RobCodeGenerator

Erzeugt von Doxygen 1.9.7

Kapitel 1

Hierarchie-Verzeichnis

1.1 Klassenhierarchie

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

Kapitel 2

Klassen-Verzeichnis

2.1 Auflistung der Klassen

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

Kapitel 3

Datei-Verzeichnis

3.1 Auflistung der Dateien

Hier folgt die Aufzählung aller dokumentierten Dateien mit einer Kurzbeschreibung:

Kapitel 4

Klassen-Dokumentation

4.1 CEulerMatrix Klassenreferenz

Öffentliche Methoden

- **CEulerMatrix** (float inputMatrix[3][3])
- void **setMatrix** (float inputMatrix[3][3])
- CEulerMatrix **getMatrix** ()
- CEulerMatrix **calculatAngel** (double A, double B, double C)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- EulerMatrix.h
- EulerMatrix.cpp

4.2 CGUI Klassenreferenz

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- GUI.h
- GUI.cpp

4.3 CInputParameter Klassenreferenz

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

```
#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
- double getSpeed (void)
  : Gibt Geschwindigkeit zurck
- bool getSpeedManual (void)
  : Gibt zurck ob hische Geschwindigkeit verwendet werden soll
- bool getOrientationManual (void)
  : Gibt zurck ob hische Orientierung verwendet werden soll
- tuple< double, double, double > getAngles (void)
  : Gibt Winkel zurck
- 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 Sprnge in den Daten
- vector< list< CInputPoint3D > > & getPath ()
  : Gibt Pfad zurck

4.3.1 Ausführliche Beschreibung

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

4.3.2 Beschreibung der Konstruktoren und Destruktoren

4.3.2.1 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)

Initialiserung der Klasse mit 0

Siehe auch

CInputParameter(double initSpeed, bool initSpeedManual, bool initOrientationManual, double initA, double initB, double initC)

~CInputParameter()

4.3.2.2 CInputParameter() [2/2]

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

: Konstruktor mit Werten

Initialisiert die Input Daten

Siehe auch

: CInputParameter(void);

Initialiserung der Klasse mit allen Werten

Parameter

initSpeed	double
initSpeedManual	bool
initOrientationManual	bool
initA	double
initB	double
initC	double

Siehe auch

CInputParameter()
~CInputParameter()

4.3.2.3 ~CInputParameter()

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

: Dekonstruktor

Initialisiert die Input Daten

Dekonstruktor

Siehe auch

CInputParameter(double initSpeed, bool initSpeedManual, bool initOrientationManual, double initA, double initB, double initC)
CInputParameter()

4.3.3 Dokumentation der Elementfunktionen

4.3.3.1 detectJump()

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

: Erkennt Sprnge in den Daten

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

4.3.3.2 getAngles()

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

: Gibt Winkel zurck

Gibt die eingegebenen Winkel als tuple zurck

4.3.3.3 getOrientationManual()

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

: Gibt zurck ob hische Orientierung verwendet werden soll

Gibt zurck ob hische Orientierung verwendet werden soll, sonst wird sie spr berechnet.

4.3.3.4 getPath()

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

: Gibt Pfad zurck

Gibt die eingegebenen Winkel als tuple zurck

4.3.3.5 getSpeed()

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

: Gibt Geschwindigkeit zurck

Gibt die eingegebene Geschwindigkeit zurck

4.3.3.6 getSpeedManual()

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

: Gibt zurck ob hische Geschwindigkeit verwendet werden soll

Gibt zurck ob hische Geschwindigekit verwendet werden soll, sonst wird sie spr berechnet.

4.3.3.7 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.

4.3.3.8 setOrientation()

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

: Setzt Orientierungs Daten

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

4.3.3.9 setSpeed()

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

: Setzt Geschwindigkeits Daten

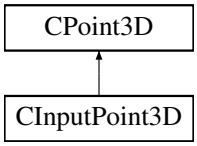
Setzt ob die Geschwindigkeit Hisch eingegeben werden soll und die Geschwindigkeit in m/s

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- InputParameter.h
- InputParameter.cpp

4.4 CInputPoint3D Klassenreferenz

Klassendiagramm für CInputPoint3D:



Öffentliche Methoden

- **CInputPoint3D** (double X, double Y, double Z, double Timestamp, CEulerMatrix Matrix)
- double **getTime** ()
- CEulerMatrix **getEulerMatrix** ()
- void **setTime** (double time)
- void **setEulerMatrix** (CEulerMatrix orientation)
- void **setPoint** (double time, double X, double Y, double Z, CEulerMatrix orientation)

