HBQT-Tutorial

Giovanni Di Maria

April 22, 2011

email: calimero22@yahoo.it

Harbour (2.1) and QT Classes (4.7) - Tutorial

Rev. [1] February 12, 2011 - Rev. [4022] April 22, 2011

www.gruppoeratostene.com/harbour/harbour-tutorials.htm

This tutorial has been created with LaTeX

1 Index of contents

Contents

2 Record of Revision

Revision	Description
1	Feb 12, 2011
10	Feb 16, 2011
50	Feb 21, 2011
100	Feb 24, 2011
200	Feb 28, 2011
300	Mar 02, 2011
400	Mar 04, 2011
500	Mar 06, 2011
600	Mar 07, 2011
700	Mar 08, 2011
800	Mar 09, 2011
900	Mar 15, 2011
1000	Mar 22, 2011
1100	Mar 23, 2011
1200	Mar 24, 2011
1300	Mar 24, 2011
1400	Mar 30, 2011
1500	Mar 30, 2011
2000	Mar 30, 2011
2500	Mar 30, 2011
3000	Apr 03, 2011
3500	Apr 18, 2011
4000	Apr 21, 2011

3 Introduction

This tutorial is a brief, continuously updated, of the use of the classes QT with Harbour language.

It is specifically written for beginners that initially encountered some difficulties in using these classes, however, extremely powerful and efficient.

The approach of the tutorial is different from other guides that are online. It simply focuses a single class or single object, so you do not get lost in the maze of the vast files of examples provided with the product.

This will easily learn to manage and plan their individual class, as needed, and, finally, to put "together" the whole.

Giovanni Di Maria
email: calimero22@yahoo.it

4 Notes for developers

- In order to make good use of the Qt classes, you should see the include file "hbqtgui.ch". The path of this include is (for windows) /hb21/include
- You should consult the Nokia's official documentation
- You should consult this tutorial in 2-pages mode
- If possible, you must not use static variables
- You should disconnect the objects that were attached with the connect method
- It is strongly recommended to use hbformat tool to format the .prg sources
- To create a source, you can copy and paste the code to your text editor and save it with .prg extension

5 Conventions

- The present tutorial shows small codes about QT classes. It organizes the arguments as follows:
  - Title (Class Name);
  - Brief description of the program;
  - Author's name;
  - Screenshot of the application;
  - Harbour source .prg;
- All the sources in this tutorial are formatted with hbformat tool
- The name of the variables follows these rules:
  - Numeric: nVariableNameClearlyDefiningThePurpose e.g. nWidth, nHeight
  - String: cVariableNameClearlyDefiningThePurpose e.g. cWndTitle, cMenuPrompt
  - Logical: lVariableNameClearlyDefiningThePurpose e.g. lVisible, lChecked
  - Date: dVariableNameClearlyDefiningThePurpose e.g. dToday, dTomorrow
  - Object: oVariableNameClearlyDefiningThePurpose e.g. oWnd, oSize

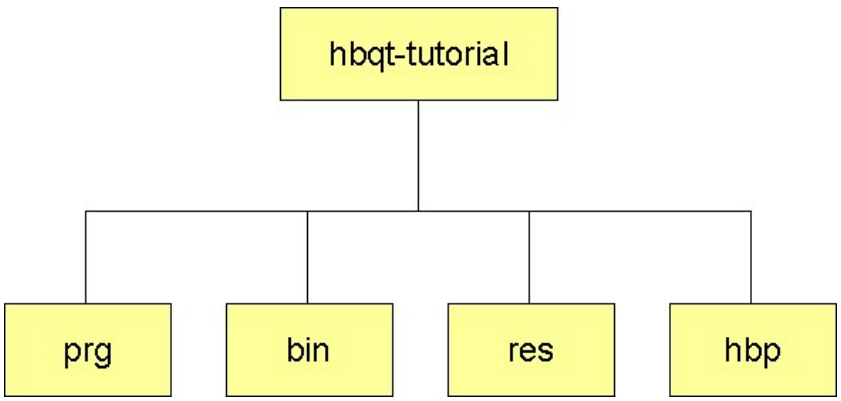
6 Compiling

6.1 Tree of hbqt-tutorial

All the files of present tutorial are stored in different folders. The main folder "hbqt-tutorial" contains the following folders:

- prg for source codes
- bin for binary executables
- res for bmp, dbf and other files
- hbp for the project files.

- The folder "prg " contains all sources in .prg format.
- The folder "bin" contains the compiled and executable files.
- The folder "res" contains all the files programs use, such as images (bmp, png), database (dbf, mdb), and any type of files used by programs.
- The folder "hbp" contains the project files to compile the sources.



6.2 Compiling in Windows

In order to compile correctly your sources, you must follow this procedure:

- Create the file named sample.hbp (or other name) as follow:

  hbqt.hbc
  -w3 -es2
  sample.prg
  otherprogrs.prg

- Create your source program, named sample.prg (or other name).
- Compile with hbmk2 sample.hbp.
- The procedure will create the file sample.exe.
- You can distribute your program with the following files:

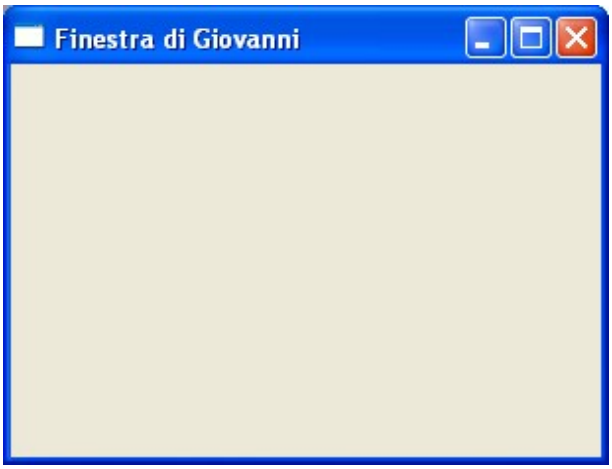
  sample.exe
  libgcc_s_dw2-1.dll
  mingwm10.dll
  QtCore4.dll
  QtGui4.dll

These DLL are located in C:/hb21/bin or in C:/hb21/comp/mingw/bin.

7 QMainWindow

7.1 Window

The following example shows how to create a simple main window. The window is resizable. *(by Giovanni Di Maria)*




```
   // SOURCE=QMainWindow.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

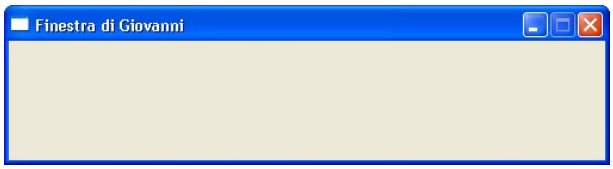
7.2 Window not resizable

The following example shows how to create a simple main window. The window is NOT resizable. *(by Giovanni Di Maria)*


```
    // SOURCE=QMainWindow.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:setFixedSize( 200, 200 )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

7.3 Window with fixed width

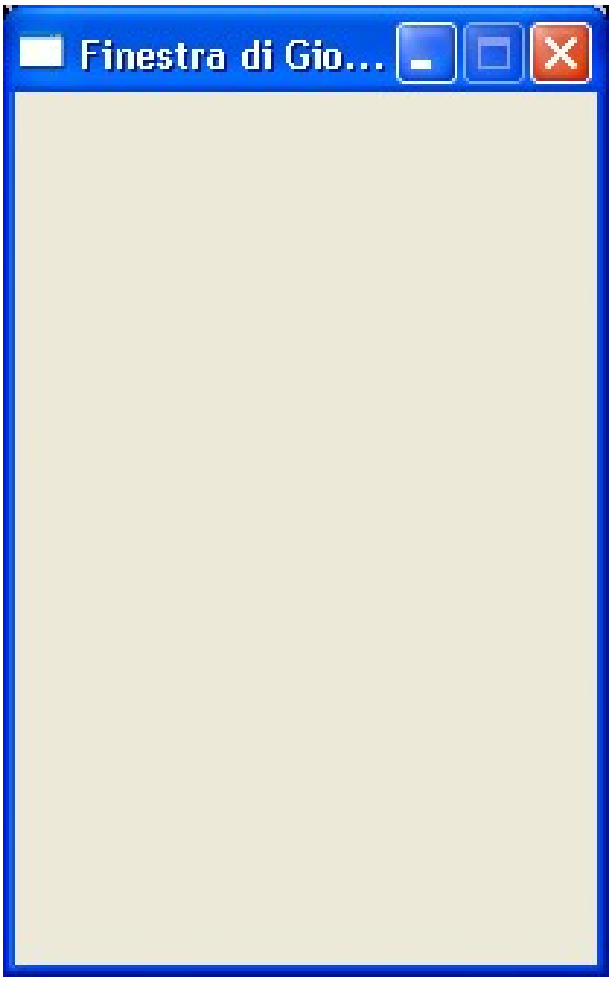
The following example shows how to create a simple main window. The width is locked and the height is resizable. *(by Giovanni Di Maria)*




```
   // SOURCE=QMainWindow.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:setFixedWidth( 500 )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

7.4 Window with fixed height

The following example shows how to create a simple main window. The height is locked and the width is resizable. *(by Giovanni Di Maria)*

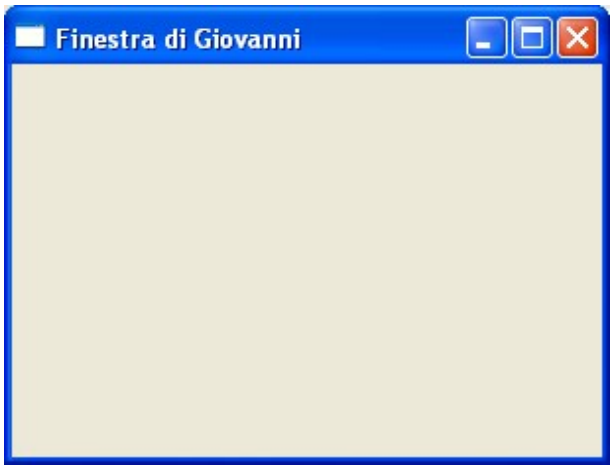


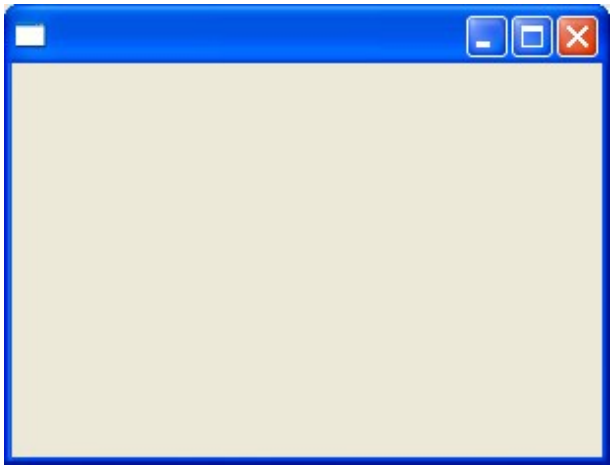

```
   // SOURCE=QMainWindow.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd

   oWnd := QMainWindow()
```

```
    oWnd:setWindowTitle( "Finestra di Giovanni" )
    oWnd:setFixedHeight( 300 )

    oWnd:show()
    QApplication():exec()

    RETURN

    // END SOURCE
```

7.5 Blinking Title Bar

The following example shows how to create a blinking title bar. *(by Giovanni Di Maria)*



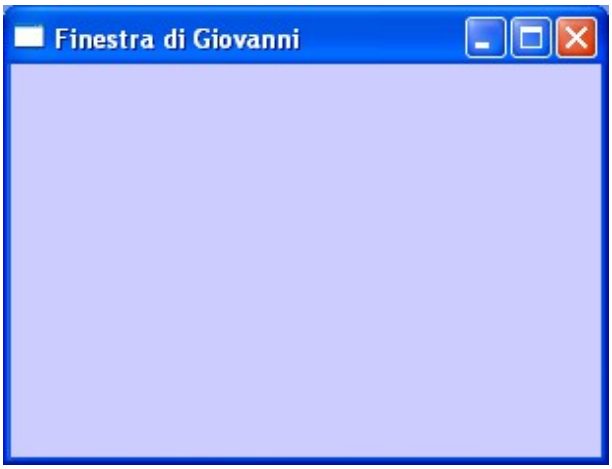


```
   // SOURCE=QMainWindow.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oClock
```

```
   oWnd := QMainWindow()
   oWnd:resize( 300, 200 )
   oWnd:setWindowTitle( "Finestra di Giovanni" )

   oClock := QTimer()
   oClock:Connect( "timeout()", { || toggle( oWnd ) } )
   oClock:start( 500 )

   oWnd:show()
   QApplication():exec()
   oClock:stop()

   RETURN

PROCEDURE toggle( o )

   IF o:WindowTitle = "Finestra di Giovanni"
      o:setWindowTitle( "" )
   ELSE
      o:setWindowTitle( "Finestra di Giovanni" )
   ENDIF

   RETURN

   // END SOURCE
```

7.6 Colored Window (QSS)

The following example shows how to create a colored main window, using QSS. *(by Giovanni Di Maria)*




```
   // SOURCE=QMainWindow.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   oWnd:setStyleSheet( " background-color: #CCCCFF; " )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

7.7 Colored Window (QPalette)

The following example shows how to create a colored main window, using Qpalette. *(by Giovanni Di Maria)*


```
  // SOURCE=QMainWindow.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oPalette

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   oPalette := QPalette()
   oPalette:SetColor( QPalette_Window, QColor(
      255,200,200 ) )
   oWnd:setPalette( oPalette )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

7.8 Window with background

The following example shows how to create a window with a background from an image. *(by Giovanni Di Maria)*


```
   // SOURCE=QMainWindow.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oButton

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 400, 300 )
   oWnd:setWindowTitle( "Finestra Giovanni" )
   oWnd:setStyleSheet( "background-image: url(../res/
       background-01.png) " );

   oButton := QPushButton( oWnd )
   oButton:setText( "Press" )
   oButton:move( 150, 50 )
   oButton:setStyleSheet( " background: #F4F4F0; " )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

