valueChanged, this, &GUI::setOrientation);
00055
00056     ui.BValue->setRange(-180, 180);
00057     ui.BValue->setSingleStep(5);
00058     ui.BValue->setValue(90);
00059     connect(ui.BValue, &QDoubleSpinBox::valueChanged, this, &GUI::setOrientation);
00060
00061     ui.CValue->setRange(-180, 180);
00062     ui.CValue->setSingleStep(5);
00063     ui.CValue->setValue(0);
00064     connect(ui.CValue, &QDoubleSpinBox::valueChanged, this, &GUI::setOrientation);
00065
00066     //Offset
00067     connect(ui.bOffset, &QCheckBox::clicked, this, &GUI::activateOffset);
00068     ui.offsetX->setRange(-400, 400);
00069     ui.offsetX->setSingleStep(10);
00070     ui.offsetX->setValue(0);
00071     connect(ui.offsetX, &QDoubleSpinBox::valueChanged, this, &GUI::setOffset);
00072
00073     ui.offsetY->setRange(-400, 400);
00074     ui.offsetY->setSingleStep(10);
00075     ui.offsetY->setValue(0);
00076     connect(ui.offsetY, &QDoubleSpinBox::valueChanged, this, &GUI::setOffset);
00077
00078     ui.offsetZ->setRange(-400, 400);
00079     ui.offsetZ->setSingleStep(10);
00080     ui.offsetZ->setValue(0);
00081     connect(ui.offsetZ, &QDoubleSpinBox::valueChanged, this, &GUI::setOffset);
00082
00083     //Logging
00084     ui.bLogging->setChecked(true);
00085     inputParameter.setOffset(0, 0, 0, false);
00086     connect(ui.bLogging, &QCheckBox::clicked, this, &GUI::activateLogging);
00087

```

```

00088     connect(ui.startCalculation, &QPushButton::clicked, this, &GUI::calculate);
00089 }
00090
00091 GUI::~GUI()
00092 {}
00093
00094 void GUI::activateLogging(void)
00095 {
00096     inputParameter.setLogging(ui.bLogging->isChecked());
00097 }
00098
00099 void GUI::setOffset(void)
00100 {
00101     inputParameter.setOffset(ui.offsetX->value(), ui.offsetY->value(), ui.offsetZ->value(),
00102         ui.bOffset->isChecked());
00103 }
00104
00105 void GUI::activateOffset(void)
00106 {
00107     if (ui.bOffset->isChecked())
00108     {
00109         ui.offset->setEnabled(true);
00110         ui.offset->setStyleSheet("background-color: rgb(67, 72, 91); color: rgb(3, 8, 14); border:
00111     lpx solid black;");
00112     }
00113     else
00114     {
00115         ui.offset->setEnabled(false);
00116         ui.offset->setStyleSheet("background-color: rgb(210,211,218); color: rgb(117,125,149)");
00117         ui.offsetX->setValue(0);
00118         ui.offsetY->setValue(0);
00119         ui.offsetZ->setValue(0);
00120         inputParameter.setOffset(0, 0, 0, false);
00121     }
00122 }
00123 void GUI::setOrientation(void)
00124 {
00125     inputParameter.setOrientation(ui.AValue->value(), ui.BValue->value(), ui.CValue->value(),
00126         ui.bManOrientation->isChecked());
00127 }
00128
00129 void GUI::activateOrientation(void)
00130 {
00131     if (ui.bManOrientation->isChecked())
00132     {
00133         ui.orientation->setEnabled(true);
00134         ui.orientation->setStyleSheet("background-color: rgb(67, 72, 91); color: rgb(3, 8, 14); border:
00135     lpx solid black;");
00136     }
00137     else
00138     {
00139         ui.orientation->setEnabled(false);
00140         ui.orientation->setStyleSheet("background-color: rgb(210,211,218); color: rgb(117,125,149)");
00141     }
00142 }
00143 void GUI::setSpeed(void)
00144 {
00145     inputParameter.setSpeed(ui.speed->value(), ui.bSpeed->isChecked());
00146 }
00147
00148 void GUI::activateSpeed(void)
00149 {
00150     if (ui.bSpeed->isChecked())
00151     {
00152         ui.speed_2->setEnabled(true);
00153         ui.speed_2->setStyleSheet("background-color: rgb(67, 72, 91); color: rgb(3, 8, 14); border:
00154     lpx solid black; ");
00155     }
00156     else
00157     {
00158         ui.speed_2->setEnabled(false);
00159         ui.speed_2->setStyleSheet("background-color: rgb(210,211,218); color: rgb(117,125,149)");
00160     }
00161 }
00162 void GUI::setMean(void)
00163 {
00164     meanLength = ui.meanLength->value();
00165 }
00166
00167
00168 void GUI::setDP(void)
00169 {
00170     dpTolerance = ui.dpToleranz->value();
00171 }

```

```

00172
00173 void GUI::setInputPath(void)
00174 {
00175     inputPathUI = QFileDialog::getOpenFileName(this);
00176     ui.pathInput->setText(inputPathUI);
00177 }
00178
00179 void GUI::setOutputPath(void)
00180 {
00181     outputPathUI = QFileDialog::getExistingDirectory(this);
00182     ui.pathOutput->setText(outputPathUI);
00183 }
00184
00185 void GUI::calculate()
00186 {
00187
00188     if (inputPathUI.isEmpty() || outputPathUI.isEmpty())
00189     {
00190         QMessageBox messageBox;
00191         messageBox.critical(0, "Error", "Keine Datei ausgewaehlt!");
00192         messageBox.setFixedSize(500, 200);
00193         return;
00194     }
00195
00196     try
00197     {
00198         string outputPath = outputPathUI.toUtf8().constData();
00199         string inputPath = inputPathUI.toUtf8().constData();
00200         ui.textBrowser->clear();
00201
00202         //logging Initialisieren
00203         CLogging logging(outputPath, inputParameter.getLoggingManual());
00204
00205         //read Data
00206         inputParameter.openFile(inputPath);
00207         ui.textBrowser->insertPlainText("Datei eingelesen\n");
00208
00209         //moving Average
00210
00211         CMeanFilter meanFilter;
00212         meanFilter.setWindowSize(meanLength);
00213         meanFilter.mean(inputParameter.getPath(), logging);
00214         ui.textBrowser->insertPlainText("Gleitender Mittelwert berechnet\n");
00215
00216         // Douglas-Peuker Algorithm
00217
00218         CSegmentApproximator segmentApproximator;
00219         segmentApproximator.setmaxDistance(dpTolerance);
00220         segmentApproximator.approx(meanFilter.getPath(), logging);
00221         ui.textBrowser->insertPlainText("Douglas-Peuker-Algorithmus berechnet\n");
00222
00223         // Puts the Segments together to one path
00224
00225         CPathBuilder pathBuilder;
00226         pathBuilder.createPath(segmentApproximator.getSegmentsApproxVector(), logging);
00227         ui.textBrowser->insertPlainText("Pfad zusammengesetzt\n");
00228
00229         // Calculates Speed, Angle and generates the Output Data
00230
00231         CRobCodeGenerator codeGenerator(inputParameter);
00232         codeGenerator.generateRobCode(pathBuilder.getPath(), outputPath, inputPath, logging);
00233         ui.textBrowser->insertPlainText("Datei erstellt\n");
00234     }
00235
00236     catch (exception& e)
00237     {
00238         cerr << e.what() << "\n";
00239     }
00240 }

```

8.25 source/InputParameter.cpp-Dateireferenz

Source File Daten Einlesen.

```

#include "../header/InputParameter.h"
#include "../header/Point3D.h"
#include "../header/EulerMatrix.h"

```

8.25.1 Ausführliche Beschreibung

Source File Daten Einlesen.

Definiert in Datei [InputParameter.cpp](#).

8.26 InputParameter.cpp

[gehe zur Dokumentation dieser Datei](#)

```

00001
00007 #include "../header/InputParameter.h"
00008 #include "../header/Point3D.h"
00009 #include "../header/EulerMatrix.h"
00010
00011 /* CInputParamameter mir Uebergabewerten initialisieren */
00012 CInputParameter::CInputParameter(double initSpeed, bool initSpeedManual, bool initOrientationManual,
double initA, double initB, double initC)
00013 {
00014     speed = initSpeed;
00015     speedManual = initSpeedManual;
00016     orientationManual = initOrientationManual;
00017     A = initA;
00018     B = initB;
00019     C = initC;
00020 }
00021 }
00022
00023 /* CInputParameter mit 0 initialisieren */
00024 CInputParameter::CInputParameter(void)
00025 {
00026     speed = 0.1;
00027     A = 0;
00028     B = 75;
00029     C = 0;
00030     speedManual = false;
00031     orientationManual = false;
00032 }
00033 }
00034
00035 CInputParameter::~CInputParameter(void)
00036 {
00037 }
00038 }
00039
00040 /* Einstellung fuer Orientierung und Winkel setzten */
00041 void CInputParameter::setOrientation(bool initOrientationManual, double initA, double initB, double
initC)
00042 {
00043     orientationManual = initOrientationManual;
00044     A = initA;
00045     B = initB;
00046     C = initC;
00047 }
00048
00049 void CInputParameter::setOffset(double X, double Y, double Z, bool initOffsetManual)
00050 {
00051     offsetManual = initOffsetManual;
00052     offsetX = X;
00053     offsetY = Y;
00054     offsetZ = Z;
00055 }
00056
00057 void CInputParameter::setLogging(bool initLoggingManual)
00058 {
00059     loggingManual = initLoggingManual;
00060 }
00061
00062 /* Einstellung fuer Geschwindigkeit und Geschwindigkeit setzen */
00063 void CInputParameter::setSpeed(double initSpeed, bool initSpeedManual)
00064 {
00065     speed = initSpeed;
00066     speedManual = initSpeedManual;
00067 }
00068
00069 vector<list<CInputPoint3D>& CInputParameter::getPath()
00070 {
00071     return initialPath;    // Path zurueck geben
00072 }

```

```

00073
00074 double CInputParameter::getSpeed(void)
00075 {
00076     return speed;          // Geschwindigkeit zurueck geben
00077 }
00078
00079 bool CInputParameter::getSpeedManual(void)
00080 {
00081     return speedManual;    // Vorgewaehlte Einstellung fuer Geschwindigkeit zurueck geben
00082 }
00083
00084 bool CInputParameter::getOrientationManual(void)
00085 {
00086     return orientationManual; // Vorgewaehlte Einstellung fuer Orientierung zurueck geben
00087 }
00088
00089 bool CInputParameter::getOffsetManual(void)
00090 {
00091     return offsetManual;    // Vorgewaehlte Einstellung fuer den Offset zurueck
00092 }
00093
00094 bool CInputParameter::getLoggingManual(void)
00095 {
00096     return loggingManual;   // Vorgewaehlte Einstellung fuer das Logging zurueck
00097 }
00098
00099 tuple <double, double, double> CInputParameter::getAngles(void)
00100 {
00101     return make_tuple(A, B, C); // Winkel zurueck geben
00102 }
00103
00104 tuple <double, double, double> CInputParameter::getOffset(void)
00105 {
00106     return make_tuple(offsetX, offsetY, offsetZ); // Offset zurueck geben
00107 }
00108
00109
00110 /* Eingabedatei oeffnen */
00111 void CInputParameter::openFile(string path)
00112 {
00113     ifstream fin(path);
00114     CInputPoint3D tmpPoint; // Zwischenspeicher zum konvertieren von tmpEuler in Point3D
00115     CEulerMatrix tmpEuler; // Zwischenspeicher zum konvertieren von DummyMatrix in EulerMatrix
00116     double x, y, z; // Punktkoordinaten
00117     double x_prev = 0, y_prev = 0, z_prev = 0; // Zwischenspeicher fuer Punktkoordinaten
00118     double timestamp; // Zeitstempel
00119     int segmentCount = -1; // Segmentzaehler
00120     float dummyMatrix[3][3]; // DummyMatrix zum speichern
00121
00122     if (!fin.is_open())
00123     {
00124         cerr << "Datei konnte nicht geoeffnet werden" << endl; // Fehler Datei konnte nicht
geoeffnet werden.
00125     }
00126     string line;
00127     while(getline(fin, line))
00128     {
00129         std::istringstream sStream (line);
00130         sStream >> timestamp >> x >> y >> z >> dummyMatrix[0][0] >> dummyMatrix[0][1] >> dummyMatrix[0][2]
// Zeile in die einzelnen Parameter zerlegen
00131         >> dummyMatrix[1][0] >> dummyMatrix[1][1] >> dummyMatrix[1][2] >> dummyMatrix[2][0] >>
dummyMatrix[2][1] >> dummyMatrix[2][2]; // und in DummyMatrix bzw. Variablen abspeichern
00132
00133         tmpEuler.setMatrix(dummyMatrix); // DummyMatrix[3][3] in
EulerMatrix speichern
00134         tmpPoint.setPoint(timestamp, x, y, z, tmpEuler.getEulerMatrix()); // Variablen und EulerWinkel
in CPoint3D speichern
00135
00136         if (detectJump(tmpPoint, x_prev, y_prev, z_prev)) // if there is a jump in the data, start da
new segment
00137         {
00138             segmentCount++; // neues Segment anlegen
00139             initialPath.push_back(list<CInputPoint3D>()); // Punkt in Segment speichern
00140         }
00141
00142         initialPath[segmentCount].push_back(tmpPoint); // Punkt in bestehendes Segment
abspeichern
00143
00144         x_prev = x; // X-Wert zwischenspeichern
00145         y_prev = y; // Y-Wert zwischenspeichern
00146         z_prev = z; // Z-Wert zwischenspeichern
00147     }
00148     fin.close(); // Datei schliessen
00149 }
00150
00151
00152

```

```

00153 bool CInputParameter::detectJump(CInputPoint3D p, double x_prev, double y_prev, double z_prev)
00154 {
00155     if(abs(p.getX() - x_prev) > difference) // Abstand zwischen Punkten groesser max
00156         Differenz??
00157         return true;
00157     else if(abs(p.getY() - y_prev) > difference) // Abstand zwischen Punkten groesser max
00158         Differenz??
00158         return true;
00159     else if(abs(p.getZ() - z_prev) > difference) // Abstand zwischen Punkten groesser max
00160         Differenz??
00160         return true;
00161     else
00162         return false;
00163 }

```

8.27 source/Line3D.cpp-Dateireferenz

Source File Line3D.

```

#include "../header/Line3D.h"
#include "../header/Point3D.h"

```

8.27.1 Ausführliche Beschreibung

Source File Line3D.

Definiert in Datei [Line3D.cpp](#).

8.28 Line3D.cpp

[gehe zur Dokumentation dieser Datei](#)

```

00001
00007 #include "../header/Line3D.h"
00008 #include "../header/Point3D.h"
00009
00010 CLine3D::CLine3D(void)
00011 {
00012 }
00013
00014 /* initialisieren mit 2 Punkten*/
00015 CLine3D::CLine3D(CPoint3D P1, CPoint3D P2)
00016 {
00017     p1 = P1;
00018     p2 = P2;
00019 }
00020
00021 CLine3D::~CLine3D(void)
00022 {
00023 }

```

8.29 source/Logging.cpp-Dateireferenz

Source File Logging.

```

#include "../header/Logging.h"

```

8.29.1 Ausführliche Beschreibung

Source File Logging.

Definiert in Datei [Logging.cpp](#).

8.30 Logging.cpp

[gehe zur Dokumentation dieser Datei](#)

```

00001
00007 #include "../header/Logging.h"
00008
00009 /* Step mit 0 initialisieren */
00010 CLogging::CLogging(void)
00011 {
00012     step = 0;
00013 }
00014
00015 /* Path mit Parameter initialisieren*/
00016 CLogging::CLogging(string Path, bool Detailed)
00017 {
00018     path = Path;
00019     detailed = Detailed;
00020 }
00021
00022 CLogging::~CLogging(void)
00023 {
00024
00025 }
00026
00027 bool CLogging::getDetailed(void)
00028 {
00029     return detailed;
00030 }
00031
00032 void CLogging::setStep(int Step)
00033 {
00034     step = Step;    // Step setzen
00035 }
00036
00037 void CLogging::logData(vector<list<CInputPoint3D>& sourcePath)
00038 {
00039     string filepath;           // file Pfad
00040     float dummyMatrix[3][3];  // dummyMatrix zum Zwischenspeichern
00041     CEulerMatrix tmpEuler;     // CEulerMatrix zum Zwischenspeichern
00042
00043     filepath = path + "/" + "0" + std::to_string(step) + "_log.csv";
00044
00045     FILE* fid = fopen(filepath.c_str(), "w");    // file oeffnen
00046
00047     if (fid == NULL)
00048     {
00049         cerr << "ERROR - Can NOT write to output file!\n";    // Fehler beim file oeffnen
00050         return;
00051     }
00052
00053     for (size_t s = 0; s < sourcePath.size(); s++) //for all segments
00054     {
00055         list<CInputPoint3D>::iterator itr = sourcePath[s].begin();
00056
00057         tmpEuler = itr->getEulerMatrix();
00058         tmpEuler.getMatrix(dummyMatrix);
00059
00060         /* Ausgeben der Punkte mit dummyMatrix */
00061         for (; itr != sourcePath[s].end(); itr++) //for all points in the segment
00062         {
00063             fprintf(fid, "%f %f %f %f %f %f %f %f %f %f %f %f\n", (double)itr->getTime(),
00064 (double)itr->getX(), (double)itr->getY(), (double)itr->getZ(),
00065                 dummyMatrix[0][0], dummyMatrix[0][1], dummyMatrix[0][2],
00066                 dummyMatrix[1][0], dummyMatrix[1][1], dummyMatrix[1][2],
00067                 dummyMatrix[2][0], dummyMatrix[2][1], dummyMatrix[2][2]);
00068         }
00069         itr--;
00070     }
00071 }
00072
00073 void CLogging::logData(vector<CInputPoint3D>& sourcePath)

```



```

00074 {
00075     string filepath;           // file Pfad
00076     float dummyMatrix[3][3];  // dummyMatrix zum Zwischenspeichern
00077     CEulerMatrix tmpEuler;    // CEulerMatrix zum Zwischenspeichern
00078
00079     filepath = path + "/" + "0" + std::to_string(step) + "_log.csv";
00080
00081     FILE* fid = fopen(filepath.c_str(), "w"); // file oeffnen
00082
00083     if (fid == NULL)
00084     {
00085         cerr << "ERROR - Can NOT write to output file!\n"; // Fehler beim file oeffnen
00086         return;
00087     }
00088
00089     /* Ausgeben der Punkte mit dummyMatrix */
00090     for (size_t s = 0; s < sourcePath.size(); s++) //for all points in the vector
00091     {
00092         tmpEuler.getMatrix(dummyMatrix);
00093
00094         fprintf(fid, "%f %f %f %f %f %f %f %f %f %f %f %f\n", (double)sourcePath[s].getTime(),
00095             (double)sourcePath[s].getX(), (double)sourcePath[s].getY(), (double)sourcePath[s].getZ(),
00096             dummyMatrix[0][0], dummyMatrix[0][1], dummyMatrix[0][2],
00097             dummyMatrix[1][0], dummyMatrix[1][1], dummyMatrix[1][2],
00098             dummyMatrix[2][0], dummyMatrix[2][1], dummyMatrix[2][2]);
00099     }
00100 }
00101
00102 void CLogging::logData(vector<COutputPoint3D>& sourcePath)
00103 {
00104     string filepath;           // file Pfad
00105     float dummyMatrix[3][3];  // dummyMatrix zum Zwischenspeichern
00106     CEulerMatrix tmpEuler;    // CEulerMatrix zum Zwischenspeichern
00107
00108     filepath = path + "/" + "0" + std::to_string(step) + "_path.csv";
00109
00110     FILE* fid = fopen(filepath.c_str(), "w"); // file oeffnen
00111
00112     if (fid == NULL)
00113     {
00114         cerr << "ERROR - Can NOT write to output file!\n"; // Fehler beim file oeffnen
00115         return;
00116     }
00117
00118     /* Ausgeben der Punkte mit dummyMatrix */
00119     for (size_t s = 0; s < sourcePath.size(); s++) //for all points in the vector
00120     {
00121         tmpEuler.getMatrix(dummyMatrix);
00122
00123         fprintf(fid, "%f %f %f %f %f %f %f %f %f %f %f %f\n", (double)sourcePath[s].getSpeed(),
00124             (double)sourcePath[s].getX(), (double)sourcePath[s].getY(), (double)sourcePath[s].getZ(),
00125             (double)sourcePath[s].getA(), (double)sourcePath[s].getB(), (double)sourcePath[s].getC());
00126     }
00127 }

```

8.31 source/MeanFilter.cpp-Dateireferenz

Source File gleitender Mittelwertfilter.

