Lissajous Curves 3D

Generated by Doxygen 1.9.5

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 ChartClass Class Reference	. 7
4.1.1 Detailed Description	. 7
4.1.2 Constructor & Destructor Documentation	. 7
4.1.2.1 ChartClass()	. 7
4.1.3 Member Function Documentation	. 8
4.1.3.1 Draw()	. 8
4.1.4 Member Data Documentation	. 10
4.1.4.1 cfg	. 10
4.1.4.2 timer	. 10
4.2 ConfigClass Class Reference	. 11
4.2.1 Detailed Description	. 12
4.2.2 Constructor & Destructor Documentation	. 13
4.2.2.1 ConfigClass()	. 13
4.2.3 Member Function Documentation	. 13
4.2.3.1 get_Animation()	. 13
4.2.3.2 get_Points()	. 13
4.2.3.3 get_Polar()	. 14
4.2.3.4 getR()	. 14
4.2.3.5 getX_A()	. 14
4.2.3.6 getX_phi()	
4.2.3.7 getX_Rot()	. 15
4.2.3.8 getX_theta()	
4.2.3.9 getY_A()	
4.2.3.10 getY_phi()	. 16
4.2.3.11 getY_Rot()	
4.2.3.12 getY_theta()	
4.2.3.13 getZ_A()	. 17
4.2.3.14 getZ_phi()	
4.2.3.15 getZ_Rot()	
4.2.3.16 getZ_theta()	
4.2.3.17 Set_Animation()	
4.2.3.18 Set_ParamType()	
4.2.3.19 Set_Points()	

4.2.3.20 SetR()	19
4.2.3.21 SetX_A()	19
4.2.3.22 SetX_phi()	20
4.2.3.23 SetX_Rot()	20
4.2.3.24 SetX_theta()	20
4.2.3.25 SetY_A()	21
4.2.3.26 SetY_phi()	21
4.2.3.27 SetY_Rot()	21
4.2.3.28 SetY_theta()	22
4.2.3.29 SetZ_A()	22
4.2.3.30 SetZ_phi()	22
4.2.3.31 SetZ_Rot()	23
4.2.3.32 SetZ_theta()	23
4.2.4 Member Data Documentation	23
4.2.4.1 animation	23
4.2.4.2 MainWindow	24
4.2.4.3 points	24
4.2.4.4 polar	24
4.2.4.5 R	24
4.2.4.6 x	24
4.2.4.7 X_A	24
4.2.4.8 X_phi	25
4.2.4.9 X_Rot	25
4.2.4.10 X_theta	25
4.2.4.11 y	25
4.2.4.12 Y_A	25
4.2.4.13 Y_phi	25
4.2.4.14 Y_Rot	26
4.2.4.15 Y_theta	26
4.2.4.16 z	26
4.2.4.17 Z_A	26
4.2.4.18 Z_phi	26
4.2.4.19 Z_Rot	26
4.2.4.20 Z_theta	27
4.3 GUIMyFrame Class Reference	27
4.3.1 Detailed Description	28
4.3.2 Constructor & Destructor Documentation	29
4.3.2.1 GUIMyFrame()	29
4.3.2.2 ∼GUIMyFrame()	29
4.3.3 Member Function Documentation	29
4.3.3.1 Animation_Updated()	29
4.3.3.2 DisplayPanelRepaint()	30

4.3.3.3 DotsLines_Updated()	 30
4.3.3.4 MainFormClose()	 31
4.3.3.5 ParamType_Updated()	 31
4.3.3.6 Repaint()	 32
4.3.3.7 XA_Updated()	 32
4.3.3.8 XPhi_Updated()	 33
4.3.3.9 XRot_Updated()	 33
4.3.3.10 XTheta_Updated()	 34
4.3.3.11 YA_Updated()	 34
4.3.3.12 YPhi_Updated()	 34
4.3.3.13 YRot_Updated()	 35
4.3.3.14 YTheta_Updated()	 35
4.3.3.15 ZA_Updated()	 36
4.3.3.16 ZPhi_Updated()	 36
4.3.3.17 ZRot_Updated()	 37
4.3.3.18 ZTheta_Updated()	 37
4.3.4 Friends And Related Function Documentation	 37
4.3.4.1 ChartClass	 37
4.3.5 Member Data Documentation	 38
4.3.5.1 cfg	 38
4.4 Matrix4 Class Reference	 38
4.4.1 Detailed Description	 38
4.4.2 Constructor & Destructor Documentation	 38
4.4.2.1 Matrix4()	 39
4.4.3 Member Function Documentation	 39
4.4.3.1 operator*()	 39
4.4.3.2 Print()	 39
4.4.4 Friends And Related Function Documentation	 39
4.4.4.1 operator*	 40
4.4.5 Member Data Documentation	 40
4.4.5.1 data	 40
4.5 MyApp Class Reference	 40
4.5.1 Detailed Description	 40
4.5.2 Member Function Documentation	 41
4.5.2.1 OnExit()	 41
4.5.2.2 Onlnit()	 41
4.6 MyFrame1 Class Reference	 41
4.6.1 Detailed Description	 43
4.6.2 Constructor & Destructor Documentation	 43
4.6.2.1 MyFrame1()	 44
4.6.2.2 ∼MyFrame1()	 49
4.6.3 Member Function Documentation	 50

4.6.3.1 Animation_Updated()	50
4.6.3.2 AnimationBreak()	51
4.6.3.3 DisplayPanelRepaint()	51
4.6.3.4 DotsLines_Updated()	51
4.6.3.5 MainFormClose()	52
4.6.3.6 ParamType_Updated()	52
4.6.3.7 Scrolls_Updated()	52
4.6.3.8 XA_Updated()	53
4.6.3.9 XPhi_Updated()	53
4.6.3.10 XRot_Updated()	53
4.6.3.11 XTheta_Updated()	54
4.6.3.12 YA_Updated()	54
4.6.3.13 YPhi_Updated()	54
4.6.3.14 YRot_Updated()	55
4.6.3.15 YTheta_Updated()	55
4.6.3.16 ZA_Updated()	55
4.6.3.17 ZPhi_Updated()	57
4.6.3.18 ZRot_Updated()	57
4.6.3.19 ZTheta_Updated()	57
4.6.4 Friends And Related Function Documentation	59
4.6.4.1 GUIMyFrame	59
4.6.5 Member Data Documentation	59
4.6.5.1 animationBox	59
4.6.5.2 m_AXText	59
4.6.5.3 m_AYText	60
4.6.5.4 m_AZText	60
4.6.5.5 m_DisplayWindow	60
4.6.5.6 m_PhiXText	60
4.6.5.7 m_PhiYText	60
4.6.5.8 m_PhiZText	61
4.6.5.9 m_radioBox1	61
4.6.5.10 m_staticText17	61
4.6.5.11 m_staticText18	61
4.6.5.12 m_staticText19	61
4.6.5.13 m_staticText20	62
4.6.5.14 m_staticText21	62
4.6.5.15 m_staticText22	62
4.6.5.16 m_staticText23	62
4.6.5.17 m_staticText4	62
4.6.5.18 m_staticText5	63
4.6.5.19 m_staticText51	63
4.6.5.20 m_staticText511	63

4.6.5.21 m_staticText512	 63
4.6.5.22 m_staticText52	 63
4.6.5.23 m_staticText521	 64
4.6.5.24 m_staticText522	 64
4.6.5.25 m_staticText53	 64
4.6.5.26 m_staticText54	 64
4.6.5.27 m_ThXText	 64
4.6.5.28 m_ThYText	 65
4.6.5.29 m_ThZText	 65
4.6.5.30 m_timer	 65
4.6.5.31 m_XRotationSlider	 65
4.6.5.32 m_YRotationSlider	 65
4.6.5.33 m_ZRotationSlider	 66
4.6.5.34 ParamBox	 66
4.7 Vector4 Class Reference	 66
4.7.1 Detailed Description	 67
4.7.2 Constructor & Destructor Documentation	 67
4.7.2.1 Vector4()	 67
4.7.3 Member Function Documentation	 67
4.7.3.1 GetX()	 67
4.7.3.2 GetY()	 67
4.7.3.3 GetZ()	 67
4.7.3.4 operator-()	 68
4.7.3.5 Print()	 68
4.7.3.6 Set()	 68
4.7.4 Friends And Related Function Documentation	 68
4.7.4.1 operator*	 68
4.7.5 Member Data Documentation	 69
4.7.5.1 data	 69
5 File Documentation	71
5.1 include/ChartClass.h File Reference	71
5.1.1 Detailed Description	71
5.1.2 Function Documentation	71
5.1.2.1 min()	71
5.2 ChartClass.h	72
5.3 include/ConfigClass.h File Reference	72
5.3.1 Detailed Description	72
5.4 ConfigClass.h	73
5.5 include/GUIMyFrame.h File Reference	73
5.5.1 Detailed Description	73 74
5.6 GUIMyFrame.h	74
	 , ,

	5./ Include/vecmat.h File Reference	/5
	5.7.1 Detailed Description	75
	5.8 vecmat.h	75
	5.9 include/Window.h File Reference	76
	5.9.1 Detailed Description	76
	5.10 Window.h	76
	5.11 main.cpp File Reference	77
	5.11.1 Function Documentation	78
	5.11.1.1 IMPLEMENT_APP()	78
	5.12 main.cpp	78
	5.13 src/ChartClass.cpp File Reference	78
	5.13.1 Function Documentation	79
	5.13.1.1 min()	79
	5.14 ChartClass.cpp	79
	5.15 src/ConfigClass.cpp File Reference	81
	5.16 ConfigClass.cpp	81
	5.17 src/GUIMyFrame.cpp File Reference	82
	5.18 GUIMyFrame.cpp	82
	5.19 src/vecmat.cpp File Reference	85
	5.19.1 Function Documentation	85
	5.19.1.1 operator*() [1/2]	85
	5.19.1.2 operator*() [2/2]	85
	5.20 vecmat.cpp	86
	5.21 src/Window.cpp File Reference	87
	5.22 Window.cpp	87
Inc	dex	95

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

tClass	7
igClass	11
ix4	38
or4	66
ор	
МуАрр	40
rame	
MyFrame1	41
GUIMvFrame	27

2 Hierarchical Index

Chapter 2

Class Index

2.1 Class List

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

ChartClass	
Contains the method that draws the curve	. 7
ConfigClass	
Contains curve parameters, variables describing how the curve is being drawn and related meth	-
ods	. 11
GUIMyFrame	
Derived class of MyFrame1, with methods that edit the curve	. 27
Matrix4	
Matrix class	
MyApp	. 40
MyFrame1	
Class MyFrame1	. 41
Vector4	
Vector class	. 66

4 Class Index

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

main.cpp	77
include/ChartClass.h	
A file containing ChartClass class	71
include/ConfigClass.h	
A file containing ConfigClass	72
include/GUIMyFrame.h	
A file containing GUYMyFrame class	73
include/vecmat.h	
A file containing Vector4 and Matrix4 classes	75
include/Window.h	
A file containing MyFrame1 class	76
src/ChartClass.cpp	78
src/ConfigClass.cpp	81
src/GUIMyFrame.cpp	82
src/vecmat.cpp	85
src/Window.cpp	87

6 File Index

Chapter 4

Class Documentation

4.1 ChartClass Class Reference

Contains the method that draws the curve.

```
#include <ChartClass.h>
```

Public Member Functions

```
    ChartClass (std::shared_ptr< ConfigClass > c)
    Constructor.
```

void Draw (wxDC *dc, int w, int h)

a normal function taking three arguments, drawing the curve

Public Attributes

wxTimer timer

Private Attributes

std::shared_ptr< ConfigClass > cfg

4.1.1 Detailed Description

Contains the method that draws the curve.

Definition at line 16 of file ChartClass.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ChartClass()

```
\label{lass:chartClass} \mbox{ChartClass::ChartClass (} \\ \mbox{std::shared\_ptr} < \mbox{ConfigClass} > c \mbox{)}
```

Constructor.

Parameters

c is a std::shared_ptr pointing at a ConfigClass object

Definition at line 7 of file ChartClass.cpp.

4.1.3 Member Function Documentation

4.1.3.1 Draw()

a normal function taking three arguments, drawing the curve

Parameters

dc	is a wxDC object, the curve is being drawn on it
W	is an inteager value, width of the window
h	is an inteager value, height of the window

Definition at line 12 of file ChartClass.cpp.

```
00013 {
00014
            dc->SetBackground(wxBrush(wxColor(255, 255, 255)));
00015
            dc->Clear();
00016
            Vector4 vector1;
00017
           Vector4 vector2;
           double r1, r2, phi1, phi2, th1, th2;
00018
00019
           Matrix4 m2;
00020
00021
            double alpha = cfg->getZ_Rot() * M_PI / 180.0;
           m2.data[0][0] = cos(alpha);
m2.data[0][1] = sin(alpha);
00022
00023
           m2.data[1][0] = -sin(alpha);
m2.data[1][1] = cos(alpha);
00024
00025
           m2.data[2][2] = 1;
00026
00027
00028
            Matrix4 m3;
00029
            alpha = cfg->getY_Rot() * M_PI / 180.0;
           m3.data[0][0] = cos(alpha);
m3.data[0][2] = -sin(alpha);
00030
00031
            m3.data[1][1] = 1;
00032
           m3.data[2][0] = sin(alpha);
m3.data[2][2] = cos(alpha);
00033
00034
00035
00036
            Matrix4 m4;
           alpha = cfg->getX_Rot() * M_PI / 180.0;
m4.data[0][0] = 1;
m4.data[1][1] = cos(alpha);
00037
00038
00039
            m4.data[1][2] = sin(alpha);
00040
00041
            m4.data[2][1] = -sin(alpha);
            m4.data[2][2] = cos(alpha);
00042
00043
00044
           Matrix4 transform1 = m4 * m3 * m2;
00045
           double minTheta = min(cfg->getX_theta(), cfg->getY_theta(), cfg->getZ_theta());
00046
            // drawing
00047
            wxPen m_pen;
```

```
00048
                          m_pen.SetColour((wxColor(200, 200, 200)));
00049
                          dc->SetPen(m_pen);
00050
                          if (!cfg->get_Animation())
00051
                                    for (double i = 0; i < ((50 * 3.14159) / minTheta); <math>i += ((2 * 3.14159) / (minTheta * 200)))
00052
00053
00054
                                               if(!cfg->get_Polar()){
00055
                                                        vector1.data[0] = cfg->getX_A() * sin(cfg->getX_theta() * i + cfg->getX_phi());
00056
                                                        vector1.data[1] = cfg->getY_A() * sin(cfg->getY_theta() * i + cfg->getY_phi());
00057
                                                        vector1.data[2] = cfg->getZ_A() * sin(cfg->getZ_theta() * i + cfg->getZ_phi());
                                                        vector1.data[3] = 1;
00058
00059
                                                        vector1 = transform1 * vector1;
00060
00061
                                                        (minTheta * 200))) + cfg->getX_phi());
00062
                                                        \texttt{vector2.data[1]} = \texttt{cfg-} \texttt{yetY\_A()} * \texttt{sin(cfg-} \texttt{yetY\_theta()} * (\texttt{i} + ((2 * 3.14159) / (2 * 3.14159))) * (\texttt{i} + ((2 * 3.14159) / (2 * 3.14159))) * (\texttt{i} + ((2 * 3.14159)
                   (minTheta * 200))) + cfg->getY_phi());
00063
                                                        vector2.data[2] = cfg->getZ A() * sin(cfg->getZ theta() * (i + ((2 * 3.14159) / )))
                   (minTheta * 200))) + cfg->getZ_phi());
00064
                                                        vector2.data[3] = 1;
00065
                                                        vector2 = transform1 * vector2;
00066
                                             else {
00067
                                                       rl = cfg->getR() * sin(cfg->getX_theta() * i + cfg->getX_phi());
thl = M_PI * sin(cfg->getY_theta() * i + cfg->getY_phi());
00068
00069
                                                       phi1 = (M_PI/2) * sin(cfg->getZ_theta() * i + cfg->getZ_phi());
00070
00071
00072
                                                        vector1.data[0] = r1 * cos(th1) * cos(phi1);
                                                        vector1.data[1] = r1 * sin(th1) * cos(phi1);
00073
                                                        vector1.data[2] = r1 * sin(phi1);
00074
00075
                                                        vector2.data[3] = 1;
00076
                                                        vector1 = transform1 * vector1;
00077
00078
                                                         \texttt{r2} = \texttt{cfg-} \texttt{>getR()} * \texttt{sin(cfg-} \texttt{>getX\_theta()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200)))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200))))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200))))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200))))) + \texttt{cfg-} \texttt{>getR()} * (i + ((2 * 3.14159) / (\texttt{minTheta} * 200))))) + \texttt{cfg-} \texttt{>getR()} * (i + ((
                  cfg->getX_phi());
00079
                                                        th2 = M_PI * sin(cfq->qetY_theta() * (i + ((2 * 3.14159) / (minTheta * 200))) +
                  cfg->getY_phi());
00080
                                                       phi2 = (M_PI / 2) * sin(cfg->getZ_theta() * (i + ((2 * 3.14159) / (minTheta * 200))) +
                  cfg->getZ_phi());
00081
00082
                                                        vector2.data[0] = r2 * cos(th2) * cos(phi2);
                                                        vector2.data[1] = r2 * sin(th2) * cos(phi2);
00083
                                                        vector2.data[2] = r2 * sin(phi2);
00084
                                                        vector2.data[3] = 1;
00085
00086
                                                        vector2 = transform1 * vector2;
00087
                                             }
00088
00089
                                              dc->SetPen(wxPen(*wxBLACK, 2));
00090
                                              if (cfg->get Points() == true)
                                                       dc->DrawCircle(wxPoint(w / 2 + vector1.data[0], h / 2 + vector1.data[1]), 1);
00091
00092
                                                       dc->DrawLine(wxPoint(w / 2 + vector1.data[0], h / 2 + vector1.data[1]), wxPoint(w / 2
00093
                  + vector2.data[0], h / 2 + vector2.data[1]));
00094
                               }
                         }
00095
00096
                         else
00097
00098
                                    static long long int startPoint = 0;
00099
                                    std::vector<std::vector<double» animationPoints;
00100
                                    std::vector<double> t;
00101
                                    if (!cfg->get Polar())
                                              for (double i = 0; i < ((50 * 3.14159) / minTheta); i += ((2 * 3.14159) / (minTheta *
00102
                  200)))
00103
00104
                                                        \verb|vector1.data[0]| = \verb|cfg->getX_A()| * \sin(\verb|cfg->getX_theta()| * i + \verb|cfg->getX_phi())|;
00105
                                                        \verb|vector1.data[1]| = \verb|cfg->getY_A()| * sin(\verb|cfg->getY_theta()| * i + \verb|cfg->getY_phi())|;
                                                        vector1.data[2] = cfg->getZ_A() * sin(cfg->getZ_theta() * i + cfg->getZ_phi());
00106
                                                        vector1.data[3] = 1;
00107
                                                        vector1 = transform1 * vector1;
00108
00109
                                                        t.push_back(vector1.data[0]);
00110
                                                         t.push_back(vector1.data[1]);
00111
                                                        animationPoints.push_back(t);
00112
                                                        t.clear();
                                            }
00113
00114
                                   else
                                             for (double i = 0; i < ((50 * 3.14159) / minTheta); <math>i += ((2 * 3.14159) / (minTheta))
                  200)))
00116
                                                        r1 = cfg->getR() * sin(cfg->getX_theta() * i + cfg->getX_phi());
th1 = M_PI * sin(cfg->getY_theta() * i + cfg->getY_phi());
00117
00118
                                                        phi1 = (M_PI / 2) * sin(cfg->getZ_theta() * i + cfg->getZ_phi());
00119
00120
                                                        vector1.data[0] = r1 * cos(th1) * cos(phi1);
vector1.data[1] = r1 * sin(th1) * cos(phi1);
00121
00122
                                                        vector1.data[2] = r1 * sin(phi1);
vector2.data[3] = 1;
00123
00124
                                                        vector1 = transform1 * vector1;
00125
```

```
t.push_back(vector1.data[0]);
  00127
                                                                                                                                t.push_back(vector1.data[1]);
 00128
                                                                                                                                 animationPoints.push_back(t);
 00129
                                                                                                                                t.clear();
 00130
                                                                                  for (int i = 0; i < animationPoints.size()-1; i++)</pre>
 00131
  00132
                                                                                  {
  00133
                                                                                                         \label{eq:continuous} $$ dc->DrawLine(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][1]), $$ $$ dc->DrawLine(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][1]), $$ $$ dc->DrawLine(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][0]), $$ $$ dc->DrawLine(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][0]), $$ $$ dc->DrawLine(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][0]), $$ $$ dc->DrawLine(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][0]), $$ $$ dc->DrawLine(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][0]), $$ $$ dc->DrawLine(wxPoints[i][0], h / 2 + animationPoints[i][0], h / 2 + animationPoints[i][
                                         wxPoint(w \ / \ 2 \ + \ animationPoints[(i \ + \ 1) \ \$ \ animationPoints.size()][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0] \ , \ h \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ 
                                         % animationPoints.size()][1]));
 00134
                                                                                }
 00135
  00136
                                                                                  for (int i = startPoint; i < startPoint + 150;)</pre>
  00137
 00138
                                                                                                         i %= animationPoints.size();
00139
                                                                                                         dc->SetPen(wxPen(wxColor(255 * (i - startPoint) / 150, 0, 255 * (150 - i + startPoint) /
                                         150), 2));
 00140
                                                                                                          if (cfg->get Points() == true)
 00141
 00142
                                                                                                                                 dc->DrawCircle(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][1]),
 00143
                                                                                                                                 i += 4;
 00144
                                                                                                        }
 00145
                                                                                                        else
 00146
                                                                                                        {
                                                                                                                               dc->DrawLine(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][1]),
 00147
                                         wxPoint(w \ / \ 2 \ + \ animationPoints[(i \ + \ 1) \ \$ \ animationPoints.size()][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1)][0], \ h \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A \ / \ A
                                         % animationPoints.size()][1]));
 00148
                                                                                                                               i++;
 00149
  00150
                                                                                                       if (i == animationPoints.size() - 1) i = 0;
  00151
  00152
                                                                                 if (cfg->get_Points()) startPoint += 4;
                                                                                  else startPoint++;
  00153
                                                                                  if (startPoint >= 4000) startPoint = 0;
 00154
                                                         }
 00155
00156 }
```

4.1.4 Member Data Documentation

4.1.4.1 cfg

```
std::shared_ptr<ConfigClass> ChartClass::cfg [private]
```

Definition at line 19 of file ChartClass.h.

4.1.4.2 timer

```
wxTimer ChartClass::timer
```

Definition at line 33 of file ChartClass.h.