8 QApplication

8.1 Beep

The following example shows how to create a simple main window, as application. At exit, the application produces a beep. This method sounds the bell, using the default volume and sound. The function is not available in Qt for Embedded Linux. *(by Giovanni Di Maria)*


```
   // SOURCE=QApplication.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 200, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   oWnd:show()
   QApplication():exec()

   QApplication():beep()
```

```
   RETURN

   // END SOURCE
```

8.2 Quitting from application

The following example shows how to quit from an application, using a method. *(by Giovanni Di Maria)*


```
   // SOURCE=QApplication.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oButton1, oWnd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Giovanni" )
   oWnd:resize( 300, 200 )

   oButton1 := QPushButton( oWnd )
   oButton1:setText( "Quit" )
   oButton1:move( 50, 50 )
   oButton1:Connect( "clicked()", { || QApplication():
       quit() } )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

9 QIcon

9.1 Widget with Icon

The following example shows how to add an icon to a widget. *(by Apolinar)*


```
   // SOURCE=QIcon.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni with icon"
        )
   oWnd:resize( 500, 200 )
   oWnd:setwindowicon( QIcon( "../res/harbour-icon.png"
       ) )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

10 QDateEdit

10.1 Editing a date

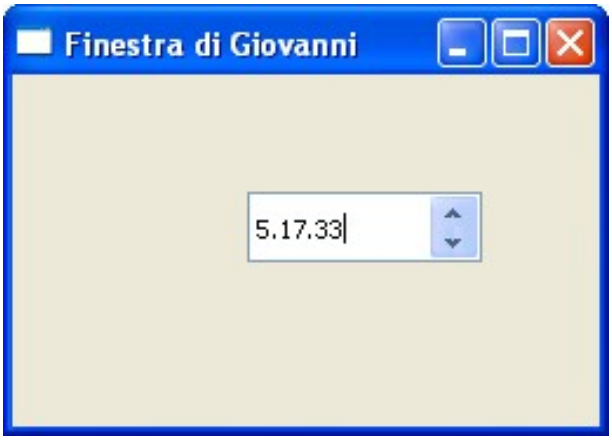
The following example shows how to enter and edit a date. *(by Giovanni Di Maria)*


```
   // SOURCE=date.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oDate

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 300, 200 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oDate := QDateEdit( oWnd )
   oDate:move( 60, 30 )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

11 QTimeEdit

11.1 Editing the time

The following example shows how to enter the time. *(by Giovanni Di Maria)*




```
   // SOURCE=QTimeEdit.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oSetting

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 250, 150 )

   oSetting := QTimeEdit( oWnd )
   oSetting:move( 100, 50 )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

12 QCalendarWidget

12.1 Calendar

The following example shows how to create a simple window containing a calendar. *(by Giovanni Di Maria)*


```
   // SOURCE=QCalendarWidget.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oCalendar

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 300 )

   oCalendar := QCalendarWidget( oWnd )
   oCalendar:resize( 250, 200 )
   oCalendar:move( 50, 50 )
   oCalendar:setFirstDayOfWeek( 1 )
   oCalendar:setGridVisible( .T. )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

12.2 Interactive Calendar

The following example shows how to create a simple window containing an interactive calendar. The buttons allow to navigate through the months and to show or hide the grid. *(by Giovanni Di Maria)*


```
   // SOURCE=QCalendarWidget.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oCal
   LOCAL oBPrev, oBNext, oBGrid

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 300 )

   oCal := QcalendarWidget( oWnd )
   oCal:resize( 300, 200 )
   oCal:move( 50, 50 )
   oCal:setFirstDayOfWeek( 1 )
   oCal:setGridVisible( .T. )

   oBPrev := QPushButton( oWnd )
   oBPrev:setText( "Prev Month" )
   oBPrev:move( 100, 5 )
   oBPrev:resize( 90, 25 )
```

```
   oBPrev:Connect( "clicked()", { || oCal:
      showPreviousMonth() } )

   oBNext := QPushButton( oWnd )
   oBNext:setText( "Next Month" )
   oBNext:move( 200, 5 )
   oBNext:resize( 90, 25 )
   oBNext:Connect( "clicked()", { || oCal:showNextMonth
      () } )

   oBGrid := QPushButton( oWnd )
   oBGrid:setText( "Show/Hide Grid" )
   oBGrid:move( 100, 260 )
   oBGrid:resize( 200, 25 )
   oBGrid:Connect( "clicked()", { || oCal:setGridVisible
      ( ! oCal:isGridVisible() ) } )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

13 QLabel

13.1 Text Label

The following example shows how to create a simple main window with a label, used as text string. *(by Giovanni Di Maria)*


```
   // SOURCE=QLabel.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oText

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   oText := Qlabel( oWnd )
   oText:setText( "Hello World" )
   oText:move( 100, 100 )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

13.2 Image

The following example shows how to view an image, using the Qlabel class. *(by Giovanni Di Maria)*


```
   // SOURCE=QSetPixmap.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oImg

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 400, 300 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oImg := QLabel( oWnd )
   oImg:move( 50, 50 )
   oImg:resize( 300, 200 )
   oImg:SetPixmap( QPixmap( "../res/sample.bmp" ) )
   oImg:setStyleSheet( "border: 2px solid #0000ff;" )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

13.3 Colored Text Label (QSS)

The following example shows how to create a simple main window with a label used as text string. The label is colored using QSS. *(by Giovanni Di Maria)*


```
   // SOURCE=QLabel.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oText

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   oText := Qlabel( oWnd )
   oText:setStyleSheet( "background-color : yellow;
       color : red;" )
   oText:setText( "Hello World" )
   oText:move( 100, 100 )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

13.4 Colored Text Label (QPalette)

The following example shows how to create a simple main window with a label used as text string. The label is colored using QPalette. *(by Giovanni Di Maria)*


```
   // SOURCE=QLabel.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oPalette
   LOCAL oText

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   oPalette := QPalette()
   oPalette:SetColor( QPalette_WindowText, QColor(
      255,0,0 ) )

   oText := Qlabel( oWnd )
   oText:setPalette( oPalette )
   oText:setText( "Hello World" )
   oText:move( 100, 100 )

   oWnd:show()
   QApplication():exec()
```

```
   RETURN

   // END SOURCE
```

13.5 Colored Text Label (HTML)

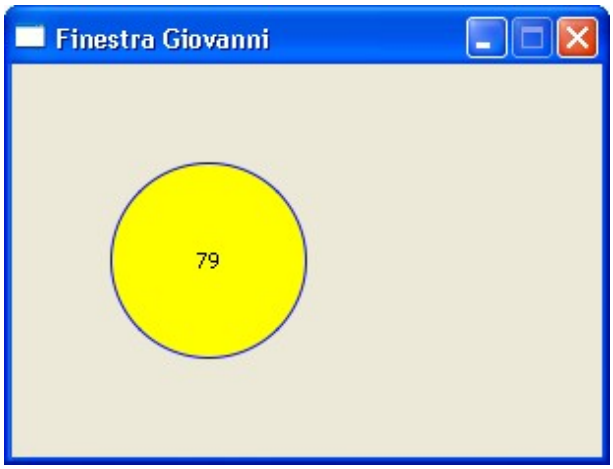
The following example shows how to create a simple main window with a label used as text string. The label is colored and formatted using HTML. *(by Giovanni Di Maria)*


```
   // SOURCE=QLabel.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oText
   LOCAL cString

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   cString = "Hello World <br><br>"
   cString = cString + "This is a complex text, colored
       and formatted "
   cString = cString + "by HTML. <br><br>"
   cString = cString + "<FONT color=#FF0000>RED</FONT><
       br>"
   cString = cString + "<FONT color=#008800>GREEN</FONT
       ><br>"
   cString = cString + "<FONT color=#0000FF>BLUE</FONT><
       br><br>"
   cString = cString + "<b>Bold Text</b> <br>"
```

```
   cString = cString + "<i>Italic Text</i> <br>"
   cString = cString + "<u>Underlined Text</u> <br"

   oText := Qlabel( oWnd )
   oText:setText( cString )
   oText:move( 20, 20 )
   oText:resize( 200, 150 )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

13.6 Circular Label

The following example shows how to create a circular label. The label is colored and formatted using QSS. *(by Giovanni Di Maria)*

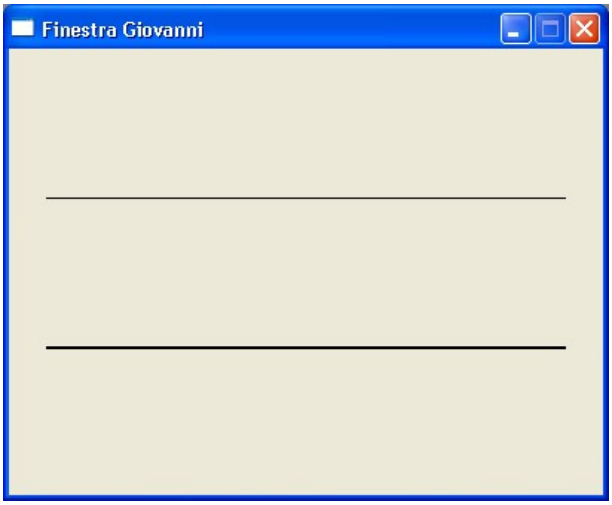



```
   // SOURCE=QLabel.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oLabel

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 300, 200 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oLabel := QLabel( oWnd )
   oLabel:move( 50, 50 )
   oLabel:resize( 100, 100 )
   oLabel:setText( "79" )
   oLabel:setStyleSheet( "border: 1px solid #0000FF;
       background-color: yellow; border-radius: 50px;" )
   oLabel:setAlignment( Qt_AlignHCenter +
       Qt_AlignVCenter )

   oWnd:show()
   QApplication():exec()

   RETURN
```

```
// END SOURCE
```

13.7 Horizontal line

The following example shows how to simulate a horizontal line. A good solution is to create a label with a thickness of 1 or 2 pixels. *(by Giovanni Di Maria)*

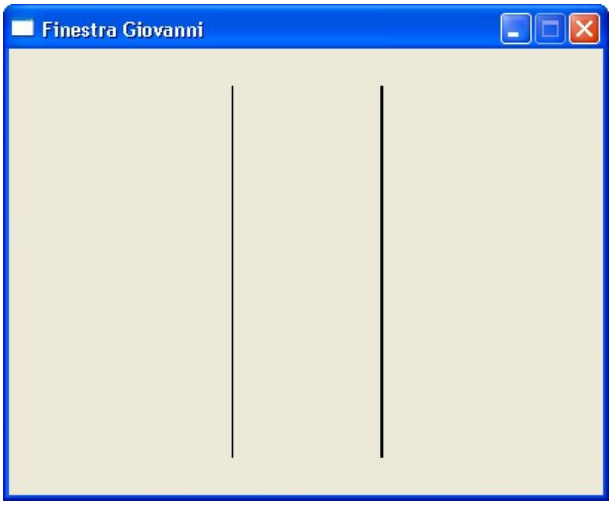



```
   // SOURCE=QLabel.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oLabel, oLabel2

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 400, 300 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oLabel := QLabel( oWnd )
   oLabel:move( 25, 100 )
   oLabel:resize( 350, 1 )
   oLabel:setStyleSheet( "background-color: black" )

   oLabel2 := QLabel( oWnd )
   oLabel2:move( 25, 200 )
   oLabel2:resize( 350, 2 )
   oLabel2:setStyleSheet( "background-color: black" )

   oWnd:show()
```

```
    QApplication():exec()

    RETURN

    // END SOURCE
```

13.8 Vertical line

The following example shows how to simulate a vertical line. A good solution is to create a label with a thickness of 1 or 2 pixels. *(by Giovanni Di Maria)*

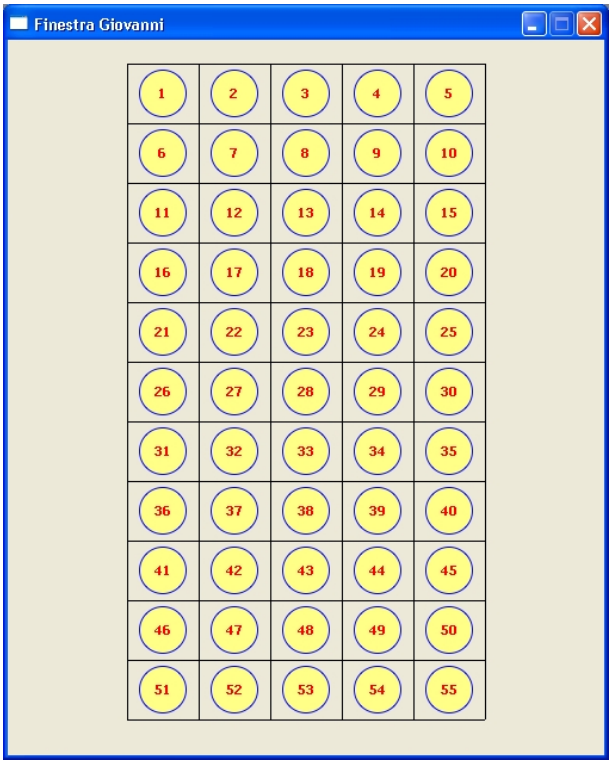



```
   // SOURCE=QLabel.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oLabel, oLabel2

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 400, 300 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oLabel := QLabel( oWnd )
   oLabel:move( 150, 25 )
   oLabel:resize( 1, 250 )
   oLabel:setStyleSheet( "background-color: black" )

   oLabel2 := QLabel( oWnd )
   oLabel2:move( 250, 25 )
   oLabel2:resize( 2, 250 )
   oLabel2:setStyleSheet( "background-color: black" )

   oWnd:show()
```

```
    QApplication():exec()

    RETURN

    // END SOURCE
```

