OpenPilot
Gauge Version 0.2.0

Technical Documentation

Gauge Widget
Plugin API Details

Tim Brocklehurst
24th Feb 2008
Revision 0

Table of Contents

Introduction

This widget is intended to provide an easy way to draw custom gauges. The widget is extendible and highly configurable, with the option to draw simple, or more complex gauges as required. The user has a considerable amount of control over the design of a base gauge, as well as being able to choose the design (plate) and arrow individually.

Installation under Linux

Here we assume that the gauge widget is unzipped to ~/gauge and the project you wish to include the gauge in is ~/project. The following commands will install the gauge widget:

```
cd ~/gauge
make
su root
make install
cd /usr/lib
ln -s /usr/lib/qt4/plugins/designer/libgauge.so
logout
```

As of version 0.2.0 this widget installs the required header files into /usr/include/qt4/OpenPilot and therefore, developers should give themselves appropriate permissons on this directory. It is also worth giving developers write permission to /usr/lib/qt4/plugins/designer, this will save a lot of time when re-building the library.

Installation under Win32

Here we assume that the gauge widget is unzipped to D:\gauge and the project you wish to include the gauge in is D:\project. The following commands will install the gauge widget:

```
cd D:\gauge
make install
copy release\gauge.dll C:\Qt\4.3.3\lib\gauge.dll
```

This process must be repeated every time the widget is updated. It is advisable to write a batch script to automate this process if you intend to develop the widget under windows.

Using The Widget

Once you have successfully installed the widget, open QTDesigner and you will see a new widget called “Gauge” in the “Display Widgets” section of QTDesigner. You can now just drag and drop the widget as you would for any other. However, you must ensure that your project file has the following lines in the global section.

```
INCLUDEPATH += $$[QT_INSTALL_HEADERS]/OpenPilot
LIBS += -lgauge
```

The header file is <gauge.h>, it is automatically included by uic when using QTDesigner.

Signals

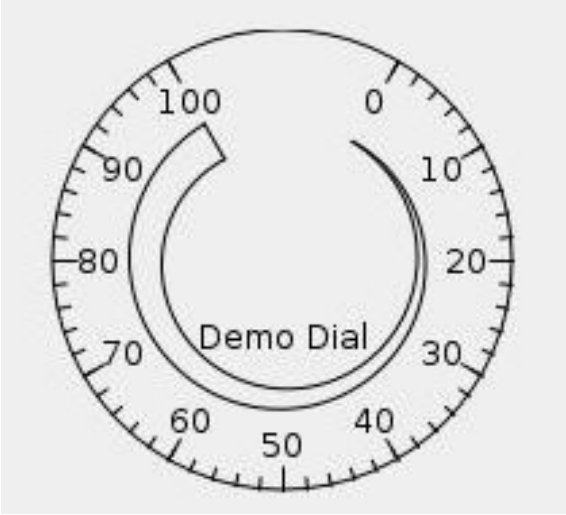
changed()

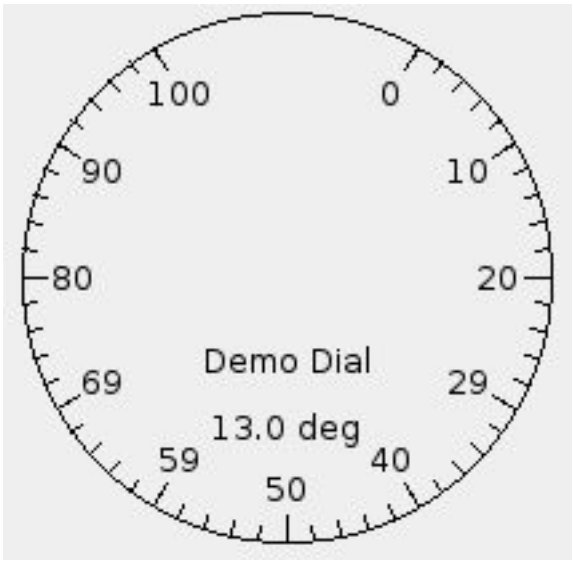
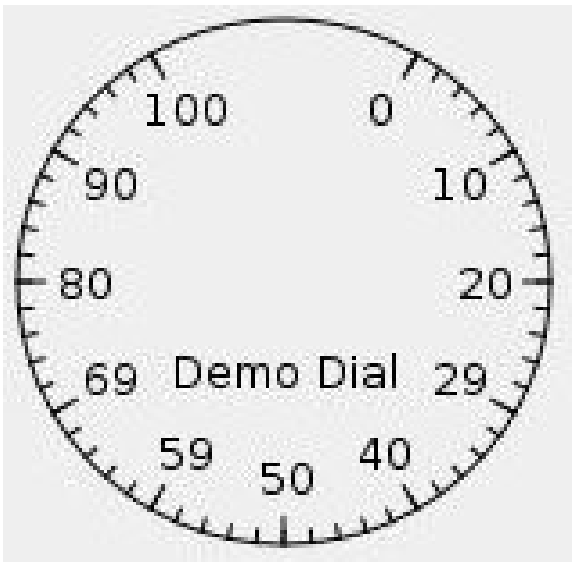
This signal tells the rest of the form that the widget has changed. This may be useful in certain circumstances.

Slots

void setPlateType(int)

This slot allows the user to choose which gauge style to use. The designation “plate” comes from the dial-plate used on “proper” gauges.