Öffentliche Methoden geerbt von CPoint3D

- **CPoint3D** (double X, double Y, double Z)
- double **getX** ()
- double **getY** ()
- double **getZ** ()
- void **setX** (double X)
- void **setY** (double Y)
- void **setZ** (double Z)
- void **set** (double X, double Y, double Z)
- double **distanceTo** (CPoint3D point)
- double **distanceTo** (CLine3D line)

Weitere Geerbte Elemente

Geschützte Attribute geerbt von CPoint3D

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

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Point3D.h
- Point3D.cpp

4.5 CLine3D Klassenreferenz

Öffentliche Methoden

- **CLine3D** (CPoint3D P1, CPoint3D P2)

Öffentliche Attribute

- CPoint3D **p1**
- CPoint3D **p2**

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Line3D.h
- Line3D.cpp

4.6 CMeanFilter Klassenreferenz

Öffentliche Methoden

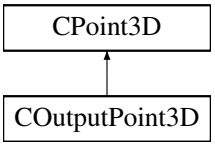
- **CMeanFilter** (int Window, long lenth)
- void **setWindowSize** (int Window)
- int **getWindowSize** ()
- vector< list< CInputPoint3D > > & **getPath** ()
- list< CInputPoint3D > **calculateMean** (list< CInputPoint3D > &segment)
- void **mean** (vector< list< CInputPoint3D > > &sourcePath)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- MeanFilter.h
- MeanFilter.cpp

4.7 COutputPoint3D Klassenreferenz

Klassendiagramm für COutputPoint3D:



Öffentliche Methoden

- **COutputPoint3D** (double Speed, double X, double Y, double Z, double A, double B, double C)
- double **getSpeed** ()
- double **getA** ()
- double **getB** ()
- double **getC** ()
- void **setSpeed** (double speed)
- void **setA** (double A)
- void **setB** (double B)
- void **setC** (double C)

Öffentliche Methoden geerbt von CPoint3D

- **CPoint3D** (double X, double Y, double Z)
- double **getX** ()
- double **getY** ()
- double **getZ** ()
- void **setX** (double X)
- void **setY** (double Y)
- void **setZ** (double Z)
- void **set** (double X, double Y, double Z)
- double **distanceTo** (CPoint3D point)
- double **distanceTo** (CLine3D line)

Weitere Geerbte Elemente

Geschützte Attribute geerbt von CPoint3D

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

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Point3D.h
- Point3D.cpp

4.8 CPathBuilder Klassenreferenz

Öffentliche Methoden

- vector< CInputPoint3D > & **getPath** ()
- void **createPath** (vector< list< CInputPoint3D > > &segments, string filename)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- PathBuilder.h
- PathBuilder.cpp

4.9 CPathPostProcessing Klassenreferenz

Öffentliche Methoden

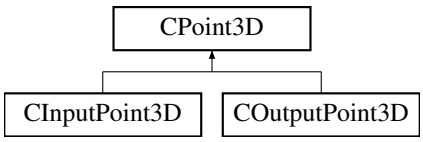
- **CPathPostProcessing** (double speedIn, bool speedManualIn, bool orientationManualIn, double AIn, double BIn, double CIn)
- vector< CPoint3D > & **getProcessedPath** (void)
- void **postProcessing** (vector< CPoint3D > &path)
- void **calculateSpeed** (COutputPoint3D &p, size_t i)
- void **calculateAngles** (COutputPoint3D &p)
- void **setData** (double speed, bool speedManual, bool orientationManual, tuple< double, double, double > angles)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- PathPostProcessing.h
- PathPostProcessing.cpp

4.10 CPoint3D Klassenreferenz

Klassendiagramm für CPoint3D:



Öffentliche Methoden

- **CPoint3D** (double X, double Y, double Z)
- double **getX** ()
- double **getY** ()
- double **getZ** ()
- void **setX** (double X)
- void **setY** (double Y)
- void **setZ** (double Z)
- void **set** (double X, double Y, double Z)
- double **distanceTo** (CPoint3D point)
- double **distanceTo** (CLine3D line)

Geschützte Attribute

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

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Point3D.h
- Point3D.cpp

4.11 CRobCodeGenerator Klassenreferenz

Öffentliche Methoden

- **CRobCodeGenerator** (double speedIn, bool speedManualIn, bool orientationManualIn, tuple< double, double, double > angles)
- void **generateRobCode** (vector< CInputPoint3D > &path, string filename)
- void **postProcessing** (vector< CInputPoint3D > &path)
- double **calculateSpeed** (CInputPoint3D &p, size_t i, double timePrev)
- void **calculateAngles** (COutputPoint3D &p, CInputPoint3D &pIn)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- RobCodeGenerator.h
- RobCodeGenerator.cpp

4.12 CSegmentApproximator Klassenreferenz

Öffentliche Methoden

- void **approx** (const vector< list< CInputPoint3D > > &Segments)
- void **setmaxDistance** (double maxDistanceSource)
- double **getmaxDistance** ()
- vector< list< CInputPoint3D > > & **getSegmentsApproxVector** ()

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- SegmentApproximator.h
- SegmentApproximator.cpp

Kapitel 5

Datei-Dokumentation

5.1 EulerMatrix.h