13.9 Grid of lines

The following example shows how to simulate grid of lines. A good solution is to create two arrays of labels, with a thickness of 1 or 2 pixels. The first array is used for horizontal lines, the second array is used for vertical lines. *(by Giovanni Di Maria)*




```
   // SOURCE=QLabel.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oHorizontalLine[12], oVerticalLine[6]
   LOCAL oNumber[11,5]
   LOCAL nR, nC, nN

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 500, 600 )
   oWnd:setWindowTitle( "Finestra Giovanni" )
```

```
nN = 0
for nR = 1 TO 11
   for nC = 1 TO 5
      nN ++
      oNumber[nR,nC] := QLabel( oWnd )
      oNumber[nR,nC]:setStyleSheet( "border: 1px
          solid #0000FF; background-color: #FFFF88;
          border-radius: 20px; color:red;" )
      oNumber[nR,nC]:setText( "<b>" + AllTrim( Str(nN
           ) ) + "</b>" )
      oNumber[nR,nC]:move( nC * 60 + 50, nR * 50 - 25
           )
      oNumber[nR,nC]:resize( 40, 40 )
      oNumber[nR,nC]:setAlignment( Qt_AlignHCenter +
          Qt_AlignVCenter )
   next nC
next nR

for nR = 1 TO 12
   oHorizontalLine[nR] := QLabel( oWnd )
   oHorizontalLine[nR]:resize( 300, 1 )
   oHorizontalLine[nR]:move( 100, nR * 50 - 30 )
   oHorizontalLine[nR]:setStyleSheet( "background-
       color:#000000;" )
next nR


for nC = 1 TO 6
   oVerticalLine[nC] := QLabel( oWnd )
   oVerticalLine[nC]:resize( 1, 550 )
   oVerticalLine[nC]:move( nC * 60 + 40, 20 )
   oVerticalLine[nC]:setStyleSheet( "background-color
       :#000000;" )
next nC

oWnd:show()
QApplication():exec()

RETURN

// END SOURCE
```

14 QMessageBox

14.1 Message Box (simple)

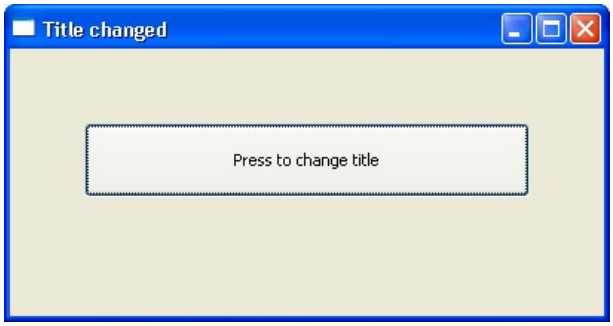
The following example shows how to create a simple window with an active button. If the button is pressed, a message box appears. *(by Giovanni Di Maria)*


```
   // SOURCE=QMessageBox.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oButton1

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 300 )

   oButton1 := QPushButton( oWnd )
   oButton1:setText( "Press for message" )
   oButton1:resize( 300, 50 )
   oButton1:move( 50, 50 )
   oButton1:Connect( "clicked()", { || message() } )

   oWnd:show()
   QApplication():exec()

   RETURN
```

```
PROCEDURE message()

   LOCAL oBox

   oBox := QMessageBox()
   oBox:setInformativeText( "attention!!! " )
   oBox:setWindowTitle( "Informazione" )

   oBox:exec()

   RETURN

   // END SOURCE
```

14.2 Message Box (Yes and No buttons)

The following example shows how to create a simple window with the Yes and No button. If the Yes button is pressed, the window title changes. *(by Giovanni Di Maria)*




```
   // SOURCE=QMessageBox.prg

#include "hbqtgui.ch"

   STATIC oWnd

PROCEDURE Main()

   LOCAL oButton1

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 180 )
```

```
   oButton1 := QPushButton( oWnd )
   oButton1:setText( "Press to change title" )
   oButton1:resize( 300, 50 )
   oButton1:move( 50, 50 )
   oButton1:Connect( "clicked()", { || MsgYesNo() } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE MsgYesNo()

   LOCAL oMB, nButtonPressed

   oMB := QMessageBox()
   oMB:setInformativeText( "Ok to change the Window
      Title ?" )
   oMB:setWindowTitle( "Information" )
   oMB:setWindowFlags( Qt_Dialog )
   oMB:setStandardButtons( QMessageBox_Yes +
      QMessageBox_No )
   oMB:setDefaultButton( QMessageBox_Yes )

   nButtonPressed := oMB:exec()

   IF nButtonPressed = QMessageBox_Yes
      oWnd:setWindowTitle( "Title changed" )
   ENDIF

   RETURN

   // END SOURCE
```

15 QPushButton

15.1 Simple button

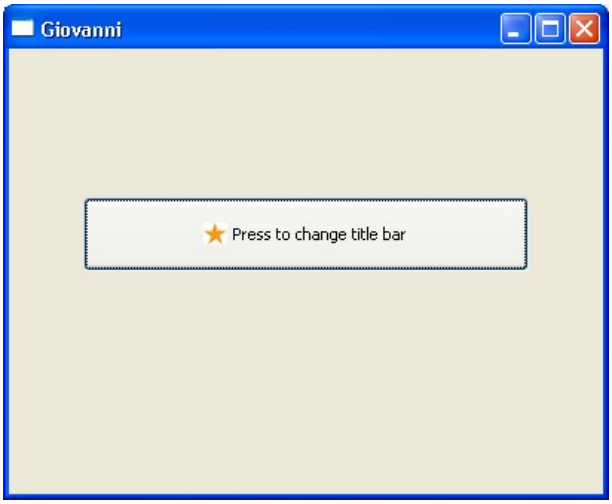
The following example shows how to create a simple button. If pressed, the window will be resized. *(by Giovanni Di Maria)*


```
   // SOURCE=QPushButton.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oButton1, oWnd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Giovanni" )
   oWnd:resize( 400, 300 )

   oButton1 := QPushButton( oWnd )
   oButton1:setText( "Resize Window" )
   oButton1:move( 50, 50 )
   oButton1:Connect( "clicked()", { || edit( oWnd ) } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE edit( oWnd )

   oWnd:resize( 500, 400 )
```

```
   RETURN

   // END SOURCE
```

15.2 Button with icon

The following example shows how to create a simple button with a icon. If pressed, the title bar changes. *(by Giovanni Di Maria)*

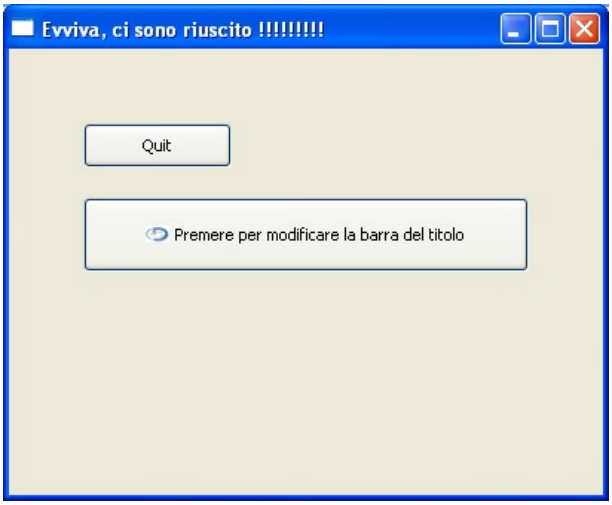



```
   // SOURCE=QPushButton.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oButton2, oWnd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Giovanni" )
   oWnd:resize( 400, 300 )

   oButton2 := QPushButton( oWnd )
   oButton2:setText( "Press to change title bar" )
   oButton2:move( 50, 100 )
   oButton2:setIcon( QIcon( "star.bmp" ) )
   oButton2:resize( 300, 50 )
   oButton2:Connect( "clicked()", { || edit( oWnd ) } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE edit( oWnd )
```

```
   oWnd:setWindowTitle( "Ok, changed" )

   RETURN

   // END SOURCE
```

15.3 Buttons

The following example shows how to create a simple window with two active buttons. If the first button is pressed, the window closes. If the second button is pressed, the title of the window changes his value. *(by Giovanni Di Maria)*

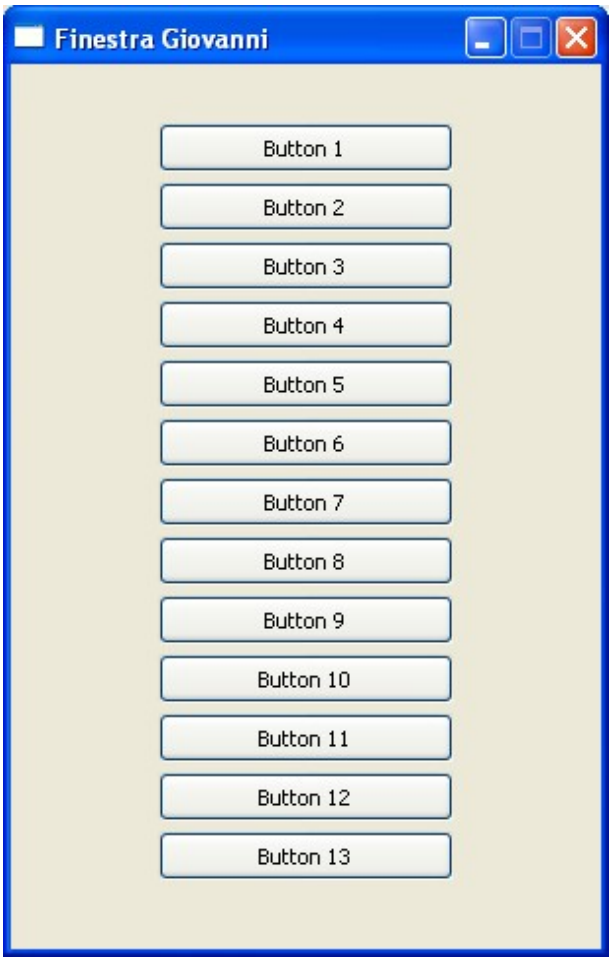



```
   // SOURCE=QPushButton.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oButton1, oButton2, oWnd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Prova dei pulsanti" )
   oWnd:resize( 640, 480 )

   oButton1 := QPushButton( oWnd )
   oButton1:setText( "Quit" )
   oButton1:move( 50, 50 )
   oButton1:Connect( "clicked()", { || QApplication():
      quit() } )

   oButton2 := QPushButton( oWnd )
   oButton2:setText( "Premere per modificare la barra
      del titolo" )
   oButton2:move( 50, 100 )
   oButton2:setIcon( QIcon( "../res/star_32.bmp" ) )
   oButton2:resize( 300, 50 )
```

```
   oButton2:Connect( "clicked()", { || edit( oWnd ) } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE edit( oWnd )

   oWnd:setWindowTitle( "Evviva, ci sono riuscito
      !!!!!!!!!" )

   RETURN

   // END SOURCE
```

15.4 Array of Buttons

The following example shows how to create many buttons, stored in an array. *(by Giovanni Di Maria)*

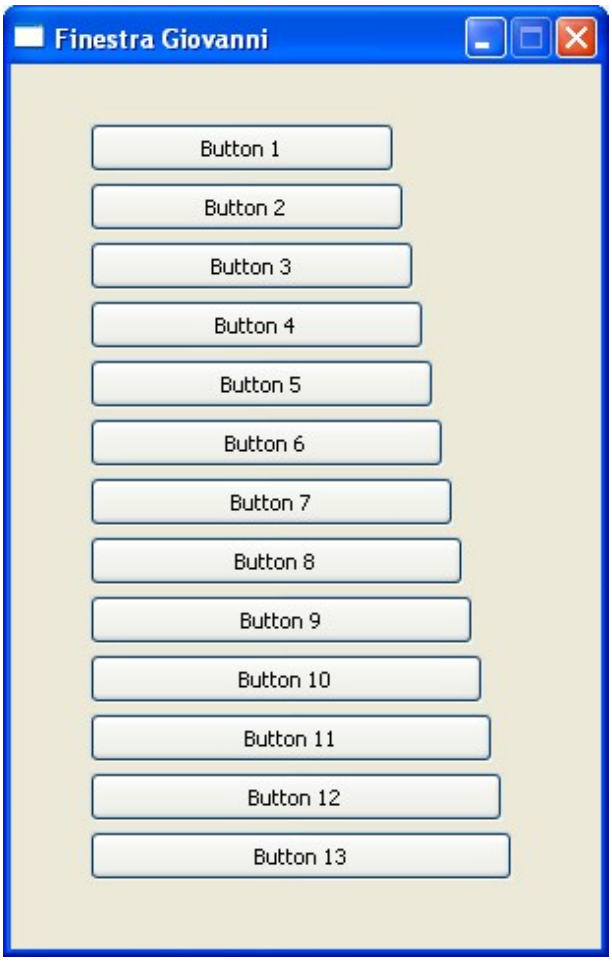



```
   // SOURCE=QPushButton.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oButton[13]
   LOCAL k
```

```
oWnd := QmainWindow()
oWnd:SetFixedSize( 300, 450 )
oWnd:setWindowTitle( "Finestra Giovanni" )


for k = 1 TO 13
   oButton[k] := QPushButton( oWnd )
   oButton[k]:resize( 150, 25 )
   oButton[k]:move( 75, 30 * k )
   oButton[k]:setText( "Button " + AllTrim( Str(k ) )
       )
next k

oWnd:show()
QApplication():exec()

RETURN

// END SOURCE
```

15.5 Array of Buttons with variable dimensions

The following example shows how to create many buttons, stored in an array. Their dimension is set by an incremental variable. *(by Giovanni Di Maria)*