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

- · include/ChartClass.h
- · src/ChartClass.cpp

4.2 ConfigClass Class Reference

Contains curve parameters, variables describing how the curve is being drawn and related methods.

```
#include <ConfigClass.h>
```

Public Member Functions

```
    ConfigClass (GUIMyFrame *wnd)

• bool get_Points ()
     a normal method returning a bool value.

    bool get_Animation ()

      a normal method returning a bool value.

    bool get_Polar ()

     a normal method returning a bool value.

    double getX_A ()

     a normal method returning a double value.
• double getY A ()
     a normal method returning a double value.

    double getZ_A ()

     a normal method returning a double value.

    double getX_theta ()

      a normal method returning a double value.

    double getY_theta ()

     a normal method returning a double value.
• double getZ_theta ()
     a normal method returning a double value.
• double getX_phi ()
     a normal method returning a double value.

    double getY_phi ()

     a normal method returning a double value.

    double getZ_phi ()

     a normal method returning a double value.

    double getX_Rot ()

     a normal method returning a double value.
· double getY_Rot ()
     a normal method returning a double value.

    double getZ_Rot ()

      a normal method returning a double value.
• double getR ()
      a normal method returning a double value.

    void Set_Points (bool a)

     a normal method taking one argument and setting points variable

    void Set Animation (bool a)

     a normal method taking one argument and setting animation variable

    void Set_ParamType (bool a)

     a normal method taking one argument and setting polar variable
```

a normal method taking one argument and setting X_Rot variable

void SetX Rot (int a)

void SetY_Rot (int a)

a normal method taking one argument and setting Y_Rot variable

void SetZ_Rot (int a)

a normal method taking one argument and setting Z_Rot variable

void SetX A (double a)

a normal method taking one argument and setting X_A variable

void SetY_A (double a)

a normal method taking one argument and setting Y_A variable

void SetZ_A (double a)

a normal method taking one argument and setting Z_A variable

void SetX_theta (double theta)

a normal method taking one argument and setting X_theta variable

void SetY_theta (double theta)

a normal method taking one argument and setting Y_theta variable

void SetZ_theta (double theta)

a normal method taking one argument and setting Z_theta variable

void SetX_phi (double phi)

a normal method taking one argument and setting X_phi variable

void SetY_phi (double phi)

a normal method taking one argument and setting Y_phi variable

void SetZ phi (double phi)

a normal method taking one argument and setting Z_phi variable

void SetR (double nR)

a normal method taking one argument and setting R variable

Protected Attributes

- GUIMyFrame * MainWindow
- double X A
- double Y_A
- double Z_A
- double X theta
- double Y_theta
- double Z theta
- double X_phi
- double Y phi
- double Z_phi
- double X Rot
- double Y Rot
- double Z Rot
- double x
- double y
- double z
- double R
- bool animation
- bool points
- · bool polar

4.2.1 Detailed Description

Contains curve parameters, variables describing how the curve is being drawn and related methods.

Definition at line 11 of file ConfigClass.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 ConfigClass()

```
ConfigClass::ConfigClass (
                    GUIMyFrame * wnd )
Definition at line 4 of file ConfigClass.cpp.
00005 {
             MainWindow = wnd;

X_A = Y_A = Z_A = 100;

X_theta = 1;

Y_theta = 1;

Z_theta = 0;

X_phi = Z_phi = 0;

Y_phi = 3.1415 / 2;
00006
00007
80000
00009
00010
00011
00012
00013
              R = 100;
00014
00015
              points = false;
00016
              animation = false;
00017
              polar = false;
00018 }
```

4.2.3 Member Function Documentation

4.2.3.1 get_Animation()

```
bool ConfigClass::get_Animation ( ) [inline]
```

a normal method returning a bool value.

Returns

is the animation radiobox checked

```
Definition at line 37 of file ConfigClass.h. 00037 { return animation; }
```

4.2.3.2 get_Points()

```
bool ConfigClass::get_Points ( ) [inline]
```

a normal method returning a bool value.

Returns

is the dots radiobox checked

```
Definition at line 32 of file ConfigClass.h.
```

```
00032 { return points; }
```

4.2.3.3 get_Polar()

```
bool ConfigClass::get_Polar ( ) [inline]
```

a normal method returning a bool value.

Returns

is the polar radiobox checked

```
Definition at line 43 of file ConfigClass.h. 00043 { return polar; };
```

4.2.3.4 getR()

```
double ConfigClass::getR ( ) [inline]
```

a normal method returning a double value.

Returns

R value

```
Definition at line 112 of file ConfigClass.h. 00112 { return R; };
```

4.2.3.5 getX_A()

```
double ConfigClass::getX_A ( ) [inline]
```

a normal method returning a double value.

Returns

A, X axis value

```
Definition at line 48 of file ConfigClass.h. 00048 { return X_A; };
```

4.2.3.6 getX_phi()

```
double ConfigClass::getX_phi ( ) [inline]
```

a normal method returning a double value.

Returns

phi, X axis value

Definition at line 80 of file ConfigClass.h. 00080 { return X_phi; };

4.2.3.7 getX_Rot()

```
double ConfigClass::getX_Rot ( ) [inline]
```

a normal method returning a double value.

Returns

rotation value, X axis

Definition at line 96 of file ConfigClass.h.

```
00096 { return X_Rot; };
```

4.2.3.8 getX_theta()

```
double ConfigClass::getX_theta ( ) [inline]
```

a normal method returning a double value.

Returns

theta, X axis value

Definition at line 64 of file ConfigClass.h.

```
00064 { return X_theta; };
```

4.2.3.9 getY_A()

```
double ConfigClass::getY_A ( ) [inline]
```

a normal method returning a double value.

Returns

A, Y axis value

Definition at line 53 of file ConfigClass.h. 00053 { return Y_A; };

4.2.3.10 getY_phi()

```
double ConfigClass::getY_phi ( ) [inline]
```

a normal method returning a double value.

Returns

phi, Y axis value

Definition at line 85 of file ConfigClass.h. 00085 { return Y_phi; };

4.2.3.11 getY_Rot()

```
double ConfigClass::getY_Rot ( ) [inline]
```

a normal method returning a double value.

Returns

rotation value, Y axis

Definition at line 101 of file ConfigClass.h. 00101 { return Y_Rot; };

4.2.3.12 getY_theta()

```
double ConfigClass::getY_theta ( ) [inline]
```

a normal method returning a double value.

Returns

theta, Y axis value

Definition at line 69 of file ConfigClass.h. 00069 { return Y_theta; };

4.2.3.13 getZ_A()

```
double ConfigClass::getZ_A ( ) [inline]
```

a normal method returning a double value.

Returns

A, Z axis value

Definition at line 58 of file ConfigClass.h. 00058 { return Z_A; };

4.2.3.14 getZ_phi()

```
double ConfigClass::getZ_phi ( ) [inline]
```

a normal method returning a double value.

Returns

phi, Z axis value

Definition at line 90 of file ConfigClass.h.

```
00090 { return Z_phi; };
```

4.2.3.15 getZ_Rot()

```
double ConfigClass::getZ_Rot ( ) [inline]
```

a normal method returning a double value.

Returns

rotation value, Z axis

Definition at line 106 of file ConfigClass.h. 00106 { return Z_Rot; };

4.2.3.16 getZ_theta()

```
double ConfigClass::getZ_theta ( ) [inline]
```

a normal method returning a double value.

Returns

theta, Z axis value

Definition at line 74 of file ConfigClass.h. 00074 { return Z_theta; };

4.2.3.17 Set_Animation()

```
void ConfigClass::Set_Animation (
          bool a ) [inline]
```

a normal method taking one argument and setting animation variable

Parameters

a bool value, is the animation radiobox checked

Definition at line 123 of file ConfigClass.h. 00123 $\{$ animation = a; $\}$

4.2.3.18 Set_ParamType()

```
void ConfigClass::Set_ParamType (
    bool a) [inline]
```

a normal method taking one argument and setting polar variable

Parameters

a bool value, is the polar radiobox checked

Definition at line 128 of file ConfigClass.h. 00128 { polar = a; }

4.2.3.19 Set Points()

```
void ConfigClass::Set_Points (
          bool a ) [inline]
```

a normal method taking one argument and setting points variable

Parameters

a bool value, is the dots radiobox checked

Definition at line 118 of file ConfigClass.h. 00118 { points = a; }

4.2.3.20 SetR()

a normal method taking one argument and setting R variable

Parameters

nR | int value, R value

Definition at line 198 of file ConfigClass.h. 00198 { R = nR; };

4.2.3.21 SetX_A()

a normal method taking one argument and setting X_A variable

Parameters

```
a int value, A value, X axis
```

Definition at line 150 of file ConfigClass.h. 00150 { X_A = a; };

4.2.3.22 SetX_phi()

a normal method taking one argument and setting X_phi variable

Parameters

```
phi int value, phi value, X axis
```

Definition at line 182 of file ConfigClass.h. 00182 { X_phi = phi; };

4.2.3.23 SetX Rot()

a normal method taking one argument and setting X_Rot variable

Parameters

a int value, rotation value, X axis

Definition at line 134 of file ConfigClass.h. 00134 { $X_{\text{Rot}} = a$; }

4.2.3.24 SetX_theta()

a normal method taking one argument and setting X_theta variable

Parameters

theta int value, theta value, X axis

Definition at line 166 of file ConfigClass.h. 00166 { X_theta = theta; };

4.2.3.25 SetY_A()

a normal method taking one argument and setting Y_A variable

Parameters

```
a int value, A value, Y axis
```

Definition at line 155 of file ConfigClass.h. 00155 { Y A = a; };

4.2.3.26 SetY_phi()

a normal method taking one argument and setting Y_phi variable

Parameters

```
phi int value, phi value, Y axis
```

Definition at line 187 of file ConfigClass.h. 00187 { Y_phi = phi; };

4.2.3.27 SetY_Rot()

a normal method taking one argument and setting Y_Rot variable

Parameters

```
a int value, rotation value, Y axis
```

```
Definition at line 139 of file ConfigClass.h. 00139 { Y_Rot = a; }
```

4.2.3.28 SetY_theta()

a normal method taking one argument and setting Y_theta variable

Parameters

```
theta int value, theta value, Y axis
```

```
Definition at line 171 of file ConfigClass.h. 00171 { Y_theta = theta; };
```

4.2.3.29 SetZ A()

a normal method taking one argument and setting Z_A variable

Parameters

```
a int value, A value, Z axis
```

```
Definition at line 160 of file ConfigClass.h. 00160 { Z_A = a; };
```

4.2.3.30 SetZ_phi()

a normal method taking one argument and setting Z_phi variable

Parameters

```
phi int value, phi value, Z axis
```

Definition at line 192 of file ConfigClass.h. 00192 { Z_phi = phi; };

4.2.3.31 SetZ Rot()

a normal method taking one argument and setting Z_Rot variable

Parameters

```
a \mid \text{int value, rotation value, Z axis}
```

Definition at line 144 of file ConfigClass.h. 00144 { Z_Rot = a; }

4.2.3.32 SetZ theta()

a normal method taking one argument and setting Z_theta variable

Parameters

theta int value, theta value, Z axis

Definition at line 176 of file ConfigClass.h. 00176 { Z_theta = theta; };

4.2.4 Member Data Documentation

4.2.4.1 animation

bool ConfigClass::animation [protected]

Definition at line 21 of file ConfigClass.h.

4.2.4.2 MainWindow

```
GUIMyFrame* ConfigClass::MainWindow [protected]
```

Definition at line 14 of file ConfigClass.h.

4.2.4.3 points

bool ConfigClass::points [protected]

Definition at line 22 of file ConfigClass.h.

4.2.4.4 polar

bool ConfigClass::polar [protected]

Definition at line 23 of file ConfigClass.h.

4.2.4.5 R

double ConfigClass::R [protected]

Definition at line 20 of file ConfigClass.h.

4.2.4.6 x

double ConfigClass::x [protected]

Definition at line 19 of file ConfigClass.h.

4.2.4.7 X_A

double ConfigClass::X_A [protected]

Definition at line 15 of file ConfigClass.h.

4.2.4.8 X_phi

```
double ConfigClass::X_phi [protected]
```

Definition at line 17 of file ConfigClass.h.

4.2.4.9 X_Rot

```
double ConfigClass::X_Rot [protected]
```

Definition at line 18 of file ConfigClass.h.

4.2.4.10 X_theta

```
double ConfigClass::X_theta [protected]
```

Definition at line 16 of file ConfigClass.h.

4.2.4.11 y

```
double ConfigClass::y [protected]
```

Definition at line 19 of file ConfigClass.h.

4.2.4.12 Y A

```
double ConfigClass::Y_A [protected]
```

Definition at line 15 of file ConfigClass.h.

4.2.4.13 Y_phi

```
double ConfigClass::Y_phi [protected]
```

Definition at line 17 of file ConfigClass.h.

4.2.4.14 Y_Rot

```
double ConfigClass::Y_Rot [protected]
```

Definition at line 18 of file ConfigClass.h.

4.2.4.15 Y_theta

```
double ConfigClass::Y_theta [protected]
```

Definition at line 16 of file ConfigClass.h.

4.2.4.16 z

```
double ConfigClass::z [protected]
```

Definition at line 19 of file ConfigClass.h.

4.2.4.17 Z_A

```
double ConfigClass::Z_A [protected]
```

Definition at line 15 of file ConfigClass.h.

4.2.4.18 Z phi

```
double ConfigClass::Z_phi [protected]
```

Definition at line 17 of file ConfigClass.h.

4.2.4.19 Z_Rot

double ConfigClass::Z_Rot [protected]

Definition at line 18 of file ConfigClass.h.

4.2.4.20 Z_theta

```
double ConfigClass::Z_theta [protected]
```

Definition at line 16 of file ConfigClass.h.

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

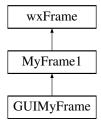
- include/ConfigClass.h
- src/ConfigClass.cpp

4.3 GUIMyFrame Class Reference

Derived class of MyFrame1, with methods that edit the curve.

```
#include <GUIMyFrame.h>
```

Inheritance diagram for GUIMyFrame:



Public Member Functions

- GUIMyFrame (wxWindow *parent)
 - Constructor.
- void Repaint ()

a normal method repainting the curve.

• ∼GUIMyFrame ()

Default destructor.

Public Attributes

• std::shared_ptr< ConfigClass > cfg

Protected Member Functions

void MainFormClose (wxCloseEvent &event)

Closes the window.

void DisplayPanelRepaint (wxUpdateUIEvent &event)

a normal member taking one argument and repainting main panel.

void XRot_Updated (wxScrollEvent &event)

a normal member taking one argument and updating X rotation.

• void YRot_Updated (wxScrollEvent &event)

a normal member taking one argument and updating Y rotation.

void ZRot_Updated (wxScrollEvent &event)

a normal member taking one argument and updating Z rotation.

void XA_Updated (wxCommandEvent &event)

a normal member taking one argument anad updating X amplitude.

void YA_Updated (wxCommandEvent &event)

a normal member taking one argument anad updating Y amplitude.

void ZA Updated (wxCommandEvent &event)

a normal member taking one argument anad updating Z amplitude.

void XTheta Updated (wxCommandEvent &event)

a normal member taking one argument anad updating X theta.

void YTheta_Updated (wxCommandEvent &event)

a normal member taking one argument anad updating Y theta.

void ZTheta_Updated (wxCommandEvent &event)

a normal member taking one argument anad updating Z theta.

void XPhi_Updated (wxCommandEvent &event)

a normal member taking one argument anad updating X phi.

void YPhi Updated (wxCommandEvent &event)

a normal member taking one argument anad updating Y phi.

void ZPhi_Updated (wxCommandEvent &event)

a normal member taking one argument anad updating Z phi.

void DotsLines_Updated (wxCommandEvent &event)

a normal method updating the drawing method of the curve (dotes/lines).

void Animation Updated (wxCommandEvent &event)

a normal method updating the drawing method of the curve (animated/static).

void ParamType_Updated (wxCommandEvent &event)

a normal method updating the drawing method of the curve (cartesian/polar).

Friends

class ChartClass

Additional Inherited Members

4.3.1 Detailed Description

Derived class of MyFrame1, with methods that edit the curve.

Definition at line 19 of file GUIMyFrame.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 GUIMyFrame()

Constructor.

Parameters

```
parent is a wxWindow
```

```
Definition at line 5 of file GUIMyFrame.cpp.
```

4.3.2.2 ∼GUIMyFrame()

```
GUIMyFrame::\sim GUIMyFrame ( )
```

Default destructor.

Definition at line 11 of file GUIMyFrame.cpp. ${\tt 00012} \ \{\,\}$

4.3.3 Member Function Documentation

4.3.3.1 Animation_Updated()

a normal method updating the drawing method of the curve (animated/static).

Parameters

```
event is a wxCommandEvent
```

Reimplemented from MyFrame1.

Definition at line 147 of file GUIMyFrame.cpp.

4.3.3.2 DisplayPanelRepaint()

a normal member taking one argument and repainting main panel.

Parameters

```
event is a wxUpdateUIEvent
```

Reimplemented from MyFrame1.

Definition at line 196 of file GUIMyFrame.cpp.

```
00197 {
00198
            static int i = 0;
00199
            if (cfg->get_Animation())
00200
00201
                if (i % 4 == 0 && cfg->get_Points())
00202
                     Repaint();
00203
                if (!cfg->get_Points())
    Repaint();
if (i > 1000) i = 0;
00204
00205
00206
00207
00208
           else
00209
                Repaint();
00210 }
```

4.3.3.3 DotsLines_Updated()

a normal method updating the drawing method of the curve (dotes/lines).

Parameters

```
event is a wxCommandEvent
```

Reimplemented from MyFrame1.

Definition at line 136 of file GUIMyFrame.cpp.

```
00137 {
00138     if (m_radioBox1->GetSelection() == 0) {
```

4.3.3.4 MainFormClose()

Closes the window.

Parameters

```
event is a wxCloseEvent
```

Reimplemented from MyFrame1.

Definition at line 14 of file GUIMyFrame.cpp.

```
00015 {
00016 Destroy();
00017 }
```

4.3.3.5 ParamType_Updated()

a normal method updating the drawing method of the curve (cartesian/polar).

Parameters

```
event is a wxCommandEvent
```

Reimplemented from MyFrame1.

Definition at line 158 of file GUIMyFrame.cpp.

```
00158
00159
          if (ParamBox->GetSelection() == 0) { //uklad kartezjanski
00160
              cfg->Set_ParamType(false);
00161
              m_AYText->SetEditable(true);
00162
              m_AZText->SetEditable(true);
00163
              m_AYText->Clear();
              m_AZText->Clear();
00164
              m_staticText5->SetLabel("A");
00165
              m_staticText53->SetLabel("A");
00166
              m_staticText54->SetLabel("A");
m_staticText51->SetLabel(wxT(""));
00167
00168
              m_staticText51->SetLabel(wxT(""));
00169
              m_staticText51->SetLabel(wxT(""));
00170
              m_staticText19->SetLabel("X");
00171
00172
              m_staticText18->SetLabel("Y");
00173
              m_staticText4->SetLabel("Z");
```

```
00175
00176
               cfg->Set_ParamType(true);
              m_AYText->SetEditable(false);
00177
               m_AYText->Clear();
00178
              m_AYText->AppendText("-");
00179
00180
              m_AZText->SetEditable(false);
00181
               m_AZText->Clear();
00182
               m_AZText->AppendText("-");
               m_staticText5->SetLabel("r");
m_staticText53->SetLabel("-");
00183
00184
               m_staticText54->SetLabel("-");
00185
               m_staticText51->SetLabel(wxT(""));
00186
               m_staticText511->SetLabel(wxT(""));
00187
00188
               m_staticText512->SetLabel(wxT(""));
               m_staticText19->SetLabel("R");
m_staticText18->SetLabel(wxT(""));
00189
00190
00191
               m_staticText4->SetLabel(wxT(""));
00192
00193
           Repaint();
00194 }
```

4.3.3.6 Repaint()

```
void GUIMyFrame::Repaint ( )
```

a normal method repainting the curve.

Definition at line 212 of file GUIMyFrame.cpp.

4.3.3.7 XA_Updated()

a normal member taking one argument anad updating X amplitude.

Parameters

```
event is a wxCommandEvent
```

Reimplemented from MyFrame1.

Definition at line 37 of file GUIMyFrame.cpp.

```
00045
00046
              if (m_AXText->GetValue().ToDouble(&v))
00047
00048
             {
00049
                 cfg->SetX_A(v);
00050
                     Repaint();
00051
00052
             else wxBell();
00053
         }
00054 }
```

4.3.3.8 XPhi_Updated()

a normal member taking one argument anad updating X phi.

Parameters

event	is a wxCommandEvent
-------	---------------------

Reimplemented from MyFrame1.

Definition at line 106 of file GUIMyFrame.cpp.

4.3.3.9 XRot_Updated()

a normal member taking one argument and updating X rotation.

Parameters

```
event is a wxScrollEvent
```

Reimplemented from MyFrame1.

Definition at line 19 of file GUIMyFrame.cpp.

4.3.3.10 XTheta_Updated()

a normal member taking one argument anad updating X theta.

Parameters

```
event is a wxCommandEvent
```

Reimplemented from MyFrame1.

Definition at line 76 of file GUIMyFrame.cpp.

```
00076
00077    double v;
00078    if (m_ThXText->GetValue().ToDouble(&v))
00079    {
        cfg->SetX_theta(v);
00081        Repaint();
00082    }
00083    else wxBell();
```

4.3.3.11 YA_Updated()

a normal member taking one argument anad updating Y amplitude.

Parameters

```
event is a wxCommandEvent
```

Reimplemented from MyFrame1.

Definition at line 56 of file GUIMyFrame.cpp.

```
00056
00057     double v;
00058     if (m_AYText->GetValue().ToDouble(&v))
00059     {
          cfg->SetY_A(v);
          Repaint();
00062     }
00063     else wxBell();
00064 }
```

4.3.3.12 YPhi Updated()

a normal member taking one argument anad updating Y phi.

Parameters

event	is a wxCommandEvent
-------	---------------------

Reimplemented from MyFrame1.

Definition at line 116 of file GUIMyFrame.cpp.

```
00116
00117
           double v;
           if (m_PhiYText->GetValue().ToDouble(&v))
00118
00119
00120
               cfg->SetY_phi(v);
00121
              Repaint();
00122
00123
          else wxBell();
00124 }
```

4.3.3.13 YRot Updated()

```
void GUIMyFrame::YRot_Updated (
            wxScrollEvent & event ) [protected], [virtual]
```

a normal member taking one argument and updating Y rotation.

Parameters

```
is a wxScrollEvent
event
```

Reimplemented from MyFrame1.