```

#include "../header/MeanFilter.h"
#include "../header/Logging.h"
#include <math.h>

```

8.31.1 Ausführliche Beschreibung

Source File gleitender Mittelwertfilter.

Definiert in Datei [MeanFilter.cpp](#).

8.32 MeanFilter.cpp

[gehe zur Dokumentation dieser Datei](#)

```

00001
00007 #include "../header/MeanFilter.h"
00008 #include "../header/Logging.h"
00009 #include <math.h>
00010
00011 CMeanFilter::CMeanFilter()
00012 {
00013     windowSize = 3;           // initialisieren mit Standardfenstergroesse 3
00014 }
00015
00016 CMeanFilter::CMeanFilter(int Window)
00017 {
00018     windowSize = Window; // initialisieren der Fenstergroesse mit Uebergabewert
00019 }
00020
00021 CMeanFilter::~CMeanFilter()
00022 {
00023 }
00024
00025 void CMeanFilter::setWindowSize(int Window)
00026 {
00027     windowSize = Window; // setzen der Fenstergroesse mit Uebergabewert
00028 }
00029
00030 int CMeanFilter::getWindowSize()
00031 {
00032     return windowSize; // Fenstergroesse zurueck geben
00033 }
00034
00035 vector<list<CInputPoint3D>& CMeanFilter::getPath()
00036 {
00037     return meanPath; // Mittelwert zurueck geben
00038 }
00039
00040 void CMeanFilter::mean(vector<list<CInputPoint3D>& sourcePath, CLogging log)
00041 {
00042     list<CInputPoint3D> dummyList;
00043     for (size_t s = 0; s < sourcePath.size(); s++)
00044     {
00045         dummyList = calculateMean(sourcePath[s]);
00046         meanPath.push_back(dummyList);
00047     }
00048     if (log.getDetailed())
00049     {
00050         log.setStep(1);
00051         log.logData(meanPath);
00052     }
00053 }
00054
00055 list<CInputPoint3D> CMeanFilter::calculateMean(list<CInputPoint3D>& segment)
00056 {
00057     double sumX = 0, sumY = 0, sumZ = 0; // Variablen zum Speichern der Summe
00058     double div = 0; // Variable zum Speichern des Teilers
00059
00060     CInputPoint3D p; //Point3D zum Zwischenspeichern
00061
00062     size_t inputSize = segment.size();
00063
00064     list<CInputPoint3D>::iterator it = segment.begin();
00065     list<CInputPoint3D> newSegment;
00066
00067     for (size_t i = 0; i < inputSize - windowSize; ++i) //For each element in the Segment
00068     {
00069         sumX = 0, sumY = 0, sumZ = 0; // Variablen zum Speichern der Summe auf 0 zurueck setzen
00070         div = 0; // Variable zum Speichern des Teilers auf 0 zurueck setzen
00071         p.setTime(it->getTime());
00072         p.setEulerMatrix(it->getEulerMatrix());
00073         for (size_t j = i; j < i + windowSize; ++j) // Build the sums for the three points
00074         {
00075             sumX += it->getX();
00076             sumY += it->getY();
00077             sumZ += it->getZ();
00078             div++;
00079             it++;
00080         }
00081         for (size_t index = windowSize; index > 0; index--) // Pain, the iterator has to be set back
00082         {
00083             it--;
00084         }
00085         p.set(sumX / div, sumY / div, sumZ / div); // Calculate smoothed values
00086         if(it != segment.end())
00087             it++;

```

```

00088         newSegment.push_back(p);
00089     }
00090     return newSegment;
00091 }

```

8.33 source/PathBuilder.cpp-Dateireferenz

Source File Segmente zu Pfad.

```
#include "../header/PathBuilder.h"
```

8.33.1 Ausführliche Beschreibung

Source File Segmente zu Pfad.

Definiert in Datei [PathBuilder.cpp](#).

8.34 PathBuilder.cpp

[gehe zur Dokumentation dieser Datei](#)

```

00001
00007 #include "../header/PathBuilder.h"
00008
00009 CPathBuilder::CPathBuilder(void)
00010 {
00011 }
00012
00013
00014 CPathBuilder::~CPathBuilder(void)
00015 {
00016 }
00017
00018 vector<CInputPoint3D>& CPathBuilder::getPath()
00019 {
00020     return path;
00021 }
00022
00023 void CPathBuilder::createPath(vector<list<CInputPoint3D>& segments, CLogging log)
00024 {
00025     CInputPoint3D point; //startpoint
00026
00027     for (size_t s = 0; s < segments.size(); s++) //for all segments
00028     {
00029         list<CInputPoint3D>::iterator itr = segments[s].begin();
00030
00031         for (; itr != segments[s].end(); itr++) //for all points in the segment
00032         {
00033             point.set((double)itr->getX(), (double)itr->getY(), (double)itr->getZ());
00034             point.setTime(itr->getTime());
00035             point.setEulerMatrix(itr->getEulerMatrix());
00036             path.push_back(point);
00037         }
00038
00039         itr--;
00040     }
00041     if (log.getDetailed())
00042     {
00043         log.setStep(3);
00044         log.logData(path);
00045     }
00046 }

```

8.35 source/Point3D.cpp-Dateireferenz

Source File Punkte.

```
#include "../header/Point3D.h"
#include "../header/Line3D.h"
#include <math.h>
```

8.35.1 Ausführliche Beschreibung

Source File Punkte.

Definiert in Datei [Point3D.cpp](#).

8.36 Point3D.cpp

[gehe zur Dokumentation dieser Datei](#)

```
00001
00007 #include "../header/Point3D.h"
00008 #include "../header/Line3D.h"
00009 #include <math.h>
00010
00011
00012 /* initialisieren des Punktes mit 0-Werten */
00013 CPoint3D::CPoint3D(void)
00014 {
00015     x = 0;
00016     y = 0;
00017     z = 0;
00018 }
00019
00020 /* initialisieren des Punktes mit Koordinaten */
00021 CPoint3D::CPoint3D(double X, double Y, double Z)
00022 {
00023     x = X;
00024     y = Y;
00025     z = Z;
00026 }
00027
00028 CPoint3D::~CPoint3D(void)
00029 {
00030 }
00031
00032 double CPoint3D::getX(void)
00033 {
00034     return x; // X-Koordinate zurueck geben
00035 }
00036
00037 double CPoint3D::getY(void)
00038 {
00039     return y; // Y-Koordinate zurueck geben
00040 }
00041
00042 double CPoint3D::getZ(void)
00043 {
00044     return z; // Z-Koordinate zurueck geben
00045 }
00046
00047 void CPoint3D::setX(double X)
00048 {
00049     x = X; // X-Koordinate setzen
00050 }
00051
00052 void CPoint3D::setY(double Y)
00053 {
00054     y = Y; // Y-Koordinate setzen
00055 }
00056
00057 void CPoint3D::setZ(double Z)
00058 {
```

```

00059     z = Z; // Z-Koordinate setzen
00060 }
00061
00062 /* X-, Y- und Z-Koordinate setzen */
00063 void CPoint3D::set(double X, double Y, double Z)
00064 {
00065     x = X; // X-Koordinate setzen
00066     y = Y; // Y-Koordinate setzen
00067     z = Z; // Z-Koordinate setzen
00068 }
00069
00070 /* Distanz zwischen Punkt und uebergebenen Punkt berechnen */
00071 double CPoint3D::distanceTo(CPoint3D point)
00072 {
00073     return sqrt(pow((double)(x - (double)point.getX()), 2) + pow((double)(y - (double)point.getY()),
00074 2) + pow((double)(z - (double)point.getZ()), 2)); // Pythagoras 3D
00075 }
00076 double CPoint3D::distanceTo(CLine3D line)
00077 {
00078     double bx, by, bz, rv_sq, dist, vp1, vp2, vp3; // Variablen Anlegen
00079
00080     /*
00081     Vermessen wird der Punkt selbst
00082
00083     bx, by, bz == Vektordifferenz
00084     rv_sq == Betrag des Linienvektors
00085     dist == Distanz von Punkt zu Linie
00086     vp1, vp2, vp3 == Vektorprodukte
00087     */
00088
00089     int rvx = line.p1.x - line.p2.x; // Parameter X des Linienvektor berechnen
00090     int rvy = line.p1.y - line.p2.y; // Parameter Y des Linienvektor berechnen
00091     int rvz = line.p1.z - line.p2.z; // Parameter Z des Linienvektor berechnen
00092
00093     rv_sq = sqrt(((double)rvx * (double)rvx) + ((double)rvy * (double)rvy) + ((double)rvz *
00094 (double)rvz)); // Betrag des Linienvektor berechnen
00095
00096     bx = x - (double)line.p1.x; // X(Punkt) - X(Aufpunkt)
00097     by = y - (double)line.p1.y; // Y(Punkt) - Y(Aufpunkt)
00098     bz = z - (double)line.p1.z; // Z(Punkt) - Z(Aufpunkt)
00099
00100     vp1 = by * rvz - bz * rvy; // Parameter X Vektorprodukt
00101     vp2 = bz * rvx - bx * rvz; // Parameter Y Vektorprodukt
00102     vp3 = bx * rvy - by * rvx; // Parameter Z Vektorprodukt
00103
00104     dist = sqrt(vp1 * vp1 + vp2 * vp2 + vp3 * vp3) / rv_sq; // Betrag des Vektors berechnen
00105     return dist;
00106 }
00107
00108 // InputPoint3D
00109
00110 CInputPoint3D::CInputPoint3D(void) : CPoint3D()
00111 {
00112     timestamp = 0; // Zeitstempel mit 0 initialisieren
00113 }
00114
00115 /* Initialisieren des Punktes mit Parameter */
00116 CInputPoint3D::CInputPoint3D(double X, double Y, double Z, double Timestamp, CEulerMatrix Matrix)
00117 {
00118     x = X;
00119     y = Y;
00120     z = Z;
00121     timestamp = Timestamp;
00122     orientationMatrix = Matrix;
00123 }
00124
00125 CInputPoint3D::~CInputPoint3D(void)
00126 {
00127 }
00128
00129 void CInputPoint3D::setEulerMatrix(CEulerMatrix orientation)
00130 {
00131     orientationMatrix = orientation; // EulerMatrix setzen
00132 }
00133
00134
00135 void CInputPoint3D::setPoint(double time, double X, double Y, double Z, CEulerMatrix orientation)
00136 {
00137     setTime(time); // Zeitstempel setzen
00138     set(X, Y, Z); // setze Punkt-Koordinaten
00139     setEulerMatrix(orientation); // EulerMatrix setzen
00140 }
00141
00142 void CInputPoint3D::setTime(double time)
00143 {

```

```
00144     timestamp = time; // Zeitstempel setzen
00145 }
00146
00147 CEulerMatrix CInputPoint3D::getEulerMatrix()
00148 {
00149     return orientationMatrix; // Rueckgabe der EulerMatrix
00150 }
00151
00152 double CInputPoint3D::getTime()
00153 {
00154     return timestamp; // Rueckgabe des Zeitstempel
00155 }
00156
00157 // OutputPoint3D
00158 /* Punkt mit 0 initialisieren */
00159 COutputPoint3D::COutputPoint3D(void) : CPoint3D()
00160 {
00161     speed = 0;
00162     a = 0;
00163     b = 0;
00164     c = 0;
00165 }
00166
00167 /* Punkt mit Parameter initialisieren*/
00168 COutputPoint3D::COutputPoint3D(double Speed, double X, double Y, double Z, double A, double B, double
C)
00169 {
00170     speed = Speed;
00171     a = A;
00172     b = B;
00173     c = C;
00174     x = X;
00175     y = Y;
00176     z = Z;
00177 }
00178
00179 COutputPoint3D::~COutputPoint3D(void)
00180 {
00181 }
00182 }
00183
00184 double COutputPoint3D::getA(void)
00185 {
00186     return a; // Rueckgabe Winkel alpha
00187 }
00188
00189 double COutputPoint3D::getB(void)
00190 {
00191     return b; // Rueckgabe Winkel beta
00192 }
00193
00194 double COutputPoint3D::getC(void)
00195 {
00196     return c; // Rueckgabe Winkel gamma
00197 }
00198
00199 double COutputPoint3D::getSpeed(void)
00200 {
00201     return speed; // Rueckgabe Geschwindigkeit
00202 }
00203
00204 void COutputPoint3D::setA(double A)
00205 {
00206     a = A; // setze Winkel alpha
00207 }
00208
00209 void COutputPoint3D::setB(double B)
00210 {
00211     b = B; // setze Winkel beta
00212 }
00213
00214 void COutputPoint3D::setC(double C)
00215 {
00216     c = C; // setze Winkel gamma
00217 }
00218
00219 void COutputPoint3D::setSpeed(double Speed)
00220 {
00221     speed = Speed; // setze Geschwindigkeit
00222 }
```

8.37 source/RobCodeGenerator.cpp-Dateireferenz

Source File Roboter Code Erstellung.

```
#include "../header/RobCodeGenerator.h"
```

8.37.1 Ausführliche Beschreibung

Source File Roboter Code Erstellung.

Definiert in Datei [RobCodeGenerator.cpp](#).

8.38 RobCodeGenerator.cpp

[gehe zur Dokumentation dieser Datei](#)

```
00001
00007 #include "../header/RobCodeGenerator.h"
00008
00009 /* CRobCodeGenerator mit 0 initialisieren */
00010 CRobCodeGenerator::CRobCodeGenerator(void)
00011 {
00012 }
00013
00014 /* CRobCodeGenerator mit Uebergabewerten initialisieren */
00015 CRobCodeGenerator::CRobCodeGenerator(CInputParameter inputParam)
00016 {
00017     input = inputParam;
00018 }
00019
00020 CRobCodeGenerator::~CRobCodeGenerator(void)
00021 {
00022 }
00023
00024 void CRobCodeGenerator::generateRobCode(vector<CInputPoint3D>& points, string filepath, string
inputPath, CLogging log)
00025 {
00026     string filename;
00027
00028     postProcessing(points); // Calculates all the necessary values
00029
00030     log.setStep(4);
00031     log.logData(processedPath);
00032
00033     errno_t err;
00034
00035     FILE* fid;
00036
00037     filename = inputPath.substr(inputPath.find_last_of("\\") + 1);
00038     filename.erase(filename.end() - 4, filename.end());
00039     filename = filename + ".src";
00040
00041     string fullPath = filepath + "/" + filename;
00042
00043     if ((err = fopen_s(&fid, fullPath.c_str(), "w")) != 0) // Errorhandling for File opening
00044     {
00045         string msg = "Open file: ";
00046         msg += filename;
00047         msg += " failed!";
00048
00049         throw exception(msg.c_str());
00050     }
00051
00052     filename.erase(filename.end()-4,filename.end()); // loescht .src
00053     fprintf(fid, "DEF %s \n", filename.c_str()); // DEF in file schreiben
00054
00055     fputs("PTP $POS_ACT\n", fid); // PTP zur aktuellen Position in file
schreiben
00056
00057     if (input.getSpeedManual()) // If the speed is set to manual, it will be defined once at the
beginning of the file
00058     {
```