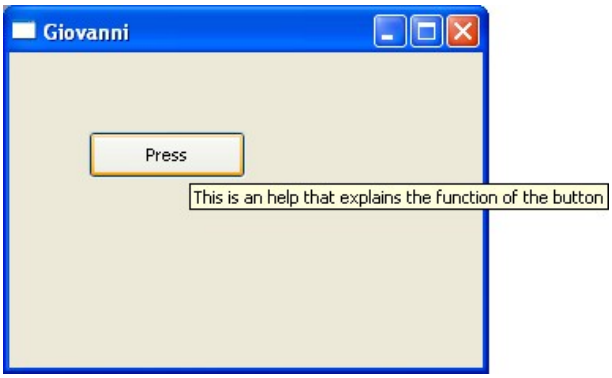



```
   // SOURCE=QPushButton.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oButton[13]
   LOCAL k
```

```
oWnd := QmainWindow()
oWnd:SetFixedSize( 300, 450 )
oWnd:setWindowTitle( "Finestra Giovanni" )


for k = 1 TO 13
   oButton[k] := QPushButton( oWnd )
   oButton[k]:resize( 150 + ( k * 5 ), 25 )
   oButton[k]:move( 40, 30 * k )
   oButton[k]:setText( "Button " + AllTrim( Str(k ) )
       )
next k

oWnd:show()
QApplication():exec()

RETURN

// END SOURCE
```

15.6 Button Tips

The following example shows how to create a button with the tip. *(by Giovanni Di Maria)*

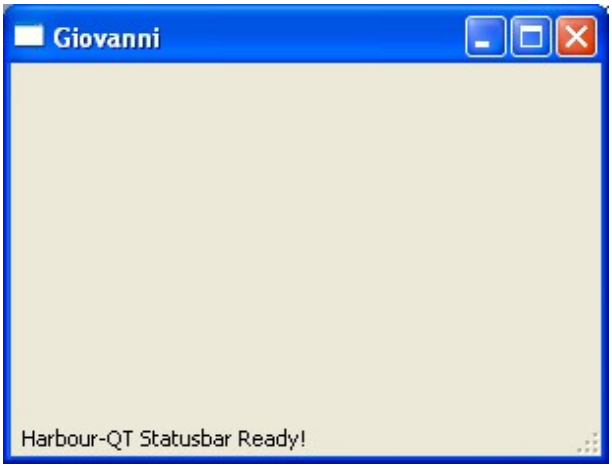



```
   // SOURCE=QPushButton.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oButton1

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Giovanni" )
   oWnd:resize( 300, 200 )

   oButton1 := QPushButton( oWnd )
   oButton1:setText( "Press" )
   oButton1:move( 50, 50 )
   oButton1:setToolTip( "This is an help that explains
      the function of the button" )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

16 QStatusBar

16.1 Status Bar

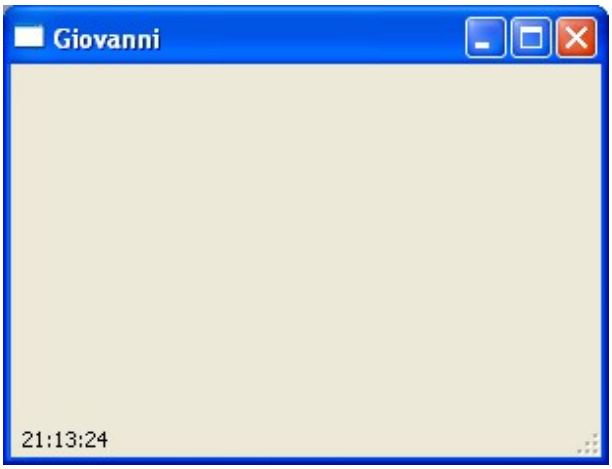
The following example shows how to create and modify the status bar, at the bottom of the window. *(by Giovanni Di Maria)*




```
   // SOURCE=QStatusBar.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oSBar

   oWnd := QMainWindow()
   oWnd:show()

   oWnd:setWindowTitle( "Giovanni" )
   oWnd:resize( 300, 200 )

   oSBar  := QStatusBar( oWnd )
   oWnd:setStatusBar( oSBar )

   oSBar:showMessage( "Harbour-QT Statusbar Ready!" )

   QApplication():exec()

   RETURN

   // END SOURCE
```

16.2 Status Bar and time clock

The following example shows how to show a clock on the status bar, at the bottom of the window. *(by Giovanni Di Maria)*




```
   // SOURCE=QStatusBar.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oSBar
   LOCAL oClock

   oWnd := QMainWindow()

   oWnd:setWindowTitle( "Giovanni" )
   oWnd:resize( 300, 200 )

   oClock := QTimer()
   oClock:Connect( "timeout()", { || oSBar:showMessage(
      Time() ) } )
   oClock:start( 1000 )

   oSBar  := QStatusBar( oWnd )
   oWnd:setStatusBar( oSBar )

   oWnd:show()
   QApplication():exec()
   oClock:stop()
```

```
   RETURN

   // END SOURCE
```

16.3 Status Bar Ping Pong

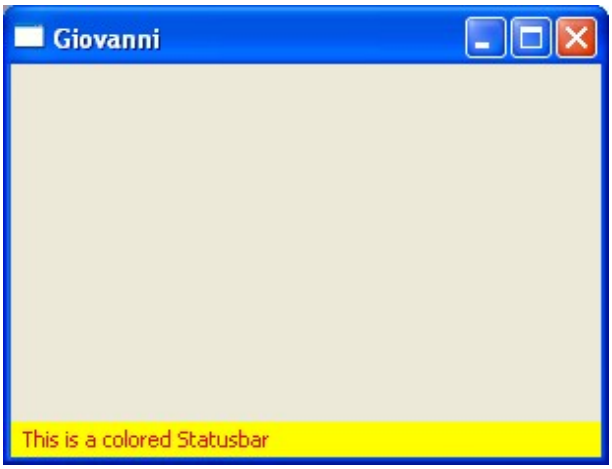
The following example shows how create a status bar with a ping pong effect. The status bar goes to left and right. *(by Giovanni Di Maria)*


```
   // SOURCE=QStatusBar.prg

#include "hbqtgui.ch"

   STATIC nSpaces := 1
   STATIC nInc := 1
   STATIC oSBar

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oClock

   oWnd := QMainWindow()

   oWnd:setWindowTitle( "Giovanni" )
   oWnd:resize( 300, 200 )

   oClock := QTimer()
   oClock:Connect( "timeout()", { || pingpong() } )
   oClock:start( 25 )

   oSBar  := QStatusBar( oWnd )
   oSBar:move( 100, 1 )

   oWnd:setStatusBar( oSBar )
```

```
   oWnd:show()
   QApplication():exec()
   oClock:stop()

   RETURN



PROCEDURE pingpong()

   LOCAL cString

   cString = Space( nSpaces ) + "Hello"
   oSBar:showMessage( cString )
   nSpaces += nInc

   IF nSpaces = 40
      nInc =- 1
   ENDIF

   IF nSpaces = 1
      nInc = 1
   ENDIF

   RETURN

   // END SOURCE
```

16.4 Colored Status Bar

The following example shows how to create and modify a colored status bar, at the bottom of the window. *(by Giovanni Di Maria)*




```
   // SOURCE=QStatusBar.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oSBar

   oWnd := QMainWindow()

   oWnd:setWindowTitle( "Giovanni" )
   oWnd:resize( 300, 200 )

   oSBar  := QStatusBar( oWnd )
   oWnd:setStatusBar( oSBar )

   oSBar:showMessage( "This is a colored Statusbar" )
   oSBar:setStyleSheet( "background-color : yellow;
       color : red;" )

   oWnd:show()
   QApplication():exec()

   RETURN
```

```
   // END SOURCE
```

17 QCursor

17.1 Cursor Managing

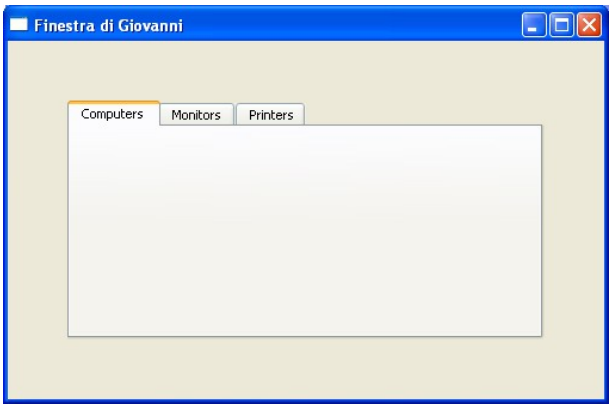
The following example shows how to modify the cursor of the mouse, over widgets. *(by Giovanni Di Maria)*


```
   // SOURCE=QCursor.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oCursor

   oCursor := QCursor()
   oCursor:setShape( Qt_WaitCursor )

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   oWnd:SetCursor( oCursor )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

18 QTabWidget

18.1 TAB Control

The following example shows how to create three Tab Widget. *(by Giovanni Di Maria)*




```
   // SOURCE=QTabWidget.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oBar
   LOCAL oComputer, oMonitor, oPrinter

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 500, 300 )

   oComputer   := QWidget()
   oMonitor    := QWidget()
   oPrinter    := QWidget()

   oBar := QTabWidget( oWnd )
   oBar:resize( 400, 200 )
   oBar:move( 50, 50 )
   oBar:addTab( oComputer, "Computers" )
   oBar:addTab( oMonitor, "Monitors" )
   oBar:addTab( oPrinter, "Printers" )

   oWnd:show()
   QApplication():exec()
```

```
   RETURN

   // END SOURCE
```

18.2 TAB Controls in a TAB Control

The following example shows how to create many Tab Widgets in three Tab Widget. *(by Giovanni Di Maria)*


```
   // SOURCE=QTabWidget.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
```

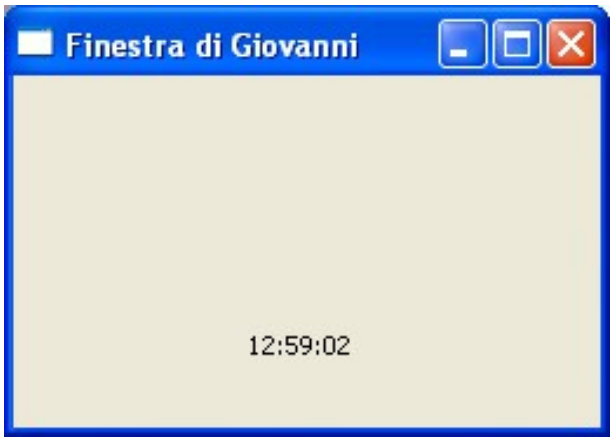
```
LOCAL oBar, oBar2, oBar3, oBar4
LOCAL oComputer, oMonitor, oPrinter
LOCAL oAmd, oIntel
LOCAL oCrt, oLcd
LOCAL oLaser, oInk, oDotmatrix, oThermal

oWnd := QMainWindow()
oWnd:setWindowTitle( "Finestra di Giovanni" )
oWnd:resize( 500, 300 )

oComputer   := QWidget()
oMonitor    := QWidget()
oPrinter    := QWidget()

oBar := QTabWidget( oWnd )
oBar:resize( 400, 200 )
oBar:move( 50, 50 )
oBar:addTab( oComputer, "Computers" )
oBar:addTab( oMonitor, "Monitors" )
oBar:addTab( oPrinter, "Printers" )

oAmd    := QWidget()
oIntel  := QWidget()
oBar2 := QTabWidget( oComputer )
oBar2:resize( 200, 100 )
oBar2:move( 50, 50 )
oBar2:addTab( oAmd, "Cpu Amd" )
oBar2:addTab( oIntel, "Cpu Intel" )

oCrt    := QWidget()
oLcd    := QWidget()
oBar3 := QTabWidget( oMonitor )
oBar3:resize( 200, 100 )
oBar3:move( 50, 50 )
oBar3:addTab( oCrt, "Monitor CRT" )
oBar3:addTab( oLcd, "Monitor LCD" )

oLaser     := QWidget()
oInk       := QWidget()
oDotmatrix := QWidget()
oThermal   := QWidget()
oBar4 := QTabWidget( oPrinter )
oBar4:resize( 380, 100 )
oBar4:move( 10, 50 )
oBar4:addTab( oLaser, "Laser Printer" )
oBar4:addTab( oInk, "Inkjet Printer" )
oBar4:addTab( oDotmatrix, "Dot-matrix Printer" )
oBar4:addTab( oThermal, "Thermal Printer" )

oWnd:show()
QApplication():exec()

RETURN
```

```
    // END SOURCE
```

19 QTimer

19.1 Timer

The following example shows a clock every 1 second, thank to Qtimer class. The time of refresh can be adjusted by the start(x) property. *(by Giovanni Di Maria)*




```
   // SOURCE=QTimer.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oClock
   LOCAL oText

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 250, 150 )

   oText := Qlabel( oWnd )
   oText:setText( "clocking..." )
   oText:move( 100, 100 )

   oClock := QTimer()
   oClock:Connect( "timeout()", { || print_clock( oText
      ) } )
   oClock:start( 1000 )

   oWnd:show()
   QApplication():exec()
   oClock:stop()
```

```
   RETURN

PROCEDURE print_clock( oT )

   oT:setText( Time() )

   RETURN

   // END SOURCE
```

19.2 Timer with controls

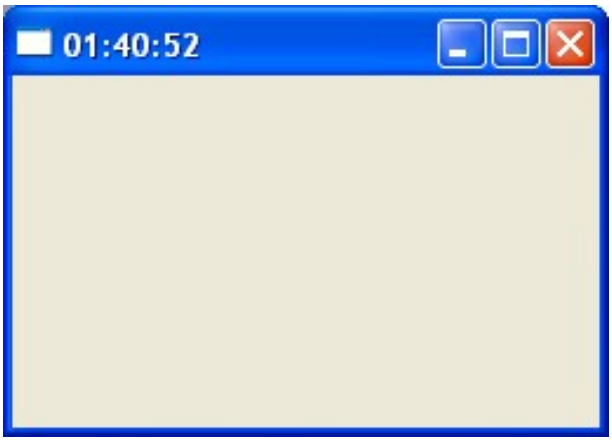
The following example shows a clock every a second, thank to Qtimer class. The time of refresh can be adjusted by the start(x) property. If the Start button is pressed, the time is updated every a second. If the Stop button is pressed, the time stops. *(by Giovanni Di Maria)*


```
   // SOURCE=QTimer.prg

#include "hbqtgui.ch"

   STATIC oText
   STATIC oClock

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oButtonStart, oButtonStop

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   oText := Qlabel( oWnd )
   oText:setText( "clocking..." )
   oText:move( 50, 50 )
   oText:resize( 200, 100 )

   oClock := QTimer()
   oClock:Connect( "timeout()", { || print_clock() } )
```

```
   oButtonStart := QPushButton( oWnd )
   oButtonStart:move( 150, 50 )
   oButtonStart:setText( "Start" )
   oButtonStart:connect( "pressed()", { || start() } )

   oButtonStop := QPushButton( oWnd )
   oButtonStop:move( 150, 100 )
   oButtonStop:setText( "Stop" )
   oButtonStop:connect( "pressed()", { || stop() } )

   oWnd:show()
   QApplication():exec()
   oClock:stop()

   RETURN



PROCEDURE print_clock()

   oText:setText( Time()  )

   RETURN



PROCEDURE start()

   oClock:start( 1000 )

   RETURN



PROCEDURE stop()

   oClock:stop()

   RETURN

   // END SOURCE
```