```
00001 using namespace std;
00002
00003 #pragma once
00004 class CEulerMatrix
00005 {
00006 public:
00007     CEulerMatrix(void);
00008     CEulerMatrix(float inputMatrix[3][3]);
00009     ~CEulerMatrix();
00010
00011     void setMatrix(float inputMatrix[3][3]);
00012     CEulerMatrix getMatrix();
00013
00014     CEulerMatrix calculatAngel(double A, double B, double C);
00015
00016 private:
00017     float eulerMatrix[3][3];
00018 };
00019
```

5.2 GUI.h

```
00001 #pragma once
00002
00003 class CGUI
00004 {
00005
00006 public:
00007     CGUI();
00008     ~CGUI();
00009 };
```

5.3 InputParameter.h-Dateireferenz

: Header File

```
#include ¨EulerMatrix.h¨
#include ¨Point3D.h¨
#include <string>
#include <vector>
#include <list>
#include <iostream>
#include <fstream>
#include <sstream>
#include <tuple>
```

Klassen

- class CInputParameter

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

5.3.1 Ausführliche Beschreibung

: Header File

5.4 InputParameter.h

gehe zur Dokumentation dieser Datei

```
00001
00007 #pragma once
00008
00009 #include "EulerMatrix.h"
00010 #include "Point3D.h"
00011 #include <string>
00012 #include <vector>
00013 #include <list>
00014 #include <iostream>
00015 #include <fstream>
00016 #include <sstream>
00017 #include <tuple>
00018
00019 using namespace std;
00020
00021 #pragma once
00022
00026 class CInputParameter
00027 {
00028 public:
00034     CInputParameter(void);
00040     CInputParameter(double initSpeed, bool initSeepManual, bool initOrientationManual, double initA,
      double initB, double initC);
00045     ~CInputParameter(void);
00046
00051     void setOrientation(bool initOrientationManual, double initA, double initB, double initC);
00056     void setSpeed(double initSpeed, bool initSpeedManual);
00057
00062     double getSpeed(void);
00067     bool getSpeedManual(void);
00072     bool getOrientationManual(void);
00077     tuple <double, double, double> getAngles(void);
00078
00083     void openFile(std::string path);
00088     bool detectJump(CInputPoint3D p, double x_prev, double  y_prev,double z_prev);
00093     vector<list<CInputPoint3D>& getPath();
00094
00095 private:
00099     vector<list<CInputPoint3D> initialPath;
00103     double speed;
00107     bool speedManual;
00111     bool orientationManual;
00115     double A;
00119     double B;
00123     double C;
00127     double difference = 20;
00128 };
00129
```

5.5 Line3D.h