Definition at line 25 of file GUIMyFrame.cpp.

```
00026 {
00027
          cfg->SetY_Rot(m_YRotationSlider->GetValue());
00028
          Repaint();
00029 }
```

4.3.3.14 YTheta_Updated()

```
void GUIMyFrame::YTheta_Updated (
            wxCommandEvent & event ) [protected], [virtual]
```

a normal member taking one argument anad updating Y theta.

Parameters

```
is a wxCommandEvent
event
```

Reimplemented from MyFrame1.

Definition at line 86 of file GUIMyFrame.cpp.

4.3.3.15 ZA_Updated()

a normal member taking one argument anad updating Z amplitude.

Parameters

```
event is a wxCommandEvent
```

Reimplemented from MyFrame1.

Definition at line 66 of file GUIMyFrame.cpp.

4.3.3.16 ZPhi_Updated()

a normal member taking one argument anad updating Z phi.

Parameters

```
event is a wxCommandEvent
```

Reimplemented from MyFrame1.

Definition at line 126 of file GUIMyFrame.cpp.

4.3.3.17 ZRot_Updated()

a normal member taking one argument and updating Z rotation.

Parameters

```
event is a wxScrollEvent
```

Reimplemented from MyFrame1.

Definition at line 31 of file GUIMyFrame.cpp.

4.3.3.18 ZTheta_Updated()

a normal member taking one argument anad updating Z theta.

Parameters

```
event is a wxCommandEvent
```

Reimplemented from MyFrame1.

Definition at line 96 of file GUIMyFrame.cpp.

4.3.4 Friends And Related Function Documentation

4.3.4.1 ChartClass

```
friend class ChartClass [friend]
```

shared_ptr with ConfigClass type

Definition at line 128 of file GUIMyFrame.h.

4.3.5 Member Data Documentation

4.3.5.1 cfg

```
std::shared_ptr<ConfigClass> GUIMyFrame::cfg
```

Definition at line 127 of file GUIMyFrame.h.

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

- include/GUIMyFrame.h
- src/GUIMyFrame.cpp

4.4 Matrix4 Class Reference

a matrix class.

```
#include <vecmat.h>
```

Public Member Functions

- Matrix4 ()
- void Print (void)
- Matrix4 operator* (const Matrix4)

Public Attributes

• double data [4][4]

Friends

• Vector4 operator* (const Matrix4, const Vector4)

4.4.1 Detailed Description

a matrix class.

Definition at line 30 of file vecmat.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Matrix4()

```
Matrix4::Matrix4 ( )
```

Definition at line 49 of file vecmat.cpp.

4.4.3 Member Function Documentation

4.4.3.1 operator*()

Definition at line 65 of file vecmat.cpp.

```
00066 {
           int i, j, k;
Matrix4 tmp;
00067
00068
00069
           for (i = 0; i < 4; i++)
    for (j = 0; j < 4; j++)</pre>
00070
00071
00072
                     tmp.data[i][j] = 0.0;
00073
00074
                     for (k = 0; k < 4; k++)
00075
                         tmp.data[i][j] = tmp.data[i][j] + (data[i][k] * gMatrix.data[k][j]);
00076
               }
00077
            return tmp;
00078 }
```

4.4.3.2 Print()

```
void Matrix4::Print (
     void )
```

Definition at line 57 of file vecmat.cpp.

```
00058 {
00059    printf("\n|%2.31f,%2.31f,%2.31f,\n", data[0][0], data[0][1], data[0][2], data[0][3]);
00060    printf("\$2.31f,%2.31f,%2.31f,\n", data[1][0], data[1][1], data[1][2], data[1][3]);
00061    printf("\$2.31f,%2.31f,%2.31f,%2.31f\\n", data[2][0], data[2][1], data[2][2], data[2][3]);
00062    printf("\$2.31f,%2.31f,%2.31f,%2.31f,\n", data[3][0], data[3][1], data[3][2], data[2][3]);
00063 }
```

4.4.4 Friends And Related Function Documentation

4.4.4.1 operator*

Definition at line 80 of file vecmat.cpp.

```
00082
          unsigned int i, j;
00083
          Vector4 tmp;
00084
00085
          for (i = 0; i < 4; i++)</pre>
00086
00087
              tmp.data[i] = 0.0;
88000
              for (j = 0; j < 4; j++) tmp.data[i] = tmp.data[i] + (gMatrix.data[i][j] * gVector.data[j]);</pre>
00089
00090
          return tmp;
00091 }
```

4.4.5 Member Data Documentation

4.4.5.1 data

```
double Matrix4::data[4][4]
```

Definition at line 33 of file vecmat.h.

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

- include/vecmat.h
- src/vecmat.cpp

4.5 MyApp Class Reference

Inheritance diagram for MyApp:



Public Member Functions

- virtual bool OnInit ()
- virtual int OnExit ()

4.5.1 Detailed Description

Definition at line 4 of file main.cpp.

4.5.2 Member Function Documentation

4.5.2.1 OnExit()

```
virtual int MyApp::OnExit ( ) [inline], [virtual]
Definition at line 8 of file main.cpp.
00008 { return 0; }
```

4.5.2.2 OnInit()

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

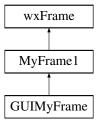
main.cpp

4.6 MyFrame1 Class Reference

Class MyFrame1.

```
#include <Window.h>
```

Inheritance diagram for MyFrame1:



Public Member Functions

MyFrame1 (wxWindow *parent, wxWindowID id=wxID_ANY, const wxString &title=wxEmptyString, const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxSize(1055, 612), long style=wxDEFAULT_← FRAME_STYLE|wxTAB_TRAVERSAL)

Constructor. Connects all of the events handlers. Initializes objects, and text,.

∼MyFrame1 ()

Default destructor. Disconnects all of the events handlers.

Public Attributes

wxPanel * m_DisplayWindow

Protected Member Functions

• virtual void MainFormClose (wxCloseEvent &event)

a pure virtual member.

virtual void DisplayPanelRepaint (wxUpdateUIEvent &event)

a pure virtual member.

· virtual void Scrolls_Updated (wxScrollEvent &event)

a pure virtual member.

virtual void XRot_Updated (wxScrollEvent &event)

a pure virtual member.

virtual void YRot_Updated (wxScrollEvent &event)

a pure virtual member.

virtual void ZRot_Updated (wxScrollEvent &event)

a pure virtual member.

virtual void XA Updated (wxCommandEvent &event)

a pure virtual member.

virtual void YA_Updated (wxCommandEvent &event)

a pure virtual member.

virtual void ZA_Updated (wxCommandEvent &event)

a pure virtual member.

virtual void XTheta_Updated (wxCommandEvent &event)

a pure virtual member.

virtual void YTheta_Updated (wxCommandEvent &event)

a pure virtual member.

virtual void ZTheta_Updated (wxCommandEvent &event)

a pure virtual member.

virtual void XPhi_Updated (wxCommandEvent &event)

a pure virtual member.

virtual void YPhi_Updated (wxCommandEvent &event)

a pure virtual member.

virtual void ZPhi_Updated (wxCommandEvent &event)

a pure virtual member.

virtual void DotsLines_Updated (wxCommandEvent &event)

a pure virtual member.

virtual void Animation_Updated (wxCommandEvent &event)

a pure virtual member.

virtual void AnimationBreak (wxCommandEvent &event)

a pure virtual member.

virtual void ParamType_Updated (wxCommandEvent &event)

a pure virtual member.

Protected Attributes

- wxStaticText * m staticText20
- wxStaticText * m_staticText21
- wxStaticText * m_staticText22
- wxStaticText * m staticText23
- wxStaticText * m_staticText17
- wxStaticText * m staticText19
- wxStaticText * m_staticText5
- wxStaticText * m_staticText51
- wxStaticText * m staticText52
- wxStaticText * m_staticText18
- wxStaticText * m staticText53
- wxStaticText * m_staticText511
- wxStaticText * m_staticText521
- wxStaticText * m_staticText4
- wxStaticText * m staticText54
- wxStaticText * m staticText512
- wxStaticText * m_staticText522
- wxSlider * m XRotationSlider
- wxSlider * m_YRotationSlider
- wxSlider * m ZRotationSlider
- wxRadioBox * animationBox
- wxRadioBox * m_radioBox1
- wxRadioBox * ParamBox
- wxTextCtrl * m_AXText
- wxTextCtrl * m_ThXText
- wxTextCtrl * m PhiXText
- wxTextCtrl * m AYText
- wxTextCtrl * m_ThYText
- wxTextCtrl * m_PhiYText
- wxTextCtrl * m_AZText
- wxTextCtrl * m_ThZText
- wxTextCtrl * m_PhiZText
- wxTimer m_timer

Friends

· class GUIMyFrame

4.6.1 Detailed Description

Class MyFrame1.

a base class for the app

Definition at line 36 of file Window.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 MyFrame1()

Constructor. Connects all of the events handlers. Initializes objects, and text,.

Definition at line 12 of file Window.cpp.

```
00012
                             : wxFrame(parent, id, title, pos, size, style), m_timer(this, 1)
00013 {
00014
          SetTitle(_("Linie Lissajous - Natalia Przetocka, Karolina Klimek i Mateusz Lewandowski"));
00015
          this->SetSizeHints(wxDefaultSize, wxDefaultSize);
00016
00017
          wxBoxSizer* bSizer1;
00018
          bSizer1 = new wxBoxSizer(wxHORIZONTAL);
00019
00020
          wxBoxSizer* bSizer2;
00021
          bSizer2 = new wxBoxSizer(wxVERTICAL);
00022
00023
          wxBoxSizer* bSizer30;
00024
          bSizer30 = new wxBoxSizer(wxVERTICAL);
00025
00026
          m_DisplayWindow = new wxPanel(this, wxID_ANY, wxDefaultPosition, wxDefaultSize, wxTAB_TRAVERSAL);
00027
          m_DisplayWindow->SetBackgroundColour(wxSystemSettings::GetColour(wxSYS_COLOUR_BTNHIGHLIGHT));
00028
00029
          bSizer30->Add(m_DisplayWindow, 5, wxALL | wxEXPAND, 5);
00030
00031
          bSizer2->Add(bSizer30, 30, wxEXPAND, 5);
00032
00033
          wxBoxSizer* bSizer37;
00034
          bSizer37 = new wxBoxSizer(wxHORIZONTAL);
00035
00036
          wxBoxSizer* bSizer38;
00037
          bSizer38 = new wxBoxSizer(wxVERTICAL);
00038
00039
          wxBoxSizer* bSizer36;
00040
          bSizer36 = new wxBoxSizer(wxVERTICAL);
00041
00042
          m_staticText20 = new wxStaticText(this, wxID_ANY, wxT("Rotacja"), wxDefaultPosition,
       wxDefaultSize, 0);
00043
         m_staticText20->Wrap(-1);
          m_staticText20->SetFont(wxFont(12, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00044
00045
00046
          bSizer36->Add(m_staticText20, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00047
00048
          bSizer38->Add(bSizer36, 0, wxEXPAND, 5);
00049
00050
          wxBoxSizer* bSizer32;
00051
          bSizer32 = new wxBoxSizer(wxHORIZONTAL);
00052
00053
          wxBoxSizer* bSizer33;
00054
          bSizer33 = new wxBoxSizer(wxVERTICAL);
00055
00056
          wxBoxSizer* bSizer40;
00057
          bSizer40 = new wxBoxSizer(wxVERTICAL);
00058
          m_staticText21 = new wxStaticText(this, wxID_ANY, wxT("O X"), wxDefaultPosition, wxDefaultSize,
00059
       0);
00060
          m_staticText21->Wrap(-1);
00061
          m_staticText21->SetFont(wxFont(12, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00062
          bSizer40->Add (m staticText21, 0, wxALL | wxALIGN CENTER HORIZONTAL, 5);
00063
00064
00065
          bSizer33->Add(bSizer40, 0, wxEXPAND, 5);
00066
00067
          wxBoxSizer* bSizer42;
00068
          bSizer42 = new wxBoxSizer(wxVERTICAL);
00069
00070
          m XRotationSlider = new wxSlider(this, wxID ANY, 0, 0, 360, wxDefaultPosition, wxSize(-1, -1),
       wxSL_HORIZONTAL | wxSL_VALUE_LABEL);
00071
          m_XRotationSlider->SetMaxSize(wxSize(300, -1));
00072
```

```
00073
          bSizer42->Add(m_XRotationSlider, 0, wxALIGN_CENTER_HORIZONTAL | wxALIGN_CENTER_VERTICAL | wxALL |
00074
00075
          bSizer33->Add(bSizer42, 0, wxEXPAND, 5);
00076
00077
          bSizer32->Add(bSizer33, 1, wxEXPAND, 5);
00078
00079
          wxBoxSizer* bSizer34;
00080
          bSizer34 = new wxBoxSizer(wxVERTICAL);
00081
          m_staticText22 = new wxStaticText(this, wxID_ANY, wxT("O Y"), wxDefaultPosition, wxDefaultSize,
00082
       0);
00083
          m staticText22->Wrap(-1);
          m_staticText22->SetFont(wxFont(12, wxFoNTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00084
       false, wxT("Cambria")));
00085
          bSizer34->Add(m staticText22, 0, wxALL | wxALIGN CENTER HORIZONTAL, 5);
00086
00087
00088
          m_YRotationSlider = new wxSlider(this, wxID_ANY, 0, 0, 360, wxDefaultPosition, wxDefaultSize,
       wxSL_HORIZONTAL | wxSL_VALUE_LABEL);
00089
          m_YRotationSlider->SetMaxSize(wxSize(300, -1));
00090
00091
          bSizer34->Add(m_YRotationSlider, 0, wxALIGN_CENTER_HORIZONTAL | wxALL | wxEXPAND, 5);
00092
00093
          bSizer32->Add(bSizer34, 1, wxEXPAND, 5);
00094
00095
          wxBoxSizer* bSizer35;
00096
          bSizer35 = new wxBoxSizer(wxVERTICAL);
00097
00098
          m_staticText23 = new wxStaticText(this, wxID_ANY, wxT("O Z"), wxDefaultPosition, wxDefaultSize,
       0);
00099
          m staticText23->Wrap(-1);
          m_staticText23->SetFont(wxFont(12, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00100
       false, wxT("Cambria")));
00101
          bSizer35->Add(m_staticText23, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00102
00103
00104
          m_ZRotationSlider = new wxSlider(this, wxID_ANY, 0, 0, 360, wxDefaultPosition, wxDefaultSize,
       wxSL_HORIZONTAL | wxSL_VALUE_LABEL);
00105
          m_ZRotationSlider->SetMaxSize(wxSize(300, -1));
00106
          bSizer35->Add(m ZRotationSlider, 0, wxALIGN CENTER HORIZONTAL | wxALL | wxEXPAND, 5);
00107
00108
00109
          bSizer32->Add(bSizer35, 1, wxEXPAND, 5);
00110
00111
          bSizer38->Add(bSizer32, 0, wxEXPAND, 5);
00112
00113
          bSizer37->Add(bSizer38, 3, wxEXPAND, 5);
00114
00115
          bSizer2->Add(bSizer37, 0, wxEXPAND, 5);
00116
00117
          bSizer1->Add(bSizer2, 7, wxALIGN_RIGHT | wxEXPAND, 5);
00118
00119
          wxBoxSizer* bSizer4;
00120
          bSizer4 = new wxBoxSizer(wxVERTICAL);
00121
          m_staticText17 = new wxStaticText(this, wxID_ANY, wxT("Parametry"), wxDefaultPosition,
00123
         m_staticText17->Wrap(-1);
          m_staticText17->SetFont(wxFont(16, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00124
       false, wxT("Cambria")));
00125
00126
          bSizer4->Add(m_staticText17, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00127
00128
          wxBoxSizer* bSizer361;
00129
          bSizer361 = new wxBoxSizer(wxVERTICAL);
00130
00131
          wxString ParamBoxChoices[] = { wxT("Kartezjaska"), wxT("Biegunowa") };
00132
          int ParamBoxNChoices = sizeof(ParamBoxChoices) / sizeof(wxString);
          ParamBox = new wxRadioBox(this, wxID_ANY, wxT("Rodzaj parametryzacji"), wxDefaultPosition,
00133
       wxDefaultSize, ParamBoxNChoices, ParamBoxChoices, 1, wxRA_SPECIFY_COLS);
          ParamBox->SetSelection(0);
00134
00135
          bSizer361->Add(ParamBox, 0, wxALIGN_CENTER_HORIZONTAL | wxALIGN_CENTER_VERTICAL | wxALL, 5);
00136
00137
00138
          bSizer4->Add(bSizer361, 0, wxEXPAND, 5);
00139
00140
          wxBoxSizer* bSizer9;
00141
          bSizer9 = new wxBoxSizer(wxVERTICAL);
00142
          m staticText19 = new wxStaticText(this, wxID ANY, wxT("X"), wxDefaultPosition, wxDefaultSize, 0);
00143
00144
          m_staticText19->Wrap(-1);
          m_staticText19->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00145
       false, wxT("Cambria")));
00146
00147
          bSizer9->Add(m_staticText19, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00148
```

```
00149
          wxBoxSizer* bSizer10;
00150
          bSizer10 = new wxBoxSizer(wxHORIZONTAL);
00151
00152
          m_staticText5 = new wxStaticText(this, wxID_ANY, wxT("A"), wxDefaultPosition, wxDefaultSize, 0);
00153
          m staticText5->Wrap(-1);
          m_staticText5->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00154
       false, wxT("Cambria")));
00155
00156
          bSizer10->Add(m_staticText5, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00157
          m_AXText = new wxTextCtrl(this, wxID_ANY, wxT("100"), wxDefaultPosition, wxSize(-1, -1), 0);
bSizer10->Add(m_AXText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00158
00159
00160
00161
          bSizer9->Add(bSizer10, 0, wxEXPAND, 5);
00162
00163
          wxBoxSizer* bSizer101;
00164
          bSizer101 = new wxBoxSizer(wxHORIZONTAL);
00165
00166
          m_staticText51 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00167
          m staticText51->Wrap(-1);
          m_staticText51->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00168
       false, wxT("Cambria")));
00169
00170
          bSizer101->Add(m staticText51, 1, wxALL | wxALIGN CENTER VERTICAL, 5);
00171
          m_ThXText = new wxTextCtrl(this, wxID_ANY, wxT("1"), wxDefaultPosition, wxDefaultSize, 0);
00172
00173
          bSizer101->Add(m_ThXText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00174
00175
          bSizer9->Add(bSizer101, 0, wxEXPAND, 5);
00176
00177
          wxBoxSizer* bSizer102:
00178
          bSizer102 = new wxBoxSizer(wxHORIZONTAL);
00179
00180
          m_staticText52 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00181
          m_staticText52->Wrap(-1);
          m_staticText52->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT NORMAL,
00182
       false, wxT("Cambria")));
00183
00184
          bSizer102->Add(m staticText52, 1, wxALL | wxALIGN CENTER VERTICAL, 5);
00185
00186
          m_PhiXText = new wxTextCtrl(this, wxID_ANY, wxT("0"), wxDefaultPosition, wxDefaultSize, 0);
          bSizer102->Add(m_PhiXText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00187
00188
00189
          bSizer9->Add(bSizer102, 0, wxEXPAND, 5);
00190
00191
          bSizer4->Add(bSizer9, 0, wxEXPAND, 5);
00192
00193
          wxBoxSizer* bSizer91;
00194
          bSizer91 = new wxBoxSizer(wxVERTICAL);
00195
00196
          m_staticText18 = new wxStaticText(this, wxID_ANY, wxT("Y"), wxDefaultPosition, wxDefaultSize, 0);
00197
          m_staticText18->Wrap(-1);
00198
          m_staticText18->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
       false, wxT("Cambria")));
00199
00200
          bSizer91->Add (m staticText18, 0, wxALL | wxALIGN CENTER HORIZONTAL, 5);
00201
00202
          wxBoxSizer* bSizer103;
00203
          bSizer103 = new wxBoxSizer(wxHORIZONTAL);
00204
          m_staticText53 = new wxStaticText(this, wxID_ANY, wxT("A"), wxDefaultPosition, wxDefaultSize, 0);
00205
          m_staticText53->Wrap(-1);
00206
00207
          m_staticText53->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
       false, wxT("Cambria")));
00208
00209
          bSizer103->Add(m_staticText53, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00210
          m_AYText = new wxTextCtrl(this, wxID_ANY, wxT("100"), wxDefaultPosition, wxDefaultSize, 0);
00211
00212
          bSizer103->Add(m_AYText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00213
00214
          bSizer91->Add(bSizer103, 1, wxEXPAND, 5);
00215
00216
          wxBoxSizer* bSizer1011;
00217
          bSizer1011 = new wxBoxSizer(wxHORIZONTAL);
00218
00219
          m_staticText511 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00220
          m_staticText511->Wrap(-1);
00221
          m_staticText511->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
       false, wxT("Cambria")));
00222
00223
          bSizer1011->Add(m staticText511, 1, wxALL | wxALIGN CENTER VERTICAL, 5);
00224
            _ThYText = new wxTextCtrl(this, wxID_ANY, wxT("1"), wxDefaultPosition, wxDefaultSize, 0);
00225
00226
          bSizer1011->Add(m_ThYText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00227
00228
          bSizer91->Add(bSizer1011, 1, wxEXPAND, 5);
00229
```

```
00230
          wxBoxSizer* bSizer1021;
00231
          bSizer1021 = new wxBoxSizer(wxHORIZONTAL);
00232
00233
          m_staticText521 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
          m staticText521->Wrap(-1):
00234
          m_staticText521->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00235
       false, wxT("Cambria")));
00236
00237
          bSizer1021->Add(m_staticText521, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00238
          m_PhiYText = new wxTextCtrl(this, wxID_ANY, wxT("1.57"), wxDefaultPosition, wxDefaultSize, 0);
00239
          bSizer1021->Add(m_PhiYText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00240
00241
00242
          bSizer91->Add(bSizer1021, 1, wxEXPAND, 5);
00243
00244
          bSizer4->Add(bSizer91, 0, wxEXPAND, 5);
00245
00246
          wxBoxSizer* bSizer92;
00247
          bSizer92 = new wxBoxSizer(wxVERTICAL);
00248
00249
          m_staticText4 = new wxStaticText(this, wxID_ANY, wxT("Z"), wxDefaultPosition, wxDefaultSize, 0);
00250
          m_staticText4->Wrap(-1);
         m_staticText4->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00251
       false, wxT("Cambria")));
00252
00253
          bSizer92->Add(m_staticText4, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00254
00255
          wxBoxSizer* bSizer104;
          bSizer104 = new wxBoxSizer(wxHORIZONTAL);
00256
00257
00258
          m staticText54 = new wxStaticText(this, wxID ANY, wxT("A"), wxDefaultPosition, wxDefaultSize, 0);
00259
         m_staticText54->Wrap(-1);
          m_staticText54->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00260
       false, wxT("Cambria")));
00261
00262
          bSizer104->Add(m staticText54, 1, wxALL | wxALIGN CENTER VERTICAL, 5);
00263
00264
          m_AZText = new wxTextCtrl(this, wxID_ANY, wxT("100"), wxDefaultPosition, wxDefaultSize, 0);
00265
          bSizer104->Add(m_AZText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00266
00267
          bSizer92->Add(bSizer104, 1, wxEXPAND, 5);
00268
00269
          wxBoxSizer* bSizer1012:
00270
          bSizer1012 = new wxBoxSizer(wxHORIZONTAL);
00271
00272
          m_staticText512 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00273
          m_staticText512->Wrap(-1);
00274
          m_staticText512->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
      false, wxT("Cambria")));
00275
00276
          bSizer1012->Add(m_staticText512, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00277
00278
           _ThZText = new wxTextCtrl(this, wxID_ANY, wxT("0"), wxDefaultPosition, wxDefaultSize, 0);
00279
          bSizer1012->Add(m_ThZText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00280
00281
          bSizer92->Add(bSizer1012, 1, wxEXPAND, 5);
00282
00283
          wxBoxSizer* bSizer1022;
00284
          bSizer1022 = new wxBoxSizer(wxHORIZONTAL);
00285
          m_staticText522 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00286
          m_staticText522->Wrap(-1);
00287
00288
          m_staticText522->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
       false, wxT("Cambria")));
00289
00290
          bSizer1022->Add(m_staticText522, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00291
          m_PhiZText = new wxTextCtrl(this, wxID_ANY, wxT("0"), wxDefaultPosition, wxDefaultSize, 0);
00292
00293
          bSizer1022->Add(m_PhiZText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00294
00295
          bSizer92->Add(bSizer1022, 1, wxEXPAND | wxALIGN_RIGHT, 5);
00296
00297
          bSizer4->Add(bSizer92, 0, wxEXPAND, 5);
00298
00299
          wxBoxSizer* bSizer27;
00300
          bSizer27 = new wxBoxSizer(wxVERTICAL);
00301
00302
          wxString animationBoxChoices[] = { wxT("Statyczna"), wxT("Animowana") };
00303
          int animationBoxNChoices = sizeof(animationBoxChoices) / sizeof(wxString);
          animationBox = new wxRadioBox(this, wxID_ANY, wxT("Rodzaj animacji"), wxDefaultPosition,
00304
       wxDefaultSize, animationBoxNChoices, animationBoxChoices, 1, wxRA_SPECIFY_COLS);
          animationBox->SetSelection(0);
00305
00306
          bSizer27->Add(animationBox, 0, wxALL | wxALIGN CENTER HORIZONTAL, 5);
00307
00308
          wxString m_radioBox1Choices[] = { wxT("Linie"), wxT("Punkty") };
          int m_radioBox1NChoices = sizeof(m_radioBox1Choices) / sizeof(wxString);
00309
00310
          m_radioBox1 = new wxRadioBox(this, wxID_ANY, wxT("Rodzaj rysowania"), wxDefaultPosition,
```

```
wxDefaultSize, m_radioBox1NChoices, m_radioBox1Choices, 1, wxRA_SPECIFY_COLS);
00311
              m_radioBox1->SetSelection(0);
00312
              m_radioBox1->SetFont(wxFont(10, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
          false, wxT("Cambria")));
00313
              bSizer27->Add(m_radioBox1, 0, wxALL | wxALIGN_CENTER_HORIZONTAL | wxALIGN_CENTER_VERTICAL, 5);
00314
00315
00316
              bSizer4->Add(bSizer27, 1, wxEXPAND, 5);
00317
00318
              bSizer1->Add(bSizer4, 1, wxALIGN LEFT | wxEXPAND, 5);
00319
00320
              this->SetSizer(bSizer1);
00321
              this->Layout();
00322
00323
              this->Centre(wxBOTH);
00324
00325
               //connect-v
              this->Connect (wxEVT CLOSE WINDOW, wxCloseEventHandler(MvFrame1::MainFormClose));
00326
00327
              m_DisplayWindow->Connect(wxEVT_UPDATE_UI, wxUpdateUIEventHandler(MyFrame1::DisplayPanelRepaint),
          NULL, this);
00328
00329
              m_XRotationSlider->Connect(wxEVT_SCROLL_TOP, wxScrollEventHandler(MyFrame1::XRot_Updated), NULL,
          this);
              m_XRotationSlider->Connect(wxEVT_SCROLL_BOTTOM, wxScrollEventHandler(MyFramel::XRot_Updated),
00330
          NULL, this);
00331
              m_XRotationSlider->Connect(wxEVT_SCROLL_LINEUP, wxScrollEventHandler(MyFramel::XRot_Updated),
          NULL, this);
00332
              m_XRotationSlider->Connect(wxEVT_SCROLL_LINEDOWN, wxScrollEventHandler(MyFramel::XRot_Updated),
          NULL, this);
00333
              m XRotationSlider->Connect(wxEVT SCROLL PAGEUP, wxScrollEventHandler(MvFrame1::XRot Updated),
          NULL, this);
00334
              m_XRotationSlider->Connect(wxEVT_SCROLL_PAGEDOWN, wxScrollEventHandler(MyFramel::XRot_Updated),
          NULL, this);
00335
              m_XRotationSlider->Connect(wxEVT_SCROLL_THUMBTRACK, wxScrollEventHandler(MyFramel::XRot_Updated),
          NULL, this);
00336
              m_XRotationSlider->Connect(wxEVT_SCROLL THUMBRELEASE,
          wxScrollEventHandler(MyFrame1::XRot Updated), NULL, this);
              m_XRotationSlider->Connect(wxEVT_SCROLL_CHANGED, wxScrollEventHandler(MyFramel::XRot_Updated),
00337
          NULL, this);
              m_YRotationSlider->Connect(wxEVT_SCROLL_TOP, wxScrollEventHandler(MyFramel::YRot_Updated), NULL,
00338
          this);
00339
              m_YRotationSlider->Connect(wxEVT_SCROLL_BOTTOM, wxScrollEventHandler(MyFramel::YRot_Updated),
          NULL, this):
00340
              m_YRotationSlider->Connect(wxEVT_SCROLL_LINEUP, wxScrollEventHandler(MyFramel::YRot_Updated),
          NULL, this);
00341
               m_YRotationSlider->Connect(wxEVT_SCROLL_LINEDOWN, wxScrollEventHandler(MyFramel::YRot_Updated),
          NULL, this);
00342
              NULL, this);
00343
              m YRotationSlider->Connect(wxEVT SCROLL PAGEDOWN, wxScrollEventHandler(MyFrame1::YRot Updated),
          NULL, this);
00344
              m_YRotationSlider->Connect(wxEVT_SCROLL_THUMBTRACK, wxScrollEventHandler(MyFrame1::YRot_Updated),
          NULL, this);
00345
              m_YRotationSlider->Connect(wxEVT_SCROLL_THUMBRELEASE,
          wxScrollEventHandler(MyFrame1::YRot_Updated), NULL, this);
              m_YRotationSlider->Connect(wxEVT_SCROLL_CHANGED, wxScrollEventHandler(MyFrame1::YRot_Updated),
00346
          NULL, this);
00347
              m ZRotationSlider->Connect(wxEVT SCROLL TOP, wxScrollEventHandler(MvFrame1::ZRot Updated), NULL,
          this);
              {\tt m\_ZRotationSlider->Connect (wxEVT\_SCROLL\_BOTTOM, wxScrollEventHandler (MyFramel::ZRot\_Updated), m_ZRotationSlider->Connect (wxEVT\_SCROLL\_BOTTOM, wxScrollEventHandler (MyFramel::ZRot_Updated), m_ZRotationSlider->Connect (wxEVT\_SCROLL_BOTTOM, wxScrollEventHandler (MyFramel::ZRot_Updated), m_ZRotationSlider->Connect (wxEVT\_SCROLL_BOTTOM, wxScrollEventHandler (MyFramel::ZRot_Updated), m_ZRotationSlider->Connect (wxEVT\_SCROLL_BOTTOM, wxScrollEventHandler (WyFramel::ZRot_Updated), m_ZRotationSlider->Connect (wxEVT\_SCROLL_BOTTOM, wxScrollEventHandler (WyFramel::ZRot_Updated), m_ZRotationSlider->Connect (wxEVT\_SCROLL_BOTTOM, wxScrollEventHandler (wxEVT\_SCROLL_BOTTOM, wxScrollEventHandler (wxEVT\_S
00348
          NULL, this);
00349
              m ZRotationSlider->Connect(wxEVT SCROLL LINEUP, wxScrollEventHandler(MvFrame1::ZRot Updated),
          NULL, this);
00350
              m ZRotationSlider->Connect(wxEVT SCROLL LINEDOWN, wxScrollEventHandler(MyFrame1::ZRot Updated),
          NULL, this);
00351
              m_ZRotationSlider->Connect(wxEVT_SCROLL_PAGEUP, wxScrollEventHandler(MyFrame1::ZRot_Updated),
          NULL, this);
00352
              m ZRotationSlider->Connect(wxEVT SCROLL PAGEDOWN, wxScrollEventHandler(MvFrame1::ZRot Updated),
          NULL, this);
00353
              m_ZRotationSlider->Connect(wxEVT_SCROLL_THUMBTRACK, wxScrollEventHandler(MyFrame1::ZRot_Updated),
          NULL, this);
00354
              m_ZRotationSlider->Connect(wxEVT_SCROLL_THUMBRELEASE,
          wxScrollEventHandler(MyFrame1::ZRot_Updated), NULL, this);
00355
              m ZRotationSlider->Connect(wxEVT SCROLL CHANGED, wxScrollEventHandler(MvFrame1::ZRot Updated),
          NULL, this);
00356
              m_AXText->Connect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFramel::XA_Updated), NULL,
00357
          this);
00358
              m_AYText->Connect (wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler (MyFrame1::YA_Updated), NULL,
          this):
00359
              m AZText->Connect (wxEVT COMMAND TEXT UPDATED, wxCommandEventHandler(MyFrame1::ZA Updated), NULL,
          this);
00360
00361
              m_ThXText->Connect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::XTheta_Updated),
          NULL, this);
00362
              m_ThYText->Connect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFramel::YTheta_Updated),
          NULL, this);
```

```
00363
          m_ThZText->Connect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::ZTheta_Updated),
       NULL, this);
00364
00365
          m_PhiXText->Connect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::XPhi_Updated),
       NULL, this);
          m_PhiYText
00366
                     ->Connect(wxEVT COMMAND TEXT UPDATED, wxCommandEventHandler(MvFramel::YPhi Updated),
       NULL, this);
00367
                     ->Connect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::ZPhi_Updated),
       NULL, this);
00368
00369
          m_radioBox1->Connect(wxEVT_COMMAND_RADIOBOX_SELECTED,
       wxCommandEventHandler(MyFramel::DotsLines_Updated), NULL, this);
00370
          animationBox->Connect(wxEVT_COMMAND_RADIOBOX_SELECTED,
       wxCommandEventHandler(MyFramel::Animation_Updated), NULL, this);
00371
          ParamBox->Connect (wxEVT_COMMAND_RADIOBOX_SELECTED,
       wxCommandEventHandler(MyFrame1::ParamType_Updated), NULL, this);
00372
00373
          m timer.Start(10);
00374 }
```