19.3 Timer in the Window Title

The following example shows a clock on the Title bar of a window. *(by Giovanni Di Maria)*

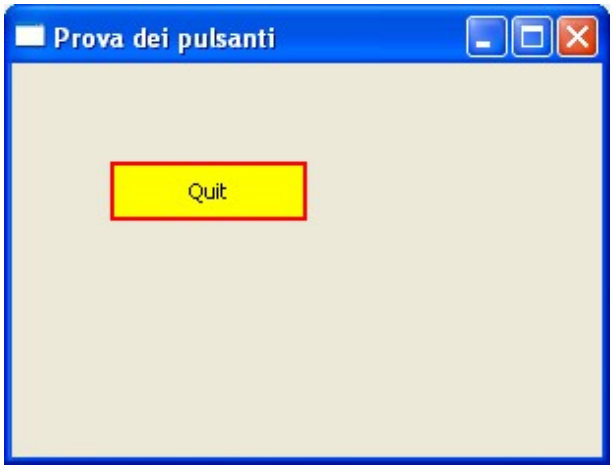



```
   // SOURCE=QTimer.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oClock

   oWnd := QMainWindow()
   oWnd:resize( 250, 150 )

   oClock := QTimer()
   oClock:Connect( "timeout()", { || oWnd:setWindowTitle
       ( Time() ) } )
   oClock:start( 1000 )

   oWnd:show()
   QApplication():exec()
   oClock:stop()

   RETURN

   // END SOURCE
```

20 QSS

20.1 QSS Style Sheet

The following example shows how we can use CSS to change any property of objects. *(by Giovanni Di Maria)*




```
  // SOURCE=setStyleSheet.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oButton1

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Prova dei pulsanti" )
   oWnd:resize( 300, 200 )

   oButton1 := QPushButton( oWnd )
   oButton1:setText( "Quit" )
   oButton1:move( 50, 50 )
   oButton1:Connect( "clicked()", { || QApplication():
      quit() } )
   oButton1:setStyleSheet( "background-color: yellow;
      border: 2px solid #FF0000;" )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

20.2 Gradient (linear: top to bottom)

The following example shows how we can use CSS to change any property of objects. In particular it shows the creation of a linear gradient. *(by Giovanni Di Maria)*


```
   // SOURCE=setStyleSheet.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oLabel

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 300, 200 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oLabel := QLabel( oWnd )
   oLabel:setText( "Gradient" )
   oLabel:move( 50, 50 )
   oLabel:resize( 100, 100 )
   oLabel:setStyleSheet( "background-color:
       qlineargradient(x1:0, y1:0, x2:0, y2:1,stop:0 #
       FF0000, stop:1 #FFFF00);" )

   oWnd:show()
   QApplication():exec()

   RETURN
```

```
   // END SOURCE
```

20.3 Gradient (linear: left to right)

The following example shows how we can use CSS to change any property of objects. In particular it shows the creation of a linear gradient. *(by Giovanni Di Maria)*


```
   // SOURCE=setStyleSheet.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oLabel

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 300, 200 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oLabel := QLabel( oWnd )
   oLabel:setText( "Gradient" )
   oLabel:move( 50, 50 )
   oLabel:resize( 100, 100 )
   oLabel:setStyleSheet( "background-color:
       qlineargradient(x1:0, y1:0, x2:1, y2:0,stop:0
       #0000FF, stop:1 #FFFFFF);" )

   oWnd:show()
   QApplication():exec()

   RETURN
```

```
   // END SOURCE
```

20.4 Gradient (linear: left-top to right-bottom)

The following example shows how we can use CSS to change any property of objects. In particular it shows the creation of a linear gradient. *(by Giovanni Di Maria)*


```
   // SOURCE=setStyleSheet.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oLabel

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 300, 200 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oLabel := QLabel( oWnd )
   oLabel:setText( "Gradient" )
   oLabel:move( 50, 50 )
   oLabel:resize( 100, 100 )
   oLabel:setStyleSheet( "background-color:
       qlineargradient(x1:0, y1:0, x2:1, y2:1,stop:0 #00
       FF00, stop:1 #004400);" )

   oWnd:show()
   QApplication():exec()

   RETURN
```

```
    // END SOURCE
```

20.5 Gradient (radial)

The following example shows how we can use CSS to change any property of objects. In particular it shows the creation of a linear gradient. *(by Giovanni Di Maria)*

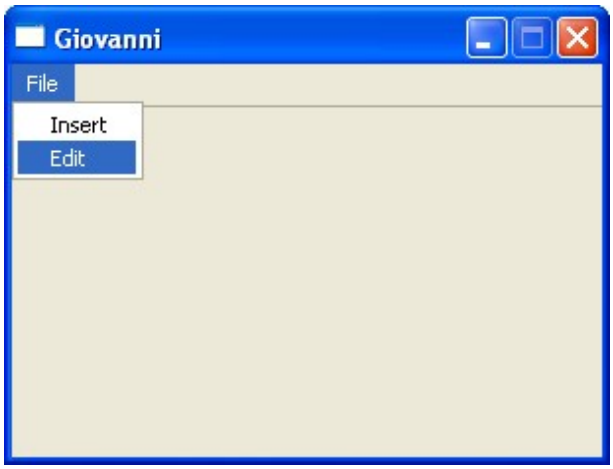

```
   // SOURCE=setStyleSheet.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oLabel

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 300, 200 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oLabel := QPushButton( oWnd )
   oLabel:setText( "Gradient" )
   oLabel:move( 50, 50 )
   oLabel:resize( 100, 100 )
   oLabel:setStyleSheet( "color: #333; border: 2px solid
        #555; border-radius: 11px; padding: 5px;
       background: qradialgradient(cx: 0.3, cy: -0.4, fx:
        0.3, fy: -0.4, radius: 1.35, stop: 0 #FFFFFF,
       stop: 1 #AAAAFF); min-width: 80px;" )

   oWnd:show()
   QApplication():exec()
```

```
   RETURN

   // END SOURCE
```

21 QMenu

21.1 Menu

The following example shows the usage of menu. Every item can be connected to any function or UDF. *(by Giovanni Di Maria)*

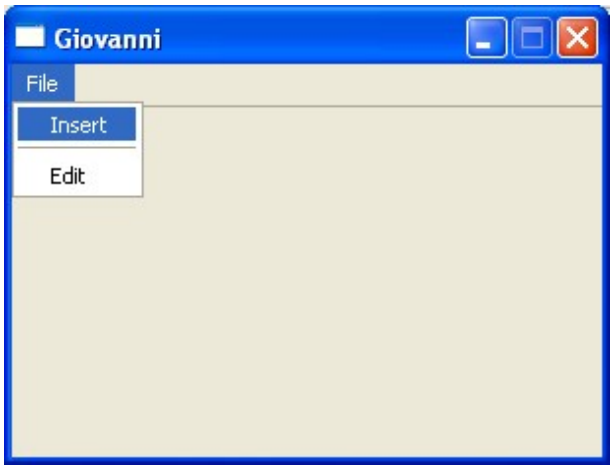



```
   // SOURCE=QMenu.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oMenuBar, oMenu1, oItemIns, oItemMod

   // ----------Window-----------
   oWnd := QmainWindow()
   oWnd:SetFixedSize( 300, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   // ----------Menu-----------
   oMenuBar := QMenuBar( oWnd )
   oMenuBar:resize( 700, 22 )

   oMenu1 := QMenu()
   oMenu1:setTitle( "File" )

   oItemIns := QAction( oMenu1 )
   oItemIns:setText( "Insert" )
   oItemIns:connect( "triggered(bool)", { || insert() }
      )
```

```
   oItemMod := QAction( oMenu1 )
   oItemMod:setText( "Edit" )
   oItemMod:connect( "triggered(bool)", { || edit() } )

   oMenu1:addAction( oItemIns )
   oMenu1:addAction( oItemMod )

   oMenuBar:addMenu( oMenu1 )

   oWnd:Show()
   QApplication():exec()

   RETURN



PROCEDURE insert()

   STATIC oWnd2

   oWnd2 := QmainWindow()
   oWnd2:SetFixedSize( 640, 480 )
   oWnd2:setWindowTitle( "Insert" )
   oWnd2:Show()

   RETURN



PROCEDURE edit()

   STATIC oWnd2

   oWnd2 := QmainWindow()
   oWnd2:SetFixedSize( 640, 480 )
   oWnd2:setWindowTitle( "Edit" )
   oWnd2:Show()

   RETURN

   // END SOURCE
```

21.2 Menu with separators

The following example shows the usage of menu. An item can separated from the other by a line separator. Every item can be connected to any function or UDF. *(by Giovanni Di Maria)*

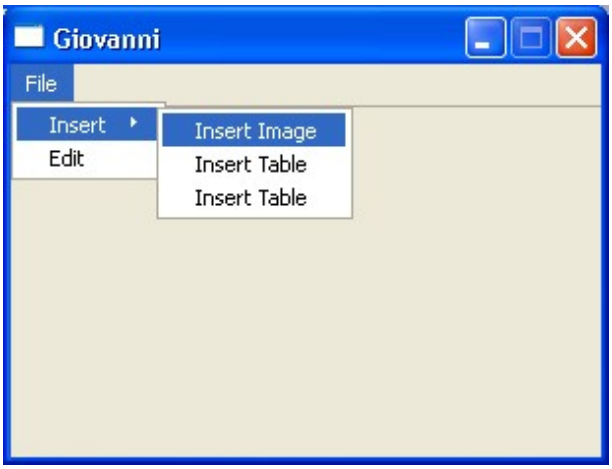



```
   // SOURCE=QMenu.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oMenuBar, oMenu1, oItemIns, oItemMod

   // ----------Window-----------
   oWnd := QmainWindow()
   oWnd:SetFixedSize( 300, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   // ----------Menu-----------
   oMenuBar := QMenuBar( oWnd )
   oMenuBar:resize( 700, 22 )

   oMenu1 := QMenu()
   oMenu1:setTitle( "File" )

   oItemIns := QAction( oMenu1 )
   oItemIns:setText( "Insert" )
   oItemIns:connect( "triggered(bool)", { || insert() }
      )
```

```
   oItemMod := QAction( oMenu1 )
   oItemMod:setText( "Edit" )
   oItemMod:connect( "triggered(bool)", { || edit() } )

   oMenu1:addAction( oItemIns )
   oMenu1:addSeparator()
   oMenu1:addAction( oItemMod )

   oMenuBar:addMenu( oMenu1 )

   oWnd:Show()
   QApplication():exec()

   RETURN



PROCEDURE insert()

   STATIC oWnd2

   oWnd2 := QmainWindow()
   oWnd2:SetFixedSize( 640, 480 )
   oWnd2:setWindowTitle( "Insert" )

   oWnd2:Show()

   RETURN



PROCEDURE edit()

   STATIC oWnd2

   oWnd2 := QmainWindow()
   oWnd2:SetFixedSize( 640, 480 )
   oWnd2:setWindowTitle( "Edit" )

   oWnd2:Show()

   RETURN

   // END SOURCE
```

21.3 Menu and sub-menu

The following example shows the usage of menu and sub-menu. The sub-menu is also a menu. *(by Giovanni Di Maria)*



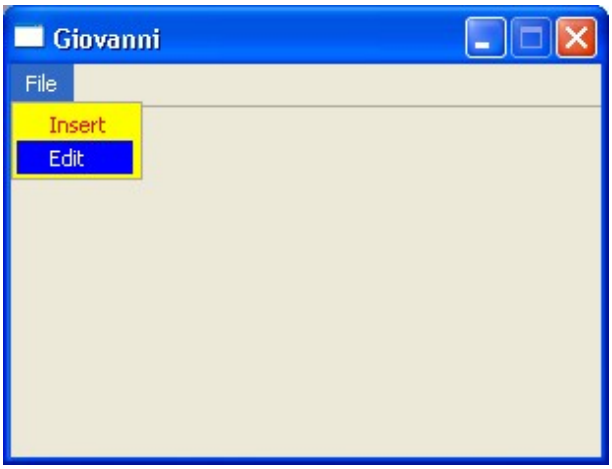

```
   // SOURCE=QMenu.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oMenuBar, oMenu1, oMenu2
   LOCAL oItemMod
   LOCAL oItemInsImage, oItemInsTable, oItemInsPage

   // ----------Window-----------
   oWnd := QmainWindow()
   oWnd:SetFixedSize( 300, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   // ----------Menu-----------
   oMenuBar := QMenuBar( oWnd )
   oMenuBar:resize( 700, 22 )

   oMenu1 := QMenu()
   oMenu1:setTitle( "File" )

   oMenu2 := QMenu()
   oMenu2:setTitle( "Insert" )
```

```
   oItemInsImage := QAction( oMenu2 )
   oItemInsImage:setText( "Insert Image" )
   oItemInsImage:connect( "triggered(bool)", { || image
      () } )
   oMenu2:addAction( oItemInsImage )

   oItemInsTable := QAction( oMenu2 )
   oItemInsTable:setText( "Insert Table" )
   oItemInsTable:connect( "triggered(bool)", { || table
      () } )
   oMenu2:addAction( oItemInsTable )

   oItemInsPage := QAction( oMenu2 )
   oItemInsPage:setText( "Insert Table" )
   oItemInsPage:connect( "triggered(bool)", { || page()
      } )
   oMenu2:addAction( oItemInsPage )

   oMenu1:addMenu( oMenu2 )

   oItemMod := QAction( oMenu1 )
   oItemMod:setText( "Edit" )
   oItemMod:connect( "triggered(bool)", { || edit() } )

   oMenu1:addAction( oItemMod )

   oMenuBar:addMenu( oMenu1 )

   oWnd:Show()
   QApplication():exec()