```
00001 #include "Point3D.h"
00002 #include <math.h>
00003
00004 using namespace std;
00005
00006 #pragma once
```

```
00007 class CLine3D
00008 {
00009 public:
00010     CLine3D(void);
00011     CLine3D(CPoint3D P1, CPoint3D P2);
00012     ~CLine3D(void);
00013
00014     CPoint3D p1;
00015     CPoint3D p2;
00016 };
00017
```

5.6 MeanFilter.h

```
00001 #include <vector>
00002 #include <list>
00003 #include "Point3D.h"
00004
00005 #pragma once
00006
00007 using namespace std;
00008
00009
00010 class CMeanFilter
00011 {
00012 public:
00013     CMeanFilter();
00014     CMeanFilter(int Window, long lenth);
00015     ~CMeanFilter();
00016
00017     void setWindowSize(int Window);
00018
00019     int getWindowSize();
00020
00021     vector<list<CInputPoint3D>& getPath();
00022
00023     list<CInputPoint3D> calculateMean(list<CInputPoint3D>& segment);
00024     void mean(vector<list<CInputPoint3D>& sourcePath);
00025
00026 private:
00027     int windowSize;
00028     int position;
00029     double sum;
00030
00031     vector<list<CInputPoint3D> meanPath;
00032 };
00033
```

5.7 PathBuilder.h

```
00001 #include <vector>
00002 #include <list>
00003 #include <iostream>
00004 #include "Point3D.h"
00005
00006 using namespace std;
00007
00008 #pragma once
00009 class CPathBuilder
00010 {
00011 public:
00012     CPathBuilder(void);
00013     ~CPathBuilder(void);
00014
00015     vector<CInputPoint3D>& getPath();
00016
00017     void createPath(vector<list<CInputPoint3D>& segments, string filename);
00018
00019 private:
00020     vector<CInputPoint3D> path;
00021 };
00022
```

5.8 PathPostProcessing.h

```
00001 #include <vector>
```

```
00002 #include <list>
00003 #include <iostream>
00004 #include <tuple>
00005 #include "Point3D.h"
00006
00007 using namespace std;
00008
00009 #pragma once
00010
00011 #define MAX_SPEED 2.0
00012
00013 class CPathPostProcessing
00014 {
00015 public:
00016     CPathPostProcessing(void);
00017     CPathPostProcessing(double speedIn, bool speedManualIn, bool orientationManualIn, double AIn,
      double BIn, double CIn);
00018     ~CPathPostProcessing(void);
00019
00020     vector<CPoint3D>& getProcessedPath(void);
00021
00022     void postProcessing(vector<CPoint3D>& path);
00023     void calculateSpeed(COutputPoint3D& p, size_t i);
00024     void calculateAngles(COutputPoint3D& p);
00025
00026     void setData(double speed, bool speedManual, bool orientationManual, tuple<double , double ,
      double> angles);
00027
00028 private:
00029     vector<COutputPoint3D> processedPath;
00030     double speed;
00031     bool speedManual;
00032     bool orientationManual;
00033     double A;
00034     double B;
00035     double C;
00036 };
00037
```

5.9 Point3D.h