4.6.2.2 \sim MyFrame1()

```
MyFramel::~MyFramel ()
```

Default destructor. Disconnects all of the events handlers.

Definition at line 376 of file Window.cpp.

```
00378
                          //disconnect-v
00379
                         this->Disconnect(wxEVT CLOSE WINDOW, wxCloseEventHandler(MyFramel::MainFormClose));
00380
                         m_DisplayWindow->Disconnect(wxEVT_UPDATE_UI,
                 wxUpdateUIEventHandler(MyFrame1::DisplayPanelRepaint), NULL, this);
00381
00382
                         m XRotationSlider->Disconnect(wxEVT SCROLL TOP, wxScrollEventHandler(MvFrame1::XRot Updated),
                 NULL, this);
                         m_XRotationSlider->Disconnect(wxEVT_SCROLL_BOTTOM, wxScrollEventHandler(MyFrame1::XRot_Updated),
00383
                 NULL, this);
00384
                         m XRotationSlider->Disconnect(wxEVT SCROLL LINEUP, wxScrollEventHandler(MyFrame1::XRot Updated),
                 NULL, this);
00385
                         m XRotationSlider->Disconnect(wxEVT SCROLL LINEDOWN, wxScrollEventHandler(MyFrame1::XRot Updated),
                 NULL, this);
00386
                         m_XRotationSlider->Disconnect(wxEVT_SCROLL_PAGEUP, wxScrollEventHandler(MyFrame1::XRot_Updated),
                 NULL, this);
00387
                         m_XRotationSlider->Disconnect(wxEVT_SCROLL_PAGEDOWN, wxScrollEventHandler(MyFrame1::XRot_Updated),
                 NULL, this);
00388
                        m XRotationSlider->Disconnect(wxEVT SCROLL THUMBTRACK,
                 wxScrollEventHandler(MyFrame1::XRot_Updated), NULL, this);
                         m_XRotationSlider->Disconnect(wxEVT_SCROLL_THUMBRELEASE,
00389
                 wxScrollEventHandler(MyFrame1::XRot_Updated), NULL, this);
00390
                        \verb|m_XRotationSlider-> \hline| Disconnect (wxEVT\_SCROLL\_CHANGED, wxScrollEventHandler (MyFrame1::XRot\_Updated), wxScrollEventHandler (MyFrame1::XRot\_Updated), wxScrollEventHandler (MyFrame1::XRot_Updated), wxScrollEve
                 NULL, this);
00391
                         m_YRotationSlider->Disconnect(wxEVT_SCROLL_TOP, wxScrollEventHandler(MyFrame1::YRot_Updated),
00392
                 NULL, this);
00393
                         m_YRotationSlider->Disconnect(wxEVT_SCROLL_BOTTOM, wxScrollEventHandler(MyFrame1::YRot_Updated),
                 NULL, this);
00394
                         m_YRotationSlider->Disconnect(wxEVT_SCROLL_LINEUP, wxScrollEventHandler(MyFrame1::YRot_Updated),
                 NULL, this);
00395
                        m YRotationSlider->Disconnect(wxEVT SCROLL LINEDOWN, wxScrollEventHandler(MyFrame1::YRot Updated),
                 NULL, this);
00396
                         m_YRotationSlider->Disconnect(wxEVT_SCROLL_PAGEUP, wxScrollEventHandler(MyFrame1::YRot_Updated),
                 NULL, this);
00397
                         m_YRotationSlider->Disconnect(wxEVT_SCROLL_PAGEDOWN, wxScrollEventHandler(MyFrame1::YRot_Updated),
                 NULL, this);
                         m YRotationSlider->Disconnect(wxEVT SCROLL THUMBTRACK,
00398
                 wxScrollEventHandler(MyFrame1::YRot_Updated), NULL, this);
                         m_YRotationSlider->Disconnect(wxEVT_SCROLL_THUMBRELEASE,
00399
                 wxScrollEventHandler(MyFrame1::YRot_Updated), NULL, this);
00400
                        \verb|m_YRotationSlider-> \hline| Disconnect (wxEVT\_SCROLL\_CHANGED, wxScrollEventHandler (MyFrame1::YRot\_Updated), wxScrollEventHandler (MyFrame1::YRot\_Updated), wxScrollEventHandler (MyFrame1::YRot_Updated), wxScrollEve
                 NULL, this);
00401
                         m_ZRotationSlider->Disconnect(wxEVT_SCROLL_TOP, wxScrollEventHandler(MyFramel::ZRot_Updated),
00402
                 NULL, this);
00403
                         m_ZRotationSlider->Disconnect(wxEVT_SCROLL_BOTTOM, wxScrollEventHandler(MyFrame1::ZRot_Updated),
                 NULL, this);
00404
                         m_ZRotationSlider->Disconnect(wxEVT_SCROLL_LINEUP, wxScrollEventHandler(MyFrame1::ZRot_Updated),
                 NULL, this);
```

```
00405
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_LINEDOWN, wxScrollEventHandler(MyFrame1::ZRot_Updated),
       NULL, this);
00406
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_PAGEUP, wxScrollEventHandler(MyFramel::ZRot_Updated),
       NULL, this);
00407
          m ZRotationSlider->Disconnect(wxEVT SCROLL PAGEDOWN, wxScrollEventHandler(MyFrame1::ZRot Updated),
       NULL, this);
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_THUMBTRACK,
00408
       wxScrollEventHandler(MyFrame1::ZRot_Updated), NULL, this);
00409
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_THUMBRELEASE,
       wxScrollEventHandler(MyFrame1::ZRot_Updated), NULL, this);
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_CHANGED, wxScrollEventHandler(MyFrame1::ZRot_Updated),
00410
       NULL, this):
00411
          m_AXText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::XA_Updated),
       NULL, this);
00413
          m_AYText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::YA_Updated),
       NULL, this);
00414
          m AZText->Disconnect(wxEVT COMMAND TEXT UPDATED, wxCommandEventHandler(MyFrame1::ZA Updated),
       NULL, this);
00415
          m_ThXText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::XTheta_Updated),
00416
       NULL, this);
00417
          m_ThYText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFramel::YTheta_Updated),
       NULL, this);
00418
          m_ThZText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::ZTheta_Updated),
       NULL, this);
00419
00420
          m_PhiXText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFramel::XPhi_Updated),
       NULL, this);
00421
          m PhiYText->Disconnect(wxEVT COMMAND TEXT UPDATED, wxCommandEventHandler(MvFrame1::YPhi Updated),
       NULL, this);
00422
          m_PhiZText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::ZPhi_Updated),
00423
00424
          m_radioBox1->Disconnect(wxEVT_COMMAND_RADIOBOX_SELECTED,
       wxCommandEventHandler(MyFramel::DotsLines_Updated), NULL, this);
          animationBox->Disconnect(wxEVT_COMMAND_RADIOBOX_SELECTED,
00425
       wxCommandEventHandler(MyFramel::Animation_Updated), NULL, this);
00426
          ParamBox->Disconnect(wxEVT_COMMAND_RADIOBOX_SELECTED,
       wxCommandEventHandler(MyFrame1::ParamType_Updated), NULL, this);
00427 }
```

4.6.3 Member Function Documentation

4.6.3.1 Animation Updated()

a pure virtual member.

Parameters

event is a wxCommandEvent

Reimplemented in GUIMyFrame.

Definition at line 163 of file Window.h. 00163 { event.Skip(); }

4.6.3.2 AnimationBreak()

a pure virtual member.

Parameters

```
event is a wxCommandEvent
```

Definition at line 168 of file Window.h. 00168 { event.Skip(); }

4.6.3.3 DisplayPanelRepaint()

a pure virtual member.

Parameters

event	is a wxUpdateUIEvent
-------	----------------------

Reimplemented in GUIMyFrame.

Definition at line 83 of file Window.h. 00083 { event.Skip(); }

4.6.3.4 DotsLines_Updated()

a pure virtual member.

Parameters

Reimplemented in GUIMyFrame.

Definition at line 158 of file Window.h. 00158 { event.Skip(); }

4.6.3.5 MainFormClose()

a pure virtual member.

phi, z axis value textctrl

Parameters

event	is a wxCloseEvent

Reimplemented in GUIMyFrame.

Definition at line 78 of file Window.h. 00078 { event.Skip(); }

4.6.3.6 ParamType_Updated()

a pure virtual member.

Parameters

event	is a wxCommandEvent
-------	---------------------

Reimplemented in GUIMyFrame.

Definition at line 173 of file Window.h. 00173 { event.Skip(); }

4.6.3.7 Scrolls_Updated()

a pure virtual member.

Parameters

event is a wxScrollEvent

Definition at line 88 of file Window.h.

```
00088 { event.Skip(); }
```

4.6.3.8 XA_Updated()

a pure virtual member.

Parameters

event	is a wxCommandEvent
-------	---------------------

Reimplemented in GUIMyFrame.

Definition at line 110 of file Window.h. 00110 { event.Skip(); }

4.6.3.9 XPhi_Updated()

a pure virtual member.

Parameters

```
event is a wxCommandEvent
```

Reimplemented in GUIMyFrame.

Definition at line 142 of file Window.h. 00142 { event.Skip(); }

4.6.3.10 XRot_Updated()

a pure virtual member.

Parameters

```
event is a wxScrollEvent
```

Reimplemented in GUIMyFrame.

```
Definition at line 94 of file Window.h. 00094 { event.Skip(); }
```

4.6.3.11 XTheta_Updated()

a pure virtual member.

Parameters

```
event is a wxCommandEvent
```

Reimplemented in GUIMyFrame.

Definition at line 126 of file Window.h. 00126 { event.Skip(); }

```
4.6.3.12 YA_Updated()
```

a pure virtual member.

Parameters

event	is a wxCommandEvent
-------	---------------------

Reimplemented in GUIMyFrame.

Definition at line 115 of file Window.h. 00115 { event.Skip(); }

4.6.3.13 YPhi_Updated()

a pure virtual member.

Parameters

event	is a wxCommandEvent
-------	---------------------

Reimplemented in GUIMyFrame.

```
Definition at line 147 of file Window.h. 00147 { event. Skip();  }
```

4.6.3.14 YRot_Updated()

a pure virtual member.

Parameters

```
event is a wxScrollEvent
```

Reimplemented in GUIMyFrame.

Definition at line 99 of file Window.h. 00099 { event.Skip(); }

4.6.3.15 YTheta_Updated()

a pure virtual member.

Parameters

```
event is a wxCommandEvent
```

Reimplemented in GUIMyFrame.

```
Definition at line 131 of file Window.h. 00131 { event.Skip(); }
```

4.6.3.16 ZA_Updated()

a pure virtual member.

Parameters

event	is a wxCommandEvent
-------	---------------------

Reimplemented in GUIMyFrame.

```
Definition at line 120 of file Window.h. 00120 { event.Skip();  }
```

4.6.3.17 ZPhi_Updated()

a pure virtual member.

Parameters

```
event is a wxCommandEvent
```

Reimplemented in GUIMyFrame.

```
Definition at line 152 of file Window.h. 00152 { event.Skip(); }
```

4.6.3.18 ZRot_Updated()

a pure virtual member.

Parameters

```
event is a wxScrollEvent
```

Reimplemented in GUIMyFrame.

```
Definition at line 104 of file Window.h. 00104 { event.Skip(); }
```

4.6.3.19 ZTheta_Updated()

a pure virtual member.