```

00059         fprintf(fid, "$VEL.CP = %f\n", input.getSpeed()); // Geschwindigkeit ein file schreiben
00060     }
00061
00062     tuple <double, double, double> offset = input.getOffset();
00063
00064     for (size_t s = 0; s < points.size(); s++)
00065     {
00066         if (!input.getSpeedManual()) // If the speed is calculated it needs to be before every LIN
command
00067             fprintf(fid, "$VEL.CP = %f\n", (float)processedPath[s].getSpeed());
00068             fprintf(fid, "LIN {X %f, Y %f, Z %f, A %f, B %f, C %f}\n", processedPath[s].getX() +
get<0>(offset), processedPath[s].getY() + get<1>(offset),
00069                 processedPath[s].getZ() + get<2>(offset), processedPath[s].getA(),
processedPath[s].getB(),
00070                 processedPath[s].getC());
00071     }
00072
00073     fputs("END", fid);
00074 }
00075
00076 void CRobCodeGenerator::postProcessing(vector<CInputPoint3D>& path)
00077 {
00078     COutputPoint3D p;
00079     double timePrev = 1;
00080
00081     for (size_t s = 0; s < path.size(); s++) // Fuer jeden Punkt in dem Vector
00082     {
00083         p.set(path[s].getX(), path[s].getY(), path[s].getZ());
00084         if (input.getSpeedManual())
00085         {
00086             if (input.getSpeed() > MAX_SPEED) //Wenn maximale Geschwindigkeit ueberschritten wird,
Geschwindigkeit begrenzen
00087                 input.setSpeed(MAX_SPEED, true);
00088         }
00089         else
00090         {
00091             if (s == 0)
00092                 p.setSpeed(1); //Der erste Punkt(0) wird mit Standardgeschwindigkeit 1m/s angefahren.
00093
00094             else
00095                 p.setSpeed(calculateSpeed(path[s], s, timePrev)); //Die Geschwindigkeit zwischen den
weiteren Punkten wird berechnet.
00096         }
00097
00098         if (input.getOrientationManual()) // Wenn der Winkel vorgegeben ist diesen setzten
00099         {
00100             tuple <double, double, double> angles;
00101             angles = input.getAngles();
00102             p.setA(get<0>(angles));
00103             p.setB(get<1>(angles));
00104             p.setC(get<2>(angles));
00105         }
00106         else // Sonst den Winkel berechnen
00107             calculateAngles(p, path[s]);
00108         timePrev = path[s].getTime();
00109         processedPath.push_back(p);
00110     }
00111 }
00112 }
00113
00114 double CRobCodeGenerator::calculateSpeed(CInputPoint3D& p, size_t s, double timePrev)
00115 {
00116     double distance = 0;
00117     double time = 0;
00118     double speed;
00119
00120     distance = processedPath[s - 1].distanceTo(p); //Strecke zwischen p und dem Punkt zuvor
00121     time = p.getTime() - timePrev; //Zeit zwischen p-1 und p
00122
00123     speed = distance / time; // Berechnug Geschwindigkeit zwischen zwei Punkten
00124
00125     if (speed > MAX_SPEED) //Begrenzung auf maximale Geschwindigkeit, falls Trackerdaten h•heren Wert
aufweisen
00126         speed = MAX_SPEED;
00127
00128     return speed; //Zuweisung der Geschwindigkeit
00129 }
00130
00131 void CRobCodeGenerator::calculateAngles(COutputPoint3D& p, CInputPoint3D& pIn)
00132 {
00133     // Funktion in Eulermatrix aufrufen die a/b/c neu berechnet
00134
00135     CEulerMatrix matrix = pIn.getEulerMatrix();
00136     tuple<double, double, double> abc;
00137
00138     abc = matrix.calculateAngels();
00139 }

```



```

00140     p.setA(get<0>(abc));
00141     p.setB(get<1>(abc));
00142     p.setC(get<2>(abc));
00143 }

```

8.39 source/RobPathEditor.cpp-Dateireferenz

Hier wird die main Funktion aufgerufen.

```

#include <iostream>
#include <ctime>
#include "../header/GUI.h"
#include <QtWidgets/QApplication>

```

Funktionen

- int [main](#) (int argc, char *argv[])

8.39.1 Ausführliche Beschreibung

Hier wird die main Funktion aufgerufen.

Definiert in Datei [RobPathEditor.cpp](#).

8.39.2 Dokumentation der Funktionen

8.39.2.1 main()

```

int main (
    int argc,
    char * argv[] )

```

Definiert in Zeile [52](#) der Datei [RobPathEditor.cpp](#).

```

00053 {
00054     QApplication a(argc, argv);
00055     GUI w;
00056     w.show();
00057     return a.exec();
00058 }

```

8.40 RobPathEditor.cpp

[gehe zur Dokumentation dieser Datei](#)

```

00001
00044 #include <iostream>
00045 #include <ctime>
00046
00047 #include "../header/GUI.h"
00048 #include <QtWidgets/QApplication>
00049
00050 using namespace std;
00051
00052 int main(int argc, char* argv[])
00053 {
00054     QApplication a(argc, argv);
00055     GUI w;
00056     w.show();
00057     return a.exec();
00058 }

```

8.41 source/SegmentApproximator.cpp-Dateireferenz

Source File Douglas-Peucker.

```
#include "../header/SegmentApproximator.h"
#include "../header/Point3D.h"
#include "../header/Line3D.h"
```

8.41.1 Ausführliche Beschreibung

Source File Douglas-Peucker.

Definiert in Datei [SegmentApproximator.cpp](#).

8.42 SegmentApproximator.cpp

[gehe zur Dokumentation dieser Datei](#)

```
00001
00007 #include "../header/SegmentApproximator.h"
00008 #include "../header/Point3D.h"
00009 #include "../header/Line3D.h"
00010
00011 CSegmentApproximator::CSegmentApproximator(void)
00012 {
00013 }
00014
00015 CSegmentApproximator::~CSegmentApproximator(void)
00016 {
00017 }
00018
00019 void CSegmentApproximator::approx(const vector<list<CInputPoint3D>& segments, CLogging log)
00020 {
00021     CInputPoint3D p;
00022
00023     segmentsApprox = segments;
00024
00025     /* Douglas Peucker fuer Segmente aufrufen*/
00026     for (size_t s = 0; s < segments.size(); s++)
00027     {
00028         douglasPeuckerRecursive(segmentsApprox[s], segmentsApprox[s].begin(),
--(segmentsApprox[s].end()));
00029     }
00030
00031     /* Logging der Daten*/
00032     if (log.getDetailed())
00033     {
00034         log.setStep(2);
00035         log.logData(segmentsApprox);
00036     }
00037 }
00038
00039 void CSegmentApproximator::setMaxDistance(double maxDistanceSource)
00040 {
00041     maxDistance = maxDistanceSource;    // setze maxDistance
00042 }
00043
00044 double CSegmentApproximator::getmaxDistance()
00045 {
00046     return maxDistance;    // Rueckgabe von maxDistance
00047 }
00048
00049 vector<list<CInputPoint3D>& CSegmentApproximator::getSegmentsApproxVector()
00050 {
00051     return segmentsApprox;    // Rueckgabe der Segmente
00052 }
00053
00054 void CSegmentApproximator::douglasPeuckerRecursive(list<CInputPoint3D>& segment,
std::list<CInputPoint3D>::iterator startItr, std::list<CInputPoint3D>::iterator endItr)
00055 {
00056     if (segment.size() < 3) return; // min Groesse pro Seg 3
```

```

00057     if (distance(startItr, endItr) == 2) return;           // Zeigerabstand == 2
00058     CInputPoint3D pStart; CInputPoint3D pEnd;           // Variablen deklarieren
00059
00060
00061     /* Startpunkt setzen */
00062     pStart.setX(startItr->getX()); pStart.setY(startItr->getY()); pStart.setZ(startItr->getZ());
00063     pStart.setTime(startItr->getTime());
00064     pStart.setEulerMatrix(startItr->getEulerMatrix());
00065
00066     /* Endpunkt setzen */
00067     pEnd.setX(endItr->getX()); pEnd.setY(endItr->getY()); pEnd.setZ(endItr->getZ());
00068     pEnd.setTime(endItr->getTime());
00069     pEnd.setEulerMatrix(endItr->getEulerMatrix());
00070
00071     double dist = 0.0, maxDist = 0.0;                   // dist und maxDist initialisieren
00072     std::list<CInputPoint3D>::iterator maxItr, itr;       // Zeiger bilden
00073
00074
00075     /* am weitesten Entfernten Punkt suchen */
00076     for (itr = startItr; itr != endItr; itr++)
00077     {
00078         CLine3D line = CLine3D(pStart, pEnd);
00079         // calc distance
00080         dist = itr->distanceTo(line);
00081         if (dist > maxDist) {
00082             maxDist = dist;
00083             maxItr = itr;
00084         }
00085     }
00086
00087     if (maxDist <= maxDistance) {
00088
00089         segment.erase(++startItr, endItr);             // Punkt loeschen
00090         return;
00091     }
00092
00093     /* Douglas Peucker erneut aufrufen */
00094     douglasPeuckerRecursive(segment, startItr, maxItr);
00095     douglasPeuckerRecursive(segment, maxItr, endItr);
00096 }

```

8.43 x64/Debug/moc/moc_GUI.cpp-Dateireferenz

```

#include ".../.../header/GUI.h"
#include <QtGui/qttextcursor.h>
#include <QtCore/qmetatype.h>
#include <memory>

```

Klassen

- struct [QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t](#)

Namensbereiche

- namespace [QT_WARNING_DISABLE_DEPRECATED](#)

Makrodefinitionen

- #define [Q_CONSTINIT](#)
- #define [QT_MOC_LITERAL](#)(ofs, len) uint(sizeof(qt_meta_stringdata_CLASSGUIENDCLASS_t::offsets↵ AndSizes) + ofs), len

8.43.1 Makro-Dokumentation

8.43.1.1 Q_CONSTINIT

```
#define Q_CONSTINIT
```

Definiert in Zeile 31 der Datei `moc_GUI.cpp`.

8.43.1.2 QT_MOC_LITERAL

```
#define QT_MOC_LITERAL(  
    ofs,  
    len )    uint(sizeof(qt_meta_stringdata_CLASSGUIENDCLASS_t::offsetsAndSizes) +  
ofs), len
```

Definiert in Zeile 75 der Datei `moc_GUI.cpp`.

8.44 moc_GUI.cpp

[gehe zur Dokumentation dieser Datei](#)

```
00001 /*****  
00002 ** Meta object code from reading C++ file 'GUI.h'  
00003 **  
00004 ** Created by: The Qt Meta Object Compiler version 68 (Qt 6.5.1)  
00005 **  
00006 ** WARNING! All changes made in this file will be lost!  
00007 *****/  
00008  
00009 #include "../header/GUI.h"  
00010 #include <QtGui/qttextcursor.h>  
00011 #include <QtCore/qmetatype.h>  
00012  
00013 #if __has_include(<QtCore/qtmochelpers.h>)  
00014 #include <QtCore/qtmochelpers.h>  
00015 #else  
00016 QT_BEGIN_MOC_NAMESPACE  
00017 #endif  
00018  
00019  
00020 #include <memory>  
00021  
00022 #if !defined(Q_MOC_OUTPUT_REVISION)  
00023 #error "The header file 'GUI.h' doesn't include <QObject>."  
00024 #elif Q_MOC_OUTPUT_REVISION != 68  
00025 #error "This file was generated using the moc from 6.5.1. It"  
00026 #error "cannot be used with the include files from this version of Qt."  
00027 #error "(The moc has changed too much.)"  
00028 #endif  
00029  
00030 #ifndef Q_CONSTINIT  
00031 #define Q_CONSTINIT  
00032 #endif  
00033  
00034 QT_WARNING_PUSH  
00035 QT_WARNING_DISABLE_DEPRECATED  
00036 QT_WARNING_DISABLE_GCC("-Wuseless-cast")  
00037 namespace {  
00038  
00039 #ifdef QT_MOC_HAS_STRINGDATA  
00040 struct qt_meta_stringdata_CLASSGUIENDCLASS_t {};  
00041 static constexpr auto qt_meta_stringdata_CLASSGUIENDCLASS = QtMocHelpers::stringData(  
00042     "GUI",  
00043     "calculate",  
00044     "",  
00045     "setInputPath",  
00046     "setOutputPath",  
00047     "setDP",  
00048     "setMean",  
00049     "activateSpeed",
```

```

00050     "setSpeed",
00051     "activateOrientation",
00052     "setOrientation",
00053     "activateOffset",
00054     "setOffset",
00055     "activateLogging"
00056 );
00057 #else // !QT_MOC_HAS_STRING_DATA
00058 struct qt_meta_stringdata_CLASSGUIENDCLASS_t {
00059     uint offsetsAndSizes[28];
00060     char stringdata0[4];
00061     char stringdata1[10];
00062     char stringdata2[1];
00063     char stringdata3[13];
00064     char stringdata4[14];
00065     char stringdata5[6];
00066     char stringdata6[8];
00067     char stringdata7[14];
00068     char stringdata8[9];
00069     char stringdata9[20];
00070     char stringdata10[15];
00071     char stringdata11[15];
00072     char stringdata12[10];
00073     char stringdata13[16];
00074 };
00075 #define QT_MOC_LITERAL(ofs, len) \
00076     uint(sizeof(qt_meta_stringdata_CLASSGUIENDCLASS_t::offsetsAndSizes) + ofs), len
00077 Q_CONSTINIT static const qt_meta_stringdata_CLASSGUIENDCLASS_t qt_meta_stringdata_CLASSGUIENDCLASS = {
00078     {
00079         QT_MOC_LITERAL(0, 3), // "GUI"
00080         QT_MOC_LITERAL(4, 9), // "calculate"
00081         QT_MOC_LITERAL(14, 0), // ""
00082         QT_MOC_LITERAL(15, 12), // "setInputPath"
00083         QT_MOC_LITERAL(28, 13), // "setOutputPath"
00084         QT_MOC_LITERAL(42, 5), // "setDP"
00085         QT_MOC_LITERAL(48, 7), // "setMean"
00086         QT_MOC_LITERAL(56, 13), // "activateSpeed"
00087         QT_MOC_LITERAL(70, 8), // "setSpeed"
00088         QT_MOC_LITERAL(79, 19), // "activateOrientation"
00089         QT_MOC_LITERAL(99, 14), // "setOrientation"
00090         QT_MOC_LITERAL(114, 14), // "activateOffset"
00091         QT_MOC_LITERAL(129, 9), // "setOffset"
00092         QT_MOC_LITERAL(139, 15) // "activateLogging"
00093     },
00094     "GUI",
00095     "calculate",
00096     "",
00097     "setInputPath",
00098     "setOutputPath",
00099     "setDP",
00100     "setMean",
00101     "activateSpeed",
00102     "setSpeed",
00103     "activateOrientation",
00104     "setOrientation",
00105     "activateOffset",
00106     "setOffset",
00107     "activateLogging"
00108 };
00109 #undef QT_MOC_LITERAL
00110 #endif // !QT_MOC_HAS_STRING_DATA
00111 } // unnamed namespace
00112
00113 Q_CONSTINIT static const uint qt_meta_data_CLASSGUIENDCLASS[] = {
00114     // content:
00115     11, // revision
00116     0, // classname
00117     0, // classinfo
00118     12, 14, // methods
00119     0, 0, // properties
00120     0, 0, // enums/sets
00121     0, 0, // constructors
00122     0, // flags
00123     0, // signalCount
00124
00125     // slots: name, argc, parameters, tag, flags, initial metatype offsets
00126     1, 0, 86, 2, 0x08, 1 /* Private */,
00127     3, 0, 87, 2, 0x08, 2 /* Private */,
00128     4, 0, 88, 2, 0x08, 3 /* Private */,
00129     5, 0, 89, 2, 0x08, 4 /* Private */,
00130     6, 0, 90, 2, 0x08, 5 /* Private */,
00131     7, 0, 91, 2, 0x08, 6 /* Private */,
00132     8, 0, 92, 2, 0x08, 7 /* Private */,
00133     9, 0, 93, 2, 0x08, 8 /* Private */,
00134     10, 0, 94, 2, 0x08, 9 /* Private */,
00135     11, 0, 95, 2, 0x08, 10 /* Private */,

```

```

00137     12,    0,   96,    2, 0x08,   11 /* Private */,
00138     13,    0,   97,    2, 0x08,   12 /* Private */,
00139
00140 // slots: parameters
00141     QMetaType::Void,
00142     QMetaType::Void,
00143     QMetaType::Void,
00144     QMetaType::Void,
00145     QMetaType::Void,
00146     QMetaType::Void,
00147     QMetaType::Void,
00148     QMetaType::Void,
00149     QMetaType::Void,
00150     QMetaType::Void,
00151     QMetaType::Void,
00152     QMetaType::Void,
00153
00154     0 // eod
00155 };
00156
00157 Q_CONSTINIT const QMetaObject GUI::staticMetaObject = { {
00158     QMetaObject::SuperData::link<QMainWindow::staticMetaObject>(),
00159     qt_meta_stringdata_CLASSGUIENDCLASS.offsetsAndSizes,
00160     qt_meta_data_CLASSGUIENDCLASS,
00161     qt_static_metacall,
00162     nullptr,
00163     qt_incomplete_metaTypeArray<qt_meta_stringdata_CLASSGUIENDCLASS_t,
00164         // Q_OBJECT / Q_GADGET
00165         QtPrivate::TypeAndForceComplete<GUI, std::true_type>,
00166         // method 'calculate'
00167         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00168         // method 'setInputPath'
00169         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00170         // method 'setOutputPath'
00171         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00172         // method 'setDP'
00173         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00174         // method 'setMean'
00175         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00176         // method 'activateSpeed'
00177         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00178         // method 'setSpeed'
00179         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00180         // method 'activateOrientation'
00181         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00182         // method 'setOrientation'
00183         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00184         // method 'activateOffset'
00185         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00186         // method 'setOffset'
00187         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00188         // method 'activateLogging'
00189         QtPrivate::TypeAndForceComplete<void, std::false_type>
00190     },
00191     nullptr
00192 } };
00193
00194 void GUI::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00195 {
00196     if (_c == QMetaObject::InvokeMetaMethod) {
00197         auto *_t = static_cast<GUI *>(_o);
00198         (void)_t;
00199         switch (_id) {
00200             case 0: _t->calculate(); break;
00201             case 1: _t->setInputPath(); break;
00202             case 2: _t->setOutputPath(); break;
00203             case 3: _t->setDP(); break;
00204             case 4: _t->setMean(); break;
00205             case 5: _t->activateSpeed(); break;
00206             case 6: _t->setSpeed(); break;
00207             case 7: _t->activateOrientation(); break;
00208             case 8: _t->setOrientation(); break;
00209             case 9: _t->activateOffset(); break;
00210             case 10: _t->setOffset(); break;
00211             case 11: _t->activateLogging(); break;
00212             default: ;
00213         }
00214     }
00215     (void)_a;
00216 }
00217
00218 const QMetaObject *GUI::metaObject() const
00219 {
00220     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00221 }
00222
00223 void *GUI::qt_metacast(const char *_cname)

```