```
00001 #include "EulerMatrix.h"
00002
00003 class CLine3D;
00004
00005 using namespace std;
00006
00007 #pragma once
00008 class CPoint3D
00009 {
00010 public:
00011     CPoint3D(void);
00012     CPoint3D(double X, double Y, double Z);
00013     ~CPoint3D(void);
00014
00015     double getX();
00016     double getY();
00017     double getZ();
00018
00019     void setX(double X);
00020     void setY(double Y);
00021     void setZ(double Z);
00022
00023     void set(double X, double Y, double Z);
00024     double distanceTo(CPoint3D point);
00025     double distanceTo(CLine3D line);
00026
00027 protected:
00028     double x, y, z;
00029 };
00030
00031 class CInputPoint3D : public CPoint3D
00032 {
00033 public:
00034     CInputPoint3D(void);
00035     CInputPoint3D(double X, double Y, double Z, double Timestamp, CEulerMatrix Matrix);
00036     ~CInputPoint3D(void);
00037
00038     double getTime();
00039     CEulerMatrix getEulerMatrix();
00040
00041     void setTime(double time);
00042     void setEulerMatrix(CEulerMatrix orientation);
```

```
00043     void setPoint(double time, double X, double Y, double Z, CEulerMatrix orientation);
00044
00045 private:
00046     double timestamp;
00047     CEulerMatrix orientationMatrix;
00048 };
00049
00050 class COutputPoint3D : public CPoint3D
00051 {
00052 public:
00053     COutputPoint3D(void);
00054     COutputPoint3D(double Speed, double X, double Y, double Z, double A, double B, double C);
00055     ~COutputPoint3D(void);
00056
00057     double getSpeed();
00058     double getA();
00059     double getB();
00060     double getC();
00061
00062     //void setPoint(double speed, double X, double Y, double Z, CEulerMatrix orientation);
00063     void setSpeed(double speed);
00064     void setA(double A);
00065     void setB(double B);
00066     void setC(double C);
00067 private:
00068     double a, b, c;
00069     double speed;
00070 };
```

5.10 RobCodeGenerator.h

```
00001 #include <vector>
00002 #include <iostream>
00003 #include "Point3D.h"
00004 #include <tuple>
00005
00006 using namespace std;
00007
00008 #pragma once
00009
00010 #define MAX_SPEED 2.0
00011
00012 class CRobCodeGenerator
00013 {
00014 public:
00015     CRobCodeGenerator(void);
00016     CRobCodeGenerator(double speedIn, bool speedManualIn, bool orientationManualIn, tuple<double,
      double, double> angles);
00017     ~CRobCodeGenerator(void);
00018
00019     void generateRobCode(vector<CInputPoint3D>& path, string filename);
00020     void postProcessing(vector<CInputPoint3D>& path);
00021     double calculateSpeed(CInputPoint3D& p, size_t i, double timePrev);
00022     void calculateAngles(COutputPoint3D& p, CInputPoint3D& pIn);
00023
00024 private:
00025     vector<COutputPoint3D> processedPath;
00026     double speed;
00027     bool speedManual;
00028     bool orientationManual;
00029     double A;
00030     double B;
00031     double C;
00032
00033 };
00034
```

5.11 SegmentApproximator.h

```
00001 #include <valarray>
00002 #include <vector>
00003 #include <list>
00004 #include <iostream>
00005 #include <math.h>
00006 #include "Point3D.h"
00007
00008 using namespace std;
00009
00010 #pragma once
```

```
00011 class CSegmentApproximator
00012 {
00013 public:
00014     CSegmentApproximator(void);
00015     ~CSegmentApproximator(void);
00016
00017     void approx(const vector<list<CInputPoint3D»& Segments);
00018     void setmaxDistance(double maxDistanceSource);
00019     double getmaxDistance();
00020
00021     vector<list<CInputPoint3D»& getSegmentsApproxVector();
00022
00023 private:
00024     vector<list<CInputPoint3D» segmentsApprox;
00025
00026     double maxDistance;
00027     void douglasPeuckerRecursive(list<CInputPoint3D>& segment, std::list<CInputPoint3D>::iterator
      startItr, std::list<CInputPoint3D>::iterator endItr, double maxDistance);
00028     double calcDist(int xS, int yS, int zS, int xE, int yE, int zE, int x, int y, int z);
00029
00030 };
```

Index