Plate	Illustration
-1	No plate. This setting is widely used when drawing gauges with more than one pointer. Since the widgets are transparent, they may be overlayed, to create more complex gauges.
0	

1	
2	

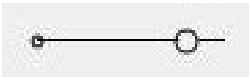
Default: 0

int plateType()

Returns the current plate.

void setArrowType(int)

This slot allows the user to choose which arrow (or “needle”) to use on the gauge.

Arrow	Illustration
0	

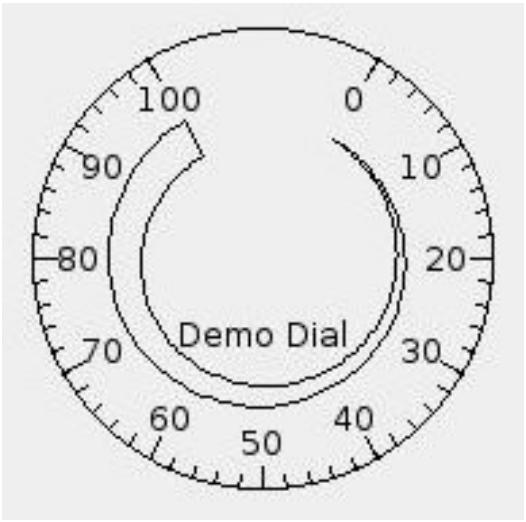
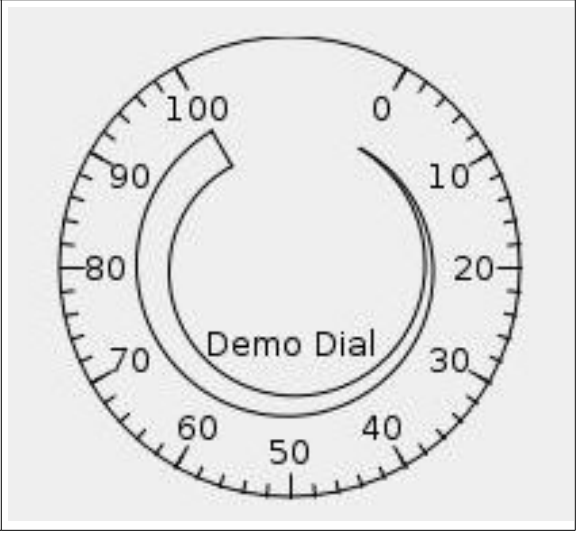
Default: 0

int arrowType()

Returns the current arrow.

void setAntialias(bool)

Allows the user to anti-alias the gauge. This produces “smoother” gauges, but requires a little more CPU power. It is recommended that anti-aliasing is used.

False (Normal)	True (Anti-aliased)
	

Default: True

bool antialias()

Returns true if anti-aliasing is used.

void setValue(int)

Sets the gauge to an integer value. This is provided to accept signals from other widgets. This is internally limited to be within the limits of the scale.

Default: 13

void setValue(double)

Sets the gauge to a double-precision value. This is the normal way to set the gauge value. This is internally limited to be within the limits of the scale.

Default: 13

double value()

Returns the current gauge value as a double-precision value.

int value_int()

Returns the gauge value to the nearest integer.

void setMaximum(double)

Sets the maximum value that can be plotted on the scale.

Default: 100

double maximum()

Returns the maximum limit of the scale.

void setMinimum(double)

Sets the minimum value that can be plotted on the scale.

Default: 0

double minimum()

Returns the minimum limit of the scale

void setLabel(QString)

Sets the gauge label.

Default: “Demo Dial”

QString label()

Returns the gauge label

void setTics(int)

Sets the number of major tick-marks around the dial.

Default: 11

int tics()

Returns the number of major tick-marks

void setSubTics(int)

Sets the number of minor tick marks which make up each major tick. For example, to space minor tick-marks at 0.2, between major tick-marks 0 and 1, use setSubTics(5).

Default: 5

int subTics()

Returns the number of minor tick-marks that make up one major tick-mark

void setStartAngle(double)

Sets the start angle for the scale. This may be greater than the end angle or greater than 360 degrees.

Default: 30

double startAngle()

Returns the start angle for the scale.

void setEndAngle(double)

Sets the end angle for the scale. This may be less than the start angle, or greater than 360 degrees. Using start angle and end angle together allows for all gauge orientations

Default: 330

double endAngle()

Returns the end angle for the scale.

void setChevronWidth(double)

Sets the current maximum width of the chevron for plate 0.

Default: 15

double chevronWidth()

Returns the current maximum width of the chevron for plate 0.

void setQuality(int)

Sets the number of points used to draw the chevron on plate 0. This should be increased as the gauge is made bigger, and may be reduced as it is made smaller.

Default: 120

int quality()

Returns the current number of points used to draw the chevron on plate 0.

void setBorder(int)

Reduces the radius of the gauge by the desired number of pixels, whilst keeping the widget-size the same to make using gauges in grid layouts easier.

Default: 0

int border()

Returns the current border.

void setGaugeColor(QColor)

Sets the current gauge foreground colour. This is particularly useful for drawing overlayed gauges.

QColor gaugeColor()

Returns the current gauge foreground colour.

void setIntegerTics(bool)

Sets the gauge to use integer labels on tick marks. Care should be taken to ensure that the tick-marks are representative if this option is used.

bool integerTics()

Returns true if tick marks use integer labels.

void setTextRadius(double)

Sets the radius at which the tick-mark labels should be drawn. The default is 0.75

double textRadius()

Returns the radius at which tick-mark labels will be drawn.

void setSuffix(QString)

Sets the textual suffix applied after the value on plate 1.

QString suffix()

Returns the textual suffix set above.