```

00224 {
00225     if (!_cname) return nullptr;
00226     if (!strcmp(_cname, qt_meta_stringdata_CLASSGUIENDCLASS.stringdata0))
00227         return static_cast<void*>(this);
00228     return QMainWindow::qt_metacast(_cname);
00229 }
00230
00231 int GUI::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00232 {
00233     _id = QMainWindow::qt_metacall(_c, _id, _a);
00234     if (_id < 0)
00235         return _id;
00236     if (_c == QMetaObject::InvokeMetaMethod) {
00237         if (_id < 12)
00238             qt_static_metacall(this, _c, _id, _a);
00239         _id -= 12;
00240     } else if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
00241         if (_id < 12)
00242             *reinterpret_cast<QMetaType *>(_a[0]) = QMetaType();
00243         _id -= 12;
00244     }
00245     return _id;
00246 }
00247 QT_WARNING_POP

```

8.45 x64/Release/moc/moc_GUI.cpp-Dateireferenz

```

#include "../.../header/GUI.h"
#include <QtGui/QtTextCursor.h>
#include <QtCore/qmetatype.h>
#include <memory>

```

Klassen

- struct [QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSGUIENDCLASS_t](#)

Namensbereiche

- namespace [QT_WARNING_DISABLE_DEPRECATED](#)

Makrodefinitionen

- [#define Q_CONSTINIT](#)
- [#define QT_MOC_LITERAL\(ofs, len\) uint\(sizeof\(qt_meta_stringdata_CLASSGUIENDCLASS_t::offsets↵ AndSizes\) + ofs\), len](#)

8.45.1 Makro-Dokumentation

8.45.1.1 Q_CONSTINIT

```
#define Q_CONSTINIT
```

Definiert in Zeile 31 der Datei [moc_GUI.cpp](#).

8.45.1.2 QT_MOC_LITERAL

```
#define QT_MOC_LITERAL(
    ofs,
    len )  uint(sizeof(qt_meta_stringdata_CLASSGUIENDCLASS_t::offsetsAndSizes) +
ofs), len
```

Definiert in Zeile 75 der Datei [moc_GUI.cpp](#).

8.46 moc_GUI.cpp

[gehe zur Dokumentation dieser Datei](#)

```
00001 /*****
00002 ** Meta object code from reading C++ file 'GUI.h'
00003 **
00004 ** Created by: The Qt Meta Object Compiler version 68 (Qt 6.5.1)
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 *****/
00008
00009 #include "../..../header/GUI.h"
00010 #include <QtGui/qtextcursor.h>
00011 #include <QtCore/qmetatype.h>
00012
00013 #if __has_include(<QtCore/qtmochelpers.h>)
00014 #include <QtCore/qtmochelpers.h>
00015 #else
00016 QT_BEGIN_MOC_NAMESPACE
00017 #endif
00018
00019
00020 #include <memory>
00021
00022 #if !defined(Q_MOC_OUTPUT_REVISION)
00023 #error "The header file 'GUI.h' doesn't include <QObject>."
00024 #elif Q_MOC_OUTPUT_REVISION != 68
00025 #error "This file was generated using the moc from 6.5.1. It"
00026 #error "cannot be used with the include files from this version of Qt."
00027 #error "(The moc has changed too much.)"
00028 #endif
00029
00030 #ifndef Q_CONSTINIT
00031 #define Q_CONSTINIT
00032 #endif
00033
00034 QT_WARNING_PUSH
00035 QT_WARNING_DISABLE_DEPRECATED
00036 QT_WARNING_DISABLE_GCC("-Wuseless-cast")
00037 namespace {
00038
00039 #ifdef QT_MOC_HAS_STRINGDATA
00040 struct qt_meta_stringdata_CLASSGUIENDCLASS_t {};
00041 static constexpr auto qt_meta_stringdata_CLASSGUIENDCLASS = QtMocHelpers::stringData(
00042     "GUI",
00043     "calculate",
00044     "",
00045     "setInputPath",
00046     "setOutputPath",
00047     "setDP",
00048     "setMean",
00049     "activateSpeed",
00050     "setSpeed",
00051     "activateOrientation",
00052     "setOrientation",
00053     "activateOffset",
00054     "setOffset",
00055     "activateLogging"
00056 );
00057 #else // !QT_MOC_HAS_STRING_DATA
00058 struct qt_meta_stringdata_CLASSGUIENDCLASS_t {
00059     uint offsetsAndSizes[28];
00060     char stringdata0[4];
00061     char stringdata1[10];
00062     char stringdata2[1];
00063     char stringdata3[13];
00064     char stringdata4[14];
00065     char stringdata5[6];
```



```

00066     char stringdata6[8];
00067     char stringdata7[14];
00068     char stringdata8[9];
00069     char stringdata9[20];
00070     char stringdata10[15];
00071     char stringdata11[15];
00072     char stringdata12[10];
00073     char stringdata13[16];
00074 };
00075 #define QT_MOC_LITERAL(ofs, len) \
00076     uint(sizeof(qt_meta_stringdata_CLASSGUIENDCLASS_t::offsetsAndSizes) + ofs), len
00077 Q_CONSTINIT static const qt_meta_stringdata_CLASSGUIENDCLASS_t qt_meta_stringdata_CLASSGUIENDCLASS = {
00078     {
00079         QT_MOC_LITERAL(0, 3),    // "GUI"
00080         QT_MOC_LITERAL(4, 9),    // "calculate"
00081         QT_MOC_LITERAL(14, 0),   // ""
00082         QT_MOC_LITERAL(15, 12),  // "setInputPath"
00083         QT_MOC_LITERAL(28, 13),  // "setOutputPath"
00084         QT_MOC_LITERAL(42, 5),   // "setDP"
00085         QT_MOC_LITERAL(48, 7),   // "setMean"
00086         QT_MOC_LITERAL(56, 13),  // "activateSpeed"
00087         QT_MOC_LITERAL(70, 8),   // "setSpeed"
00088         QT_MOC_LITERAL(79, 19),  // "activateOrientation"
00089         QT_MOC_LITERAL(99, 14),  // "setOrientation"
00090         QT_MOC_LITERAL(114, 14), // "activateOffset"
00091         QT_MOC_LITERAL(129, 9),  // "setOffset"
00092         QT_MOC_LITERAL(139, 15)  // "activateLogging"
00093     },
00094     "GUI",
00095     "calculate",
00096     "",
00097     "setInputPath",
00098     "setOutputPath",
00099     "setDP",
00100     "setMean",
00101     "activateSpeed",
00102     "setSpeed",
00103     "activateOrientation",
00104     "setOrientation",
00105     "activateOffset",
00106     "setOffset",
00107     "activateLogging"
00108 };
00109 #undef QT_MOC_LITERAL
00110 #endif // !QT_MOC_HAS_STRING_DATA
00111 } // unnamed namespace
00112
00113 Q_CONSTINIT static const uint qt_meta_data_CLASSGUIENDCLASS[] = {
00114
00115     // content:
00116     11,        // revision
00117     0,         // classname
00118     0, 0,     // classinfo
00119     12, 14,   // methods
00120     0, 0,     // properties
00121     0, 0,     // enums/sets
00122     0, 0,     // constructors
00123     0,        // flags
00124     0,        // signalCount
00125
00126     // slots: name, argc, parameters, tag, flags, initial metatype offsets
00127     1, 0, 86, 2, 0x08, 1 /* Private */,
00128     3, 0, 87, 2, 0x08, 2 /* Private */,
00129     4, 0, 88, 2, 0x08, 3 /* Private */,
00130     5, 0, 89, 2, 0x08, 4 /* Private */,
00131     6, 0, 90, 2, 0x08, 5 /* Private */,
00132     7, 0, 91, 2, 0x08, 6 /* Private */,
00133     8, 0, 92, 2, 0x08, 7 /* Private */,
00134     9, 0, 93, 2, 0x08, 8 /* Private */,
00135     10, 0, 94, 2, 0x08, 9 /* Private */,
00136     11, 0, 95, 2, 0x08, 10 /* Private */,
00137     12, 0, 96, 2, 0x08, 11 /* Private */,
00138     13, 0, 97, 2, 0x08, 12 /* Private */,
00139
00140     // slots: parameters
00141     QMetaType::Void,
00142     QMetaType::Void,
00143     QMetaType::Void,
00144     QMetaType::Void,
00145     QMetaType::Void,
00146     QMetaType::Void,
00147     QMetaType::Void,
00148     QMetaType::Void,
00149     QMetaType::Void,
00150     QMetaType::Void,
00151     QMetaType::Void,
00152     QMetaType::Void,

```

```

00153
00154     0          // eod
00155 };
00156
00157 Q_CONSTINIT const QMetaObject GUI::staticMetaObject = { {
00158     QMetaObject::SuperData::link<QMainWindow::staticMetaObject>(),
00159     qt_meta_stringdata_CLASSGUIENDCLASS.offsetsAndSizes,
00160     qt_meta_data_CLASSGUIENDCLASS,
00161     qt_static_metacall,
00162     nullptr,
00163     qt_incomplete_metaTypeArray<qt_meta_stringdata_CLASSGUIENDCLASS_t,
00164         // Q_OBJECT / Q_GADGET
00165         QtPrivate::TypeAndForceComplete<GUI, std::true_type>,
00166         // method 'calculate'
00167         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00168         // method 'setInputPath'
00169         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00170         // method 'setOutputPath'
00171         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00172         // method 'setDP'
00173         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00174         // method 'setMean'
00175         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00176         // method 'activateSpeed'
00177         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00178         // method 'setSpeed'
00179         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00180         // method 'activateOrientation'
00181         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00182         // method 'setOrientation'
00183         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00184         // method 'activateOffset'
00185         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00186         // method 'setOffset'
00187         QtPrivate::TypeAndForceComplete<void, std::false_type>,
00188         // method 'activateLogging'
00189         QtPrivate::TypeAndForceComplete<void, std::false_type>
00190     },
00191     nullptr
00192 } };
00193
00194 void GUI::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00195 {
00196     if (_c == QMetaObject::InvokeMetaMethod) {
00197         auto *_t = static_cast<GUI *>(_o);
00198         (void)_t;
00199         switch (_id) {
00200             case 0: _t->calculate(); break;
00201             case 1: _t->setInputPath(); break;
00202             case 2: _t->setOutputPath(); break;
00203             case 3: _t->setDP(); break;
00204             case 4: _t->setMean(); break;
00205             case 5: _t->activateSpeed(); break;
00206             case 6: _t->setSpeed(); break;
00207             case 7: _t->activateOrientation(); break;
00208             case 8: _t->setOrientation(); break;
00209             case 9: _t->activateOffset(); break;
00210             case 10: _t->setOffset(); break;
00211             case 11: _t->activateLogging(); break;
00212             default: ;
00213         }
00214     }
00215     (void)_a;
00216 }
00217
00218 const QMetaObject *GUI::metaObject() const
00219 {
00220     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00221 }
00222
00223 void *GUI::qt_metacast(const char *_cname)
00224 {
00225     if (!_cname) return nullptr;
00226     if (!strcmp(_cname, qt_meta_stringdata_CLASSGUIENDCLASS.stringdata0))
00227         return static_cast<void*>(this);
00228     return QMainWindow::qt_metacast(_cname);
00229 }
00230
00231 int GUI::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00232 {
00233     _id = QMainWindow::qt_metacall(_c, _id, _a);
00234     if (_id < 0)
00235         return _id;
00236     if (_c == QMetaObject::InvokeMetaMethod) {
00237         if (_id < 12)
00238             qt_static_metacall(this, _c, _id, _a);
00239         _id -= 12;

```

```

00240     } else if (_c == QMetaObject::RegisterMethodArgumentMetaType) {
00241         if (_id < 12)
00242             *reinterpret_cast<QMetaType *>(_a[0]) = QMetaType();
00243         _id -= 12;
00244     }
00245     return _id;
00246 }
00247 QT_WARNING_POP

```

8.47 x64/Debug/moc/moc_switch.cpp-Dateireferenz

```

#include "../.../switch.h"
#include <QtGui/qtextcursor.h>
#include <QScreen>
#include <QtCore/qmetatype.h>
#include <memory>

```

Klassen

- struct [QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSwitchENDCLASS_t](#)

Namensbereiche

- namespace [QT_WARNING_DISABLE_DEPRECATED](#)

Makrodefinitionen

- #define [Q_CONSTINIT](#)
- #define [QT_MOC_LITERAL](#)(ofs, len) uint(sizeof(qt_meta_stringdata_CLASSSwitchENDCLASS_t::offsetsAndSizes) + ofs), len

8.47.1 Makro-Dokumentation

8.47.1.1 Q_CONSTINIT

```
#define Q_CONSTINIT
```

Definiert in Zeile [32](#) der Datei [moc_switch.cpp](#).

8.47.1.2 QT_MOC_LITERAL

```

#define QT_MOC_LITERAL(
    ofs,
    len )  uint(sizeof(qt_meta_stringdata_CLASSSwitchENDCLASS_t::offsetsAndSizes) +
ofs), len

```

Definiert in Zeile [54](#) der Datei [moc_switch.cpp](#).

8.48 moc_switch.cpp

[gehe zur Dokumentation dieser Datei](#)

```

00001 /*****
00002 ** Meta object code from reading C++ file 'switch.h'
00003 **
00004 ** Created by: The Qt Meta Object Compiler version 68 (Qt 6.5.1)
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 *****/
00008
00009 #include ".././././switch.h"
00010 #include <QtGui/qttextcursor.h>
00011 #include <QScreen>
00012 #include <QtCore/qmetatype.h>
00013
00014 #if __has_include(<QtCore/qtmochelpers.h>)
00015 #include <QtCore/qtmochelpers.h>
00016 #else
00017 QT_BEGIN_MOC_NAMESPACE
00018 #endif
00019
00020
00021 #include <memory>
00022
00023 #if !defined(Q_MOC_OUTPUT_REVISION)
00024 #error "The header file 'switch.h' doesn't include <QObject>."
00025 #elif Q_MOC_OUTPUT_REVISION != 68
00026 #error "This file was generated using the moc from 6.5.1. It"
00027 #error "cannot be used with the include files from this version of Qt."
00028 #error "(The moc has changed too much.)"
00029 #endif
00030
00031 #ifndef Q_CONSTINIT
00032 #define Q_CONSTINIT
00033 #endif
00034
00035 QT_WARNING_PUSH
00036 QT_WARNING_DISABLE_DEPRECATED
00037 QT_WARNING_DISABLE_GCC("-Wuseless-cast")
00038 namespace {
00039
00040 #ifdef QT_MOC_HAS_STRINGDATA
00041 struct qt_meta_stringdata_CLASSSwitchENDCLASS_t {};
00042 static constexpr auto qt_meta_stringdata_CLASSSwitchENDCLASS = QtMocHelpers::stringData(
00043     "Switch",
00044     "offset",
00045     "brush"
00046 );
00047 #else // !QT_MOC_HAS_STRING_DATA
00048 struct qt_meta_stringdata_CLASSSwitchENDCLASS_t {
00049     uint offsetsAndSizes[6];
00050     char stringdata0[7];
00051     char stringdata1[7];
00052     char stringdata2[6];
00053 };
00054 #define QT_MOC_LITERAL(ofs, len) \
00055     uint(sizeof(qt_meta_stringdata_CLASSSwitchENDCLASS_t::offsetsAndSizes) + ofs), len
00056 Q_CONSTINIT static const qt_meta_stringdata_CLASSSwitchENDCLASS_t
00057 qt_meta_stringdata_CLASSSwitchENDCLASS = {
00058     {
00059         QT_MOC_LITERAL(0, 6), // "Switch"
00060         QT_MOC_LITERAL(7, 6), // "offset"
00061         QT_MOC_LITERAL(14, 5) // "brush"
00062     },
00063     "Switch",
00064     "offset",
00065     "brush"
00066 };
00067 #undef QT_MOC_LITERAL
00068 #endif // !QT_MOC_HAS_STRING_DATA
00069 } // unnamed namespace
00070
00071 Q_CONSTINIT static const uint qt_meta_data_CLASSSwitchENDCLASS[] = {
00072     // content:
00073     11, // revision
00074     0, // classname
00075     0, 0, // classinfo
00076     0, 0, // methods
00077     2, 14, // properties
00078     0, 0, // enums/sets
00079     0, 0, // constructors
00080     0, // flags
00081     0, // signalCount

```

