Skynet Robot Controller

T1000

Generated by Doxygen 1.9.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

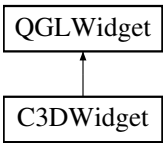
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 C3DWidget Class Reference

Inheritance diagram for C3DWidget:



Public Member Functions

- **C3DWidget** (QWidget *parent=nullptr)
- void **button_rotate** (bool value)
- void **erase_all** ()
- void **erase_selected** ()
- void **rotate_line** ()
- void **translate_line** ()
- void **draw_line** (gp_Pnt point1, gp_Pnt point2)
- void **draw_3p_arc** (gp_Pnt point1, gp_Pnt point2, gp_Pnt point3)
- void **draw_circle** (gp_Pnt center, double radius)
- void **draw_cp_arc** (gp_Pnt center, gp_Pnt point1, gp_Pnt point2)
- void **zoom_extends** ()
- void **convert_parsed_gcode** (std::vector< gcode > data)
- void **readstepfile** (const std::string &theStepName, int i)
- void update_joint_slider (int project, int joint, int degrees)
- void **get_various_positions** ()
- void **set_orthographic** ()
- void **set_perspective** ()
- void visit (const TDF_Label &theLabel, int i)

Protected Types

- enum **CurrentAction3d** { **CurAction3d_Nothing**, **CurAction3d_DynamicPanning**, **CurAction3d_DynamicZooming**, **CurAction3d_DynamicRotation** }

Protected Member Functions

- void **paintEvent** (QPaintEvent ∗)
- void **resizeEvent** (QResizeEvent ∗)
- void **mousePressEvent** (QMouseEvent ∗event)
- void **mouseReleaseEvent** (QMouseEvent ∗event)
- void **mouseMoveEvent** (QMouseEvent ∗event)
- void **wheelEvent** (QWheelEvent ∗event)

3.1.1 Member Function Documentation

3.1.1.1 update_joint_slider()

```
void C3DWidget::update_joint_slider (
            int project,
            int joint,
            int degrees )
```

A Forward kinematic done by OpenCascade's transform matrix. A joint can be readed as a chain. All code example's are compatible with the Kuka robot project template.

A Example to rotate a AIS_Shape, this is related to the stepfile objects viewed by the OpenCascade screen :

Set up a transformation matrix.

```
gp_Trsf MyTrsf;
```

Perform transformation.

```
MyTrsf.SetRotation(gp_Ax1(gp_Pnt(xpos,ypos,zpos), gp_Dir(
        xflag,
        yflag,
        zflag)),rotation_value_in_degrees * M_PI /180);
```

Apply the transformation to a AIS_Shape

```
ProjectVec.at(0).ChainVec.at(0).Ais_ShapeVec.at(0)->SetLocalTransformation(MyTrsf);
```

Testcase

```
gp_Trsf test;
test.SetRotation(gp_Ax1(gp_Pnt(0,0,0),gp_Dir(0,0,1)),degrees * M_PI /180);
```

project is a integer projectnumber. joint is a integer joint. Can be readed as a chain. degrees is the joint rotation value in degrees. We take the transformation matrix from the ProjectVec[i] struct.

ProjectVec[i] -> ChainVec[i] -> MyTrsf={};

From the ProjectVec we hold the data in :

- POINT FrameVectorPoint ={0.0,0.0,0.0};
- bool RotationflagX=0;
- bool RotationflagY=0;
- bool RotationflagZ=0;

Testcase:

for(unsigned int i=0; i< ProjectVec.at(project).ChainVec.at(joint).Ais_ShapeVec.size(); i++){ ProjectVec.↩ at(project).ChainVec.at(joint).Ais_ShapeVec.at(i)->SetLocalTransformation(ProjectVec.at(project).ChainVec.↩ at(joint).MyTrsf); }

Set local tranformation.

Example Apply transformation for one shape

```
for(unsigned int i=0; i< ProjectVec.at(project).ChainVec.at(joint).Ais_ShapeVec.size(); i++){
    ProjectVec.at(project).ChainVec.at(joint).Ais_ShapeVec.at(i)->SetLocalTransformation(ProjectVec.at(project
}
```

Example Apply transformation to all unique shapes

```
for(unsigned int i=0; i<ProjectVec.size(); i++){
    for(unsigned int j=0; j<ProjectVec.at(i).ChainVec.size(); j++){
        for(unsigned int k=0; k<ProjectVec.at(i).ChainVec.at(j).Ais_ShapeVec.size(); k++){

            ProjectVec.at(i).ChainVec.at(j).Ais_ShapeVec.at(k)->SetLocalTransformation(ProjectVec.at(i).ChainV
        }
    }
}
```