Parameters

event	is a wxCommandEvent
-------	---------------------

Reimplemented in GUIMyFrame.

Definition at line 136 of file Window.h. 00136 { event.Skip(); }

4.6.4 Friends And Related Function Documentation

4.6.4.1 GUIMyFrame

```
friend class GUIMyFrame [friend]
```

Definition at line 38 of file Window.h.

4.6.5 Member Data Documentation

4.6.5.1 animationBox

```
wxRadioBox* MyFramel::animationBox [protected]
```

Z rotation slider

Definition at line 60 of file Window.h.

4.6.5.2 m_AXText

```
wxTextCtrl* MyFrame1::m_AXText [protected]
```

adiobox: cartesian or polar coordinates curve

Definition at line 63 of file Window.h.

4.6.5.3 m_AYText

```
wxTextCtrl* MyFramel::m_AYText [protected]
```

phi, x axis value textctrl

Definition at line 66 of file Window.h.

4.6.5.4 m_AZText

```
wxTextCtrl* MyFramel::m_AZText [protected]
```

phi, y axis value textctrl

Definition at line 69 of file Window.h.

4.6.5.5 m_DisplayWindow

```
wxPanel* MyFramel::m_DisplayWindow
```

Definition at line 178 of file Window.h.

4.6.5.6 m_PhiXText

```
wxTextCtrl* MyFramel::m_PhiXText [protected]
```

theta, x axis value textctrl

Definition at line 65 of file Window.h.

4.6.5.7 m_PhiYText

```
wxTextCtrl* MyFramel::m_PhiYText [protected]
```

theta, y axis value textctrl

Definition at line 68 of file Window.h.

4.6.5.8 m_PhiZText

```
wxTextCtrl* MyFramel::m_PhiZText [protected]
```

theta, z axis value textctrl

Definition at line 71 of file Window.h.

4.6.5.9 m_radioBox1

```
wxRadioBox* MyFrame1::m_radioBox1 [protected]
```

radiobox: animated or static curve

Definition at line 61 of file Window.h.

4.6.5.10 m_staticText17

```
wxStaticText* MyFramel::m_staticText17 [protected]
```

Z axis text rotation

Definition at line 44 of file Window.h.

4.6.5.11 m_staticText18

```
wxStaticText* MyFrame1::m_staticText18 [protected]
```

phi parameter text X axis

Definition at line 49 of file Window.h.

4.6.5.12 m_staticText19

```
wxStaticText* MyFramel::m_staticText19 [protected]
```

parameters text

Definition at line 45 of file Window.h.

4.6.5.13 m_staticText20

```
wxStaticText* MyFramel::m_staticText20 [protected]
```

Definition at line 40 of file Window.h.

4.6.5.14 m_staticText21

```
wxStaticText* MyFrame1::m_staticText21 [protected]
```

rotation text

Definition at line 41 of file Window.h.

4.6.5.15 m_staticText22

```
wxStaticText* MyFramel::m_staticText22 [protected]
```

X axis text rotation

Definition at line 42 of file Window.h.

4.6.5.16 m_staticText23

```
wxStaticText* MyFramel::m_staticText23 [protected]
```

Y axis text rotation

Definition at line 43 of file Window.h.

4.6.5.17 m_staticText4

```
wxStaticText* MyFrame1::m_staticText4 [protected]
```

phi parameter text Y axis

Definition at line 53 of file Window.h.

4.6.5.18 m_staticText5

wxStaticText* MyFramel::m_staticText5 [protected]

X axis text parameters

Definition at line 46 of file Window.h.

4.6.5.19 m_staticText51

wxStaticText* MyFramel::m_staticText51 [protected]

A parameter text X axis

Definition at line 47 of file Window.h.

4.6.5.20 m_staticText511

wxStaticText* MyFrame1::m_staticText511 [protected]

A parameter text Y axis

Definition at line 51 of file Window.h.

4.6.5.21 m_staticText512

wxStaticText* MyFrame1::m_staticText512 [protected]

A parameter text Z axis

Definition at line 55 of file Window.h.

4.6.5.22 m_staticText52

wxStaticText* MyFrame1::m_staticText52 [protected]

theta parameter text X axis

Definition at line 48 of file Window.h.

4.6.5.23 m_staticText521

```
wxStaticText* MyFrame1::m_staticText521 [protected]
```

theta parameter text Y axis

Definition at line 52 of file Window.h.

4.6.5.24 m_staticText522

```
wxStaticText* MyFrame1::m_staticText522 [protected]
```

theta parameter text Z axis

Definition at line 56 of file Window.h.

4.6.5.25 m staticText53

```
wxStaticText* MyFramel::m_staticText53 [protected]
```

Y axis text parameters

Definition at line 50 of file Window.h.

4.6.5.26 m_staticText54

```
wxStaticText* MyFrame1::m_staticText54 [protected]
```

Y axis text parameters

Definition at line 54 of file Window.h.

4.6.5.27 m_ThXText

```
wxTextCtrl* MyFrame1::m_ThXText [protected]
```

A, x axis value textctrl

Definition at line 64 of file Window.h.

4.6.5.28 m_ThYText

```
wxTextCtrl* MyFrame1::m_ThYText [protected]
```

A, y axis value textctrl

Definition at line 67 of file Window.h.

4.6.5.29 m_ThZText

```
wxTextCtrl* MyFrame1::m_ThZText [protected]
```

A, z axis value textctrl

Definition at line 70 of file Window.h.

4.6.5.30 m_timer

```
wxTimer MyFramel::m_timer [protected]
```

Definition at line 175 of file Window.h.

4.6.5.31 m_XRotationSlider

```
wxSlider* MyFramel::m_XRotationSlider [protected]
```

phi parameter text Z axis

Definition at line 57 of file Window.h.

4.6.5.32 m_YRotationSlider

```
wxSlider* MyFramel::m_YRotationSlider [protected]
```

X rotation slider

Definition at line 58 of file Window.h.

66 Class Documentation

4.6.5.33 m_ZRotationSlider

```
wxSlider* MyFramel::m_ZRotationSlider [protected]
```

Y rotation slider

Definition at line 59 of file Window.h.

4.6.5.34 ParamBox

```
wxRadioBox* MyFrame1::ParamBox [protected]
```

radiobox: lines or dots curve

Definition at line 62 of file Window.h.

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

- include/Window.h
- src/Window.cpp

4.7 Vector4 Class Reference

a vector class.

```
#include <vecmat.h>
```

Public Member Functions

- Vector4 ()
- void Print (void)
- void Set (double d1, double d2, double d3)
- double GetX ()
- double GetY ()
- double GetZ ()
- Vector4 operator- (const Vector4 &)

Public Attributes

• double data [4]

Friends

• Vector4 operator* (const Vector4 &, double)

4.7.1 Detailed Description

a vector class.

Definition at line 13 of file vecmat.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Vector4()

4.7.3 Member Function Documentation

4.7.3.1 GetX()

```
double Vector4::GetX ( )

Definition at line 18 of file vecmat.cpp.
00019 {
00020     return data[0];
00021 }
```

4.7.3.2 GetY()

```
Definition at line 23 of file vecmat.cpp.

00024 {
00025 return data[1];
```

double Vector4::GetY ()

4.7.3.3 GetZ()

00026 }

```
double Vector4::GetZ ( )

Definition at line 28 of file vecmat.cpp.
00029 {
00030     return data[2];
00031 }
```

68 Class Documentation

4.7.3.4 operator-()

4.7.3.5 Print()

Definition at line 8 of file vecmat.cpp.

```
00009 {
00010    printf("(%2.31f,%2.31f,%2.31f)\n", data[0], data[1], data[2], data[3]);
00011 }
```

4.7.3.6 Set()

Definition at line 13 of file vecmat.cpp.

4.7.4 Friends And Related Function Documentation

4.7.4.1 operator*

Definition at line 41 of file vecmat.cpp.

4.7.5 Member Data Documentation

4.7.5.1 data

double Vector4::data[4]

Definition at line 16 of file vecmat.h.

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

- include/vecmat.h
- src/vecmat.cpp

70 Class Documentation

Chapter 5

File Documentation

5.1 include/ChartClass.h File Reference

A file containing ChartClass class.

```
#include <memory>
#include "ConfigClass.h"
#include <wx/utils.h>
#include <wx/timer.h>
#include <vector>
```

Classes

· class ChartClass

Contains the method that draws the curve.

Functions

double min (const double a, const double b, const double c)
 a normal function taking three values, returning a double value

5.1.1 Detailed Description

A file containing ChartClass class.

Definition in file ChartClass.h.

5.1.2 Function Documentation

5.1.2.1 min()

```
double min (  {\rm const\ double}\ a, \\ {\rm const\ double}\ b, \\ {\rm const\ double}\ c\ )
```

a normal function taking three values, returning a double value

Parameters

а	a const double value
b	a const double value
С	a const double value

Returns

the smallest value of the three parameters

Definition at line 158 of file ChartClass.cpp.

5.2 ChartClass.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <memory>
00003 #include "ConfigClass.h"
00004 #include <wx/utils.h>
00005 #include <wx/timer.h>
00006 #include <vector>
00007
00016 class ChartClass 00017 {
00018 private:
           std::shared_ptr<ConfigClass> cfg;
00020 public:
      ChartClass(std::shared_ptr<ConfigClass> c);
00025
00032
           void Draw(wxDC* dc, int w, int h);
00033
           wxTimer timer;
00034 };
00035
00043 double min(const double a, const double b, const double c);
```

5.3 include/ConfigClass.h File Reference

A file containing ConfigClass.

```
#include "GUIMyFrame.h"
```

Classes

· class ConfigClass

Contains curve parameters, variables describing how the curve is being drawn and related methods.

5.3.1 Detailed Description

A file containing ConfigClass.

Definition in file ConfigClass.h.

5.4 ConfigClass.h 73

5.4 ConfigClass.h

```
Go to the documentation of this file.
00001 #pragma once
00002 #pragma warning(disable: 4996)
00003 #include "GUIMyFrame.h"
00011 class ConfigClass
00012 {
00013 protected:
00014
            GUIMyFrame* MainWindow;
            double X_A, Y_A, Z_A;
double X_theta, Y_theta, Z_theta;
double X_phi, Y_phi, Z_phi;
double X_Rot, Y_Rot, Z_Rot;
00015
00016
00018
            double x, y, z;
00019
00020
            double R;
            bool animation;
00021
00022
            bool points;
00023
            bool polar;
00024
00025 public:
00026
           ConfigClass(GUIMyFrame* wnd);
00027
            bool get_Points() { return points; }
bool get_Animation() { return animation; }
00032
00038
            bool get_Polar() { return polar; };
double getX_A() { return X_A; };
double getY_A() { return Y_A; };
00043
00048
00053
00058
            double getZ_A() { return Z_A; };
00059
00064
            double getX_theta() { return X_theta; };
            double getY_theta() { return Y_theta; };
double getZ_theta() { return Z_theta; };
00069
00074
00075
            double getX_phi() { return X_phi; };
00080
00085
            double getY_phi() { return Y_phi; };
double getZ_phi() { return Z_phi; };
00091
00096
             double getX_Rot() { return X_Rot; };
            double getY_Rot() { return Y_Rot; };
double getZ_Rot() { return Z_Rot; };
00101
00106
00107
00112
             double getR() { return R; };
00113
00118
             void Set_Points(bool a) { points = a; }
00123
             void Set_Animation(bool a) { animation = a; }
            void Set_ParamType(bool a) { polar = a; }
00128
00129
             void SetX_Rot(int a) { X_Rot = a;
00134
00139
             void SetY_Rot(int a) { Y_Rot = a; }
             void SetZ_Rot(int a) { Z_Rot = a; }
00145
            void SetX_A(double a) { X_A = a; };
void SetY_A(double a) { Y_A = a; };
void SetZ_A(double a) { Z_A = a; };
00150
00155
00160
00161
             void SetX_theta(double theta) { X_theta = theta; };
00171
             void SetY_theta(double theta) { Y_theta = theta; };
00176
            void SetZ_theta(double theta) { Z_theta = theta; };
00177
            void SetX_phi(double phi) { X_phi = phi; };
void SetY_phi(double phi) { Y_phi = phi; };
00182
00192
             void SetZ_phi(double phi) { Z_phi = phi; };
00193
00198
             void SetR(double nR) { R = nR; };
00199
00200 1:
```

5.5 include/GUIMyFrame.h File Reference

A file containing GUYMyFrame class.

```
#include "Window.h"
#include <wx/filedlg.h>
#include <wx/dcmemory.h>
```

```
#include <wx/dcclient.h>
#include <wx/dcbuffer.h>
#include <wx/colourdata.h>
#include <wx/colordlg.h>
#include <memory>
```

Classes

class GUIMyFrame

Derived class of MyFrame1, with methods that edit the curve.

5.5.1 Detailed Description

A file containing GUYMyFrame class.

Definition in file GUIMyFrame.h.

5.6 GUIMyFrame.h

```
00001 #pragma once
00002 #include "Window.h"
00003 #include <wx/filedlg.h>
00004 #include <wx/dcmemory.h>
00005 #include <wx/dcclient.h>
00006 #include <wx/dcbuffer.h>
00007 #include <wx/colourdata.h>
00008 #include <wx/colordlg.h>
00009 #include <memory>
00014 class ConfigClass;
00015
00019 class GUIMyFrame : public MyFrame1
00021 protected:
00026
          void MainFormClose(wxCloseEvent& event);
00031
          void DisplayPanelRepaint(wxUpdateUIEvent& event);
00032
00037
          void XRot Updated(wxScrollEvent& event);
00042
          void YRot_Updated(wxScrollEvent& event);
00047
          void ZRot_Updated(wxScrollEvent& event);
00048
00053
          void XA_Updated(wxCommandEvent& event);
00058
          void YA_Updated(wxCommandEvent& event);
00063
          void ZA_Updated(wxCommandEvent& event);
00064
00069
          void XTheta_Updated(wxCommandEvent& event);
00074
          void YTheta_Updated(wxCommandEvent& event);
00079
          void ZTheta_Updated(wxCommandEvent& event);
08000
00085
          void XPhi_Updated(wxCommandEvent& event);
00090
          void YPhi_Updated(wxCommandEvent& event);
          void ZPhi_Updated(wxCommandEvent& event);
00096
00101
          void DotsLines_Updated(wxCommandEvent& event);
00106
          void Animation_Updated(wxCommandEvent& event);
00111
          void ParamType_Updated(wxCommandEvent& event);
00112
00113 public:
          GUIMyFrame(wxWindow* parent);
00122
          void Repaint();
00126
          ~GUIMyFrame();
          std::shared_ptr<ConfigClass> cfg;
00127
00128
          friend class ChartClass:
00129 };
```

5.7 include/vecmat.h File Reference

A file containing Vector4 and Matrix4 classes.

```
#include <stdio.h>
```

Classes

- class Vector4
 - a vector class.
- · class Matrix4

a matrix class.

5.7.1 Detailed Description

A file containing Vector4 and Matrix4 classes.

Definition in file vecmat.h.

5.8 vecmat.h

```
00001 #pragma once
00002 //Extremely simple vector and matrix classes by Janusz Malinowski.
00003 #include <stdio.h>
00004 #pragma once
00005
00013 class Vector4
00014 {
00015 public:
        double data[4];
00016
00017
          Vector4();
00018
          void Print(void);
00019
          void Set(double d1, double d2, double d3);
00020
          double GetX();
00021
          double GetY();
          double GetZ();
Vector4 operator-(const Vector4&);
00022
00024
          friend Vector4 operator*(const Vector4&, double);
00025 };
00026
00030 class Matrix4
00031 {
00032 public:
         double data[4][4];
00034
          Matrix4();
          void Print(void);
00035
00036
          Matrix4 operator*(const Matrix4);
          friend Vector4 operator*(const Matrix4, const Vector4);
00037
00038 };
```

5.9 include/Window.h File Reference

A file containing MyFrame1 class.

```
#include <wx/artprov.h>
#include <wx/xrc/xmlres.h>
#include <wx/panel.h>
#include <wx/gdicmn.h>
#include <wx/font.h>
#include <wx/colour.h>
#include <wx/settings.h>
#include <wx/string.h>
#include <wx/string.h>
#include <wx/sizer.h>
#include <wx/stattext.h>
#include <wx/slider.h>
#include <wx/textctrl.h>
#include <wx/radiobox.h>
#include <wx/frame.h>
#include <wx/timer.h>
```

Classes

• class MyFrame1

Class MyFrame1.

5.9.1 Detailed Description

A file containing MyFrame1 class.

Definition in file Window.h.

5.10 Window.h

```
00002 // C++ code generated with wxFormBuilder (version 3.10.1-0-g8feb16b3)
00003 // http://www.wxformbuilder.org/
00004 //
00005 // PLEASE DO *NOT* EDIT THIS FILE!
00008 #pragma once
00009
00010 #include <wx/artprov.h>
00011 #include <wx/xrc/xmlres.h>
00012 #include <wx/panel.h>
00013 #include <wx/gdicmn.h>
00014 #include <wx/font.h>
00015 #include <wx/colour.h>
00016 #include <wx/settings.h>
00017 #include <wx/string.h>
00018 #include <wx/sizer.h>
00019 #include <wx/stattext.h>
00020 #include <wx/slider.h>
00021 #include <wx/textctrl.h>
00022 #include <wx/radiobox.h>
00023 #include <wx/frame.h>
00024 #include <wx/timer.h>
00025
```

```
00036 class MyFrame1 : public wxFrame
00038
          friend class GUIMyFrame;
00039 protected:
         wxStaticText* m_staticText20;
00040
          wxStaticText* m_staticText21;
00041
          wxStaticText* m_staticText22;
00043
          wxStaticText* m_staticText23;
00044
          wxStaticText* m_staticText17;
00045
          wxStaticText* m_staticText19;
          wxStaticText* m_staticText5;
00046
          wxStaticText* m_staticText51;
00047
00048
          wxStaticText* m_staticText52;
          wxStaticText* m_staticText18;
00049
00050
          wxStaticText* m_staticText53;
00051
          wxStaticText* m_staticText511;
00052
          wxStaticText* m_staticText521;
00053
          wxStaticText* m_staticText4;
          wxStaticText* m_staticText54;
00055
          wxStaticText* m_staticText512;
00056
          wxStaticText* m_staticText522;
00057
          wxSlider* m_XRotationSlider;
00058
          wxSlider* m_YRotationSlider;
          wxSlider* m_ZRotationSlider;
00059
00060
          wxRadioBox* animationBox;
          wxRadioBox* m_radioBox1;
00061
00062
          wxRadioBox* ParamBox;
00063
          wxTextCtrl* m_AXText;
00064
          wxTextCtrl* m_ThXText;
00065
          wxTextCtrl* m PhiXText;
          wxTextCtrl* m_AYText;
00066
00067
          wxTextCtrl* m_ThYText;
00068
          wxTextCtrl* m_PhiYText;
00069
          wxTextCtrl* m_AZText;
00070
          wxTextCtrl* m_ThZText;
00071
          wxTextCtrl* m PhiZText;
00078
          virtual void MainFormClose(wxCloseEvent& event) { event.Skip(); }
          virtual void DisplayPanelRepaint(wxUpdateUIEvent& event) { event.Skip(); }
00083
00088
          virtual void Scrolls_Updated(wxScrollEvent& event) { event.Skip(); }
00089
00094
          virtual void XRot_Updated(wxScrollEvent& event) { event.Skip();
00099
          virtual void YRot_Updated(wxScrollEvent& event) { event.Skip();
00104
          virtual void ZRot Updated(wxScrollEvent& event) { event.Skip(); }
00105
          virtual void XA_Updated(wxCommandEvent& event) { event.Skip(); }
00110
00115
          virtual void YA_Updated(wxCommandEvent& event) { event.Skip();
00120
          virtual void ZA_Updated(wxCommandEvent& event) { event.Skip(); }
00121
          virtual void XTheta Updated (wxCommandEvent& event) { event.Skip(); }
00126
00131
          virtual void YTheta_Updated(wxCommandEvent& event) { event.Skip();
          virtual void ZTheta_Updated(wxCommandEvent& event) { event.Skip(); }
00136
00137
00142
          virtual void XPhi_Updated(wxCommandEvent& event) { event.Skip();
00147
          virtual void YPhi_Updated(wxCommandEvent& event) { event.Skip();
00152
          virtual void ZPhi_Updated(wxCommandEvent& event) { event.Skip(); }
00153
          virtual void DotsLines_Updated(wxCommandEvent& event) {  event.Skip(); }
00163
          virtual void Animation_Updated(wxCommandEvent& event) { event.Skip(); }
00168
          virtual void AnimationBreak(wxCommandEvent& event) { event.Skip();
00173
          virtual void ParamType_Updated(wxCommandEvent& event) { event.Skip(); }
00174
00175
          wxTimer m timer;
00177 public:
00178
          wxPanel* m_DisplayWindow;
00183
          MyFramel(wxWindow* parent, wxWindowID id = wxID_ANY, const wxString& title = wxEmptyString, const
       wxPoint& pos = wxDefaultPosition, const wxSize& size = wxSize(1055, 612), long style
wxDeFAULT_FRAME_STYLE | wxTAB_TRAVERSAL);
00188
          ~MvFrame1();
00189 };
```

5.11 main.cpp File Reference

```
#include <wx/wx.h>
#include "GUIMyFrame.h"
```

Classes

class MyApp

Functions

IMPLEMENT_APP (MyApp)

5.11.1 Function Documentation

5.11.1.1 IMPLEMENT_APP()

```
IMPLEMENT_APP (
          MyApp )
```

5.12 main.cpp

Go to the documentation of this file.