```

00082
00083 // properties: name, type, flags
00084     1, QMetaType::Int, 0x00015103, uint(-1), 0,
00085     2, QMetaType::QBrush, 0x00015103, uint(-1), 0,
00086
00087     0 // eod
00088 };
00089
00090 Q_CONSTINIT const QMetaObject Switch::staticMetaObject = { {
00091     QMetaObject::SuperData::link<QAbstractButton::staticMetaObject>(),
00092     qt_meta_stringdata_CLASSSwitchENDCLASS.offsetsAndSizes,
00093     qt_meta_data_CLASSSwitchENDCLASS,
00094     qt_static_metacall,
00095     nullptr,
00096     qt_incomplete_metaTypeArray<qt_meta_stringdata_CLASSSwitchENDCLASS_t,
00097         // property 'offset'
00098         QtPrivate::TypeAndForceComplete<int, std::true_type>,
00099         // property 'brush'
00100         QtPrivate::TypeAndForceComplete<QBrush, std::true_type>,
00101         // Q_OBJECT / Q_GADGET
00102         QtPrivate::TypeAndForceComplete<Switch, std::true_type>
00103     >,
00104     nullptr
00105 } };
00106
00107 void Switch::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00108 {
00109     if (_c == QMetaObject::ReadProperty) {
00110         auto *_t = static_cast<Switch *>(_o);
00111         (void)_t;
00112         void *_v = _a[0];
00113         switch (_id) {
00114             case 0: *reinterpret_cast< int*>(_v) = _t->offset(); break;
00115             case 1: *reinterpret_cast< QBrush*>(_v) = _t->brush(); break;
00116             default: break;
00117         }
00118     } else if (_c == QMetaObject::WriteProperty) {
00119         auto *_t = static_cast<Switch *>(_o);
00120         (void)_t;
00121         void *_v = _a[0];
00122         switch (_id) {
00123             case 0: _t->setOffset(*reinterpret_cast< int*>(_v)); break;
00124             case 1: _t->setBrush(*reinterpret_cast< QBrush*>(_v)); break;
00125             default: break;
00126         }
00127     } else if (_c == QMetaObject::ResetProperty) {
00128     } else if (_c == QMetaObject::BindableProperty) {
00129     }
00130     (void)_o;
00131     (void)_id;
00132     (void)_c;
00133     (void)_a;
00134 }
00135
00136 const QMetaObject *Switch::metaObject() const
00137 {
00138     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00139 }
00140
00141 void *Switch::qt_metacast(const char *_cname)
00142 {
00143     if (!_cname) return nullptr;
00144     if (!strcmp(_cname, qt_meta_stringdata_CLASSSwitchENDCLASS.stringdata0))
00145         return static_cast<void*>(this);
00146     return QAbstractButton::qt_metacast(_cname);
00147 }
00148
00149 int Switch::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00150 {
00151     _id = QAbstractButton::qt_metacall(_c, _id, _a);
00152     if (_id < 0)
00153         return _id;
00154     if (_c == QMetaObject::ReadProperty || _c == QMetaObject::WriteProperty
00155         || _c == QMetaObject::ResetProperty || _c == QMetaObject::BindableProperty
00156         || _c == QMetaObject::RegisterPropertyMetaType) {
00157         qt_static_metacall(this, _c, _id, _a);
00158         _id -= 2;
00159     }
00160     return _id;
00161 }
00162 QT_WARNING_POP

```

8.49 x64/Release/moc/moc_switch.cpp-Dateireferenz

```
#include "../../switch.h"
#include <QtGui/QtTextCursor.h>
#include <QScreen>
#include <QtCore/qmetatype.h>
#include <memory>
```

Klassen

- struct [QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSwitchENDCLASS_t](#)

Namensbereiche

- namespace [QT_WARNING_DISABLE_DEPRECATED](#)

Makrodefinitionen

- #define [Q_CONSTINIT](#)
- #define [QT_MOC_LITERAL](#)(ofs, len) uint(sizeof(qt_meta_stringdata_CLASSSwitchENDCLASS_t::offsets↵AndSizes) + ofs), len

8.49.1 Makro-Dokumentation

8.49.1.1 Q_CONSTINIT

```
#define Q_CONSTINIT
```

Definiert in Zeile [32](#) der Datei [moc_switch.cpp](#).

8.49.1.2 QT_MOC_LITERAL

```
#define QT_MOC_LITERAL(  
    ofs,  
    len )  uint(sizeof(qt_meta_stringdata_CLASSSwitchENDCLASS_t::offsetsAndSizes) +  
ofs), len
```

Definiert in Zeile [54](#) der Datei [moc_switch.cpp](#).

8.50 moc_switch.cpp

[gehe zur Dokumentation dieser Datei](#)

```

00001 /*****
00002 ** Meta object code from reading C++ file 'switch.h'
00003 **
00004 ** Created by: The Qt Meta Object Compiler version 68 (Qt 6.5.1)
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 *****/
00008
00009 #include ".././././switch.h"
00010 #include <QtGui/qttextcursor.h>
00011 #include <QScreen>
00012 #include <QtCore/qmetatype.h>
00013
00014 #if __has_include(<QtCore/qtmochelpers.h>)
00015 #include <QtCore/qtmochelpers.h>
00016 #else
00017 QT_BEGIN_MOC_NAMESPACE
00018 #endif
00019
00020
00021 #include <memory>
00022
00023 #if !defined(Q_MOC_OUTPUT_REVISION)
00024 #error "The header file 'switch.h' doesn't include <QObject>."
00025 #elif Q_MOC_OUTPUT_REVISION != 68
00026 #error "This file was generated using the moc from 6.5.1. It"
00027 #error "cannot be used with the include files from this version of Qt."
00028 #error "(The moc has changed too much.)"
00029 #endif
00030
00031 #ifndef Q_CONSTINIT
00032 #define Q_CONSTINIT
00033 #endif
00034
00035 QT_WARNING_PUSH
00036 QT_WARNING_DISABLE_DEPRECATED
00037 QT_WARNING_DISABLE_GCC("-Wuseless-cast")
00038 namespace {
00039
00040 #ifdef QT_MOC_HAS_STRINGDATA
00041 struct qt_meta_stringdata_CLASSSwitchENDCLASS_t {};
00042 static constexpr auto qt_meta_stringdata_CLASSSwitchENDCLASS = QtMocHelpers::stringData(
00043     "Switch",
00044     "offset",
00045     "brush"
00046 );
00047 #else // !QT_MOC_HAS_STRING_DATA
00048 struct qt_meta_stringdata_CLASSSwitchENDCLASS_t {
00049     uint offsetsAndSizes[6];
00050     char stringdata0[7];
00051     char stringdata1[7];
00052     char stringdata2[6];
00053 };
00054 #define QT_MOC_LITERAL(ofs, len) \
00055     uint(sizeof(qt_meta_stringdata_CLASSSwitchENDCLASS_t::offsetsAndSizes) + ofs), len
00056 Q_CONSTINIT static const qt_meta_stringdata_CLASSSwitchENDCLASS_t
00057 qt_meta_stringdata_CLASSSwitchENDCLASS = {
00058     {
00059         QT_MOC_LITERAL(0, 6), // "Switch"
00060         QT_MOC_LITERAL(7, 6), // "offset"
00061         QT_MOC_LITERAL(14, 5) // "brush"
00062     },
00063     "Switch",
00064     "offset",
00065     "brush"
00066 };
00067 #undef QT_MOC_LITERAL
00068 #endif // !QT_MOC_HAS_STRING_DATA
00069 } // unnamed namespace
00070
00071 Q_CONSTINIT static const uint qt_meta_data_CLASSSwitchENDCLASS[] = {
00072     // content:
00073     11, // revision
00074     0, // classname
00075     0, 0, // classinfo
00076     0, 0, // methods
00077     2, 14, // properties
00078     0, 0, // enums/sets
00079     0, 0, // constructors
00080     0, // flags
00081     0, // signalCount

```

```

00082
00083 // properties: name, type, flags
00084     1, QMetaType::Int, 0x00015103, uint(-1), 0,
00085     2, QMetaType::QBrush, 0x00015103, uint(-1), 0,
00086
00087     0 // eod
00088 };
00089
00090 Q_CONSTINIT const QMetaObject Switch::staticMetaObject = { {
00091     QMetaObject::SuperData::link<QAbstractButton::staticMetaObject>(),
00092     qt_meta_stringdata_CLASSSwitchENDCLASS.offsetsAndSizes,
00093     qt_meta_data_CLASSSwitchENDCLASS,
00094     qt_static_metacall,
00095     nullptr,
00096     qt_incomplete_metaTypeArray<qt_meta_stringdata_CLASSSwitchENDCLASS_t,
00097         // property 'offset'
00098         QtPrivate::TypeAndForceComplete<int, std::true_type>,
00099         // property 'brush'
00100         QtPrivate::TypeAndForceComplete<QBrush, std::true_type>,
00101         // Q_OBJECT / Q_GADGET
00102         QtPrivate::TypeAndForceComplete<Switch, std::true_type>
00103     >,
00104     nullptr
00105 } };
00106
00107 void Switch::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
00108 {
00109     if (_c == QMetaObject::ReadProperty) {
00110         auto *_t = static_cast<Switch *>(_o);
00111         (void)_t;
00112         void *_v = _a[0];
00113         switch (_id) {
00114             case 0: *reinterpret_cast< int*>(_v) = _t->offset(); break;
00115             case 1: *reinterpret_cast< QBrush*>(_v) = _t->brush(); break;
00116             default: break;
00117         }
00118     } else if (_c == QMetaObject::WriteProperty) {
00119         auto *_t = static_cast<Switch *>(_o);
00120         (void)_t;
00121         void *_v = _a[0];
00122         switch (_id) {
00123             case 0: _t->setOffset(*reinterpret_cast< int*>(_v)); break;
00124             case 1: _t->setBrush(*reinterpret_cast< QBrush*>(_v)); break;
00125             default: break;
00126         }
00127     } else if (_c == QMetaObject::ResetProperty) {
00128     } else if (_c == QMetaObject::BindableProperty) {
00129     }
00130     (void)_o;
00131     (void)_id;
00132     (void)_c;
00133     (void)_a;
00134 }
00135
00136 const QMetaObject *Switch::metaObject() const
00137 {
00138     return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() : &staticMetaObject;
00139 }
00140
00141 void *Switch::qt_metacast(const char *_cname)
00142 {
00143     if (!_cname) return nullptr;
00144     if (!strcmp(_cname, qt_meta_stringdata_CLASSSwitchENDCLASS.stringdata0))
00145         return static_cast<void*>(this);
00146     return QAbstractButton::qt_metacast(_cname);
00147 }
00148
00149 int Switch::qt_metacall(QMetaObject::Call _c, int _id, void **_a)
00150 {
00151     _id = QAbstractButton::qt_metacall(_c, _id, _a);
00152     if (_id < 0)
00153         return _id;
00154     if (_c == QMetaObject::ReadProperty || _c == QMetaObject::WriteProperty
00155         || _c == QMetaObject::ResetProperty || _c == QMetaObject::BindableProperty
00156         || _c == QMetaObject::RegisterPropertyMetaType) {
00157         qt_static_metacall(this, _c, _id, _a);
00158         _id -= 2;
00159     }
00160     return _id;
00161 }
00162 QT_WARNING_POP

```


8.51 x64/Debug/rcc/qrc_GUI.cpp-Dateireferenz

Makrodefinitionen

- `#define QT_RCC_PREPEND_NAMESPACE(name) name`
- `#define QT_RCC_MANGLE_NAMESPACE(name) name`

Funktionen

- `int QT_RCC_MANGLE_NAMESPACE() qInitResources_GUI ()`
- `int QT_RCC_MANGLE_NAMESPACE() qCleanupResources_GUI ()`

8.51.1 Makro-Dokumentation

8.51.1.1 QT_RCC_MANGLE_NAMESPACE

```
#define QT_RCC_MANGLE_NAMESPACE(  
    name ) name
```

Definiert in Zeile 18 der Datei [qrc_GUI.cpp](#).

8.51.1.2 QT_RCC_PREPEND_NAMESPACE

```
#define QT_RCC_PREPEND_NAMESPACE(  
    name ) name
```

Definiert in Zeile 17 der Datei [qrc_GUI.cpp](#).

8.51.2 Dokumentation der Funktionen

8.51.2.1 qCleanupResources_GUI()

```
int QT_RCC_MANGLE_NAMESPACE() qCleanupResources_GUI ( )
```

Definiert in Zeile 36 der Datei [qrc_GUI.cpp](#).

```
00037 {  
00038     return 1;  
00039 }
```

8.51.2.2 qInitResources_GUI()

```
int QT_RCC_MANGLE_NAMESPACE() qInitResources_GUI ( )
```

Definiert in Zeile 30 der Datei [qrc_GUI.cpp](#).

```
00031 {  
00032     return 1;  
00033 }
```

8.52 qrc_GUI.cpp

[gehe zur Dokumentation dieser Datei](#)

```

00001 /*****
00002 ** Resource object code
00003 **
00004 ** Created by: The Resource Compiler for Qt version 6.5.1
00005 **
00006 ** WARNING! All changes made in this file will be lost!
00007 *****/
00008
00009 #ifdef QT_NAMESPACE
00010 #   define QT_RCC_PREPEND_NAMESPACE(name) ::QT_NAMESPACE::name
00011 #   define QT_RCC_MANGLE_NAMESPACE0(x) x
00012 #   define QT_RCC_MANGLE_NAMESPACE1(a, b) a##_##b
00013 #   define QT_RCC_MANGLE_NAMESPACE2(a, b) QT_RCC_MANGLE_NAMESPACE1(a,b)
00014 #   define QT_RCC_MANGLE_NAMESPACE(name) QT_RCC_MANGLE_NAMESPACE2( \
00015       QT_RCC_MANGLE_NAMESPACE0(name), QT_RCC_MANGLE_NAMESPACE0(QT_NAMESPACE))
00016 #else
00017 #   define QT_RCC_PREPEND_NAMESPACE(name) name
00018 #   define QT_RCC_MANGLE_NAMESPACE(name) name
00019 #endif
00020
00021 #ifdef QT_NAMESPACE
00022 namespace QT_NAMESPACE {
00023 #endif
00024
00025 #ifdef QT_NAMESPACE
00026 }
00027 #endif
00028
00029 int QT_RCC_MANGLE_NAMESPACE(qInitResources_GUI)();
00030 int QT_RCC_MANGLE_NAMESPACE(qInitResources_GUI)()
00031 {
00032     return 1;
00033 }
00034
00035 int QT_RCC_MANGLE_NAMESPACE(qCleanupResources_GUI)();
00036 int QT_RCC_MANGLE_NAMESPACE(qCleanupResources_GUI)()
00037 {
00038     return 1;
00039 }
00040
00041 #ifdef __clang__
00042 #   pragma clang diagnostic push
00043 #   pragma clang diagnostic ignored "-Wexit-time-destructors"
00044 #endif
00045
00046 namespace {
00047     struct initializer {
00048         initializer() { QT_RCC_MANGLE_NAMESPACE(qInitResources_GUI)(); }
00049         ~initializer() { QT_RCC_MANGLE_NAMESPACE(qCleanupResources_GUI)(); }
00050     } dummy;
00051 }
00052
00053 #ifdef __clang__
00054 #   pragma clang diagnostic pop
00055 #endif

```

8.53 x64/Release/rcc/qrc_GUI.cpp-Dateireferenz

Makrodefinitionen

- `#define QT_RCC_PREPEND_NAMESPACE(name) name`
- `#define QT_RCC_MANGLE_NAMESPACE(name) name`

Funktionen

- `int QT_RCC_MANGLE_NAMESPACE() qInitResources_GUI ()`
- `int QT_RCC_MANGLE_NAMESPACE() qCleanupResources_GUI ()`

8.53.1 Makro-Dokumentation

8.53.1.1 QT_RCC_MANGLE_NAMESPACE

```
#define QT_RCC_MANGLE_NAMESPACE(  
    name ) name
```

Definiert in Zeile 18 der Datei [qrc_GUI.cpp](#).

8.53.1.2 QT_RCC_PREPEND_NAMESPACE

```
#define QT_RCC_PREPEND_NAMESPACE(  
    name ) name
```

Definiert in Zeile 17 der Datei [qrc_GUI.cpp](#).

8.53.2 Dokumentation der Funktionen

8.53.2.1 qCleanupResources_GUI()

```
int QT_RCC_MANGLE_NAMESPACE() qCleanupResources_GUI ( )
```

Definiert in Zeile 36 der Datei [qrc_GUI.cpp](#).

```
00037 {  
00038     return 1;  
00039 }
```

8.53.2.2 qInitResources_GUI()

```
int QT_RCC_MANGLE_NAMESPACE() qInitResources_GUI ( )
```

Definiert in Zeile 30 der Datei [qrc_GUI.cpp](#).

```
00031 {  
00032     return 1;  
00033 }
```

8.54 qrc_GUI.cpp

[gehe zur Dokumentation dieser Datei](#)

```
00001 /*****  
00002 ** Resource object code  
00003 **  
00004 ** Created by: The Resource Compiler for Qt version 6.5.1  
00005 **  
00006 ** WARNING! All changes made in this file will be lost!  
00007 *****/  
00008  
00009 #ifndef QT_NAMESPACE  
00010 # define QT_RCC_PREPEND_NAMESPACE(name) ::QT_NAMESPACE::name  
00011 # define QT_RCC_MANGLE_NAMESPACE0(x) x  
00012 # define QT_RCC_MANGLE_NAMESPACE1(a, b) a##_##b  
00013 # define QT_RCC_MANGLE_NAMESPACE2(a, b) QT_RCC_MANGLE_NAMESPACE1(a,b)  
00014 # define QT_RCC_MANGLE_NAMESPACE(name) QT_RCC_MANGLE_NAMESPACE2( \  
00015     QT_RCC_MANGLE_NAMESPACE0(name), QT_RCC_MANGLE_NAMESPACE0(QT_NAMESPACE))  
00016 #else  
00017 # define QT_RCC_PREPEND_NAMESPACE(name) name  
00018 # define QT_RCC_MANGLE_NAMESPACE(name) name
```

```

00019 #endif
00020
00021 #ifdef QT_NAMESPACE
00022 namespace QT_NAMESPACE {
00023 #endif
00024
00025 #ifdef QT_NAMESPACE
00026 }
00027 #endif
00028
00029 int QT_RCC_MANGLE_NAMESPACE(qInitResources_GUI)();
00030 int QT_RCC_MANGLE_NAMESPACE(qInitResources_GUI)()
00031 {
00032     return 1;
00033 }
00034
00035 int QT_RCC_MANGLE_NAMESPACE(qCleanupResources_GUI)();
00036 int QT_RCC_MANGLE_NAMESPACE(qCleanupResources_GUI)()
00037 {
00038     return 1;
00039 }
00040
00041 #ifdef __clang__
00042 #   pragma clang diagnostic push
00043 #   pragma clang diagnostic ignored "-Wexit-time-destructors"
00044 #endif
00045
00046 namespace {
00047     struct initializer {
00048         initializer() { QT_RCC_MANGLE_NAMESPACE(qInitResources_GUI)(); }
00049         ~initializer() { QT_RCC_MANGLE_NAMESPACE(qCleanupResources_GUI)(); }
00050     } dummy;
00051 }
00052
00053 #ifdef __clang__
00054 #   pragma clang diagnostic pop
00055 #endif

```

8.55 x64/Debug/uic/ui_GUI.h-Dateireferenz

