

My Project

Generated by Doxygen 1.9.2

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Aorta Class Reference	7
4.1.1 Member Function Documentation	7
4.1.1.1 g_dangerD()	7
4.1.1.2 g_deadD()	8
4.1.1.3 g_safety()	8
4.1.1.4 s_deadD()	8
4.1.1.5 s_point()	9
4.1.1.6 s_points()	9
4.2 Catheter Class Reference	9
4.3 MainLoop Class Reference	10
4.4 Visualizer Class Reference	11
5 File Documentation	13
5.1 aorta.h	13
5.2 catheter.h	14
5.3 mainloop.h	14
5.4 robot.h	15
5.5 visualizer.h	15
Index	17

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Aorta	7
Catheter	9
Visualizer	11
vtkCommand	
MainLoop	10

Chapter 2

Class Index

2.1 Class List

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

Aorta	7
Catheter	9
MainLoop	10
Visualizer	11

Chapter 3

File Index

3.1 File List

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

include/ aorta.h	13
include/ catheter.h	14
include/ mainloop.h	14
include/ robot.h	15
include/ visualizer.h	15

Chapter 4

Class Documentation

4.1 Aorta Class Reference

Public Member Functions

- bool [s_point](#) (Eigen::Matrix4d point)
Adds a single point to the point cloud of the [Aorta](#).
- bool [s_points](#) (Eigen::MatrixXd points)
Adds a group of points to the point cloud of the [Aorta](#).
- bool [s_deadD](#) (double deadD)
Sets the dead distance for the [Aorta](#).
- bool [s_dangerD](#) (double dangerD)
- bool [s_safeD](#) (double safeD)
- void [clear](#) ()
- void [checkDistance](#) (Eigen::Matrix4d ee)
does somthegn
- int [g_safety](#) ()
Returns the calculated safety value of the current simulation.
- double [g_maxDist](#) ()
- double [g_deadD](#) ()
Returns the dead distance.
- double [g_dangerD](#) ()
Returns the danger distance.
- Eigen::MatrixXd [g_points](#) ()
Returns all points set by [s_point\(\)](#) and [s_points\(\)](#)

4.1.1 Member Function Documentation

4.1.1.1 [g_dangerD\(\)](#)

```
double Aorta::g_dangerD ( )
```

Returns the danger distance.

Parameters

out		
-----	--	--

verbatim Returns -1 if `s_dangerD()` has not yet been called Returns dangerD otherwise

4.1.1.2 `g_deadD()`

```
double Aorta::g_deadD ( )
```

Returns the dead distance.

Parameters

out		
-----	--	--

verbatim Returns -1 if `s_deadD()` has not yet returned true. Returns deadD otherwise

4.1.1.3 `g_safety()`

```
int Aorta::g_safety ( )
```

Returns the calculated safety value of the current simulation.

Calculates a safety value from a single point and the every part of the point cloud. The safety value is based on the distance to the cloud

Parameters

out	<i>int;</i>	ranging from -1 to 2 -1 if <code>checkDistance()</code> has not yet been run and <code>s_deadD()</code> and <code>s_dangerD()</code> have both not yet returned true 0 if the distance is less than the Dead Distance 1 if the distance is greater than Dead Distance but less than Danger Distance 2 if the distance is greater than Danger Distance
-----	-------------	---

4.1.1.4 `s_deadD()`

```
bool Aorta::s_deadD (
    double deadD )
```

Sets the dead distance for the [Aorta](#).

The minimum distance something can come to the aorta.

Parameters

in	<i>deadD</i>	
----	--------------	--

Parameters

out	<i>True/False</i>	True if the distance is within the predefined acceptable range. False if it falls outside the predefined acceptable range.
-----	-------------------	---

4.1.1.5 s_point()

```
bool Aorta::s_point (
    Eigen::Matrix4d point )
```

Adds a single point to the point cloud of the [Aorta](#).