```
00001 #include <wx/wx.h>
00002 #include "GUIMyFrame.h"
00003
00004 class MyApp : public wxApp {
00005 public:
00006
           virtual bool OnInit();
virtual int OnExit() { return 0; }
00007
80000
00009
00010 };
00011
00012 IMPLEMENT_APP (MyApp); 00013
00014 bool MyApp::OnInit()
00015 {
           GUIMyFrame* mainFrame = new GUIMyFrame(NULL);
00017
00018
           mainFrame->Show(true);
00019
           SetTopWindow(mainFrame);
00020
00021
           return true;
00022 }
```

5.13 src/ChartClass.cpp File Reference

```
#include "ChartClass.h"
#include "vecmat.h"
#include <memory>
#include <vector>
#include <wx/timer.h>
```

Functions

• double min (const double a, const double b, const double c)

a normal function taking three values, returning a double value

5.14 ChartClass.cpp 79

5.13.1 Function Documentation

5.13.1.1 min()

```
double min (  \begin{tabular}{ll} $\operatorname{const}$ double $a$, \\ $\operatorname{const}$ double $b$, \\ $\operatorname{const}$ double $c$ ) \end{tabular}
```

a normal function taking three values, returning a double value

Parameters

а	a const double value
b	a const double value
С	a const double value

Returns

the smallest value of the three parameters

Definition at line 158 of file ChartClass.cpp.

5.14 ChartClass.cpp

```
00001 #include "ChartClass.h"
00002 #include "vecmat.h"
00003 #include <memory>
00004 #include <vector>
00005 #include <wx/timer.h>
00006
00007 ChartClass::ChartClass(std::shared_ptr<ConfigClass> c)
00008 {
00009
           cfg = std::move(c);
00010 }
00011
00012 void ChartClass::Draw(wxDC* dc, int w, int h)
00013 {
           dc->SetBackground(wxBrush(wxColor(255, 255, 255)));
00014
00015
           dc->Clear();
00016
           Vector4 vector1;
00017
           Vector4 vector2;
00018
           double r1, r2, phi1, phi2, th1, th2;
00019
00020
           Matrix4 m2:
           double alpha = cfg->getZ_Rot() * M_PI / 180.0;
m2.data[0][0] = cos(alpha);
m2.data[0][1] = sin(alpha);
00021
00022
00023
           m2.data[1][0] = -sin(alpha);
00024
           m2.data[1][1] = cos(alpha);
m2.data[2][2] = 1;
00025
00026
00027
00028
           Matrix4 m3;
00029
           alpha = cfg->getY_Rot() * M_PI / 180.0;
```

```
m3.data[0][0] = cos(alpha);
                              m3.data[0][2] = -sin(alpha);
 00031
                              m3.data[1][1] = 1;
 00032
00033
                              m3.data[2][0] = sin(alpha);
                              m3.data[2][2] = cos(alpha);
 00034
 00035
                              Matrix4 m4;
 00037
                              alpha = cfg->getX_Rot() * M_PI / 180.0;
                              m4.data[0][0] = 1;
m4.data[1][1] = cos(alpha);
 00038
 00039
                              m4.data[1][2] = sin(alpha);
00040
                              m4.data[2][1] = -sin(alpha);
 00041
 00042
                              m4.data[2][2] = cos(alpha);
 00043
 00044
                              Matrix4 transform1 = m4 * m3 * m2;
 00045
                              double minTheta = min(cfg->getX_theta() , cfg->getY_theta(), cfg->getZ_theta());
 00046
                              // drawing
                              wxPen m_pen;
 00047
 00048
                              m_pen.SetColour((wxColor(200, 200, 200)));
 00049
                              dc->SetPen(m_pen);
 00050
                               if (!cfg->get Animation())
 00051
                                          for (double i = 0; i < ((50 * 3.14159) / minTheta); i += ((2 * 3.14159) / (minTheta * 200)))
00052
 00053
                                          {
 00054
                                                       if(!cfg->get_Polar()){
 00055
                                                                  vector1.data[0] = cfg->getX_A() * sin(cfg->getX_theta() * i + cfg->getX_phi());
 00056
                                                                  \label{eq:continuous} \textbf{vector1.data[1]} = \textbf{cfg->getY\_A()} * \textbf{sin(cfg->getY\_theta()} * \textbf{i} + \textbf{cfg->getY\_phi());}
                                                                  vector1.data[2] = cfg->getZ_A() * sin(cfg->getZ_theta() * i + cfg->getZ_phi());
vector1.data[3] = 1;
00057
00058
00059
                                                                  vector1 = transform1 * vector1;
00060
00061
                                                                  vector2.data[0] = cfg - sqetX_A() * sin(cfg - sqetX_theta() * (i + ((2 * 3.14159) / squares + (2 * 3.14159) / squares + (3 * 3.14159) / squares + 
                       (minTheta * 200))) + cfg->getX_phi());
00062
                                                                  vector2.data[1] = cfg->getY_A() * sin(cfg->getY_theta() * (i + ((2 * 3.14159) / (2.14159)))) * (i + ((2 * 3.14159) / (2.14159))) * (i + 
                       (minTheta * 200))) + cfg->getY_phi());
00063
                                                                  \texttt{vector2.data[2]} = \texttt{cfg-} \texttt{yet2\_A()} * \texttt{sin(cfg-} \texttt{yet2\_theta()} * (\texttt{i} + ((2 * 3.14159) / (2 * 3.14159))) * (\texttt{i} + ((2 * 3.14159) / (2 * 3.14159))) * (\texttt{i} + ((2 * 3.14159)
                       (minTheta * 200))) + cfg->getZ_phi());
                                                                  vector2.data[3] = 1;
 00064
 00065
                                                                  vector2 = transform1 * vector2;
 00066
 00067
                                                       else {
                                                                 r1 = cfg->getR() * sin(cfg->getX_theta() * i + cfg->getX_phi());
 00068
                                                                  th1 = M_PI * sin(cfg->getY_theta() * i + cfg->getY_phi());
 00069
 00070
                                                                 phi1 = (M_PI/2) * sin(cfg->getZ_theta() * i + cfg->getZ_phi());
 00071
 00072
                                                                  vector1.data[0] = r1 * cos(th1) * cos(phi1);
00073
                                                                  vector1.data[1] = r1 * sin(th1) * cos(phi1);
 00074
                                                                  vector1.data[2] = r1 * sin(phi1);
00075
                                                                  vector2.data[3] = 1;
00076
                                                                  vector1 = transform1 * vector1;
 00077
00078
                                                                  r2 = cfq - yqetR() * sin(cfq - yqetX_theta() * (i + ((2 * 3.14159) / (minTheta * 200))) +
                     cfg->getX_phi());
00079
                                                                  th2 = M_PI * sin(cfg->getY_theta() * (i + ((2 * 3.14159) / (minTheta * 200))) +
                     cfg->getY_phi());
08000
                                                                  cfg->getZ_phi());
 00081
 00082
                                                                  vector2.data[0] = r2 * cos(th2) * cos(phi2);
                                                                  vector2.data[1] = r2 * sin(th2) * cos(phi2);
 00083
 00084
                                                                  vector2.data[2] = r2 * sin(phi2);
                                                                  vector2.data[3] = 1;
 00085
 00086
                                                                  vector2 = transform1 * vector2;
 00087
 00088
00089
                                                      dc->SetPen(wxPen(*wxBLACK, 2));
00090
                                                      if (cfg->get_Points() == true)
00091
                                                                 dc->DrawCircle(wxPoint(w / 2 + vector1.data[0], h / 2 + vector1.data[1]), 1);
 00092
00093
                                                                  dc->DrawLine(wxPoint(w / 2 + vectorl.data[0], h / 2 + vectorl.data[1]), wxPoint(w / 2
                      + vector2.data[0], h / 2 + vector2.data[1]));
 00094
                                         }
 00095
 00096
                             else
 00097
 00098
                                          static long long int startPoint = 0;
 00099
                                          std::vector<std::vector<double> animationPoints;
 00100
                                           std::vector<double> t;
00101
                                          if (!cfg->get_Polar())
                                                      for (double i = 0; i < ((50 * 3.14159) / minTheta); i += ((2 * 3.14159) / (minTheta *
00102
                     200)))
 00103
                                                                 vector1.data[0] = cfg->getX_A() * sin(cfg->getX_theta() * i + cfg->getX_phi());
vector1.data[1] = cfg->getY_A() * sin(cfg->getY_theta() * i + cfg->getY_phi());
 00104
 00105
                                                                 vector1.data[2] = cfg->getZ_A() * sin(cfg->getZ_theta() * i + cfg->getZ_phi());
vector1.data[3] = 1;
 00106
 00107
00108
                                                                  vector1 = transform1 * vector1;
```

```
t.push_back(vector1.data[0]);
00110
                                                                  t.push_back(vector1.data[1]);
00111
                                                                  animationPoints.push_back(t);
00112
                                                                  t.clear();
00113
                                                     }
00114
                                         else
00115
                                                     for (double i = 0; i < ((50 * 3.14159) / minTheta); i += ((2 * 3.14159) / (minTheta *
                     200)))
00116
00117
                                                                  r1 = cfg - yetR() * sin(cfg - yetX_theta() * i + cfg - yetX_phi());
                                                                 th1 = M_PI * sin(cfg->getY_theta() * i + cfg->getY_phi());
00118
                                                                 phi1 = (M_PI / 2) * sin(cfg->getZ_theta() * i + cfg->getZ_phi());
00119
00120
                                                                  vector1.data[0] = r1 * cos(th1) * cos(phi1);
00121
00122
                                                                  vector1.data[1] = r1 * sin(th1) * cos(phi1);
                                                                 vector1.data[2] = r1 * sin(phi1);
vector2.data[3] = 1;
00123
00124
00125
                                                                 vector1 = transform1 * vector1;
                                                                 t.push_back(vector1.data[0]);
00127
                                                                  t.push_back(vector1.data[1]);
                                                                  animationPoints.push_back(t);
00128
00129
                                                                  t.clear();
00130
                                          for (int i = 0; i < animationPoints.size()-1; i++)</pre>
00131
00132
                                          {
                                                      dc->DrawLine(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][1]),
00133
                     wxPoint(w \ / \ 2 \ + \ animationPoints[(i \ + \ 1) \ \% \ animationPoints.size()][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1) \ \% \ animationPoints.size()][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1) \ \% \ animationPoints][(i \ + \ 1) \ \% \ animationPoints.size()][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1) \ \% \ animationPoints][(i \ + \ 1) \ \% \ animationPoints.size()][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1) \ \% \ animationPoints][(i \ + \ 1) \ \% \ animationPoints.size()][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1) \ \% \ animationPoints][(i \ + \ 1) \ \% \ animationPoints.size()][0], \ h \ / \ 2 \ + \ animationPoints[(i \ + \ 1) \ \% \ animationPoints][(i \ + \ 1) \ \% \ animationPoints[(i \ + \ 1) \ \% \ animationPoints][(i \ + \ 1) \ \% \ animationPoints[(i \ + \ 1) \ \% \ anim
                     % animationPoints.size()][1]));
00134
                                         }
00135
00136
                                          for (int i = startPoint; i < startPoint + 150;)</pre>
00137
                                         {
00138
                                                      i %= animationPoints.size();
00139
                                                     dc->SetPen(wxPen(wxColor(255 * (i - startPoint) / 150, 0, 255 * (150 - i + startPoint) /
                     150), 2));
00140
                                                      if (cfg->get_Points() == true)
00141
                                                      {
                                                                 dc->DrawCircle(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][1]),
                     1);
00143
                                                                 i += 4;
00144
00145
                                                     else
00146
                                                     {
00147
                                                                 dc->DrawLine(wxPoint(w / 2 + animationPoints[i][0], h / 2 + animationPoints[i][1]),
                      wxPoint(w / 2 + animationPoints[(i + 1) % animationPoints.size()][0], h / 2 + animationPoints[(i + 1) % animationPoints
                     % animationPoints.size()][1]));
00148
                                                                 i++;
00149
                                                      if (i == animationPoints.size() - 1) i = 0;
00150
00151
00152
                                          if (cfg->get_Points()) startPoint += 4;
00153
                                          else startPoint++;
00154
                                          if (startPoint >= 4000) startPoint = 0;
00155
                            }
00156 }
00157
00158 double min(const double a, const double b, const double c)
00159 {
00160
                              double min = a;
                             if (b < min && b != 0) min = b;
if (c < min && c != 0) min = c;</pre>
00161
00162
00163
                             return min;
00164 }
```

5.15 src/ConfigClass.cpp File Reference

```
#include <fstream>
#include "ConfigClass.h"
```

5.16 ConfigClass.cpp

Go to the documentation of this file.

00001 #include <fstream>

```
00002 #include "ConfigClass.h"
00004 ConfigClass::ConfigClass(GUIMyFrame* wnd)
00005 {
00006
           MainWindow = wnd;
           X_A = Y_A = Z_A = 100;
00007
           X_{theta} = 1;
00009
           Y_{theta} = 1;
00010
           Z_{theta} = 0;
          X_phi = Z_phi = 0;
Y_phi = 3.1415 / 2;
00011
00012
00013
          R = 100:
00014
00015
          points = false;
00016
           animation = false;
00017
          polar = false;
00018 }
```

5.17 src/GUIMyFrame.cpp File Reference

```
#include "GUIMyFrame.h"
#include "ConfigClass.h"
#include "ChartClass.h"
```

5.18 GUIMyFrame.cpp

```
00001 #include "GUIMyFrame.h"
00002 #include "ConfigClass.h"
00003 #include "ChartClass.h"
00004
00005 GUIMyFrame::GUIMyFrame(wxWindow* parent) : MyFrame1(parent)
00006 {
00007
          cfg = std::make_shared<ConfigClass>(this);
80000
          Repaint();
00009 }
00010
00011 GUIMyFrame::~GUIMyFrame()
00012 {}
00013
00014 void GUIMyFrame::MainFormClose(wxCloseEvent& event)
00015 {
00016
00017 }
00018
00019 void GUIMyFrame::XRot_Updated(wxScrollEvent& event)
00021
          cfg->SetX_Rot (m_XRotationSlider->GetValue());
00022
          Repaint();
00023 }
00024
00025 void GUIMyFrame::YRot_Updated(wxScrollEvent& event)
00026 {
          cfg->SetY_Rot(m_YRotationSlider->GetValue());
00028
          Repaint();
00029 }
00030
00031 void GUIMyFrame::ZRot_Updated(wxScrollEvent& event)
00032 {
00033
          cfg->SetZ_Rot (m_ZRotationSlider->GetValue());
00034
          Repaint();
00035 }
00036 // -----
00037 void GUIMyFrame::XA_Updated(wxCommandEvent& event) {
00038
       double v;
          if (cfg->get_Polar()) {
00040
              if (m_AXText->GetValue().ToDouble(&v))
00041
              {
00042
                  cfg->SetR(v);
00043
                  Repaint();
00044
              }
00045
00046
          else {
```

```
00047
              if (m_AXText->GetValue().ToDouble(&v))
00048
00049
                  cfg->SetX_A(v);
00050
                      Repaint();
00051
00052
              else wxBell();
00053
          }
00054 }
00055
00056 void GUIMyFrame::YA_Updated(wxCommandEvent& event) {
00057
         double v:
          if (m_AYText->GetValue().ToDouble(&v))
00058
00059
          {
00060
              cfg->SetY_A(v);
00061
              Repaint();
00062
          else wxBell():
00063
00064 }
00065
00066 void GUIMyFrame::ZA_Updated(wxCommandEvent& event) {
00067
         double v;
00068
          if (m_AZText->GetValue().ToDouble(&v))
00069
00070
              cfq->SetZ A(v);
00071
             Repaint();
00072
00073
          else wxBell();
00074 }
00075 // ---
00076 void GUIMyFrame::XTheta_Updated(wxCommandEvent& event) {
00077
         double v:
00078
          if (m_ThXText->GetValue().ToDouble(&v))
00079
08000
              cfg->SetX_theta(v);
00081
             Repaint();
00082
00083
         else wxBell();
00084 }
00085
00086 void GUIMyFrame::YTheta_Updated(wxCommandEvent& event) {
00087
         double v;
          if (m_ThYText->GetValue().ToDouble(&v))
00088
00089
          {
00090
              cfg->SetY_theta(v);
00091
              Repaint();
00092
00093
          else wxBell();
00094 }
00095
00096 void GUIMyFrame::ZTheta_Updated(wxCommandEvent& event) {
00097
         double v;
00098
          if (m_ThZText->GetValue().ToDouble(&v))
00099
00100
              cfg->SetZ_theta(v);
00101
              Repaint();
00102
00103
          else wxBell();
00104 }
00105 // -----
00106 void GUIMyFrame::XPhi_Updated(wxCommandEvent& event) {
00107
         double v:
00108
          if (m_PhiXText->GetValue().ToDouble(&v))
00109
          {
00110
              cfg->SetX_phi(v);
00111
              Repaint();
00112
00113
          else wxBell();
00114 }
00115
00116 void GUIMyFrame::YPhi_Updated(wxCommandEvent& event) {
00117
         double v;
00118
          if (m_PhiYText->GetValue().ToDouble(&v))
00119
              cfg->SetY_phi(v);
00120
00121
              Repaint();
00122
          else wxBell();
00123
00124 }
00125
00126 void GUIMyFrame::ZPhi_Updated(wxCommandEvent& event) {
00127
         double v;
          if (m_PhiZText->GetValue().ToDouble(&v))
00128
00129
00130
              cfg->SetZ_phi(v);
00131
              Repaint();
00132
00133
          else wxBell();
```

```
00134 }
00135 // -----
00136 void GUIMyFrame::DotsLines_Updated(wxCommandEvent& event)
00137 {
00138
          if (m radioBox1->GetSelection() == 0) {
00139
             cfg->Set_Points(false);
00140
00141
          else {
00142
            cfg->Set_Points(true);
00143
          Repaint();
00144
00145 }
00146
00147 void GUIMyFrame::Animation_Updated(wxCommandEvent& event)
00148 {
00149
          if (animationBox->GetSelection() == 0) {
00150
              cfq->Set_Animation(false);
00151
00152
          else {
00153
              cfg->Set_Animation(true);
00154
00155
          Repaint();
00156 }
00157
00158 void GUIMyFrame::ParamType_Updated(wxCommandEvent& event) {
        if (ParamBox->GetSelection() == 0) { //uklad kartezjanski
00160
              cfg->Set_ParamType(false);
00161
              m_AYText->SetEditable(true);
00162
              m_AZText->SetEditable(true);
              m_AYText->Clear();
00163
00164
              m_AZText->Clear();
00165
              m_staticText5->SetLabel("A");
00166
              m_staticText53->SetLabel("A");
00167
              m_staticText54->SetLabel("A");
00168
              m_staticText51->SetLabel(wxT(""));
              m_staticText51->SetLabel(wxT(""));
00169
00170
              m_staticText51->SetLabel(wxT(""));
00171
              m_staticText19->SetLabel("X");
00172
              m_staticText18->SetLabel("Y");
00173
              m_staticText4->SetLabel("Z");
00174
00175
          else {
00176
             cfg->Set_ParamType(true);
00177
              m_AYText->SetEditable(false);
00178
              m_AYText->Clear();
00179
              m_AYText->AppendText("-");
00180
              m_AZText->SetEditable(false);
00181
              m AZText->Clear();
              m_AZText->AppendText("-");
00182
00183
              m_staticText5->SetLabel("r");
              m_staticText53->SetLabel("-");
00184
00185
              m_staticText54->SetLabel("-");
00186
              m_staticText51->SetLabel(wxT(""));
              m_staticText511->SetLabel(wxT(""));
00187
              m_staticText512->SetLabel(wxT(""));
00188
              m_staticText19->SetLabel("R");
m_staticText18->SetLabel(wxT(""));
00189
00190
00191
              m_staticText4->SetLabel(wxT(""));
00192
00193
          Repaint();
00194 }
00195
00196 void GUIMyFrame::DisplayPanelRepaint(wxUpdateUIEvent& event)
00197 {
00198
          static int i = 0:
00199
          if (cfg->get_Animation())
00200
00201
              i++;
              if (i % 4 == 0 && cfg->get_Points())
00202
00203
                  Repaint();
00204
              if (!cfg->get_Points())
00205
                  Repaint();
              if (i > 1000) i = 0;
00206
00207
00208
          else
00209
              Repaint();
00210 }
00211
00212 void GUIMyFrame::Repaint()
00213 {
00214
          wxClientDC dc1(m_DisplayWindow);
          wxBufferedDC dc(&dc1);
00216
00217
          ChartClass MyChart(cfg);
00218
          int w, h;
          m_DisplayWindow->GetSize(&w, &h);
00219
00220
          MyChart.Draw(&dc, w, h);
```

00221 }

5.19 src/vecmat.cpp File Reference

```
#include "vecmat.h"
```

Functions

- Vector4 operator* (const Vector4 &gVector, double val)
- Vector4 operator* (const Matrix4 gMatrix, const Vector4 gVector)

5.19.1 Function Documentation

5.19.1.1 operator*() [1/2]

Definition at line 80 of file vecmat.cpp.

5.19.1.2 operator*() [2/2]

```
Vector4 operator* (  {\tt const\ Vector4\ \&\ } gVector,   {\tt double\ } val\ )
```

Definition at line 41 of file vecmat.cpp.

5.20 vecmat.cpp

```
Go to the documentation of this file.
```

```
00001 #include "vecmat.h"
00002
00003 Vector4::Vector4()
00004 {
00005
          data[0] = 0.0; data[1] = 0.0; data[2] = 0.0; data[3] = 1.0;
00006 }
00007
00008 void Vector4::Print(void)
00009 {
         printf("(%2.31f, %2.31f, %2.31f, %2.31f) \n", data[0], data[1], data[2], data[3]);
00012
00013 void Vector4::Set (double d1, double d2, double d3)
00014 {
00015
         data[0] = d1; data[1] = d2; data[2] = d3;
00016 }
00018 double Vector4::GetX()
00019 {
00020
          return data[0]:
00021 }
00022
00023 double Vector4::GetY()
00024 {
00025
          return data[1];
00026 }
00027
00028 double Vector4::GetZ()
00029 {
00030
          return data[2];
00031 }
00032
00033 Vector4 Vector4::operator- (const Vector4& gVector)
00034 {
00035
          unsigned int i;
00036
          Vector4 Result;
00037
          for (i = 0; i < 4; i++) Result.data[i] = data[i] - gVector.data[i];</pre>
00038
          return Result;
00039 }
00040
00041 Vector4 operator* (const Vector4& gVector, double val)
00042 {
00043
          unsigned int i;
00044
          Vector4 Result;
         for (i = 0; i < 4; i++) Result.data[i] = gVector.data[i] * val;</pre>
00045
00046
         return Result:
00047 }
00049 Matrix4::Matrix4()
00050 {
          data[0][0] = 0.0; data[0][1] = 0.0; data[0][2] = 0.0; data[0][3] = 0.0;
00051
         data[1][0] = 0.0; data[1][1] = 0.0; data[1][2] = 0.0; data[1][3] = 0.0; data[2][0] = 0.0; data[2][1] = 0.0; data[2][2] = 0.0; data[2][3] = 0.0;
00052
00053
         data[3][0] = 0.0; data[3][1] = 0.0; data[3][2] = 0.0; data[3][3] = 1.0;
00054
00055 }
00056
00057 void Matrix4::Print(void)
00058 {
         00059
00061
00062
00063 }
00064
00065 Matrix4 Matrix4::operator* (const Matrix4 gMatrix)
00066 {
00067
          int i, j, k;
00068
         Matrix4 tmp;
00069
00070
          for (i = 0; i < 4; i++)
              for (j = 0; j < 4; j++)
00071
00072
                  tmp.data[i][j] = 0.0;
00074
                  for (k = 0; k < 4; k++)
00075
                      tmp.data[i][j] = tmp.data[i][j] + (data[i][k] * gMatrix.data[k][j]);
00076
00077
          return tmp;
00078 }
00080 Vector4 operator* (const Matrix4 gMatrix, const Vector4 gVector)
00081 {
00082
          unsigned int i, j;
```

5.21 src/Window.cpp File Reference