```

#include <QtCore/QVariant>
#include <QtWidgets/QApplication>
#include <QtWidgets/QCheckBox>
#include <QtWidgets/QDoubleSpinBox>
#include <QtWidgets/QFrame>
#include <QtWidgets/QLabel>
#include <QtWidgets/QMainWindow>
#include <QtWidgets/QPushButton>
#include <QtWidgets/QSpinBox>
#include <QtWidgets/QTextBrowser>
#include <QtWidgets/QWidget>

```

Klassen

- class [Ui_GUIClass](#)
- class [Ui::GUIClass](#)

Namensbereiche

- namespace [Ui](#)

8.56 ui_GUI.h

[gehe zur Dokumentation dieser Datei](#)

```

00001 /*****
00002 ** Form generated from reading UI file 'GUI.ui'
00003 **
00004 ** Created by: Qt User Interface Compiler version 6.5.1
00005 **
00006 ** WARNING! All changes made in this file will be lost when recompiling UI file!
00007 *****/
00008
00009 #ifndef UI_GUI_H
00010 #define UI_GUI_H
00011
00012 #include <QtCore/QVariant>
00013 #include <QtWidgets/QApplication>
00014 #include <QtWidgets/QCheckBox>
00015 #include <QtWidgets/QDoubleSpinBox>
00016 #include <QtWidgets/QFrame>
00017 #include <QtWidgets/QLabel>
00018 #include <QtWidgets/QMainWindow>
00019 #include <QtWidgets/QPushButton>
00020 #include <QtWidgets/QSpinBox>
00021 #include <QtWidgets/QTextBrowser>
00022 #include <QtWidgets/QWidget>
00023
00024 QT_BEGIN_NAMESPACE
00025
00026 class Ui_GUIClass
00027 {
00028 public:
00029     QWidget *centralWidget;
00030     QLabel *pathInput;
00031     QFrame *frame;
00032     QWidget *speed_2;
00033     QDoubleSpinBox *speed;
00034     QLabel *label_5;
00035     QCheckBox *bSpeed;
00036     QWidget *orientation;
00037     QLabel *label_10;
00038     QLabel *label_11;
00039     QLabel *label_12;
00040     QDoubleSpinBox *AValue;
00041     QDoubleSpinBox *BValue;
00042     QDoubleSpinBox *CValue;
00043     QCheckBox *bManOrientation;
00044     QSpinBox *meanLength;
00045     QLabel *label_dp;
00046     QLabel *label_4;
00047     QSpinBox *dpToleranz;
00048     QCheckBox *bLogging;
00049     QCheckBox *bOffset;
00050     QWidget *offset;
00051     QLabel *label_13;
00052     QLabel *label_14;
00053     QLabel *label_15;
00054     QDoubleSpinBox *offsetX;
00055     QDoubleSpinBox *offsetY;
00056     QDoubleSpinBox *offsetZ;
00057     QPushButton *pushInput;
00058     QPushButton *pushOutput;
00059     QTextBrowser *textBrowser;
00060     QPushButton *startCalculation;
00061     QLabel *pathOutput;
00062
00063     void setupUi(QMainWindow *GUIClass)
00064     {
00065         if (GUIClass->objectName().isEmpty())
00066             GUIClass->setObjectName("GUIClass");
00067         GUIClass->resize(355, 700);
00068         QSizePolicy sizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);
00069         sizePolicy.setHorizontalStretch(0);
00070         sizePolicy.setVerticalStretch(0);
00071         sizePolicy.setHeightForWidth(GUIClass->sizePolicy().hasHeightForWidth());
00072         GUIClass->setSizePolicy(sizePolicy);
00073         GUIClass->setMinimumSize(QSize(355, 530));
00074         GUIClass->setMaximumSize(QSize(355, 700));
00075         QFont font;
00076         font.setFamilies({QString::fromUtf8("Rubik")});
00077         GUIClass->setFont(font);
00078         GUIClass->setAnimated(true);
00079         GUIClass->setTabShape(QTabWidget::Rounded);
00080         GUIClass->setUnifiedTitleAndToolBarOnMac(false);
00081         centralWidget = new QWidget(GUIClass);
00082         centralWidget->setObjectName("centralWidget");

```

```

00083         QFont font1;
00084         font1.setFamilies({QString::fromUtf8("Rubik")});
00085         font1.setBold(false);
00086         font1.setItalic(false);
00087         centralWidget->setFont(font1);
00088         centralWidget->setStyleSheet(QString::fromUtf8("background-color: rgb(3, 8, 14);\n"
00089 "color: rgb(3, 8, 14);\n"
00090 "));
00091         pathInput = new QLabel(centralWidget);
00092         pathInput->setObjectName("pathInput");
00093         pathInput->setGeometry(QRect(10, 10, 331, 31));
00094         pathInput->setFont(font1);
00095         pathInput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00096 "border-radius: 10px;\n"
00097 "background-color: rgb(210, 211, 218);\n"
00098 "color: rgb(3, 8, 14);"));
00099         frame = new QFrame(centralWidget);
00100         frame->setObjectName("frame");
00101         frame->setGeometry(QRect(10, 90, 331, 491));
00102         frame->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00103 "border-radius: 10px;\n"
00104 "background-color: rgb(117, 125, 149)"));
00105         frame->setFrameShape(QFrame::Box);
00106         frame->setFrameShadow(QFrame::Raised);
00107         frame->setLineWidth(1);
00108         speed_2 = new QWidget(frame);
00109         speed_2->setObjectName("speed_2");
00110         speed_2->setEnabled(false);
00111         speed_2->setGeometry(QRect(15, 110, 301, 41));
00112         QFont font2;
00113         font2.setFamilies({QString::fromUtf8("Rubik")});
00114         font2.setBold(true);
00115         speed_2->setFont(font2);
00116         speed_2->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00117 "color: rgb(117, 125, 149);\n"
00118 "border: 1px solid rgb(67, 72, 91); "));
00119         speed = new QDoubleSpinBox(speed_2);
00120         speed->setObjectName("speed");
00121         speed->setGeometry(QRect(159, 12, 136, 20));
00122         speed->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00123 "\n"
00124 "border-radius: 6px;"));
00125         speed->setAlignment(Qt::AlignCenter);
00126         label_5 = new QLabel(speed_2);
00127         label_5->setObjectName("label_5");
00128         label_5->setGeometry(QRect(11, 12, 93, 16));
00129         label_5->setFont(font1);
00130         label_5->setStyleSheet(QString::fromUtf8("border: 0px"));
00131         bSpeed = new QCheckBox(frame);
00132         bSpeed->setObjectName("bSpeed");
00133         bSpeed->setGeometry(QRect(15, 80, 171, 20));
00134         bSpeed->setFont(font1);
00135         bSpeed->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00136 "border: 0px"));
00137         orientation = new QWidget(frame);
00138         orientation->setObjectName("orientation");
00139         orientation->setEnabled(false);
00140         orientation->setGeometry(QRect(15, 190, 301, 111));
00141         orientation->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00142 "color: rgb(117, 125, 149);\n"
00143 "border: 1px solid rgb(67, 72, 91); "));
00144         label_10 = new QLabel(orientation);
00145         label_10->setObjectName("label_10");
00146         label_10->setGeometry(QRect(10, 15, 18, 16));
00147         label_10->setFont(font1);
00148         label_10->setStyleSheet(QString::fromUtf8("border: 0px"));
00149         label_11 = new QLabel(orientation);
00150         label_11->setObjectName("label_11");
00151         label_11->setGeometry(QRect(11, 47, 33, 16));
00152         label_11->setFont(font1);
00153         label_11->setStyleSheet(QString::fromUtf8("\n"
00154 "border: 0px"));
00155         label_12 = new QLabel(orientation);
00156         label_12->setObjectName("label_12");
00157         label_12->setGeometry(QRect(11, 78, 17, 16));
00158         label_12->setFont(font1);
00159         label_12->setStyleSheet(QString::fromUtf8("border: 0px"));
00160         AValue = new QDoubleSpinBox(orientation);
00161         AValue->setObjectName("AValue");
00162         AValue->setGeometry(QRect(153, 15, 141, 20));
00163         AValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00164 "\n"
00165 "border-radius: 6px;"));
00166         BValue = new QDoubleSpinBox(orientation);
00167         BValue->setObjectName("BValue");
00168         BValue->setGeometry(QRect(154, 47, 141, 20));
00169         BValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"

```

```

00170 "\n"
00171 "border-radius: 6px;"));
00172     CValue = new QDoubleSpinBox(orientation);
00173     CValue->setObjectName("CValue");
00174     CValue->setGeometry(QRect(154, 78, 141, 20));
00175     CValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00176 "\n"
00177 "border-radius: 6px;"));
00178     label_12->raise();
00179     label_10->raise();
00180     AValue->raise();
00181     BValue->raise();
00182     CValue->raise();
00183     label_11->raise();
00184     bManOrientation = new QCheckBox(frame);
00185     bManOrientation->setObjectName("bManOrientation");
00186     bManOrientation->setGeometry(QRect(15, 160, 151, 20));
00187     bManOrientation->setFont(font1);
00188     bManOrientation->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00189 "border: 0px;"));
00190     meanLength = new QSpinBox(frame);
00191     meanLength->setObjectName("meanLength");
00192     meanLength->setGeometry(QRect(200, 48, 116, 20));
00193     meanLength->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00194 "\n"
00195 "border-radius: 6px;"));
00196     label_dp = new QLabel(frame);
00197     label_dp->setObjectName("label_dp");
00198     label_dp->setGeometry(QRect(14, 22, 144, 16));
00199     label_dp->setFont(font1);
00200     label_dp->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00201 "border: 0px;"));
00202     label_4 = new QLabel(frame);
00203     label_4->setObjectName("label_4");
00204     label_4->setGeometry(QRect(14, 48, 180, 16));
00205     label_4->setFont(font1);
00206     label_4->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00207 "border: 0px;"));
00208     dpToleranz = new QSpinBox(frame);
00209     dpToleranz->setObjectName("dpToleranz");
00210     dpToleranz->setGeometry(QRect(200, 22, 116, 20));
00211     dpToleranz->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00212 "\n"
00213 "border-radius: 6px;"));
00214     dpToleranz->setFrame(true);
00215     dpToleranz->setButtonSymbols(QAbstractSpinBox::UpDownArrows);
00216     dpToleranz->setAccelerated(false);
00217     bLogging = new QCheckBox(frame);
00218     bLogging->setObjectName("bLogging");
00219     bLogging->setGeometry(QRect(15, 460, 151, 20));
00220     bLogging->setFont(font1);
00221     bLogging->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00222 "border: 0px;"));
00223     bOffset = new QCheckBox(frame);
00224     bOffset->setObjectName("bOffset");
00225     bOffset->setGeometry(QRect(15, 310, 151, 20));
00226     bOffset->setFont(font1);
00227     bOffset->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00228 "border: 0px;"));
00229     offset = new QWidget(frame);
00230     offset->setObjectName("offset");
00231     offset->setEnabled(false);
00232     offset->setGeometry(QRect(15, 340, 301, 111));
00233     offset->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00234 "color: rgb(117, 125, 149);\n"
00235 "border: 1px solid rgb(67, 72, 91); "));
00236     label_13 = new QLabel(offset);
00237     label_13->setObjectName("label_13");
00238     label_13->setGeometry(QRect(10, 15, 18, 16));
00239     label_13->setFont(font1);
00240     label_13->setStyleSheet(QString::fromUtf8("border: 0px;"));
00241     label_14 = new QLabel(offset);
00242     label_14->setObjectName("label_14");
00243     label_14->setGeometry(QRect(11, 47, 33, 16));
00244     label_14->setFont(font1);
00245     label_14->setStyleSheet(QString::fromUtf8("\n"
00246 "border: 0px;"));
00247     label_15 = new QLabel(offset);
00248     label_15->setObjectName("label_15");
00249     label_15->setGeometry(QRect(11, 78, 17, 16));
00250     label_15->setFont(font1);
00251     label_15->setStyleSheet(QString::fromUtf8("border: 0px;"));
00252     offsetX = new QDoubleSpinBox(offset);
00253     offsetX->setObjectName("offsetX");
00254     offsetX->setGeometry(QRect(153, 15, 141, 20));
00255     offsetX->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00256 "\n"

```

```

00257 "border-radius: 6px;"));
00258     offsetY = new QDoubleSpinBox(offset);
00259     offsetY->setObjectName("offsetY");
00260     offsetY->setGeometry(QRect(154, 47, 141, 20));
00261     offsetY->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00262 "\n"
00263 "border-radius: 6px;"));
00264     offsetZ = new QDoubleSpinBox(offset);
00265     offsetZ->setObjectName("offsetZ");
00266     offsetZ->setGeometry(QRect(154, 78, 141, 20));
00267     offsetZ->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00268 "\n"
00269 "border-radius: 6px;"));
00270     pushInput = new QPushButton(centralWidget);
00271     pushInput->setObjectName("pushInput");
00272     pushInput->setGeometry(QRect(240, 10, 101, 31));
00273     pushInput->setFont(font1);
00274     pushInput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00275 "border-radius: 10px;\n"
00276 "background-color: rgb(117, 125, 149);\n"
00277 "color: rgb(3, 8, 14);"));
00278     pushOutput = new QPushButton(centralWidget);
00279     pushOutput->setObjectName("pushOutput");
00280     pushOutput->setGeometry(QRect(240, 50, 101, 31));
00281     pushOutput->setFont(font1);
00282     pushOutput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00283 "border-radius: 10px;\n"
00284 "background-color: rgb(117, 125, 149);\n"
00285 "color: rgb(3, 8, 14);"));
00286     textBrowser = new QTextBrowser(centralWidget);
00287     textBrowser->setObjectName("textBrowser");
00288     textBrowser->setGeometry(QRect(0, 630, 355, 71));
00289     textBrowser->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00290 "border-radius: 0px;"));
00291     startCalculation = new QPushButton(centralWidget);
00292     startCalculation->setObjectName("startCalculation");
00293     startCalculation->setGeometry(QRect(10, 590, 331, 31));
00294     sizePolicy.setHeightForWidth(startCalculation->sizePolicy().hasHeightForWidth());
00295     startCalculation->setSizePolicy(sizePolicy);
00296     QFont font3;
00297     font3.setFamilies({QString::fromUtf8("Rubik")});
00298     font3.setPointSize(10);
00299     font3.setBold(false);
00300     font3.setItalic(false);
00301     startCalculation->setFont(font3);
00302     startCalculation->setStyleSheet(QString::fromUtf8("border: 1px solid black;\n"
00303 "border-radius: 10px;\n"
00304 "background-color: rgb(67, 72, 91);\n"
00305 "color: rgb(210, 211, 218)"));
00306     pathOutput = new QLabel(centralWidget);
00307     pathOutput->setObjectName("pathOutput");
00308     pathOutput->setGeometry(QRect(10, 50, 331, 31));
00309     pathOutput->setFont(font1);
00310     pathOutput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00311 "border-radius: 10px;\n"
00312 "background-color: rgb(210, 211, 218);\n"
00313 "color: rgb(3, 8, 14);"));
00314     pathOutput->setTextFormat(Qt::MarkdownText);
00315     GUIClass->setCentralWidget(centralWidget);
00316     pathInput->raise();
00317     frame->raise();
00318     textBrowser->raise();
00319     startCalculation->raise();
00320     pathOutput->raise();
00321     pushOutput->raise();
00322     pushInput->raise();
00323
00324     retranslateUi(GUIClass);
00325
00326     QMetaObject::connectSlotsByName(GUIClass);
00327 } // setupUi
00328
00329 void retranslateUi(QMainWindow *GUIClass)
00330 {
00331     GUIClass->setWindowTitle(QCoreApplication::translate("GUIClass", "Roboter Pfad Editor",
00332 nullptr));
00333     pathInput->setText(QCoreApplication::translate("GUIClass", "Eingabedatei", nullptr));
00334     speed->setSuffix(QCoreApplication::translate("GUIClass", " m/s", nullptr));
00335     label_5->setText(QCoreApplication::translate("GUIClass", "Geschwindigkeit", nullptr));
00336     bSpeed->setText(QCoreApplication::translate("GUIClass", "Manuelle Geschwindigkeit", nullptr));
00337     label_10->setText(QCoreApplication::translate("GUIClass", "Phi", nullptr));
00338     label_11->setText(QCoreApplication::translate("GUIClass", "Theta", nullptr));
00339     label_12->setText(QCoreApplication::translate("GUIClass", "Psi", nullptr));
00340     bManOrientation->setText(QCoreApplication::translate("GUIClass", "Manuelle Ausrichtung",
00341 nullptr));
00342     label_dp->setText(QCoreApplication::translate("GUIClass", "Douglas-Peuker-Toleranz",
00343 nullptr));

```



```

00341     label_4->setText(QCoreApplication::translate("GUIClass", "Filterlaenge gleitender Mittelwert",
    nullptr));
00342     dpToleranz->setSpecialValueText(QString());
00343     dpToleranz->setSuffix(QString());
00344     bLogging->setText(QCoreApplication::translate("GUIClass", "Detailliertes Logging", nullptr));
00345     bOffset->setText(QCoreApplication::translate("GUIClass", "Einstellung Offset", nullptr));
00346     label_13->setText(QCoreApplication::translate("GUIClass", "X", nullptr));
00347     label_14->setText(QCoreApplication::translate("GUIClass", "Y", nullptr));
00348     label_15->setText(QCoreApplication::translate("GUIClass", "Z", nullptr));
00349     pushInput->setText(QCoreApplication::translate("GUIClass", "Datei waehlen", nullptr));
00350     pushOutput->setText(QCoreApplication::translate("GUIClass", "Pfad waehlen", nullptr));
00351     startCalculation->setText(QCoreApplication::translate("GUIClass", "Datei erstellen",
    nullptr));
00352     pathOutput->setText(QCoreApplication::translate("GUIClass", "Ausgabeordner", nullptr));
00353     } // retranslateUi
00354
00355 };
00356
00357 namespace Ui {
00358     class GUIClass: public Ui_GUIClass {};
00359 } // namespace Ui
00360
00361 QT_END_NAMESPACE
00362
00363 #endif // UI_GUI_H

```

8.57 x64/Release/uic/ui_GUI.h-Dateireferenz