   RETURN

PROCEDURE edit()

   // Type your code here

   RETURN

PROCEDURE image()

   // Type your code here

   RETURN

PROCEDURE table()

   // Type your code here

   RETURN

PROCEDURE page()

   // Type your code here
```

```
   RETURN

   // END SOURCE
```

21.4 Colored Menu

The following example shows the usage of a colored menu. Every item can be connected to any function or UDF. *(by Giovanni Di Maria)*

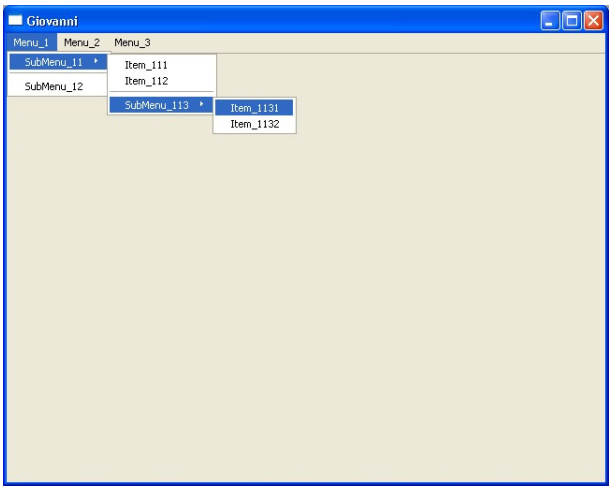



```
   // SOURCE=QMenu.prg

#include "hbqtgui.ch"
#define cSTYLE "background-color : yellow; color : red;
    selection-color: white; selection-background-color:
    blue;"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oMenuBar, oMenu1, oItemIns, oItemMod

   // ----------Window-----------
   oWnd := QmainWindow()
   oWnd:SetFixedSize( 300, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   // ----------Menu-----------
   oMenuBar := QMenuBar( oWnd )
   oMenuBar:resize( 700, 22 )


   oMenu1 := QMenu()
   oMenu1:setTitle( "File" )
   oMenu1:setStyleSheet( cSTYLE )
```

```
   oItemIns := QAction( oMenu1 )
   oItemIns:setText( "Insert" )

   oItemIns:connect( "triggered(bool)", { || insert() }
      )

   oItemMod := QAction( oMenu1 )
   oItemMod:setText( "Edit" )
   oItemMod:connect( "triggered(bool)", { || edit() } )

   oMenu1:addAction( oItemIns )
   oMenu1:addAction( oItemMod )

   oMenuBar:addMenu( oMenu1 )

   oWnd:Show()
   QApplication():exec()

   RETURN

PROCEDURE insert()

   // Type your code here

   RETURN

PROCEDURE edit()

   // Type your code here

   RETURN

   // END SOURCE
```

21.5 A complex Menu

The following example shows how to create a complex menu. *(by Mario Wan Stadnik (Qatan))*




```
   // SOURCE=QMenu.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oMenuBar
   LOCAL oMenu1, oMenu2, oMenu3
   LOCAL oSubMenu11, oSubMenu12, oSubMenu21, oSubMenu31
   LOCAL oSubMenu113
   LOCAL oItem111, oItem112
   LOCAL oItem1131, oItem1132
   LOCAL oItem211, oItem212

   // ---------- Window -----------
   oWnd := QmainWindow()
   oWnd:reSize( 640, 480 )
   oWnd:setWindowTitle( "Giovanni" )

   // ---------- Menu -----------
   oMenuBar := QMenuBar( oWnd )
   oMenuBar:resize( 3000, 22 )

   // ---------- Menu 1 ----------
```

```
    oMenu1 := QMenu()
    oMenu1:setTitle( "Menu_1" )
    // ---------- SubMenu 11 ----------
    oSubMenu11 := QMenu()
    oSubMenu11:setTitle( "SubMenu_11" )
    // --------- Items ---------
    oItem111 := QAction( oSubMenu11 )
    oItem111:setText( "Item_111" )
    oItem111:connect( "triggered(bool)", { || Item_111()
       } )
    oSubMenu11:addAction( oItem111 )
    oItem112 := QAction( oSubMenu11 )
    oItem112:setText( "Item_112" )
    oItem112:connect( "triggered(bool)", { || Item_112()
       } )
    oSubMenu11:addAction( oItem112 )
    oSubMenu11:addSeparator() //====================
    // ---------- SubMenu 113 ----------
    oSubMenu113 := QMenu()
    oSubMenu113:setTitle( "SubMenu_113" )
    // --------- Items ---------
    oItem1131 := QAction( oSubMenu113 )
    oItem1131:setText( "Item_1131" )
    oItem1131:connect( "triggered(bool)", { ||Item_1131()
        } )
    oSubMenu113:addAction( oItem1131 )
    oItem1132 := QAction( oSubMenu113 )
    oItem1132:setText( "Item_1132" )
    oItem1132:connect( "triggered(bool)", { ||Item_1132()
        } )
    oSubMenu113:addAction( oItem1132 )
    oSubMenu11:addMenu( oSubMenu113 )
    oMenu1:addMenu( oSubMenu11 )
    oMenu1:addSeparator() //====================
    // ---------- SubMenu 12 ----------
    oSubMenu12 := QAction( oMenu1 )
    oSubMenu12:setText( "SubMenu_12" )
    oSubMenu12:connect( "triggered(bool)", { ||
       SubMenu_12() } )
    oMenu1:addAction( oSubMenu12 )
    oMenuBar:addMenu( oMenu1 )
    // ---------- Menu 2 ----------
    oMenu2 := QMenu()
    oMenu2:setTitle( "Menu_2" )
    // ---------- SubMenu 21 ----------
    oSubMenu21 := QMenu()
    oSubMenu21:setTitle( "SubMenu_21" )
    // --------- Items ---------
    oItem211 := QAction( oSubMenu21 )
    oItem211:setText( "Item_211" )
    oItem211:connect( "triggered(bool)", { || Item_211()
       } )
    oSubMenu21:addAction( oItem211 )
    oItem212 := QAction( oSubMenu21 )
```

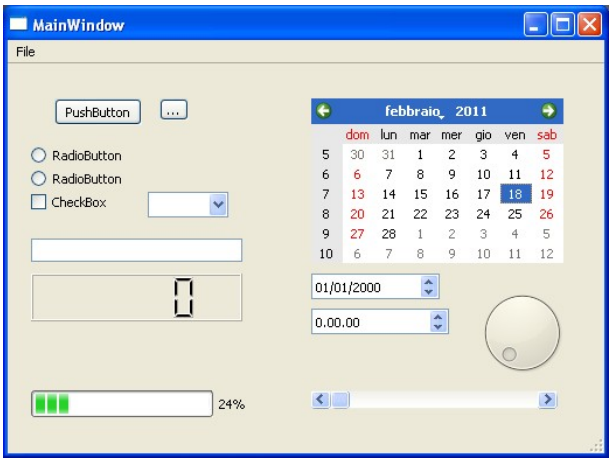
```
   oItem212:setText( "Item_212" )
   oItem212:connect( "triggered(bool)", { || Item_212()
      } )
   oSubMenu21:addAction( oItem212 )
   oMenu2:addMenu( oSubMenu21 )
   oMenuBar:addMenu( oMenu2 )
   // ---------- Menu 3 ----------
   oMenu3 := QMenu()
   oMenu3:setTitle( "Menu_3" )
   // ---------- SubMenu 31 ----------
   oSubMenu31 := QAction( oMenu3 )
   oSubMenu31:setText( "SubMenu_31" )
   oSubMenu31:connect( "triggered(bool)", { ||
      SubMenu_31() } )
   oMenu3:addAction( oSubMenu31 )
   oMenuBar:addMenu( oMenu3 )

   oWnd:Show()
   QApplication():exec()

   RETURN

PROCEDURE Item_111()

   // Type your code here

   RETURN

PROCEDURE Item_112()

   // Type your code here

   RETURN

PROCEDURE Item_1131()

   // Type your code here

   RETURN

PROCEDURE Item_1132()

   // Type your code here

   RETURN

PROCEDURE SubMenu_12()

   // Type your code here

   RETURN

PROCEDURE Item_211()
```

```
   // Type your code here

   RETURN

PROCEDURE Item_212()

   // Type your code here

   RETURN

PROCEDURE SubMenu_31()

   // Type your code here

   RETURN

   // END SOURCE
```

22 UI

22.1 UI file created with QT Creator

The following example shows how to draw a window or a complete set of widget, from a UI file created with QT Creator program. *(by Giovanni Di Maria)*




```
   // SOURCE=QUiLoader.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oUi, oFile

   oFile := QFile( "../res/sample.ui" )
   oFile:open( 1 )

   oUi := QUiLoader()
   oWnd := oUi:load( oFile )
   oFile:close()

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

22.2 A fully functional example with UI file

The following example uses a file .UI created with QT Creator program. Pressing the button, the title changes. *(by Giovanni Di Maria)*


```
 // SOURCE=qt-creator.prg
   // HBP=form.ui

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd

   oWnd := hbqtui_form()
   oWnd:setWindowTitle( "Old Title" )
   oWnd:q_pushButton:Connect( "clicked()",  { ||oWnd:
       setWindowTitle( "Changed" ) } )

   oWnd:show()
   QApplication():exec()

   RETURN
```

```
    // END SOURCE
```

This is the form.ui file, created with QT Creator

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Form</class>
 <widget class="QWidget" name="Form">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>325</width>
    <height>129</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Form</string>
  </property>
  <widget class="QPushButton" name="pushButton">
   <property name="geometry">
    <rect>
     <x>50</x>
     <y>50</y>
     <width>211</width>
     <height>23</height>
    </rect>
   </property>
   <property name="text">
    <string>Change Title Bar</string>
   </property>
  </widget>
 </widget>
 <resources/>
 <connections/>
</ui>
```

To work correctly, the programmer must remember the following rules:

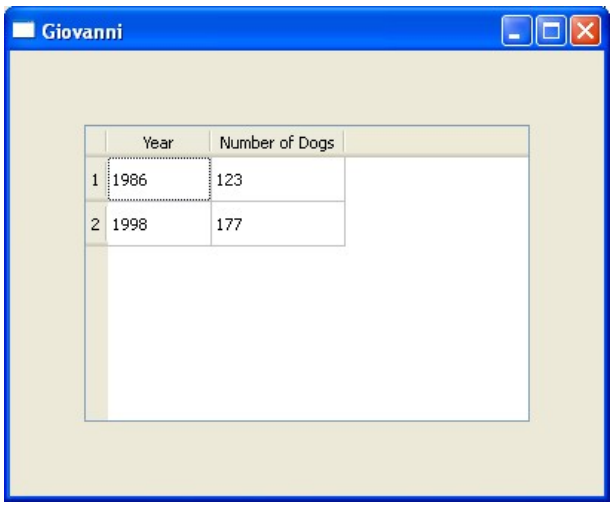
1. You must create the file "form.ui" (or other name) with QT Creator
2. You must include the "form.ui" file in the compilation (directly or in the file .HBP)
3. To refer to children of the widget, you must "rename" the objects with "q_" at the beginning of the line. For example: pushButton –¿ q_pushButton
4. The "form.ui" file is not necessary for the execution of the program. It's compiled in the .exe file

5. If the name of the .UI file is "form.ui", then you must call the hbqtui_form() function. If the name of the .UI file is "example.ui", then you must call the hbqtui_example() function.

23 QTableWidget

23.1 Simple table

The following example shows how to create a small table, with 2 rows and 2 columns. *(by Giovanni Di Maria)*

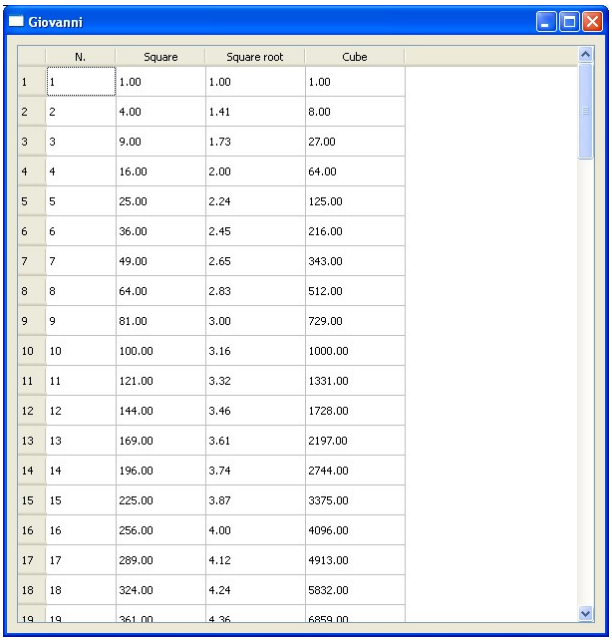



```
   // SOURCE=QTableWidget.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oTable
   LOCAL oCell, oLabel

   oWnd := QMainWindow()
   oWnd:resize( 400, 300 )
   oWnd:setWindowTitle( "Giovanni" )
   //------------Table--------------
   oTable := QTableWidget( oWnd )
   oTable:move( 50, 50 )
   oTable:resize( 300, 200 )
   oTable:setRowCount( 2 )
   oTable:setColumnCount( 2 )
   oTable:setColumnWidth( 0, 70 )
   oTable:setColumnWidth( 1, 90 )
   //------------Fill Table--------------
   oCell := QTableWidgetItem()
   oCell:setText( "1986" )
   oTable:setItem( 0, 0, oCell )
```