```
#include "Window.h"
```

5.22 Window.cpp

```
00002 // C++ code generated with wxFormBuilder (version 3.10.1-0-g8feb16b3)
00003 // http://www.wxformbuilder.org/
00004 //
00005 // PLEASE DO *NOT* EDIT THIS FILE!
00007
00008 #include "Window.h"
00009
00011
00012 MyFrame1::MyFrame1(wxWindow* parent, wxWindowID id, const wxString& title, const wxPoint& pos, const
       wxSize& size, long style) : wxFrame(parent, id, title, pos, size, style), m_timer(this, 1)
00013 {
00014
          SetTitle(_("Linie Lissajous - Natalia Przetocka, Karolina Klimek i Mateusz Lewandowski"));
00015
          this->SetSizeHints(wxDefaultSize, wxDefaultSize);
00016
00017
          wxBoxSizer* bSizer1;
00018
          bSizer1 = new wxBoxSizer(wxHORIZONTAL);
00019
00020
          wxBoxSizer* bSizer2;
00021
          bSizer2 = new wxBoxSizer(wxVERTICAL);
00022
00023
          wxBoxSizer* bSizer30;
00024
          bSizer30 = new wxBoxSizer(wxVERTICAL);
00025
00026
          m_DisplayWindow = new wxPanel(this, wxID_ANY, wxDefaultPosition, wxDefaultSize, wxTAB_TRAVERSAL);
00027
          m_DisplayWindow->SetBackgroundColour(wxSystemSettings::GetColour(wxSYS_COLOUR_BTNHIGHLIGHT));
00028
00029
          bSizer30->Add(m_DisplayWindow, 5, wxALL | wxEXPAND, 5);
00030
00031
          bSizer2->Add(bSizer30, 30, wxEXPAND, 5);
00032
00033
          wxBoxSizer* bSizer37;
00034
          bSizer37 = new wxBoxSizer(wxHORIZONTAL);
00035
00036
          wxBoxSizer* bSizer38;
00037
          bSizer38 = new wxBoxSizer(wxVERTICAL);
00038
00039
          wxBoxSizer* bSizer36;
00040
          bSizer36 = new wxBoxSizer(wxVERTICAL);
00041
00042
          m_staticText20 = new wxStaticText(this, wxID_ANY, wxT("Rotacja"), wxDefaultPosition,
       wxDefaultSize, 0);
m_staticText20->Wrap(-1);
00043
          m_staticText20->SetFont(wxFont(12, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00044
       false, wxT("Cambria")));
00045
00046
          bSizer36->Add(m_staticText20, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00047
00048
          bSizer38->Add(bSizer36, 0, wxEXPAND, 5);
00049
00050
          wxBoxSizer* bSizer32;
00051
          bSizer32 = new wxBoxSizer(wxHORIZONTAL);
00052
00053
          wxBoxSizer* bSizer33;
00054
          bSizer33 = new wxBoxSizer(wxVERTICAL);
00055
00056
          wxBoxSizer* bSizer40;
```

```
00057
                bSizer40 = new wxBoxSizer(wxVERTICAL);
00058
00059
                m_staticText21 = new wxStaticText(this, wxID_ANY, wxT("O X"), wxDefaultPosition, wxDefaultSize,
           0);
00060
                m staticText21->Wrap(-1):
                m_staticText21->SetFont(wxFont(12, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00061
            false, wxT("Cambria")));
00062
00063
                bSizer40->Add(m_staticText21, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00064
00065
                bSizer33->Add(bSizer40, 0, wxEXPAND, 5);
00066
00067
                 wxBoxSizer* bSizer42;
00068
                bSizer42 = new wxBoxSizer(wxVERTICAL);
00069
                 \texttt{m\_XRotationSlider} = \texttt{new wxSlider(this, wxID\_ANY, 0, 0, 360, wxDefaultPosition, wxSize(-1, -1), and the state of th
00070
           wxSL_HORIZONTAL | wxSL_VALUE LABEL);
00071
                m XRotationSlider->SetMaxSize(wxSize(300, -1));
00072
00073
                bSizer42->Add(m_XRotationSlider, 0, wxALIGN_CENTER_HORIZONTAL | wxALIGN_CENTER_VERTICAL | wxALL |
           wxEXPAND, 5);
00074
00075
                bSizer33->Add(bSizer42, 0, wxEXPAND, 5);
00076
00077
                bSizer32->Add(bSizer33, 1, wxEXPAND, 5);
00078
00079
                 wxBoxSizer* bSizer34;
00080
                bSizer34 = new wxBoxSizer(wxVERTICAL);
00081
00082
                m staticText22 = new wxStaticText(this, wxID ANY, wxT("O Y"), wxDefaultPosition, wxDefaultSize,
           0);
00083
                m_staticText22->Wrap(-1);
                 m_staticText22->SetFont(wxFont(12, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00084
            false, wxT("Cambria")));
00085
00086
                bSizer34->Add(m_staticText22, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00087
00088
                m_YRotationSlider = new wxSlider(this, wxID_ANY, 0, 0, 360, wxDefaultPosition, wxDefaultSize,
           wxSL_HORIZONTAL | wxSL_VALUE_LABEL);
00089
                m_YRotationSlider->SetMaxSize(wxSize(300, -1));
00090
00091
                bSizer34->Add(m YRotationSlider, 0, wxALIGN CENTER HORIZONTAL | wxALL | wxEXPAND, 5);
00092
00093
                bSizer32->Add(bSizer34, 1, wxEXPAND, 5);
00094
00095
                 wxBoxSizer* bSizer35;
00096
                bSizer35 = new wxBoxSizer(wxVERTICAL);
00097
                m staticText23 = new wxStaticText(this, wxID ANY, wxT("O Z"), wxDefaultPosition, wxDefaultSize,
00098
           0);
00099
                m_staticText23->Wrap(-1);
                 m_staticText23->SetFont(wxFont(12, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00100
            false, wxT("Cambria")));
00101
                bSizer35->Add(m_staticText23, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00102
00103
00104
                m_ZRotationSlider = new wxSlider(this, wxID_ANY, 0, 0, 360, wxDefaultPosition, wxDefaultSize,
           wxSL_HORIZONTAL | wxSL_VALUE_LABEL);
00105
                m_ZRotationSlider->SetMaxSize(wxSize(300, -1));
00106
00107
                bSizer35->Add(m ZRotationSlider, 0, wxALIGN CENTER HORIZONTAL | wxALL | wxEXPAND, 5);
00108
00109
                bSizer32->Add(bSizer35, 1, wxEXPAND, 5);
00110
00111
                bSizer38->Add(bSizer32, 0, wxEXPAND, 5);
00112
00113
                bSizer37->Add(bSizer38, 3, wxEXPAND, 5);
00114
00115
                bSizer2->Add(bSizer37, 0, wxEXPAND, 5);
00116
00117
                bSizer1->Add(bSizer2, 7, wxALIGN_RIGHT | wxEXPAND, 5);
00118
00119
                wxBoxSizer* bSizer4;
00120
                bSizer4 = new wxBoxSizer(wxVERTICAL);
00121
00122
                m_staticText17 = new wxStaticText(this, wxID_ANY, wxT("Parametry"), wxDefaultPosition,
           wxDefaultSize, 0);
                m_staticText17->Wrap(-1);
m_staticText17->SetFont(wxFont(16, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00123
00124
            false, wxT("Cambria")));
00125
00126
                bSizer4->Add(m_staticText17, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00127
00128
                 wxBoxSizer* bSizer361;
00129
                bSizer361 = new wxBoxSizer(wxVERTICAL);
00130
00131
                wxString ParamBoxChoices[] = { wxT("Kartezjaska"), wxT("Biegunowa") };
```

5.22 Window.cpp 89

```
00132
          int ParamBoxNChoices = sizeof(ParamBoxChoices) / sizeof(wxString);
          ParamBox = new wxRadioBox(this, wxID_ANY, wxT("Rodzaj parametryzacji"), wxDefaultPosition,
00133
       wxDefaultSize, ParamBoxNChoices, ParamBoxChoices, 1, wxRA_SPECIFY_COLS);
00134
          ParamBox->SetSelection(0);
00135
          bSizer361->Add(ParamBox, 0, wxALIGN CENTER HORIZONTAL | wxALIGN CENTER VERTICAL | wxALL, 5);
00136
00137
00138
          bSizer4->Add(bSizer361, 0, wxEXPAND, 5);
00139
00140
          wxBoxSizer* bSizer9;
          bSizer9 = new wxBoxSizer(wxVERTICAL);
00141
00142
00143
          m_staticText19 = new wxStaticText(this, wxID_ANY, wxT("X"), wxDefaultPosition, wxDefaultSize, 0);
          m_staticText19->Wrap(-1);
00144
00145
          m_staticText19->SetFont(wxFont(14, wxFonTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
       false, wxT("Cambria")));
00146
00147
          bSizer9->Add(m staticText19, 0, wxALL | wxALIGN CENTER HORIZONTAL, 5);
00148
00149
          wxBoxSizer* bSizer10;
00150
          bSizer10 = new wxBoxSizer(wxHORIZONTAL);
00151
00152
          m_staticText5 = new wxStaticText(this, wxID_ANY, wxT("A"), wxDefaultPosition, wxDefaultSize, 0);
          m staticText5->Wrap(-1):
00153
          m_staticText5->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00154
       false, wxT("Cambria")));
00155
00156
          bSizer10->Add(m_staticText5, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00157
          m_AXText = new wxTextCtrl(this, wxID_ANY, wxT("100"), wxDefaultPosition, wxSize(-1, -1), 0);
bSizer10->Add(m_AXText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00158
00159
00160
00161
          bSizer9->Add(bSizer10, 0, wxEXPAND, 5);
00162
00163
          wxBoxSizer* bSizer101;
          bSizer101 = new wxBoxSizer(wxHORIZONTAL);
00164
00165
00166
          m_staticText51 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00167
          m staticText51->Wrap(-1);
          m_staticText51->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00168
       false, wxT("Cambria")));
00169
00170
          bSizer101->Add(m staticText51, 1, wxALL | wxALIGN CENTER VERTICAL, 5):
00171
            _ThXText = new wxTextCtrl(this, wxID_ANY, wxT("1"), wxDefaultPosition, wxDefaultSize, 0);
00172
00173
          bSizer101->Add(m_ThXText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00174
00175
          bSizer9->Add(bSizer101, 0, wxEXPAND, 5);
00176
00177
          wxBoxSizer* bSizer102:
00178
          bSizer102 = new wxBoxSizer(wxHORIZONTAL);
00179
00180
          m_staticText52 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00181
          m_staticText52->Wrap(-1);
          m_staticText52->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00182
       false, wxT("Cambria")));
00183
00184
          bSizer102->Add(m_staticText52, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00185
          m_PhiXText = new wxTextCtrl(this, wxID_ANY, wxT("0"), wxDefaultPosition, wxDefaultSize, 0);
00186
00187
          bSizer102->Add(m_PhiXText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00188
00189
          bSizer9->Add(bSizer102, 0, wxEXPAND, 5);
00190
00191
          bSizer4->Add(bSizer9, 0, wxEXPAND, 5);
00192
00193
          wxBoxSizer* bSizer91;
00194
          bSizer91 = new wxBoxSizer(wxVERTICAL);
00195
00196
          m_staticText18 = new wxStaticText(this, wxID_ANY, wxT("Y"), wxDefaultPosition, wxDefaultSize, 0);
00197
          m_staticText18->Wrap(-1);
00198
          m_staticText18->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
       false, wxT("Cambria")));
00199
00200
          bSizer91->Add(m staticText18, 0, wxALL | wxALIGN CENTER HORIZONTAL, 5);
00201
00202
          wxBoxSizer* bSizer103;
00203
          bSizer103 = new wxBoxSizer(wxHORIZONTAL);
00204
00205
          m staticText53 = new wxStaticText(this, wxID ANY, wxT("A"), wxDefaultPosition, wxDefaultSize, 0);
          m_staticText53->Wrap(-1);
00206
00207
          m_staticText53->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
       false, wxT("Cambria")));
00208
00209
          bSizer103->Add(m_staticText53, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00210
00211
          m_AYText = new wxTextCtrl(this, wxID_ANY, wxT("100"), wxDefaultPosition, wxDefaultSize, 0);
```

```
bSizer103->Add(m_AYText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00213
00214
          bSizer91->Add(bSizer103, 1, wxEXPAND, 5);
00215
00216
          wxBoxSizer* bSizer1011;
00217
          bSizer1011 = new wxBoxSizer(wxHORIZONTAL);
00218
00219
          m_staticText511 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00220
          m_staticText511->Wrap(-1);
00221
          m_staticText511->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
       false, wxT("Cambria")));
00222
00223
          bSizer1011->Add (m staticText511, 1, wxALL | wxALIGN CENTER VERTICAL, 5);
00224
00225
           _ThYText = new wxTextCtrl(this, wxID_ANY, wxT("1"), wxDefaultPosition, wxDefaultSize, 0);
00226
          bSizer1011->Add(m_ThYText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00227
00228
          bSizer91->Add(bSizer1011, 1, wxEXPAND, 5);
00229
00230
          wxBoxSizer* bSizer1021;
00231
          bSizer1021 = new wxBoxSizer(wxHORIZONTAL);
00232
          m_staticText521 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00233
          m staticText521->Wrap(-1):
00234
00235
          m_staticText521->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
       false, wxT("Cambria")));
00236
00237
          bSizer1021->Add(m_staticText521, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00238
          m_PhiYText = new wxTextCtrl(this, wxID_ANY, wxT("1.57"), wxDefaultPosition, wxDefaultSize, 0);
00239
00240
          bSizer1021->Add(m PhiYText, 5, wxALL | wxALIGN RIGHT | wxALIGN CENTER VERTICAL, 5);
00241
00242
          bSizer91->Add(bSizer1021, 1, wxEXPAND, 5);
00243
00244
          bSizer4->Add(bSizer91, 0, wxEXPAND, 5);
00245
00246
          wxBoxSizer* bSizer92;
          bSizer92 = new wxBoxSizer(wxVERTICAL);
00247
00248
00249
          m_staticText4 = new wxStaticText(this, wxID_ANY, wxT("Z"), wxDefaultPosition, wxDefaultSize, 0);
00250
          m_staticText4->Wrap(-1);
          m_staticText4->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00251
       false. wxT("Cambria"))):
00252
00253
          bSizer92->Add(m_staticText4, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00254
00255
          wxBoxSizer* bSizer104;
00256
          bSizer104 = new wxBoxSizer(wxHORIZONTAL);
00257
00258
          m staticText54 = new wxStaticText(this, wxID ANY, wxT("A"), wxDefaultPosition, wxDefaultSize, 0);
          m_staticText54->Wrap(-1);
00259
          m_staticText54->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00260
       false, wxT("Cambria")));
00261
00262
          bSizer104->Add(m_staticText54, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00263
          m_AZText = new wxTextCtrl(this, wxID_ANY, wxT("100"), wxDefaultPosition, wxDefaultSize, 0);
00264
00265
          bSizer104->Add(m_AZText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00266
00267
          bSizer92->Add(bSizer104, 1, wxEXPAND, 5);
00268
00269
          wxBoxSizer* bSizer1012;
00270
          bSizer1012 = new wxBoxSizer(wxHORIZONTAL);
00271
00272
          m_staticText512 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00273
          m_staticText512->Wrap(-1);
00274
          m_staticText512->SetFont(wxFont(14, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
       false, wxT("Cambria")));
00275
00276
          bSizer1012->Add(m_staticText512, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00277
00278
          m_ThZText = new wxTextCtrl(this, wxID_ANY, wxT("0"), wxDefaultPosition, wxDefaultSize, 0);
00279
          bSizer1012->Add(m_ThZText, 5, wxALL | wxALIGN_RIGHT | wxALIGN_CENTER_VERTICAL, 5);
00280
00281
          bSizer92->Add(bSizer1012, 1, wxEXPAND, 5);
00282
00283
          wxBoxSizer* bSizer1022;
00284
          bSizer1022 = new wxBoxSizer(wxHORIZONTAL);
00285
          m_staticText522 = new wxStaticText(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDefaultSize, 0);
00286
          m_staticText522->Wrap(-1);
00287
          m_staticText522->SetFont(wxFont(14, wxFoNTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
00288
       false, wxT("Cambria")));
00289
00290
          bSizer1022->Add(m_staticText522, 1, wxALL | wxALIGN_CENTER_VERTICAL, 5);
00291
00292
          m PhiZText = new wxTextCtrl(this, wxID ANY, wxT("0"), wxDefaultPosition, wxDefaultSize, 0);
```

5.22 Window.cpp 91

```
00293
                       bSizer1022->Add(m PhiZText, 5, wxall | wxalign Right | wxalign Center Vertical, 5);
00294
00295
                       bSizer92->Add(bSizer1022, 1, wxEXPAND | wxALIGN_RIGHT, 5);
00296
00297
                       bSizer4->Add(bSizer92, 0, wxEXPAND, 5):
00298
00299
                        wxBoxSizer* bSizer27;
00300
                       bSizer27 = new wxBoxSizer(wxVERTICAL);
00301
00302
                       wxString animationBoxChoices[] = { wxT("Statyczna"), wxT("Animowana") };
                       int animationBoxNChoices = sizeof(animationBoxChoices) / sizeof(wxString);
00303
                       animationBox = new wxRadioBox(this, wxID_ANY, wxT("Rodzaj animacji"), wxDefaultPosition,
00304
                wxDefaultSize, animationBoxNChoices, animationBoxChoices, 1, wxRA_SPECIFY_COLS);
00305
                       animationBox->SetSelection(0);
00306
                       bSizer27->Add(animationBox, 0, wxALL | wxALIGN_CENTER_HORIZONTAL, 5);
00307
                       wxString m_radioBox1Choices[] = { wxT("Linie"), wxT("Punkty") };
00308
                       int m_radioBox1NChoices = sizeof(m_radioBox1Choices) / sizeof(wxString);
00309
00310
                       m_radioBox1 = new wxRadioBox(this, wxID_ANY, wxT("Rodzaj rysowania"), wxDefaultPosition,
                wxDefaultSize, m_radioBox1NChoices, m_radioBox1Choices, 1, wxRA_SPECIFY_COLS);
00311
                       m_radioBox1->SetSelection(0);
00312
                       m_radioBox1->SetFont(wxFont(10, wxFONTFAMILY_ROMAN, wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL,
                false, wxT("Cambria")));
00313
00314
                       bSizer27->Add(m_radioBox1, 0, wxALL | wxALIGN_CENTER_HORIZONTAL | wxALIGN_CENTER_VERTICAL, 5);
00315
00316
                       bSizer4->Add(bSizer27, 1, wxEXPAND, 5);
00317
00318
                       bSizer1->Add(bSizer4, 1, wxALIGN_LEFT | wxEXPAND, 5);
00319
00320
                       this->SetSizer(bSizer1);
00321
                      this->Layout();
00322
00323
                       this->Centre (wxBOTH);
00324
00325
                        //connect-v
                       this->Connect (wxEVT CLOSE WINDOW, wxCloseEventHandler(MyFrame1::MainFormClose));
00326
00327
                       m_DisplayWindow->Connect(wxEVT_UPDATE_UI, wxUpdateUIEventHandler(MyFrame1::DisplayPanelRepaint),
                NULL, this);
00328
00329
                       \verb|m_XRotationSlider->Connect(wxEVT_SCROLL_TOP, wxScrollEventHandler(MyFramel::XRot_Updated), NULL, wxScrollEventHandler(
                this);
                       m XRotationSlider->Connect(wxEVT SCROLL BOTTOM, wxScrollEventHandler(MvFramel::XRot Updated).
00330
                NULL, this);
00331
                       m_XRotationSlider->Connect(wxEVT_SCROLL_LINEUP, wxScrollEventHandler(MyFrame1::XRot_Updated),
                NULL, this);
00332
                      m_XRotationSlider->Connect(wxEVT_SCROLL_LINEDOWN, wxScrollEventHandler(MyFrame1::XRot_Updated),
                NULL, this);
00333
                       m XRotationSlider->Connect(wxEVT SCROLL PAGEUP, wxScrollEventHandler(MvFrame1::XRot Updated),
                NULL, this):
00334
                       m_XRotationSlider->Connect(wxEVT_SCROLL_PAGEDOWN, wxScrollEventHandler(MyFrame1::XRot_Updated),
                NULL, this);
00335
                       m_XRotationSlider->Connect(wxEVT_SCROLL_THUMBTRACK, wxScrollEventHandler(MyFrame1::XRot_Updated),
                NULL, this);
                      m_XRotationSlider->Connect(wxEVT_SCROLL_THUMBRELEASE,
00336
                wxScrollEventHandler(MyFramel::XRot_Updated), NULL, this);
                       m_XRotationSlider->Connect(wxEVT_SCROLL_CHANGED, wxScrollEventHandler(MyFramel::XRot_Updated),
00337
                NULL, this);
                       m_YRotationSlider->Connect(wxEVT_SCROLL_TOP, wxScrollEventHandler(MyFramel::YRot_Updated), NULL,
00338
                this);
                      \verb|m_YRotationSlider->Connect(wxEVT_SCROLL_BOTTOM, wxScrollEventHandler(MyFrame1::YRot_Updated)|, wxScrollEventHandler
00339
                NULL, this);
00340
                       m_YRotationSlider->Connect(wxEVT_SCROLL_LINEUP, wxScrollEventHandler(MyFrame1::YRot_Updated),
                NULL, this);
00341
                       m_YRotationSlider->Connect(wxEVT_SCROLL_LINEDOWN, wxScrollEventHandler(MyFramel::YRot_Updated),
                NULL, this);
00342
                       m_YRotationSlider->Connect(wxEVT_SCROLL_PAGEUP, wxScrollEventHandler(MyFrame1::YRot_Updated),
                NULL, this);
00343
                      m YRotationSlider->Connect(wxEVT SCROLL PAGEDOWN, wxScrollEventHandler(MvFrame1::YRot Updated),
                NULL, this);
                        m_YRotationSlider->Connect(wxEVT_SCROLL_THUMBTRACK, wxScrollEventHandler(MyFramel::YRot_Updated),
00344
                NULL, this);
00345
                      m_YRotationSlider->Connect(wxEVT_SCROLL_THUMBRELEASE,
                wxScrollEventHandler(MyFrame1::YRot_Updated), NULL, this);
                       m_YRotationSlider->Connect(wxEVT_SCROLL_CHANGED, wxScrollEventHandler(MyFrame1::YRot_Updated),
00346
                NULL, this);
00347
                       m_ZRotationSlider->Connect(wxEVT_SCROLL_TOP, wxScrollEventHandler(MyFrame1::ZRot_Updated), NULL,
                this);
00348
                       NULL, this);
                       m ZRotationSlider->Connect(wxEVT SCROLL LINEUP, wxScrollEventHandler(MyFramel::ZRot Updated),
00349
                NULL, this);
00350
                       m ZRotationSlider->Connect(wxEVT SCROLL LINEDOWN, wxScrollEventHandler(MyFrame1::ZRot Updated),
                NULL, this);
00351
                      \verb|m_ZRotationSlider->Connect(wxEVT_SCROLL_PAGEUP, wxScrollEventHandler(MyFramel::ZRot_Updated)|, wxScrollEventHandler
                NULL, this);
00352
                      m ZRotationSlider->Connect(wxEVT SCROLL PAGEDOWN, wxScrollEventHandler(MvFrame1::ZRot Updated),
```