Example Multiply each transformation

```
gp_Trsf level0=ProjectVec.at(0).ChainVec.at(0).MyTrsf;
gp_Trsf level1=ProjectVec.at(0).ChainVec.at(1).MyTrsf;
gp_Trsf level2=ProjectVec.at(0).ChainVec.at(2).MyTrsf;
gp_Trsf level3=ProjectVec.at(0).ChainVec.at(3).MyTrsf;
gp_Trsf level4=ProjectVec.at(0).ChainVec.at(4).MyTrsf;
gp_Trsf level5=ProjectVec.at(0).ChainVec.at(5).MyTrsf;
gp_Trsf level6=ProjectVec.at(0).ChainVec.at(6).MyTrsf;

        level0=level0;
gp_Trsf level0x1=level0*level1;
gp_Trsf level0x1x2 = level0x1*level2;
gp_Trsf level0x1x2x3 = level0x1x2*level3;
gp_Trsf level0x1x2x3x4 = level0x1x2x3*level4;
gp_Trsf level0x1x2x3x4x5 = level0x1x2x3x4*level5;
gp_Trsf level0x1x2x3x4x5x6 = level0x1x2x3x4x5*level6;

Apply multipied transformation.
ProjectVec.at(0).ChainVec.at(0).Ais_ShapeVec.at(0)->SetLocalTransformation(level0);
ProjectVec.at(0).ChainVec.at(1).Ais_ShapeVec.at(0)->SetLocalTransformation(level0x1);
ProjectVec.at(0).ChainVec.at(2).Ais_ShapeVec.at(0)->SetLocalTransformation(level0x1x2);
ProjectVec.at(0).ChainVec.at(3).Ais_ShapeVec.at(0)->SetLocalTransformation(level0x1x2x3);
ProjectVec.at(0).ChainVec.at(4).Ais_ShapeVec.at(0)->SetLocalTransformation(level0x1x2x3x4);
ProjectVec.at(0).ChainVec.at(5).Ais_ShapeVec.at(0)->SetLocalTransformation(level0x1x2x3x4x5);
ProjectVec.at(0).ChainVec.at(6).Ais_ShapeVec.at(0)->SetLocalTransformation(level0x1x2x3x4x5x6);
```

3.1.1.2 visit()

```
void C3DWidget::visit (
            const TDF_Label & theLabel,
            int i )
```

Example how to load a shape to the screen : Handle(AIS_Shape) ais_Shape = new AIS_Shape(aShape); ais_Shape->SetColor(col); ais_Shape->SetDisplayMode(AIS_Shaded); ais_Shape->Attributes()->SetFace↩ BoundaryDraw(true); ais_Shape->Attributes()->SetFaceBoundaryAspect(new Prs3d_LineAspect(Quantity_↩ NOC_BLACK, Aspect_TOL_SOLID, 1.)); ais_Shape->Attributes()->SetIsoOnTriangulation(true);

Convert TopoDS_Shape to AIS_Shape and copy it to the project vector. Set the viewstyle to shade with wireframe.

Preview the AIS_Shape to the screen with m_context.

Repeat the visit function for each childmember.

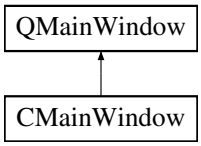
Zoom extends

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

- c3dwidget.h
- c3dwidget.cpp

3.2 CMainWindow Class Reference

Inheritance diagram for CMainWindow:



Public Member Functions

- **CMainWindow** (QWidget ∗parent=0)

Public Attributes

- C3DWidget ∗ **m_3d_widget**

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

- cmainwindow.h
- cmainwindow.cpp

3.3 color Struct Reference

Public Attributes

- double **red** =0
- double **green** =0
- double **blue** =0

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

- c3dwidget.cpp

3.4 FWKIN_DEG Struct Reference

Public Attributes

- double **J1** =0
- double **J2** =0
- double **J3** =0
- double **J4** =0
- double **J5** =0
- double **J6** =0
- double **Xee** =0
- double **Yee** =0
- double **Zee** =0
- double **EulerZ** =0
- double **EulerY** =0
- double **EulerX** =0
- double **Xtrans** =0
- double **Ytrans** =0
- double **Ztrans** =0

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

- kinematics.h

3.5 gcode Struct Reference

Public Attributes

- std::string **type** ="g0"
- double **xs** =0
- double **ys** =0
- double **zs** =0
- double **xe** =0
- double **ye** =0
- double **ze** =0
- double **xc** =0

- double **yc** =0
- double **zc** =0
- double **xcon** =0
- double **ycon** =0
- double **zcon** =0
- double **d** =0
- double **pi_start** =0
- double **pi_end** =0

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

- read_ngc.h

3.6 kinematics Class Reference

Public Member Functions

- struct FWKIN_DEG **kinematics_fwd** (struct FWKIN_DEG)
- struct FWKIN_DEG **kinematics_inv** (struct FWKIN_DEG)

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

- kinematics.h
- kinematics.cpp

3.7 read_ngc Class Reference

Public Member Functions

- void **read_ngc_file** (std::string filename)
- void **parse_gcode** (std::vector< std::string > stringvector)
- std::vector< std::string > **split_line** (std::string string)
- void **print_gcode_result** (std::vector< gcode > data)
- gcode **create_arc_controlpoint** (gcode data)

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

- read_ngc.h
- read_ngc.cpp

Index