Parameters

in	<i>point</i>	
out	<i>True/False</i>	True if the point does not already exist. False if the point does already exist.

4.1.1.6 s_points()

```
bool Aorta::s_points (
    Eigen::MatrixXd points )
```

Adds a group of points to the point cloud of the [Aorta](#).

Parameters

in	<i>points</i>	
out	<i>True/False</i>	True if all the points did not yet exist. False if any of the points did already exist.

The documentation for this class was generated from the following files:

- include/aorta.h
- src/aorta.cpp

4.2 Catheter Class Reference

Public Member Functions

- bool **s_baseFrame** (Eigen::Matrix4d baseFrame)

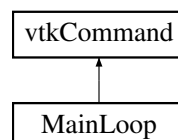
- bool **s_nseg** (int nseg)
- bool **s_nq** (int nq)
- bool **s_pps** (int pps)
- bool **s_rad** (double rad)
- bool **s_bbLen** (double bbLen)
- void **fkine** (Eigen::MatrixXd q)
- double **g_distEE** ()
- double **g_rad** ()
- double **g_q1change** ()
- double **g_q2change** ()
- double **g_q3change** ()
- double **g_qChange** (int qkind)
- Eigen::MatrixXd **g_q** ()
- Eigen::MatrixXd **g_baseFrame** ()
- Eigen::Matrix4d **g_eeFrame** ()
- Eigen::MatrixXd **g_backbone** ()

The documentation for this class was generated from the following files:

- include/catheter.h
- src/catheter.cpp

4.3 MainLoop Class Reference

Inheritance diagram for MainLoop:



Public Member Functions

- **MainLoop** ([Visualizer](#) vis)
- virtual void **Execute** (vtkObject *vtkNotUsed(caller), unsigned long eventId, void *vtkNotUsed(callData))

The documentation for this class was generated from the following files:

- include/mainloop.h
- src/mainloop.cpp

4.4 Visualizer Class Reference

Public Member Functions

- void **drawCath** (Eigen::MatrixXd bb, double rad)
- void **drawAorta** (Eigen::MatrixXd wall, double dead, double danger)
- void **update** ()
- void **clearCath** ()
- void **clearAorta** ()
- void **clear** ()
- vtkSmartPointer< vtkRenderWindow > **g_renderWindow** ()

The documentation for this class was generated from the following files:

- include/visualizer.h
- src/visualizer.cpp

Chapter 5

File Documentation

5.1 aorta.h

```
1
2
3
4
5
6 // Include
7 #include <robot.h>
8
9 //stl
10 #include <cmath>
11 #include <iostream>
12 #include <ctime>
13
14 //Eigen
15 #include <Eigen/Dense>
16
17 class Aorta{
18     private:
19         double m_deadD;
20         double m_dangerD;
21         double m_safeD;
22
23         // For Checking
24         double m_maxD;
25         int m_safety;
26
27         // For Drawing
28         Eigen::MatrixX<double> m_points;
29
30     public:
31         // Initialize
32         Aorta();
33         ~Aorta();
34
35         // Setting Functions
36         bool s_point(Eigen::Matrix4<double> point);
37         bool s_points(Eigen::MatrixX<double> points);
38         bool s_deadD(double deadD);
39         bool s_dangerD(double dangerD);
40         bool s_safeD(double safeD);
41
42         // Doing Functions
43         void clear();
44         void checkDistance(Eigen::Matrix4<double> ee);
45
46         // Getting Functions
47         int g_safety();
48         double g_maxDist();
49         double g_deadD();
50         double g_dangerD();
51         Eigen::MatrixX<double> g_points();
52 };
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
```