```
NULL, this);
00353
               m_ZRotationSlider->Connect(wxEVT_SCROLL_THUMBTRACK, wxScrollEventHandler(MyFrame1::ZRot_Updated),
          NULL, this);
00354
              m_ZRotationSlider->Connect(wxEVT_SCROLL_THUMBRELEASE,
          wxScrollEventHandler(MyFrame1::ZRot_Updated), NULL, this);
              m_ZRotationSlider->Connect(wxEVT_SCROLL_CHANGED, wxScrollEventHandler(MyFramel::ZRot_Updated),
00355
          NULL, this);
00356
              m_AXText->Connect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::XA_Updated), NULL,
00357
          this);
00358
              m AYText->Connect (wxEVT COMMAND TEXT UPDATED, wxCommandEventHandler (MvFrame1::YA Updated), NULL,
          this):
00359
              m AZText->Connect (wxEVT COMMAND TEXT UPDATED, wxCommandEventHandler (MyFrame1:: ZA Updated), NULL,
          this);
00360
              \verb|m_ThXText-> Connect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::XTheta_Updated)|, wxCommandEventHandler(MyFrame1::XTheta_Updat
00361
          NULL, this);
00362
              m ThYText->Connect (wxEVT COMMAND TEXT UPDATED, wxCommandEventHandler(MyFrame1::YTheta Updated),
          NULL, this);
00363
              m_ThZText->Connect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::ZTheta_Updated),
          NULL, this);
00364
00365
              m PhiXText->Connect (wxEVT COMMAND TEXT UPDATED, wxCommandEventHandler (MyFrame1::XPhi Updated),
          NULL, this);
00366
              m_PhiYText->Connect (wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::YPhi_Updated),
          NULL, this);
00367
              m_PhiZText
                             ->Connect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::ZPhi_Updated),
          NULL, this);
00368
00369
              m radioBox1->Connect(wxEVT COMMAND RADIOBOX SELECTED.
          wxCommandEventHandler(MvFramel::DotsLines Updated), NULL, this);
00370
              animationBox->Connect (wxEVT_COMMAND_RADIOBOX_SELECTED,
          wxCommandEventHandler(MyFramel::Animation_Updated), NULL, this);
00371
              ParamBox->Connect (wxEVT_COMMAND_RADIOBOX_SELECTED,
          wxCommandEventHandler(MyFrame1::ParamType_Updated), NULL, this);
00372
00373
              m timer.Start(10);
00374 }
00375
00376 MyFrame1::~MyFrame1()
00377 {
00378
               //disconnect-v
00379
              this->Disconnect (wxEVT CLOSE WINDOW, wxCloseEventHandler(MvFramel::MainFormClose)):
              m_DisplayWindow->Disconnect(wxEVT_UPDATE_UI,
00380
          wxUpdateUIEventHandler(MyFrame1::DisplayPanelRepaint), NULL, this);
00381
00382
              m_XRotationSlider->Disconnect(wxEVT_SCROLL_TOP, wxScrollEventHandler(MyFrame1::XRot_Updated),
          NULL, this);
00383
              m XRotationSlider->Disconnect(wxEVT SCROLL BOTTOM, wxScrollEventHandler(MvFrame1::XRot Updated),
          NULL, this);
00384
              m_XRotationSlider->Disconnect(wxEVT_SCROLL_LINEUP, wxScrollEventHandler(MyFramel::XRot_Updated),
          NULL, this);
00385
              m_XRotationSlider->Disconnect(wxEVT_SCROLL_LINEDOWN, wxScrollEventHandler(MyFrame1::XRot_Updated),
          NULL, this);
              00386
          NULL, this);
00387
              m XRotationSlider->Disconnect(wxEVT SCROLL PAGEDOWN, wxScrollEventHandler(MvFrame1::XRot Updated),
          NULL, this);
              m_XRotationSlider->Disconnect(wxEVT_SCROLL_THUMBTRACK,
00388
          wxScrollEventHandler(MyFrame1::XRot_Updated), NULL, this);
00389
              m XRotationSlider->Disconnect(wxEVT SCROLL THUMBRELEASE,
          wxScrollEventHandler(MyFrame1::XRot Updated), NULL, this);
00390
              m_XRotationSlider->Disconnect(wxEVT_SCROLL_CHANGED, wxScrollEventHandler(MyFrame1::XRot_Updated),
00391
00392
              m_YRotationSlider->Disconnect(wxEVT_SCROLL_TOP, wxScrollEventHandler(MyFrame1::YRot_Updated),
          NULL, this);
00393
              m YRotationSlider->Disconnect(wxEVT SCROLL BOTTOM, wxScrollEventHandler(MvFramel::YRot Updated),
          NULL, this);
00394
              m_YRotationSlider->Disconnect(wxEVT_SCROLL_LINEUP, wxScrollEventHandler(MyFrame1::YRot_Updated),
          NULL, this);
00395
              m_YRotationSlider->Disconnect(wxEVT_SCROLL_LINEDOWN, wxScrollEventHandler(MyFrame1::YRot_Updated),
          NULL, this);
00396
              m YRotationSlider->Disconnect(wxEVT SCROLL PAGEUP, wxScrollEventHandler(MvFrame1::YRot Updated),
          NULL, this);
00397
              m_YRotationSlider->Disconnect(wxEVT_SCROLL_PAGEDOWN, wxScrollEventHandler(MyFrame1::YRot_Updated),
          NULL, this);
00398
              m_YRotationSlider->Disconnect(wxEVT_SCROLL_THUMBTRACK,
          wxScrollEventHandler (MyFrame1::YRot_Updated), NULL, this);
              m_YRotationSlider->Disconnect(wxEVT_SCROLL_THUMBRELEASE,
00399
          wxScrollEventHandler(MyFrame1::YRot Updated), NULL, this);
00400
              m_YRotationSlider->Disconnect(wxEVT_SCROLL_CHANGED, wxScrollEventHandler(MyFramel::YRot_Updated),
          NULL, this);
00401
00402
              m_ZRotationSlider->Disconnect(wxEVT_SCROLL_TOP, wxScrollEventHandler(MyFrame1::ZRot_Updated),
          NULL, this):
00403
              m ZRotationSlider->Disconnect(wxEVT SCROLL BOTTOM, wxScrollEventHandler(MvFrame1::ZRot Updated),
```

5.22 Window.cpp 93

```
NULL, this);
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_LINEUP, wxScrollEventHandler(MyFrame1::ZRot_Updated),
00404
       NULL, this);
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_LINEDOWN, wxScrollEventHandler(MyFrame1::ZRot_Updated),
00405
       NULL, this);
00406
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_PAGEUP, wxScrollEventHandler(MyFramel::ZRot_Updated),
       NULL, this);
00407
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_PAGEDOWN, wxScrollEventHandler(MyFramel::ZRot_Updated),
       NULL, this);
00408
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_THUMBTRACK,
       {\tt wxScrollEventHandler(MyFrame1::ZRot\_Updated),\ NULL,\ this);}
00409
          m ZRotationSlider->Disconnect(wxEVT SCROLL THUMBRELEASE.
       wxScrollEventHandler(MyFrame1::ZRot_Updated), NULL, this);
00410
          m_ZRotationSlider->Disconnect(wxEVT_SCROLL_CHANGED, wxScrollEventHandler(MyFrame1::ZRot_Updated),
       NULL, this);
00411
          m_AXText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::XA_Updated),
00412
       NULL, this);
00413
          m_AYText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::YA_Updated),
       NULL, this);
          m_AZText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::ZA_Updated),
00414
       NULL, this);
00415
          m_ThXText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::XTheta_Updated),
00416
       NULL, this);
00417
          m_ThYText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::YTheta_Updated),
       NULL, this);
00418
          m_ThZText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFramel::ZTheta_Updated),
       NULL, this);
00419
          m PhiXText->Disconnect(wxEVT COMMAND TEXT UPDATED, wxCommandEventHandler(MvFrame1::XPhi Updated),
00420
       NULL, this);
00421
          m_PhiYText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFrame1::YPhi_Updated),
       NULL, this);
00422
          m_PhiZText->Disconnect(wxEVT_COMMAND_TEXT_UPDATED, wxCommandEventHandler(MyFramel::ZPhi_Updated),
       NULL, this);
00423
00424
          m_radioBox1->Disconnect(wxEVT_COMMAND_RADIOBOX_SELECTED,
       wxCommandEventHandler(MyFramel::DotsLines_Updated), NULL, this);
00425
          animationBox->Disconnect(wxEVT_COMMAND_RADIOBOX_SELECTED,
       wxCommandEventHandler(MyFramel::Animation_Updated), NULL, this);
          ParamBox->Disconnect(wxEVT_COMMAND_RADIOBOX_SELECTED,
00426
       wxCommandEventHandler(MyFramel::ParamType_Updated), NULL, this);
00427 }
```

Index

o GUMvFrame	R, 24
~GUIMyFrame GUIMyFrame, 29	· ·
~MyFrame1	Set_Animation, 18 Set ParamType, 18
MyFrame1, 49	Set_Points, 19
animation	SetR, 19
ConfigClass, 23	SetX_A, 19
Animation_Updated	SetX_phi, 20
GUIMyFrame, 29	SetX_Rot, 20
MyFrame1, 50	SetX_theta, 20
animationBox	SetY_A, 21
	SetY_phi, 21
MyFrame1, 59	SetY_Rot, 21
AnimationBreak	SetY_theta, 22
MyFrame1, 50	SetZ_A, 22
cfg	SetZ_phi, 22
ChartClass, 10	SetZ_Rot, 23
GUIMyFrame, 38	SetZ_theta, 23
ChartClass, 7	x, 24
	X_A, 24
cfg, 10	X_phi, 24
ChartClass, 7	X_Rot, 25
Draw, 8	X_theta, 25
GUIMyFrame, 37	y, 25
timer, 10	Y_A, 25
ChartClass.cpp	Y_phi, 25
min, 79	Y_Rot, 25
ChartClass.h	Y_theta, 26
min, 71	z, 26
ConfigClass, 11	Z_A, 26
animation, 23	Z_phi, 26
ConfigClass, 13	Z_Rot, 26
get_Animation, 13	Z_theta, 26
get_Points, 13	
get_Polar, 13	data
getR, 14	Matrix4, 40
getX_A, 14	Vector4, 69
getX_phi, 14	DisplayPanelRepaint
getX_Rot, 15	GUIMyFrame, 30
getX_theta, 15	MyFrame1, 51
getY_A, 15	DotsLines_Updated
getY_phi, 16	GUIMyFrame, 30
getY_Rot, 16	MyFrame1, 51
getY_theta, 16	Draw
getZ_A, 17	ChartClass, 8
getZ_phi, 17	
getZ_Rot, 17	get_Animation
getZ_theta, 18	ConfigClass, 13
MainWindow, 23	get_Points
points, 24	ConfigClass, 13
polar, 24	get_Polar
	- -

ConfigClass, 13	main.cpp, 78
getR	include/ChartClass.h, 71, 72
ConfigClass, 14	include/ConfigClass.h, 72, 73
GetX	include/GUIMyFrame.h, 73, 74
Vector4, 67	include/vecmat.h, 75
getX_A	include/Window.h, 76
ConfigClass, 14	- AVTout
getX_phi	m_AXText
ConfigClass, 14	MyFrame1, 59
getX_Rot	m_AYText
ConfigClass, 15	MyFrame1, 59
getX_theta	m_AZText
ConfigClass, 15	MyFrame1, 60
GetY	m_DisplayWindow
Vector4, 67	MyFrame1, 60
getY_A	m_PhiXText
ConfigClass, 15	MyFrame1, 60
getY_phi	m_PhiYText
ConfigClass, 16	MyFrame1, 60
getY_Rot	m_PhiZText
ConfigClass, 16	MyFrame1, 60
getY_theta	m_radioBox1
ConfigClass, 16	MyFrame1, 61
GetZ	m_staticText17
Vector4, 67	MyFrame1, 61
getZ_A	m_staticText18
ConfigClass, 17	MyFrame1, 61
getZ_phi	m_staticText19
ConfigClass, 17	MyFrame1, 61
getZ_Rot	m_staticText20
ConfigClass, 17	MyFrame1, 61
getZ_theta	m_staticText21
ConfigClass, 18	MyFrame1, 62
GUIMyFrame, 27	m_staticText22
~GUIMyFrame, 29	MyFrame1, 62
Animation_Updated, 29	m_staticText23
cfg, 38	MyFrame1, 62
ChartClass, 37	m_staticText4
DisplayPanelRepaint, 30	MyFrame1, 62
DotsLines_Updated, 30	m_staticText5
GUIMyFrame, 29	MyFrame1, 62
MainFormClose, 31	m staticText51
MyFrame1, 59	MyFrame1, 63
ParamType Updated, 31	m staticText511
Repaint, 32	MyFrame1, 63
XA Updated, 32	m_staticText512
	MyFrame1, 63
XPhi_Updated, 33	m staticText52
XRot_Updated, 33	MyFrame1, 63
XTheta_Updated, 33	m staticText521
YA_Updated, 34	MyFrame1, 63
YPhi_Updated, 34	m staticText522
YRot_Updated, 35	MyFrame1, 64
YTheta_Updated, 35	m_staticText53
ZA_Updated, 36	MyFrame1, 64
ZPhi_Updated, 36	m staticText54
ZRot_Updated, 36	MyFrame1, 64
ZTheta_Updated, 37	-
IMDI EMENT ADD	m_ThXText
IMPLEMENT_APP	MyFrame1, 64

m_ThYText	m_staticText52, 63
MyFrame1, 64	m_staticText521, 63
m_ThZText	m_staticText522, 64
MyFrame1, 65	m_staticText53, 64
m_timer	m_staticText54, 64
MyFrame1, 65	m_ThXText, 64
m_XRotationSlider	m_ThYText, 64
MyFrame1, 65	m_ThZText, 65
m YRotationSlider	m_timer, 65
MyFrame1, 65	m_XRotationSlider, 65
m ZRotationSlider	m YRotationSlider, 65
MyFrame1, 65	m ZRotationSlider, 65
main.cpp, 77	MainFormClose, 51
IMPLEMENT_APP, 78	MyFrame1, 43
MainFormClose	ParamBox, 66
GUIMyFrame, 31	ParamType_Updated, 52
MyFrame1, 51	Scrolls_Updated, 52
MainWindow	XA_Updated, 53
ConfigClass, 23	XPhi_Updated, 53
Matrix4, 38	XRot Updated, 53
data, 40	XTheta_Updated, 54
Matrix4, 38	YA_Updated, 54
	<u> </u>
operator*, 39	YPhi_Updated, 54
Print, 39	YRot_Updated, 55
min 70	YTheta_Updated, 55
ChartClass.cpp, 79	ZA_Updated, 55
ChartClass.h, 71	ZPhi_Updated, 57
MyApp, 40	ZRot_Updated, 57
OnExit, 41	ZTheta_Updated, 57
Onlnit, 41	
	0
MyFrame1, 41	OnExit
∼MyFrame1, 49	MyApp, 41
-	MyApp, 41 OnInit
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59	MyApp, 41 OnInit MyApp, 41
~MyFrame1, 49 Animation_Updated, 50	MyApp, 41 OnInit MyApp, 41 operator*
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59	MyApp, 41 OnInit MyApp, 41
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator-
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiYText, 60	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiXText, 60 m_PhiZText, 60 m_PhiZText, 60 m_PhiZText, 60	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiXText, 60 m_PhiZText, 60 m_PhiZText, 60 m_radioBox1, 61	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiYText, 60 m_PhiZText, 60 m_radioBox1, 61 m_staticText17, 61	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24 polar
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiXText, 60 m_PhiZText, 60 m_PhiZText, 60 m_radioBox1, 61 m_staticText17, 61 m_staticText19, 61	MyApp, 41 OnInit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24 polar ConfigClass, 24
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiYText, 60 m_PhiZText, 60 m_radioBox1, 61 m_staticText17, 61 m_staticText19, 61 m_staticText20, 61	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24 polar ConfigClass, 24 Print
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiXText, 60 m_PhiZext, 60 m_PhiZext, 60 m_radioBox1, 61 m_staticText17, 61 m_staticText18, 61 m_staticText19, 61 m_staticText20, 61 m_staticText21, 62	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24 polar ConfigClass, 24 Print Matrix4, 39
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiYText, 60 m_PhiZText, 60 m_radioBox1, 61 m_staticText17, 61 m_staticText19, 61 m_staticText20, 61 m_staticText21, 62 m_staticText22, 62	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24 polar ConfigClass, 24 Print
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiXText, 60 m_PhiZText, 60 m_radioBox1, 61 m_staticText17, 61 m_staticText18, 61 m_staticText20, 61 m_staticText21, 62 m_staticText22, 62 m_staticText23, 62	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24 polar ConfigClass, 24 Print Matrix4, 39 Vector4, 68
~MyFrame1, 49 Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiXText, 60 m_PhiZText, 60 m_radioBox1, 61 m_staticText17, 61 m_staticText18, 61 m_staticText20, 61 m_staticText21, 62 m_staticText23, 62 m_staticText4, 62	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24 polar ConfigClass, 24 Print Matrix4, 39 Vector4, 68
Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AZText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiXText, 60 m_PhiZText, 60 m_radioBox1, 61 m_staticText17, 61 m_staticText18, 61 m_staticText20, 61 m_staticText21, 62 m_staticText23, 62 m_staticText4, 62 m_staticText5, 62	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24 polar ConfigClass, 24 Print Matrix4, 39 Vector4, 68 R ConfigClass, 24
Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AYText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiXText, 60 m_PhiZText, 60 m_radioBox1, 61 m_staticText17, 61 m_staticText18, 61 m_staticText19, 61 m_staticText20, 61 m_staticText21, 62 m_staticText23, 62 m_staticText4, 62 m_staticText51, 63	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24 polar ConfigClass, 24 Print Matrix4, 39 Vector4, 68 R ConfigClass, 24 Repaint
Animation_Updated, 50 animationBox, 59 AnimationBreak, 50 DisplayPanelRepaint, 51 DotsLines_Updated, 51 GUIMyFrame, 59 m_AXText, 59 m_AZText, 59 m_AZText, 60 m_DisplayWindow, 60 m_PhiXText, 60 m_PhiXText, 60 m_PhiZText, 60 m_radioBox1, 61 m_staticText17, 61 m_staticText18, 61 m_staticText20, 61 m_staticText21, 62 m_staticText23, 62 m_staticText4, 62 m_staticText5, 62	MyApp, 41 Onlnit MyApp, 41 operator* Matrix4, 39 vecmat.cpp, 85 Vector4, 68 operator- Vector4, 67 ParamBox MyFrame1, 66 ParamType_Updated GUIMyFrame, 31 MyFrame1, 52 points ConfigClass, 24 polar ConfigClass, 24 Print Matrix4, 39 Vector4, 68 R ConfigClass, 24

Caralla Undeted	ConfigClass 04
Scrolls_Updated MyFrame1, 52	ConfigClass, 24 X A
Set	ConfigClass, 24
Vector4, 68	X_phi
Set Animation	ConfigClass, 24
ConfigClass, 18	X Rot
Set_ParamType	ConfigClass, 25
ConfigClass, 18	X theta
Set_Points	ConfigClass, 25
ConfigClass, 19	XA_Updated
SetR	GUIMyFrame, 32
ConfigClass, 19	MyFrame1, 53
SetX_A	XPhi_Updated
ConfigClass, 19	GUIMyFrame, 33
SetX_phi	MyFrame1, 53
ConfigClass, 20	XRot_Updated
SetX_Rot	GUIMyFrame, 33
ConfigClass, 20	MyFrame1, 53
SetX_theta ConfigClass, 20	XTheta_Updated GUIMyFrame, 33
SetY A	MyFrame1, 54
ConfigClass, 21	wyr ramer, 54
SetY_phi	у
ConfigClass, 21	ConfigClass, 25
SetY Rot	Y_A
ConfigClass, 21	ConfigClass, 25
SetY_theta	Y_phi
ConfigClass, 22	ConfigClass, 25
SetZ_A	Y_Rot
ConfigClass, 22	ConfigClass, 25
SetZ_phi	Y_theta
ConfigClass, 22	ConfigClass, 26
SetZ_Rot	YA_Updated
ConfigClass, 23	GUIMyFrame, 34 MyFrame1, 54
SetZ_theta	YPhi_Updated
ConfigClass, 23	GUIMyFrame, 34
src/ChartClass.cpp, 78, 79 src/ConfigClass.cpp, 81	MyFrame1, 54
src/GUIMyFrame.cpp, 82	YRot Updated
src/vecmat.cpp, 85, 86	GUIMyFrame, 35
src/Window.cpp, 87	MyFrame1, 55
	YTheta_Updated
timer	GUIMyFrame, 35
ChartClass, 10	MyFrame1, 55
versus de la companya del companya del companya de la companya de	_
vecmat.cpp	Z ConfigClass 00
operator*, 85	ConfigClass, 26
Vector4, 66	Z_A ConfigClass, 26
data, 69 GetX, 67	Z_phi
GetX, 67 GetY, 67	ConfigClass, 26
Get7, 67	Z Rot
operator*, 68	ConfigClass, 26
operator-, 67	Z_theta
Print, 68	ConfigClass, 26
Set, 68	ZA_Updated
Vector4, 67	GUIMyFrame, 36
	MyFrame1, 55
X	ZPhi_Updated

GUIMyFrame, 36 MyFrame1, 57 ZRot_Updated GUIMyFrame, 36 MyFrame1, 57 ZTheta_Updated GUIMyFrame, 37 MyFrame1, 57