```
oCell := QTableWidgetItem()
oCell:setText( "123" )
oTable:setItem( 0, 1, oCell )

oCell := QTableWidgetItem()
oCell:setText( "1998" )
oTable:setItem( 1, 0, oCell )

oCell := QTableWidgetItem()
oCell:setText( "177" )
oTable:setItem( 1, 1, oCell )

oLabel := QStringList()
oLabel:append( "Year" )
oLabel:append( "Number of Dogs" )
oTable:setHorizontalHeaderLabels( oLabel )

oWnd:show()
QApplication():exec()

RETURN

// END SOURCE
```

23.2 Arithmetic tables

The following example shows how to create an Arithmetic table, with 100 rows and 4 columns. The first column contains the number, the second column contains the square of the number, the third column contains the square root of the number and the fourth column contains the cube of the number. The numbers are generated by a loop. *(by Giovanni Di Maria)*

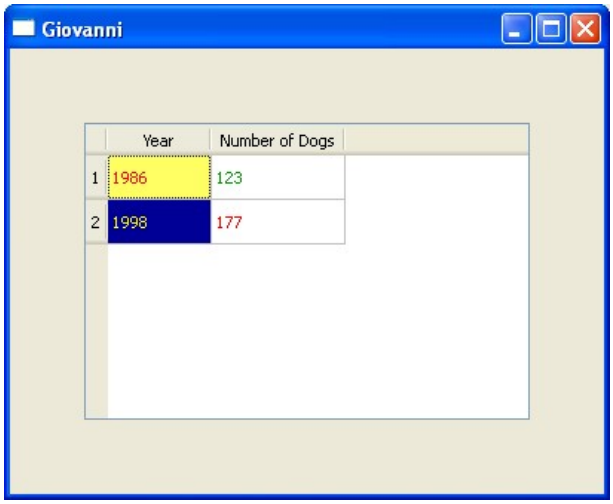



```
   // SOURCE=QTableWidget.prg

#include "hbqtgui.ch"

   STATIC oTable

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oCell, oLabel
   LOCAL k

   oWnd := QMainWindow()
   oWnd:resize( 600, 600 )
   oWnd:setWindowTitle( "Giovanni" )

   //------------Table--------------
```

```
oTable := QTableWidget( oWnd )
oTable:move( 10, 10 )
oTable:resize( 580, 580 )
oTable:setRowCount( 100 )
oTable:setColumnCount( 4 )
oTable:setColumnWidth( 0, 70 )
oTable:setColumnWidth( 1, 90 )

//------------Fill Table--------------
for k = 1 TO 100
   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(k ) ) )
   oTable:setItem( k - 1, 0, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(k ^ 2 ) ) )
   oTable:setItem( k - 1, 1, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(Sqrt(k ) ) ) )
   oTable:setItem( k - 1, 2, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(k ^ 3 ) ) )
   oTable:setItem( k - 1, 3, oCell )
next k

oLabel := QStringList()
oLabel:append( "N." )
oLabel:append( "Square" )
oLabel:append( "Square root" )
oLabel:append( "Cube" )
oTable:setHorizontalHeaderLabels( oLabel )

oWnd:show()
QApplication():exec()

RETURN

// END SOURCE
```

23.3 Colored Cells

The following example shows how to create a small table, with 2 rows and 2 columns. The cells are colored. *(by Giovanni Di Maria)*




```
   // SOURCE=QTableWidget.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oTable
   LOCAL oCell, oLabel

   oWnd := QMainWindow()
   oWnd:resize( 400, 300 )
   oWnd:setWindowTitle( "Giovanni" )

   //------------Table--------------
   oTable := QTableWidget( oWnd )
   oTable:move( 50, 50 )
   oTable:resize( 300, 200 )
   oTable:setRowCount( 2 )
   oTable:setColumnCount( 2 )
   oTable:setColumnWidth( 0, 70 )
   oTable:setColumnWidth( 1, 90 )
   //------------Fill Table--------------
   oCell := QTableWidgetItem()
   oCell:setText( "1986" )
```

```
   oCell:setForeground( QBrush( QColor(255,0,0 ) ) )
   oCell:setBackground( QBrush( QColor(255,255,100 ) ) )
   oTable:setItem( 0, 0, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( "123" )
   oCell:setForeground( QBrush( QColor(0,150,0 ) ) )
   oTable:setItem( 0, 1, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( "1998" )
   oCell:setForeground( QBrush( QColor(255,255,0 ) ) )
   oCell:setBackground( QBrush( QColor(0,0,150 ) ) )
   oTable:setItem( 1, 0, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( "177" )
   oCell:setForeground( QBrush( QColor(255,0,0 ) ) )
   oTable:setItem( 1, 1, oCell )

   oLabel := QStringList()
   oLabel:append( "Year" )
   oLabel:append( "Number of Dogs" )

   oTable:setHorizontalHeaderLabels( oLabel )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

23.4 Arithmetic tables (colored columns)

The following example shows how to create an Arithmetic table, with 100 rows and 4 columns. The first column contains the number, the second column contains the square of the number, the third column contains the square root of the number and the fourth column contains the cube of the number. The numbers are generated by a loop. The columns are colored. *(by Giovanni Di Maria)*


```
   // SOURCE=QTableWidget.prg

#include "hbqtgui.ch"

   STATIC oTable

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oCell, oLabel
   LOCAL k

   oWnd := QMainWindow()
   oWnd:resize( 600, 600 )
   oWnd:setWindowTitle( "Giovanni" )
```

```
//------------Table--------------
oTable := QTableWidget( oWnd )
oTable:move( 10, 10 )
oTable:resize( 580, 580 )
oTable:setRowCount( 100 )
oTable:setColumnCount( 4 )
oTable:setColumnWidth( 0, 70 )
oTable:setColumnWidth( 1, 90 )

//------------Fill Table--------------
for k = 1 TO 100
   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(k ) ) )
   oCell:setForeground( QBrush( QColor(255,0,0 ) ) )
   oCell:setBackground( QBrush( QColor(255,255,100 )
      ) )
   oTable:setItem( k - 1, 0, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(k ^ 2 ) ) )
   oCell:setForeground( QBrush( QColor(255,255,255 )
      ) )
   oCell:setBackground( QBrush( QColor(0,0,250 ) ) )
   oTable:setItem( k - 1, 1, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(Sqrt(k ) ) ) )
   oCell:setForeground( QBrush( QColor(255,255,255 )
      ) )
   oCell:setBackground( QBrush( QColor(200,50,200 ) )
       )
   oTable:setItem( k - 1, 2, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(k ^ 3 ) ) )
   oCell:setForeground( QBrush( QColor(0,100,0 ) ) )
   oCell:setBackground( QBrush( QColor(200,255,200 )
      ) )
   oTable:setItem( k - 1, 3, oCell )
next k

oLabel := QStringList()
oLabel:append( "N." )
oLabel:append( "Square" )
oLabel:append( "Square root" )
oLabel:append( "Cube" )
oTable:setHorizontalHeaderLabels( oLabel )

oWnd:show()
QApplication():exec()

RETURN

// END SOURCE
```

23.5 Arithmetic tables (conditional coloring of rows)

The following example shows how to create an Arithmetic table, with 100 rows and 4 columns. The first column contains the number, the second column contains the square of the number, the third column contains the square root of the number and the fourth column contains the cube of the number. The numbers are generated by a loop. The rows are colored following a condition (even and odd). *(by Giovanni Di Maria)*


```
   // SOURCE=QTableWidget.prg

#include "hbqtgui.ch"

   STATIC oTable

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oCell, oLabel
   LOCAL k
   LOCAL oBrushForeground, oBrushBackground

   oWnd := QMainWindow()
   oWnd:resize( 600, 600 )
   oWnd:setWindowTitle( "Giovanni" )
```

```
//------------Table--------------
oTable := QTableWidget( oWnd )
oTable:move( 10, 10 )
oTable:resize( 580, 580 )
oTable:setRowCount( 100 )
oTable:setColumnCount( 4 )
oTable:setColumnWidth( 0, 70 )
oTable:setColumnWidth( 1, 90 )

//------------Fill Table--------------

for k = 1 TO 100

   IF k % 2 = 0
      oBrushForeground := QBrush( QColor( 0,0,200 ) )
      oBrushBackground := QBrush( QColor( 220,255,220
          ) )
   ELSE
      oBrushForeground := QBrush( QColor( 0,0,200 ) )
      oBrushBackground := QBrush( QColor( 220,220,255
          ) )
   ENDIF

   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(k ) ) )
   oCell:setForeground( oBrushForeground )
   oCell:setBackground( oBrushBackground )
   oTable:setItem( k - 1, 0, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(k ^ 2 ) ) )
   oCell:setForeground( oBrushForeground )
   oCell:setBackground( oBrushBackground )
   oTable:setItem( k - 1, 1, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(Sqrt(k ) ) ) )
   oCell:setForeground( oBrushForeground )
   oCell:setBackground( oBrushBackground )
   oTable:setItem( k - 1, 2, oCell )

   oCell := QTableWidgetItem()
   oCell:setText( AllTrim( Str(k ^ 3 ) ) )
   oCell:setForeground( oBrushForeground )
   oCell:setBackground( oBrushBackground )
   oTable:setItem( k - 1, 3, oCell )
next k

oLabel := QStringList()
oLabel:append( "N." )
oLabel:append( "Square" )
oLabel:append( "Square root" )
oLabel:append( "Cube" )
```

```
    oTable:setHorizontalHeaderLabels( oLabel )

    oWnd:show()
    QApplication():exec()

    RETURN

    // END SOURCE
```

24 QInputDialog

24.1 Input Dialog window

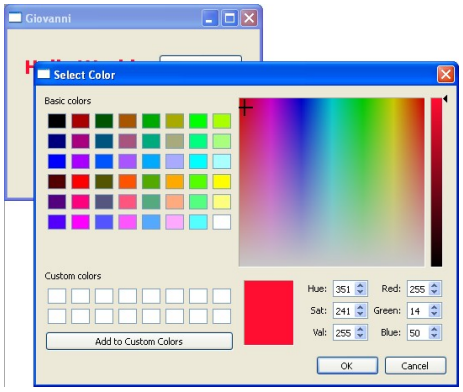
The following example uses a input dialog window to insert values into the program. *(by Giovanni Di Maria)*


```
   // SOURCE=QInputDialog.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL cString
   LOCAL oDialog

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 300 )

   oDialog := QInputDialog()
   cString = oDialog:getText( oWnd, "Title", "What's
      your name?" )
   oWnd:setWindowTitle( cString )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

25 QColorDialog

25.1 Color Dialog

The following example shows how to use a color dialog to change the color of a text label. *(by Giovanni Di Maria)*



```
   // SOURCE=QColorDialog.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oButton
   LOCAL oColorDialog, oText

   oWnd := QMainWindow()
   oWnd:resize( 300, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   oText := Qlabel( oWnd )
   oText:setText( "<h1>Hello World</h1>" )
   oText:move( 20, 20 )
   oText:resize( 250, 50 )
   oButton := QPushButton( oWnd )
   oButton:move( 180, 30 )
   oButton:setText( "Change Color" )
   oButton:Connect( "clicked()", { || oColorDialog:open
      () } )

   oColorDialog := QColorDialog( oWnd )
   oColorDialog:Connect( "currentColorChanged(QColor)",
      { ||change( oColorDialog,oText ) } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE change( oCd, cT )

   LOCAL oPalette

   oPalette := QPalette()
   oPalette:SetColor( QPalette_WindowText, oCd:
      currentColor() )
   cT:setPalette( oPalette )

   RETURN

   // END SOURCE
```

26 QProgressBar

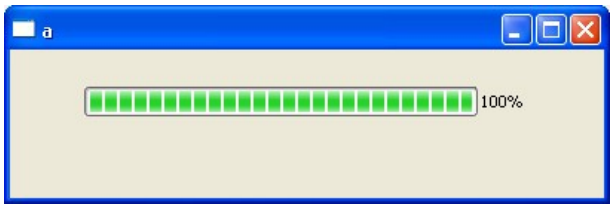
26.1 Progress Bar

The following example shows use of a progress bar. It' useful during a long counting or elaboration of big archives. *(by Giovanni Di Maria)*


```
   // SOURCE=QProgressBar.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oBar
   LOCAL k

   oBar := QProgressBar()
   oBar:resize( 400, 50 )
   oBar:move( 50, 50 )
   oBar:setRange( 1, 500000 )
   oBar:setWindowTitle( "Wait..." )
   oBar:Show()
   oBar:repaint()

   for k = 1 TO 500000
      oBar:setValue( k )
   next k

   oBar:quit()
   QApplication():exec()

   RETURN

   // END SOURCE
```

26.2 Progress Bar in a widget

The following example shows use of a progress bar in a widget. *(by Giovanni Di Maria)*

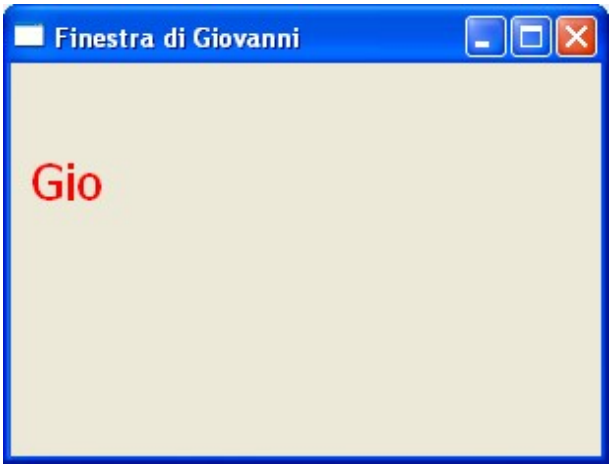



```
  // SOURCE=QProgressBar.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oBar
   LOCAL k

   oWnd := QMainWindow()
   oWnd:resize( 400, 100 )
   oWnd:show()

   oBar := QProgressBar( oWnd )
   oBar:resize( 300, 20 )
   oBar:move( 50, 25 )
   oBar:setRange( 1, 500000 )
   oBar:Show()

   for k = 1 TO 500000
      oBar:setValue( k )
   next k

   QApplication():exec()