```

#include <QtCore/QVariant>
#include <QtWidgets/QApplication>
#include <QtWidgets/QCheckBox>
#include <QtWidgets/QDoubleSpinBox>
#include <QtWidgets/QFrame>
#include <QtWidgets/QLabel>
#include <QtWidgets/QMainWindow>
#include <QtWidgets/QPushButton>
#include <QtWidgets/QSpinBox>
#include <QtWidgets/QTextBrowser>
#include <QtWidgets/QWidget>

```

Klassen

- class [Ui_GUIClass](#)
- class [Ui::GUIClass](#)

Namensbereiche

- namespace [Ui](#)

8.58 ui_GUI.h

[gehe zur Dokumentation dieser Datei](#)

```

00001 /*****
00002 ** Form generated from reading UI file 'GUI.ui'
00003 **
00004 ** Created by: Qt User Interface Compiler version 6.5.1
00005 **
00006 ** WARNING! All changes made in this file will be lost when recompiling UI file!
00007 *****/
00008

```

```

00009 #ifndef UI_GUI_H
00010 #define UI_GUI_H
00011
00012 #include <QtCore/QVariant>
00013 #include <QtWidgets/QApplication>
00014 #include <QtWidgets/QCheckBox>
00015 #include <QtWidgets/QDoubleSpinBox>
00016 #include <QtWidgets/QFrame>
00017 #include <QtWidgets/QLabel>
00018 #include <QtWidgets/QMainWindow>
00019 #include <QtWidgets/QPushButton>
00020 #include <QtWidgets/QSpinBox>
00021 #include <QtWidgets/QTextBrowser>
00022 #include <QtWidgets/QWidget>
00023
00024 QT_BEGIN_NAMESPACE
00025
00026 class Ui_GUIClass
00027 {
00028 public:
00029     QWidget *centralWidget;
00030     QLabel *pathInput;
00031     QFrame *frame;
00032     QWidget *speed_2;
00033     QDoubleSpinBox *speed;
00034     QLabel *label_5;
00035     QCheckBox *bSpeed;
00036     QWidget *orientation;
00037     QLabel *label_10;
00038     QLabel *label_11;
00039     QLabel *label_12;
00040     QDoubleSpinBox *AValue;
00041     QDoubleSpinBox *BValue;
00042     QDoubleSpinBox *CValue;
00043     QCheckBox *bManOrientation;
00044     QSpinBox *meanLength;
00045     QLabel *label_dp;
00046     QLabel *label_4;
00047     QSpinBox *dpToleranz;
00048     QCheckBox *bLogging;
00049     QCheckBox *bOffset;
00050     QWidget *offset;
00051     QLabel *label_13;
00052     QLabel *label_14;
00053     QLabel *label_15;
00054     QDoubleSpinBox *offsetX;
00055     QDoubleSpinBox *offsetY;
00056     QDoubleSpinBox *offsetZ;
00057     QPushButton *pushInput;
00058     QPushButton *pushOutput;
00059     QTextBrowser *textBrowser;
00060     QPushButton *startCalculation;
00061     QLabel *pathOutput;
00062
00063     void setupUi(QMainWindow *GUIClass)
00064     {
00065         if (GUIClass->objectName().isEmpty())
00066             GUIClass->setObjectName("GUIClass");
00067         GUIClass->resize(355, 700);
00068         QSizePolicy sizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);
00069         sizePolicy.setHorizontalStretch(0);
00070         sizePolicy.setVerticalStretch(0);
00071         sizePolicy.setHeightForWidth(GUIClass->sizePolicy().hasHeightForWidth());
00072         GUIClass->setSizePolicy(sizePolicy);
00073         GUIClass->setMinimumSize(QSize(355, 530));
00074         GUIClass->setMaximumSize(QSize(355, 700));
00075         QFont font;
00076         font.setFamilies({QString::fromUtf8("Rubik")});
00077         GUIClass->setFont(font);
00078         GUIClass->setAnimated(true);
00079         GUIClass->setTabShape(QTabWidget::Rounded);
00080         GUIClass->setUnifiedTitleAndToolBarOnMac(false);
00081         centralWidget = new QWidget(GUIClass);
00082         centralWidget->setObjectName("centralWidget");
00083         QFont font1;
00084         font1.setFamilies({QString::fromUtf8("Rubik")});
00085         font1.setBold(false);
00086         font1.setItalic(false);
00087         centralWidget->setFont(font1);
00088         centralWidget->setStyleSheet(QString::fromUtf8("background-color: rgb(3, 8, 14);\n"
00089 "color: rgb(3, 8, 14)\n"
00090 "));
00091         pathInput = new QLabel(centralWidget);
00092         pathInput->setObjectName("pathInput");
00093         pathInput->setGeometry(QRect(10, 10, 331, 31));
00094         pathInput->setFont(font1);
00095         pathInput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"

```

```

00096 "border-radius: 10px;\n"
00097 "background-color: rgb(210, 211, 218);\n"
00098 "color: rgb(3, 8, 14);");
00099     frame = new QFrame(centralWidget);
00100     frame->setObjectName("frame");
00101     frame->setGeometry(QRect(10, 90, 331, 491));
00102     frame->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00103 "border-radius: 10px;\n"
00104 "background-color: rgb(117, 125, 149)"));
00105     frame->setFrameShape(QFrame::Box);
00106     frame->setFrameShadow(QFrame::Raised);
00107     frame->setLineWidth(1);
00108     speed_2 = new QWidget(frame);
00109     speed_2->setObjectName("speed_2");
00110     speed_2->setEnabled(false);
00111     speed_2->setGeometry(QRect(15, 110, 301, 41));
00112     QFont font2;
00113     font2.setFamilies({QString::fromUtf8("Rubik")});
00114     font2.setBold(true);
00115     speed_2->setFont(font2);
00116     speed_2->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00117 "color: rgb(117, 125, 149);\n"
00118 "border: 1px solid rgb(67, 72, 91); ");
00119     speed = new QDoubleSpinBox(speed_2);
00120     speed->setObjectName("speed");
00121     speed->setGeometry(QRect(159, 12, 136, 20));
00122     speed->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00123 "\n"
00124 "border-radius: 6px;"));
00125     speed->setAlignment(Qt::AlignCenter);
00126     label_5 = new QLabel(speed_2);
00127     label_5->setObjectName("label_5");
00128     label_5->setGeometry(QRect(11, 12, 93, 16));
00129     label_5->setFont(font1);
00130     label_5->setStyleSheet(QString::fromUtf8("border: 0px"));
00131     bSpeed = new QCheckBox(frame);
00132     bSpeed->setObjectName("bSpeed");
00133     bSpeed->setGeometry(QRect(15, 80, 171, 20));
00134     bSpeed->setFont(font1);
00135     bSpeed->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00136 "border: 0px"));
00137     orientation = new QWidget(frame);
00138     orientation->setObjectName("orientation");
00139     orientation->setEnabled(false);
00140     orientation->setGeometry(QRect(15, 190, 301, 111));
00141     orientation->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00142 "color: rgb(117, 125, 149);\n"
00143 "border: 1px solid rgb(67, 72, 91); ");
00144     label_10 = new QLabel(orientation);
00145     label_10->setObjectName("label_10");
00146     label_10->setGeometry(QRect(10, 15, 18, 16));
00147     label_10->setFont(font1);
00148     label_10->setStyleSheet(QString::fromUtf8("border: 0px"));
00149     label_11 = new QLabel(orientation);
00150     label_11->setObjectName("label_11");
00151     label_11->setGeometry(QRect(11, 47, 33, 16));
00152     label_11->setFont(font1);
00153     label_11->setStyleSheet(QString::fromUtf8("\n"
00154 "border: 0px"));
00155     label_12 = new QLabel(orientation);
00156     label_12->setObjectName("label_12");
00157     label_12->setGeometry(QRect(11, 78, 17, 16));
00158     label_12->setFont(font1);
00159     label_12->setStyleSheet(QString::fromUtf8("border: 0px"));
00160     AValue = new QDoubleSpinBox(orientation);
00161     AValue->setObjectName("AValue");
00162     AValue->setGeometry(QRect(153, 15, 141, 20));
00163     AValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00164 "\n"
00165 "border-radius: 6px;"));
00166     BValue = new QDoubleSpinBox(orientation);
00167     BValue->setObjectName("BValue");
00168     BValue->setGeometry(QRect(154, 47, 141, 20));
00169     BValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00170 "\n"
00171 "border-radius: 6px;"));
00172     CValue = new QDoubleSpinBox(orientation);
00173     CValue->setObjectName("CValue");
00174     CValue->setGeometry(QRect(154, 78, 141, 20));
00175     CValue->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00176 "\n"
00177 "border-radius: 6px;"));
00178     label_12->raise();
00179     label_10->raise();
00180     AValue->raise();
00181     BValue->raise();
00182     CValue->raise();

```

```

00183         label_11->raise();
00184         bManOrientation = new QCheckBox(frame);
00185         bManOrientation->setObjectName("bManOrientation");
00186         bManOrientation->setGeometry(QRect(15, 160, 151, 20));
00187         bManOrientation->setFont(font1);
00188         bManOrientation->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00189 "border: 0px"));
00190         meanLength = new QSpinBox(frame);
00191         meanLength->setObjectName("meanLength");
00192         meanLength->setGeometry(QRect(200, 48, 116, 20));
00193         meanLength->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00194 "\n"
00195 "border-radius: 6px;"));
00196         label_dp = new QLabel(frame);
00197         label_dp->setObjectName("label_dp");
00198         label_dp->setGeometry(QRect(14, 22, 144, 16));
00199         label_dp->setFont(font1);
00200         label_dp->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00201 "border: 0px"));
00202         label_4 = new QLabel(frame);
00203         label_4->setObjectName("label_4");
00204         label_4->setGeometry(QRect(14, 48, 180, 16));
00205         label_4->setFont(font1);
00206         label_4->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00207 "border: 0px"));
00208         dpToleranz = new QSpinBox(frame);
00209         dpToleranz->setObjectName("dpToleranz");
00210         dpToleranz->setGeometry(QRect(200, 22, 116, 20));
00211         dpToleranz->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00212 "\n"
00213 "border-radius: 6px;"));
00214         dpToleranz->setFrame(true);
00215         dpToleranz->setButtonSymbols(QAbstractSpinBox::UpDownArrows);
00216         dpToleranz->setAccelerated(false);
00217         bLogging = new QCheckBox(frame);
00218         bLogging->setObjectName("bLogging");
00219         bLogging->setGeometry(QRect(15, 460, 151, 20));
00220         bLogging->setFont(font1);
00221         bLogging->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00222 "border: 0px"));
00223         bOffset = new QCheckBox(frame);
00224         bOffset->setObjectName("bOffset");
00225         bOffset->setGeometry(QRect(15, 310, 151, 20));
00226         bOffset->setFont(font1);
00227         bOffset->setStyleSheet(QString::fromUtf8("color: rgb(3, 8, 14);\n"
00228 "border: 0px"));
00229         offset = new QWidget(frame);
00230         offset->setObjectName("offset");
00231         offset->setEnabled(false);
00232         offset->setGeometry(QRect(15, 340, 301, 111));
00233         offset->setStyleSheet(QString::fromUtf8("background-color: rgb(210,211,218);\n"
00234 "color: rgb(117, 125, 149);\n"
00235 "border: 1px solid rgb(67, 72, 91);"));
00236         label_13 = new QLabel(offset);
00237         label_13->setObjectName("label_13");
00238         label_13->setGeometry(QRect(10, 15, 18, 16));
00239         label_13->setFont(font1);
00240         label_13->setStyleSheet(QString::fromUtf8("border: 0px"));
00241         label_14 = new QLabel(offset);
00242         label_14->setObjectName("label_14");
00243         label_14->setGeometry(QRect(11, 47, 33, 16));
00244         label_14->setFont(font1);
00245         label_14->setStyleSheet(QString::fromUtf8("\n"
00246 "border: 0px"));
00247         label_15 = new QLabel(offset);
00248         label_15->setObjectName("label_15");
00249         label_15->setGeometry(QRect(11, 78, 17, 16));
00250         label_15->setFont(font1);
00251         label_15->setStyleSheet(QString::fromUtf8("border: 0px"));
00252         offsetX = new QDoubleSpinBox(offset);
00253         offsetX->setObjectName("offsetX");
00254         offsetX->setGeometry(QRect(153, 15, 141, 20));
00255         offsetX->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00256 "\n"
00257 "border-radius: 6px;"));
00258         offsetY = new QDoubleSpinBox(offset);
00259         offsetY->setObjectName("offsetY");
00260         offsetY->setGeometry(QRect(154, 47, 141, 20));
00261         offsetY->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00262 "\n"
00263 "border-radius: 6px;"));
00264         offsetZ = new QDoubleSpinBox(offset);
00265         offsetZ->setObjectName("offsetZ");
00266         offsetZ->setGeometry(QRect(154, 78, 141, 20));
00267         offsetZ->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00268 "\n"
00269 "border-radius: 6px;"));

```

```

00270     pushInput = new QPushButton(centralWidget);
00271     pushInput->setObjectName("pushInput");
00272     pushInput->setGeometry(QRect(240, 10, 101, 31));
00273     pushInput->setFont(font1);
00274     pushInput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00275 "border-radius: 10px;\n"
00276 "background-color: rgb(117, 125, 149);\n"
00277 "color: rgb(3, 8, 14);"));
00278     pushOutput = new QPushButton(centralWidget);
00279     pushOutput->setObjectName("pushOutput");
00280     pushOutput->setGeometry(QRect(240, 50, 101, 31));
00281     pushOutput->setFont(font1);
00282     pushOutput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00283 "border-radius: 10px;\n"
00284 "background-color: rgb(117, 125, 149);\n"
00285 "color: rgb(3, 8, 14);"));
00286     textBrowser = new QTextBrowser(centralWidget);
00287     textBrowser->setObjectName("textBrowser");
00288     textBrowser->setGeometry(QRect(0, 630, 355, 71));
00289     textBrowser->setStyleSheet(QString::fromUtf8("background-color: rgb(210, 211, 218);\n"
00290 "border-radius: 0px;"));
00291     startCalculation = new QPushButton(centralWidget);
00292     startCalculation->setObjectName("startCalculation");
00293     startCalculation->setGeometry(QRect(10, 590, 331, 31));
00294     sizePolicy.setHeightForWidth(startCalculation->sizePolicy().hasHeightForWidth());
00295     startCalculation->setSizePolicy(sizePolicy);
00296     QFont font3;
00297     font3.setFamilies({QString::fromUtf8("Rubik")});
00298     font3.setPointSize(10);
00299     font3.setBold(false);
00300     font3.setItalic(false);
00301     startCalculation->setFont(font3);
00302     startCalculation->setStyleSheet(QString::fromUtf8("border: 1px solid black;\n"
00303 "border-radius: 10px;\n"
00304 "background-color: rgb(67, 72, 91);\n"
00305 "color: rgb(210, 211, 218);"));
00306     pathOutput = new QLabel(centralWidget);
00307     pathOutput->setObjectName("pathOutput");
00308     pathOutput->setGeometry(QRect(10, 50, 331, 31));
00309     pathOutput->setFont(font1);
00310     pathOutput->setStyleSheet(QString::fromUtf8("border: 1px solid black; \n"
00311 "border-radius: 10px;\n"
00312 "background-color: rgb(210, 211, 218);\n"
00313 "color: rgb(3, 8, 14);"));
00314     pathOutput->setTextFormat(Qt::MarkdownText);
00315     GUIClass->setCentralWidget(centralWidget);
00316     pathInput->raise();
00317     frame->raise();
00318     textBrowser->raise();
00319     startCalculation->raise();
00320     pathOutput->raise();
00321     pushOutput->raise();
00322     pushInput->raise();
00323
00324     retranslateUi(GUIClass);
00325
00326     QMetaObject::connectSlotsByName(GUIClass);
00327 } // setupUi
00328
00329 void retranslateUi(QMainWindow *GUIClass)
00330 {
00331     GUIClass->setWindowTitle(QCoreApplication::translate("GUIClass", "Roboter Pfad Editor",
00332 nullptr));
00333     pathInput->setText(QCoreApplication::translate("GUIClass", "Eingabedatei", nullptr));
00334     speed->setSuffix(QCoreApplication::translate("GUIClass", " m/s", nullptr));
00335     label_5->setText(QCoreApplication::translate("GUIClass", "Geschwindigkeit", nullptr));
00336     bSpeed->setText(QCoreApplication::translate("GUIClass", "Manuelle Geschwindigkeit", nullptr));
00337     label_10->setText(QCoreApplication::translate("GUIClass", "A", nullptr));
00338     label_11->setText(QCoreApplication::translate("GUIClass", "B", nullptr));
00339     label_12->setText(QCoreApplication::translate("GUIClass", "C", nullptr));
00340     bManOrientation->setText(QCoreApplication::translate("GUIClass", "Manuelle Ausrichtung",
00341 nullptr));
00342     label_dp->setText(QCoreApplication::translate("GUIClass", "Douglas-Peuker-Toleranz",
00343 nullptr));
00344     label_4->setText(QCoreApplication::translate("GUIClass", "Filterlaenge gleitender Mittelwert",
00345 nullptr));
00346     dpToleranz->setSpecialValueText(QString());
00347     dpToleranz->setSuffix(QString());
00348     bLogging->setText(QCoreApplication::translate("GUIClass", "Detailliertes Logging", nullptr));
00349     bOffset->setText(QCoreApplication::translate("GUIClass", "Einstellung Offset", nullptr));
00350     label_13->setText(QCoreApplication::translate("GUIClass", "X", nullptr));
00351     label_14->setText(QCoreApplication::translate("GUIClass", "Y", nullptr));
00352     label_15->setText(QCoreApplication::translate("GUIClass", "Z", nullptr));
00353     pushInput->setText(QCoreApplication::translate("GUIClass", "Datei waehlen", nullptr));
00354     pushOutput->setText(QCoreApplication::translate("GUIClass", "Pfad waehlen", nullptr));
00355     startCalculation->setText(QCoreApplication::translate("GUIClass", "Datei erstellen",
00356 nullptr));

```