5.2 catheter.h

```

1 #define _USE_MATH_DEFINES
2 #pragma once
3
4 #include "robot.h"
5
6 //stl
7 #include <cmath>
8 #include <iostream>
9 #include <ctime>
10
11 //Eigen
12 #include <Eigen/Dense>
13
14 // A catheter has the functionality of a TDCR but is drawn as if it is a CTCR
15 class Catheter
16 {
17     private:
18         int m_nseg;
19         int m_nq;
20         int m_nqps;
21         int m_pps;
22         double m_rad;
23         double m_bbLen;
24
25
26         double m_q1change;
27         double m_q2change;
28         double m_q3change;
29
30         //Member variables defining the parameters of the TDCR
31         Eigen::MatrixXd m_q;
32
33         // Matrixes
34         Eigen::Matrix4d m_baseFrame;
35         Eigen::Matrix4d m_eeFrame;
36         Eigen::MatrixXd m_backbone;
37
38     public:
39         Catheter();
40         ~Catheter();
41
42         // Setting Functions
43         bool s_baseFrame(Eigen::Matrix4d baseFrame);
44         bool s_nseg(int nseg);
45         bool s_nq(int nq);
46         bool s_pps(int pps);
47         bool s_rad(double rad);
48         bool s_bbLen(double bbLen);
49
50
51         // Doing Functions
52         void fkine(Eigen::MatrixXd q);
53
54
55         // Getting Functions
56         double g_distEE();
57         double g_rad();
58         double g_q1change();
59         double g_q2change();
60         double g_q3change();
61         double g_qChange(int qkind);
62         Eigen::MatrixXd g_q();
63         Eigen::Matrix4d g_baseFrame();
64         Eigen::Matrix4d g_eeFrame();
65         Eigen::MatrixXd g_backbone();
66
67 };

```

5.3 mainloop.h

```

1 #pragma once
2
3 #define _USE_MATH_DEFINES
4
5 #include <visualizer.h>
6 #include <catheter.h>
7 #include <aorta.h>
8
9 //stl
10 #include <ctime>
11 #include <cmath>

```

```

12 #include <fstream>
13
14 //vtk
15 #include <vtkSmartPointer.h>
16 #include <vtkCommand.h>
17 #include <vtkRenderWindowInteractor.h>
18
19 //Eigen
20 #include <Eigen/Dense>
21
22 // Class that implements the main simulation loop
23 class MainLoop : public vtkCommand
24 {
25     private:
26         Visualizer mp_vis;
27         Catheter m_cath;
28         Aorta m_aorta;
29
30         Eigen::MatrixXd m_wall;
31
32         bool m_engaged;
33         bool m_aortaEn;
34         bool m_cathEn;
35         Eigen::MatrixXd m_q;
36         double m_rotated;
37         double m_bended;
38
39     public:
40         MainLoop(Visualizer vis);
41         ~MainLoop();
42
43         // Execution Function
44         virtual void Execute(vtkObject *vtkNotUsed(caller), unsigned long eventId, void
45             *vtkNotUsed(callData));
46 };

```

5.4 robot.h

```

1 #pragma once
2
3 //stl
4 #include <cmath>
5 #include <iostream>
6 #include <ctime>
7
8 //Eigen
9 #include <Eigen/Dense>
10
11 Eigen::MatrixXd arc2x(Eigen::Matrix4d baseFrame, Eigen::MatrixXd kappa, Eigen::MatrixXd length,
12     Eigen::MatrixXd phi, int n);
13
14 // Returns the distance between two different points in space.
15 double difference(Eigen::Matrix4d p1, Eigen::Matrix4d p2);
16
17 //Eigen::MatrixXd arc_to_x(Eigen::Matrix4d init_frame, std::vector<double> kappa, std::vector<double> l,
18     std::vector<double> phi, int n, bool bishop);
19
20 //Eigen::Matrix4d matrix_log(Eigen::Matrix4d T);
21
22 //Eigen::MatrixXd calculate_desired_body_twist(Eigen::Matrix4d T_target, Eigen::Matrix4d T_cur);
23