```
00352         pathOutput->setText (QCoreApplication::translate("GUIClass", "Ausgabeordner", nullptr));
00353     } // retranslateUi
00354
00355 };
00356
00357 namespace Ui {
00358     class GUIClass: public Ui_GUIClass {};
00359 } // namespace Ui
00360
00361 QT_END_NAMESPACE
00362
00363 #endif // UI_GUI_H
```

Index

- _USE_MATH_DEFINES
 - EulerMatrix.h, [105](#)
 - ~CEulerMatrix
 - CEulerMatrix, [14](#)
 - ~CInputParameter
 - CInputParameter, [22](#)
 - ~CInputPoint3D
 - CInputPoint3D, [33](#)
 - ~CLine3D
 - CLine3D, [37](#)
 - ~CLogging
 - CLogging, [40](#)
 - ~CMeanFilter
 - CMeanFilter, [45](#)
 - ~COutputPoint3D
 - COutputPoint3D, [51](#)
 - ~CPathBuilder
 - CPathBuilder, [56](#)
 - ~CPoint3D
 - CPoint3D, [60](#)
 - ~CRobCodeGenerator
 - CRobCodeGenerator, [67](#)
 - ~CSegmentApproximator
 - CSegmentApproximator, [72](#)
 - ~GUI
 - GUI, [78](#)
- A
 - CInputParameter, [28](#)
- a
 - COutputPoint3D, [54](#)
- activateLogging
 - GUI, [78](#)
- activateOffset
 - GUI, [78](#)
- activateOrientation
 - GUI, [79](#)
- activateSpeed
 - GUI, [79](#)
- angels2mat
 - CEulerMatrix, [15](#)
- approx
 - CSegmentApproximator, [72](#)
- AValue
 - Ui_GUIClass, [98](#)
- B
 - CInputParameter, [28](#)
- b
 - COutputPoint3D, [54](#)

- Beschreibung Roboter Path Editor, [1](#)
- bLogging
 - Ui_GUIClass, [98](#)
- bManOrientation
 - Ui_GUIClass, [98](#)
- bOffset
 - Ui_GUIClass, [99](#)
- bSpeed
 - Ui_GUIClass, [99](#)
- BValue
 - Ui_GUIClass, [99](#)
- C
 - CInputParameter, [28](#)
- c
 - COutputPoint3D, [54](#)
- calculate
 - GUI, [80](#)
- calculateAngels
 - CEulerMatrix, [15](#)
- calculateAngles
 - CRobCodeGenerator, [67](#)
- calculateMean
 - CMeanFilter, [45](#)
- calculateSpeed
 - CRobCodeGenerator, [68](#)
- centralWidget
 - Ui_GUIClass, [99](#)
- CEulerMatrix, [13](#)
 - ~CEulerMatrix, [14](#)
 - angels2mat, [15](#)
 - calculateAngels, [15](#)
 - CEulerMatrix, [14](#)
 - eulerMatrix, [18](#)
 - getEulerMatrix, [16](#)
 - getMatrix, [16](#)
 - setMatrix, [18](#)
- CInputParameter, [19](#)
 - ~CInputParameter, [22](#)
 - A, [28](#)
 - B, [28](#)
 - C, [28](#)
 - CInputParameter, [20](#)
 - detectJump, [22](#)
 - difference, [29](#)
 - getAngles, [23](#)
 - getLoggingManual, [23](#)
 - getOffset, [23](#)
 - getOffsetManual, [24](#)
 - getOrientationManual, [24](#)

- getPath, 24
- getSpeed, 25
- getSpeedManual, 25
- initialPath, 29
- loggingManual, 29
- offsetManual, 29
- offsetX, 29
- offsetY, 30
- offsetZ, 30
- openFile, 25
- orientationManual, 30
- setLogging, 26
- setOffset, 27
- setOrientation, 27
- setSpeed, 28
- speed, 30
- speedManual, 30
- CInputPoint3D, 31
 - ~CInputPoint3D, 33
 - CInputPoint3D, 33
 - getEulerMatrix, 34
 - getTime, 34
 - orientationMatrix, 36
 - setEulerMatrix, 34
 - setPoint, 35
 - setTime, 35
 - timestamp, 36
- CLine3D, 36
 - ~CLine3D, 37
 - CLine3D, 37
 - p1, 38
 - p2, 38
- CLogging, 38
 - ~CLogging, 40
 - CLogging, 39
 - detailed, 43
 - getDetailed, 40
 - logData, 40–42
 - path, 43
 - setStep, 42
 - step, 43
- CMeanFilter, 44
 - ~CMeanFilter, 45
 - calculateMean, 45
 - CMeanFilter, 44
 - getPath, 46
 - getWindowSize, 46
 - mean, 46
 - meanPath, 48
 - setWindowSize, 47
 - windowSize, 48
- COutputPoint3D, 48
 - ~COutputPoint3D, 51
 - a, 54
 - b, 54
 - c, 54
 - COutputPoint3D, 50
 - getA, 51
 - getB, 51
 - getC, 52
 - getSpeed, 52
 - setA, 52
 - setB, 53
 - setC, 53
 - setSpeed, 53
 - speed, 54
- CPathBuilder, 55
 - ~CPathBuilder, 56
 - CPathBuilder, 56
 - createPath, 56
 - getPath, 57
 - path, 57
- CPoint3D, 58
 - ~CPoint3D, 60
 - CPoint3D, 59
 - distanceTo, 60, 61
 - getX, 61
 - getY, 61
 - getZ, 62
 - set, 62
 - setX, 63
 - setY, 63
 - setZ, 63
 - x, 65
 - y, 65
 - z, 65
- createPath
 - CPathBuilder, 56
- CRobCodeGenerator, 66
 - ~CRobCodeGenerator, 67
 - calculateAngles, 67
 - calculateSpeed, 68
 - CRobCodeGenerator, 66
 - generateRobCode, 68
 - input, 70
 - postProcessing, 70
 - processedPath, 71
- CSegmentApproximator, 71
 - ~CSegmentApproximator, 72
 - approx, 72
 - CSegmentApproximator, 72
 - douglasPeuckerRecursive, 73
 - getmaxDistance, 74
 - getSegmentsApproxVector, 74
 - maxDistance, 75
 - segmentsApprox, 75
 - setmaxDistance, 74
- CValue
 - Ui_GUIClass, 99
- detailed
 - CLogging, 43
- detectJump
 - CInputParameter, 22
- difference
 - CInputParameter, 29
- distanceTo

- CPoint3D, 60, 61
- douglasPeuckerRecursive
 - CSegmentApproximator, 73
- dpTolerance
 - GUI, 83
- dpToleranz
 - Ui_GUIClass, 99
- eulerMatrix
 - CEulerMatrix, 18
- EulerMatrix.h
 - _USE_MATH_DEFINES, 105
- frame
 - Ui_GUIClass, 100
- generateRobCode
 - CRobCodeGenerator, 68
- getA
 - COutputPoint3D, 51
- getAngles
 - CInputParameter, 23
- getB
 - COutputPoint3D, 51
- getC
 - COutputPoint3D, 52
- getDetailed
 - CLogging, 40
- getEulerMatrix
 - CEulerMatrix, 16
 - CInputPoint3D, 34
- getLoggingManual
 - CInputParameter, 23
- getMatrix
 - CEulerMatrix, 16
- getmaxDistance
 - CSegmentApproximator, 74
- getOffset
 - CInputParameter, 23
- getOffsetManual
 - CInputParameter, 24
- getOrientationManual
 - CInputParameter, 24
- getPath
 - CInputParameter, 24
 - CMeanFilter, 46
 - CPathBuilder, 57
- getSegmentsApproxVector
 - CSegmentApproximator, 74
- getSpeed
 - CInputParameter, 25
 - COutputPoint3D, 52
- getSpeedManual
 - CInputParameter, 25
- getTime
 - CInputPoint3D, 34
- getWindowSize
 - CMeanFilter, 46
- getX
 - CPoint3D, 61
- getY
 - CPoint3D, 61
- getZ
 - CPoint3D, 62
- GUI, 76
 - ~GUI, 78
 - activateLogging, 78
 - activateOffset, 78
 - activateOrientation, 79
 - activateSpeed, 79
 - calculate, 80
 - dpTolerance, 83
 - GUI, 77
 - inputParameter, 83
 - inputPathUI, 83
 - meanLength, 84
 - outputPathUI, 84
 - setDP, 81
 - setInputPath, 81
 - setMean, 81
 - setOffset, 82
 - setOrientation, 82
 - setOutputPath, 82
 - setSpeed, 83
 - ui, 84
- header/EulerMatrix.h, 105, 106
- header/GUI.h, 106, 107
- header/InputParameter.h, 107, 108
- header/Line3D.h, 108, 109
- header/Logging.h, 109, 110
- header/MeanFilter.h, 110, 111
- header/PathBuilder.h, 111, 112
- header/Point3D.h, 112, 113
- header/RobCodeGenerator.h, 113, 114
- header/SegmentApproximator.h, 115
- initialPath
 - CInputParameter, 29
- input
 - CRobCodeGenerator, 70
- inputParameter
 - GUI, 83
- inputPathUI
 - GUI, 83
- label_10
 - Ui_GUIClass, 100
- label_11
 - Ui_GUIClass, 100
- label_12
 - Ui_GUIClass, 100
- label_13
 - Ui_GUIClass, 100
- label_14
 - Ui_GUIClass, 100
- label_15
 - Ui_GUIClass, 101

- label_4
 - Ui_GUIClass, [101](#)
- label_5
 - Ui_GUIClass, [101](#)
- label_dp
 - Ui_GUIClass, [101](#)
- logData
 - CLogging, [40–42](#)
- loggingManual
 - CInputParameter, [29](#)
- main
 - RobPathEditor.cpp, [133](#)
- MAX_SPEED
 - RobCodeGenerator.h, [114](#)
- maxDistance
 - CSegmentApproximator, [75](#)
- mean
 - CMeanFilter, [46](#)
- meanLength
 - GUI, [84](#)
 - Ui_GUIClass, [101](#)
- meanPath
 - CMeanFilter, [48](#)
- moc_GUI.cpp
 - Q_CONSTINIT, [136, 139](#)
 - QT_MOC_LITERAL, [136, 139](#)
- moc_switch.cpp
 - Q_CONSTINIT, [143, 146](#)
 - QT_MOC_LITERAL, [143, 146](#)
- offset
 - Ui_GUIClass, [101](#)
- offsetManual
 - CInputParameter, [29](#)
- offsetsAndSizes
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_GUIClass, [86](#)
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_GUIClass, [89](#)
- offsetX
 - CInputParameter, [29](#)
 - Ui_GUIClass, [102](#)
- offsetY
 - CInputParameter, [30](#)
 - Ui_GUIClass, [102](#)
- offsetZ
 - CInputParameter, [30](#)
 - Ui_GUIClass, [102](#)
- openFile
 - CInputParameter, [25](#)
- orientation
 - Ui_GUIClass, [102](#)
- orientationManual
 - CInputParameter, [30](#)
- orientationMatrix
 - CInputPoint3D, [36](#)
- outputPathUI
 - GUI, [84](#)
- p1
 - CLine3D, [38](#)
- p2
 - CLine3D, [38](#)
- path
 - CLogging, [43](#)
 - CPathBuilder, [57](#)
- pathInput
 - Ui_GUIClass, [102](#)
- pathOutput
 - Ui_GUIClass, [102](#)
- postProcessing
 - CRobCodeGenerator, [70](#)
- processedPath
 - CRobCodeGenerator, [71](#)
- pushInput
 - Ui_GUIClass, [103](#)
- pushOutput
 - Ui_GUIClass, [103](#)
- Q_CONSTINIT
 - moc_GUI.cpp, [136, 139](#)
 - moc_switch.cpp, [143, 146](#)
- qCleanupResources_GUI
 - qrc_GUI.cpp, [149, 151](#)
- qInitResources_GUI
 - qrc_GUI.cpp, [149, 151](#)
- qrc_GUI.cpp
 - qCleanupResources_GUI, [149, 151](#)
 - qInitResources_GUI, [149, 151](#)
 - QT_RCC_MANGLE_NAMESPACE, [149, 151](#)
 - QT_RCC_PREPEND_NAMESPACE, [149, 151](#)
- QT_MOC_LITERAL
 - moc_GUI.cpp, [136, 139](#)
 - moc_switch.cpp, [143, 146](#)
- QT_RCC_MANGLE_NAMESPACE
 - data_GUIClass, [149, 151](#)
- QT_RCC_PREPEND_NAMESPACE
 - data_GUIClass, [149, 151](#)
- QT_WARNING_DISABLE_DEPRECATED, [11](#)
- QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSSwitch, [86](#)
- offsetsAndSizes, [86](#)
- stringdata0, [86](#)
- stringdata1, [86](#)
- stringdata10, [87](#)
- stringdata11, [87](#)
- stringdata12, [87](#)
- stringdata13, [87](#)
- stringdata2, [87](#)
- stringdata3, [87](#)
- stringdata4, [87](#)
- stringdata5, [87](#)
- stringdata6, [88](#)
- stringdata7, [88](#)
- stringdata8, [88](#)
- stringdata9, [88](#)
- QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASSSSwitch, [88](#)

- offsetsAndSizes, [89](#)
- stringdata0, [89](#)
- stringdata1, [89](#)
- stringdata2, [89](#)
- retranslateUi
 - Ui_GUIClass, [91](#)
- RobCodeGenerator.h
 - MAX_SPEED, [114](#)
- RobPathEditor.cpp
 - main, [133](#)
- segmentsApprox
 - CSegmentApproximator, [75](#)
- set
 - CPoint3D, [62](#)
- setA
 - COutputPoint3D, [52](#)
- setB
 - COutputPoint3D, [53](#)
- setC
 - COutputPoint3D, [53](#)
- setDP
 - GUI, [81](#)
- setEulerMatrix
 - CInputPoint3D, [34](#)
- setInputPath
 - GUI, [81](#)
- setLogging
 - CInputParameter, [26](#)
- setMatrix
 - CEulerMatrix, [18](#)
- setMaxDistance
 - CSegmentApproximator, [74](#)
- setMean
 - GUI, [81](#)
- setOffset
 - CInputParameter, [27](#)
 - GUI, [82](#)
- setOrientation
 - CInputParameter, [27](#)
 - GUI, [82](#)
- setOutputPath
 - GUI, [82](#)
- setPoint
 - CInputPoint3D, [35](#)
- setSpeed
 - CInputParameter, [28](#)
 - COutputPoint3D, [53](#)
 - GUI, [83](#)
- setStep
 - CLogging, [42](#)
- setTime
 - CInputPoint3D, [35](#)
- setUpUi
 - Ui_GUIClass, [92](#), [95](#)
- setWindowSize
 - CMeanFilter, [47](#)
- setX

- CPoint3D, [63](#)
- setY
 - CPoint3D, [63](#)
- setZ
 - CPoint3D, [63](#)
- source/EulerMatrix.cpp, [116](#)
- source/GUI.cpp, [117](#), [118](#)
- source/InputParameter.cpp, [120](#), [121](#)
- source/Line3D.cpp, [123](#)
- source/Logging.cpp, [123](#), [124](#)
- source/MeanFilter.cpp, [125](#), [126](#)
- source/PathBuilder.cpp, [127](#)
- source/Point3D.cpp, [128](#)
- source/RobCodeGenerator.cpp, [131](#)
- source/RobPathEditor.cpp, [133](#)
- source/SegmentApproximator.cpp, [134](#)
- speed
 - CInputParameter, [30](#)
 - COutputPoint3D, [54](#)
 - Ui_GUIClass, [103](#)
- speed_2
 - Ui_GUIClass, [103](#)
- speedManual
 - CInputParameter, [30](#)
- startCalculation
 - Ui_GUIClass, [103](#)
- step
 - CLogging, [43](#)
- stringdata0
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[86](#)
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[89](#)
- stringdata1
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[86](#)
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[89](#)
- stringdata10
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[87](#)
- stringdata11
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[87](#)
- stringdata12
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[87](#)
- stringdata13
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[87](#)
- stringdata2
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[87](#)
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[89](#)
- stringdata3
 - QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLA
[87](#)

stringdata4
 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASS_GUIENDCLASS_t, 87
 stringdata5
 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASS_GUIENDCLASS_t, 87
 stringdata6
 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASS_GUIENDCLASS_t, 88
 stringdata7
 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASS_GUIENDCLASS_t, 88
 stringdata8
 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASS_GUIENDCLASS_t, 88
 stringdata9
 QT_WARNING_DISABLE_DEPRECATED::qt_meta_stringdata_CLASS_GUIENDCLASS_t, 88
 textBrowser
 Ui_GUIClass, 103
 timestamp
 CInputPoint3D, 36
 Ui, 11
 ui
 GUI, 84
 Ui::GUIClass, 85
 Ui_GUIClass, 89
 AValue, 98
 bLogging, 98
 bManOrientation, 98
 bOffset, 99
 bSpeed, 99
 BValue, 99
 centralWidget, 99
 CValue, 99
 dpToleranz, 99
 frame, 100
 label_10, 100
 label_11, 100
 label_12, 100
 label_13, 100
 label_14, 100
 label_15, 101
 label_4, 101
 label_5, 101
 label_dp, 101
 meanLength, 101
 offset, 101
 offsetX, 102
 offsetY, 102
 offsetZ, 102
 orientation, 102
 pathInput, 102
 pathOutput, 102
 pushInput, 103
 pushOutput, 103
 retranslateUi, 91
 setupUi, 92, 95
 stateClass, 103
 speed_2, 103
 startCalculation, 103
 textBrowser, 103
 windowSize
 CMainWindow, 48
 COpenGLWidget, 48
 x
 CPoint3D, 65
 x64/Debug/moc/moc_GUI.cpp, 135, 136
 x64/Debug/moc/moc_switch.cpp, 143, 144
 x64/Debug/rcc/qrc_GUI.cpp, 148, 150
 x64/Debug/uic/ui_GUI.h, 152, 153
 x64/Release/moc/moc_GUI.cpp, 139, 140
 x64/Release/moc/moc_switch.cpp, 146, 147
 x64/Release/rcc/qrc_GUI.cpp, 150, 151
 x64/Release/uic/ui_GUI.h, 157
 y
 CPoint3D, 65
 z
 CPoint3D, 65