```

5.5 visualizer.h

```

1 #pragma once
2
3 #define _USE_MATH_DEFINES
4
5 //stl
6 #include <vector>
7 #include <array>
8
9 //vtk
10 #include <vtkRenderer.h>
11 #include <vtkRenderWindow.h>
12 #include <vtkSmartPointer.h>

```

```

13 #include <vtkActor.h>
14 #include <vtkAxesActor.h>
15 #include <vtkPolyData.h>
16 #include <vtkPolyDataMapper.h>
17 #include <vtkProperty.h>
18 #include <vtkTransform.h>
19 #include <vtkCellArray.h>
20 #include <vtkOpenGLLight.h>
21 #include <vtkUnstructuredGrid.h>
22 #include <vtkGeometryFilter.h>
23 #include <vtkMatrix4x4.h>
24 #include <vtkCamera.h>
25 #include <vtkTubeFilter.h>
26
27 // Shapes
28 #include <vtkLine.h>
29 #include <vtkLineSource.h>
30 #include <vtkTriangle.h>
31 #include <vtkCubeSource.h>
32 #include <vtkSphereSource.h>
33 #include <vtkCylinderSource.h>
34
35 #include <vtkNamedColors.h>
36 #include <vtkRenderWindowInteractor.h>
37
38 //Eigen
39 #include <Eigen/Dense>
40
41 // Class that implements the visualizer of the simulator using VTK
42 class Visualizer{
43     private:
44         vtkSmartPointer<vtkRenderer> mp_Ren;
45         vtkSmartPointer<vtkRenderWindow> mp_RenWin;
46         vtkSmartPointer<vtkAxesActor> mp_target_frame;
47         vtkSmartPointer<vtkNamedColors> mp_colors; // This is giving me trouble look into it later
48         //std::vector<vtkSmartPointer<vtkActor> mp_curves;
49         std::vector<vtkSmartPointer<vtkActor> m_curveActors;
50         std::vector<vtkSmartPointer<vtkActor> m_sphereActors;
51         std::vector<vtkSmartPointer<vtkActor> m_frameActors;
52         std::vector<vtkSmartPointer<vtkActor> m_cathActors;
53         std::vector<vtkSmartPointer<vtkActor> m_aortaActors;
54
55         // Drawing Functions
56         void drawCurves(Eigen::MatrixXd curve, double rad);
57         void drawFrames(Eigen::MatrixXd frames);
58         void drawPoints(Eigen::MatrixXd points, double rad, char color);
59         void drawSphere(Eigen::MatrixXd points, std::vector<vtkSmartPointer<vtkActor> &actors, double
rad, std::vector<double> color, double trans);
60
61     public:
62         // Init functions
63         Visualizer();
64         ~Visualizer();
65
66         // Set Functions
67
68         // Do functions
69         // Draw
70         void drawCath(Eigen::MatrixXd bb, double rad);
71         void drawAorta(Eigen::MatrixXd wall, double dead, double danger);
72         void update();
73
74         // Clear
75         void clearCath();
76         void clearAorta();
77         void clear();
78
79         // Get Functions
80         vtkSmartPointer<vtkRenderWindow> g_renderWindow();
81 };

```

Index

Aorta, [7](#)
 g_dangerD, [7](#)
 g_deadD, [8](#)
 g_safety, [8](#)
 s_deadD, [8](#)
 s_point, [9](#)
 s_points, [9](#)

Catheter, [9](#)

g_dangerD
 Aorta, [7](#)
g_deadD
 Aorta, [8](#)
g_safety
 Aorta, [8](#)

include/aorta.h, [13](#)
include/catheter.h, [14](#)
include/mainloop.h, [14](#)
include/robot.h, [15](#)
include/visualizer.h, [15](#)

MainLoop, [10](#)

s_deadD
 Aorta, [8](#)
s_point
 Aorta, [9](#)
s_points
 Aorta, [9](#)

Visualizer, [11](#)