   RETURN

   // END SOURCE
```

27 HTML

27.1 Tag HTML

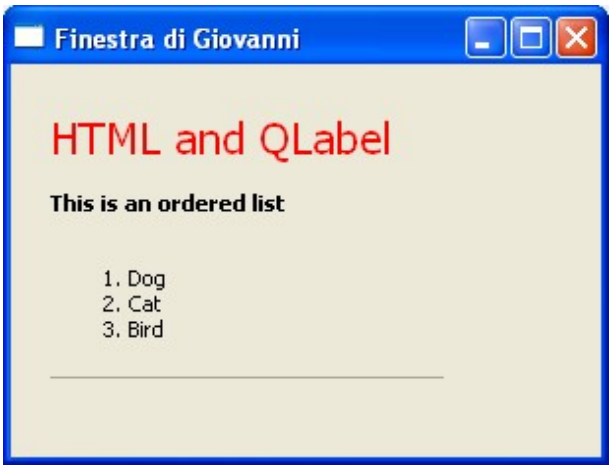
The following example shows how to use tag HTML to change the color, size and aspect of text of other objects. *(by Giovanni Di Maria)*




```
   // SOURCE=tag-html.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oText

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   oText := Qlabel( oWnd )
   oText:setText( "<font color=#FF0000 size=7>Gio</font
      >" )
   oText:move( 10, 10 )
   oText:resize( 280, 100 )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

27.2 Tag HTML

The following example shows how to use tag HTML, like ordered list, text formatting and break row. *(by Giovanni Di Maria)*




```
   // SOURCE=tag-html.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oLabel, cString

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   cString = "<font color=#FF0000 size=6>HTML and QLabel
      </font>"
   cString = cString + "<br><br>"
   cString = cString + "<b>This is an ordered list</b><
      br>"
   cString = cString + "<ol>"
   cString = cString + "<li>Dog</li>"
   cString = cString + "<li>Cat</li>"
   cString = cString + "<li>Bird</li>"
   cString = cString + "</ol>"
   cString = cString + "<hr>"

   oLabel := QLabel( oWnd )
   oLabel:resize( 200, 150 )
```

```
    oLabel:move( 20, 20 )
    oLabel:setText( cString )

    oWnd:show()
    QApplication():exec()

    RETURN

    // END SOURCE
```

27.3 Tag HTML and Images

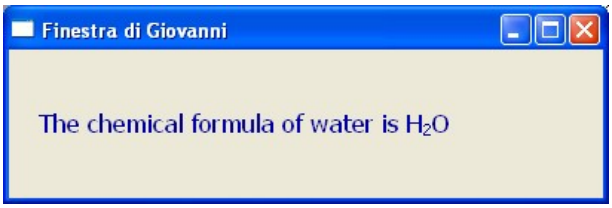
The following example shows how to insert an image with the ¡IMG¿ tag HTML. *(by Giovanni Di Maria)*


```
   // SOURCE=tag-html-img.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oLabel
   LOCAL cString

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )

   cString = "<img src='../res/logo_google.bmp'>"
   cString = cString + "<br><br>"
   cString = cString + "<img src='../res/logo_google.bmp
      ' width=70>"

   oLabel := QLabel( oWnd )
   oLabel:resize( 200, 150 )
   oLabel:move( 20, 20 )
   oLabel:setText( cString )

   oWnd:show()
   QApplication():exec()
```

```
   RETURN

   // END SOURCE
```

27.4 Tag HTML and subscript

The following example shows how to use a subscript with the text. *(by Giovanni Di Maria)*




```
   // SOURCE=tag-html-sub.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oLabel
   LOCAL oString

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 100 )

   oString = "<font size=5 color=#0000AA>"
   oString = oString + "The chemical formula of water is
       H<sub>2</sub>O"
   oString = oString + "</font>"

   oLabel := QLabel( oWnd )
   oLabel:resize( 300, 60 )
   oLabel:move( 20, 20 )
   oLabel:setText( oString )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

28 QSlider

28.1 Slider with value

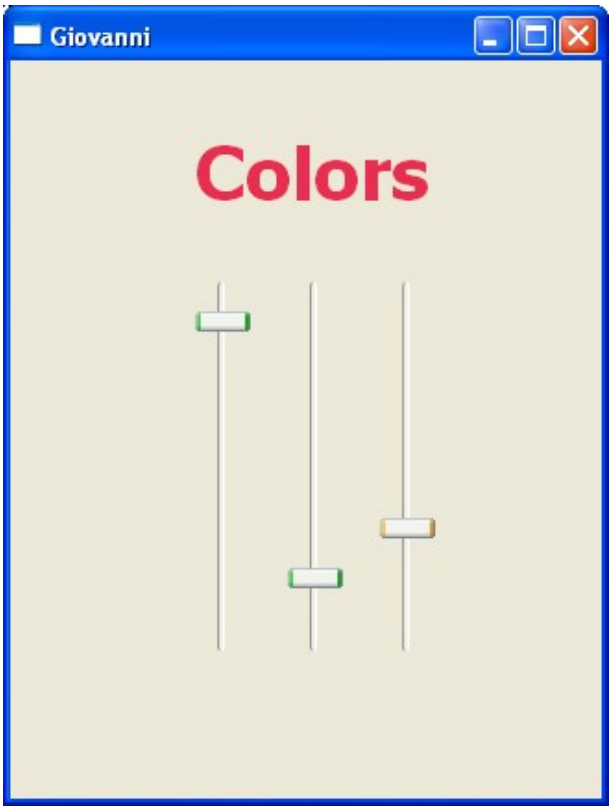
The following example shows a slider that changes the value of a Lcd display. Values are between 0 and 99. *(by Giovanni Di Maria)*


```
   // SOURCE=QSlider.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oSlider
   LOCAL oLcd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 240, 200 )

   oLcd := QLCDNumber( oWnd )
   oLcd:resize( 200, 120 )
   oLcd:move( 20, 20 )

   oSlider := QSlider( oWnd )
   oSlider:resize( 211, 21 )
   oSlider:move( 10, 160 )
   oSlider:setMinimum( 0 )
```

```
    oSlider:setMaximum( 99 )
    oSlider:setSingleStep( 1 )
    oSlider:setValue( 0 )
    oSlider:setOrientation( Qt_Horizontal )
    oSlider:Connect( "valueChanged(int)", { |x| oLcd:
       display( x ) } )

    oWnd:show()
    QApplication():exec()

    RETURN

    // END SOURCE
```

28.2 Slider RGB

The following example uses three sliders to change the RGB color of a text. *(by Giovanni Di Maria)*




```
   // SOURCE=QSlider.prg

#include "hbqtgui.ch"

   STATIC oPalette
   STATIC oSliderRed, oSliderGreen, oSliderBlu
   STATIC oText

PROCEDURE Main()

   LOCAL window
   LOCAL font

   window := QMainWindow()
   window:resize( 320, 400 )
```

```
   window:setWindowTitle( "Giovanni" )

   font := QFont()
   font:setPointSize( 30 )
   font:setBold( .T. )

   oText := QLabel( window )
   oText:setText( "Colors" )
   oText:move( 100, 10 )
   oText:resize( 200, 100 )
   oText:setfont( font )

   oSliderRed := QSlider( window )
   oSliderRed:resize( 30, 200 )
   oSliderRed:move( 100, 120 )
   oSliderRed:setMinimum( 0 )
   oSliderRed:setMaximum( 255 )
   oSliderRed:setSingleStep( 1 )
   oSliderRed:setPageStep( 10 )
   oSliderRed:setValue( 0 )
   oSliderRed:Connect( "valueChanged(int)", { ||
      change_colors() } )

   oSliderGreen := QSlider( window )
   oSliderGreen:resize( 30, 200 )
   oSliderGreen:move( 150, 120 )
   oSliderGreen:setMinimum( 0 )
   oSliderGreen:setMaximum( 255 )
   oSliderGreen:setSingleStep( 1 )
   oSliderGreen:setPageStep( 10 )
   oSliderGreen:setValue( 0 )
   oSliderGreen:Connect( "valueChanged(int)", { ||
      change_colors() } )

   oSliderBlu := QSlider( window )
   oSliderBlu:resize( 30, 200 )
   oSliderBlu:move( 200, 120 )
   oSliderBlu:setMinimum( 0 )
   oSliderBlu:setMaximum( 255 )
   oSliderBlu:setSingleStep( 1 )
   oSliderBlu:setPageStep( 10 )
   oSliderBlu:setValue( 0 )
   oSliderBlu:Connect( "valueChanged(int)", { ||
      change_colors() } )

   window:show()
   QApplication():exec()

   RETURN

PROCEDURE change_colors()

   oPalette := QPalette()
   oPalette:SetColor( QPalette_WindowText, QColor(
```

```
        oSliderRed:value , oSliderGreen:value , oSliderBlu
        :value ) )

    oText:setPalette( oPalette )

    RETURN

    // END SOURCE
```

28.3 Sliders synchronization

The following example shows how to synchronize two sliders, so that moving a slider, it moves the other. *(by Giovanni Di Maria)*

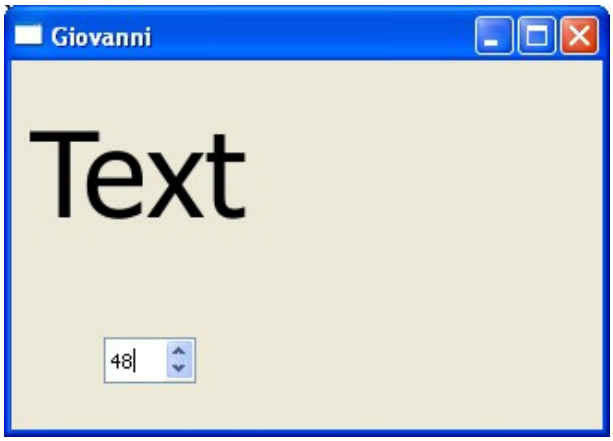

```
   // SOURCE=QSlider.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oSlider1, oSlider2

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 240, 200 )

   oSlider1 := QSlider( oWnd )
   oSlider1:resize( 211, 21 )
   oSlider1:move( 10, 50 )
   oSlider1:setMinimum( 0 )
   oSlider1:setMaximum( 99 )
   oSlider1:setSingleStep( 1 )
   oSlider1:setValue( 0 )
   oSlider1:setOrientation( Qt_Horizontal )
   oSlider1:Connect( "valueChanged(int)", { |x|oSlider2:
      setValue( x ) } )
```

```
oSlider2 := QSlider( oWnd )
oSlider2:resize( 211, 21 )
oSlider2:move( 10, 100 )
oSlider2:setMinimum( 0 )
oSlider2:setMaximum( 99 )
oSlider2:setSingleStep( 1 )
oSlider2:setValue( 0 )
oSlider2:setOrientation( Qt_Horizontal )
oSlider2:Connect( "valueChanged(int)", { |x|oSlider1:
    setValue( x ) } )

oWnd:show()
QApplication():exec()

RETURN

// END SOURCE
```

29 QSpinBox

29.1 Spin Control

The following example uses a spin control, to modify the size of a text. Values can be typed or chosen with the arrows. *(by Giovanni Di Maria)*




```
   // SOURCE=QSpinBox.prg

#include "hbqtgui.ch"

   STATIC oFont
   STATIC oText
   STATIC oChanger

PROCEDURE Main()

   LOCAL oWnd

   oWnd := QMainWindow()
   oWnd:resize( 320, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   oFont := QFont()
   oFont:setPointSize( 30 )
   oText := QLabel( oWnd )
   oText:setText( "Text" )
   oText:move( 10, 10 )
   oText:resize( 280, 100 )
   oText:setFont( oFont )

   oChanger := QSpinBox( oWnd )
   oChanger:move( 50, 150 )
   oChanger:resize( 50, 25 )
```

```
   oChanger:Connect( "valueChanged(int)", { ||
      change_dimension() } )
   oChanger:setMinimum( 1 )
   oChanger:setMaximum( 72 )
   oChanger:setSingleStep( 1 )
   oChanger:setValue( 30 )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE change_dimension()

   oFont:setPointSize( oChanger:value )
   oText:setFont( oFont )

   RETURN

   // END SOURCE
```

30 QComboBox

30.1 ComboBox

The following example creates a combobox, with many items. User can select any item. *(by Giovanni Di Maria)*


```
   // SOURCE=QComboBox.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oCombo

   oWnd := QMainWindow()
   oWnd:resize( 320, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   oCombo := QComboBox( oWnd )
   oCombo:move( 100, 50 )
   oCombo:resize( 100, 25 )
   oCombo:addItem( "Francia" )
   oCombo:addItem( "Italia" )
   oCombo:addItem( "U.S.A." )
   oCombo:addItem( "Germania" )
   oCombo:addItem( "Belgio" )
   oCombo:addItem( "Spagna" )
   oCombo:addItem( "Portogallo" )
   oCombo:addItem( "Islanda" )

   oWnd:show()
   QApplication():exec()
```

```
   RETURN

   // END SOURCE
```

30.2 ComboBox with Update

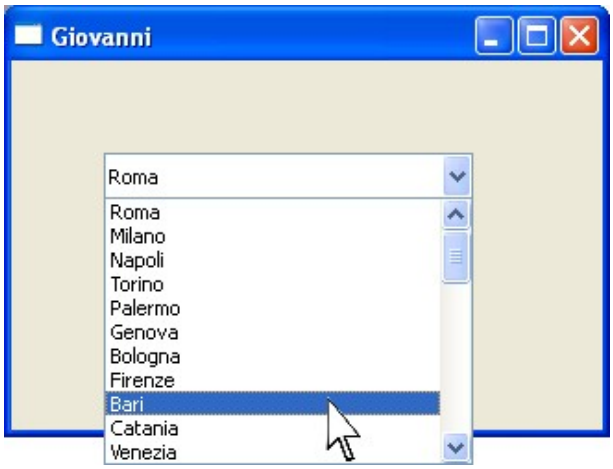
The following example creates a combobox, with many items. User can select any item. When select, it's shown in a label. *(by Giovanni Di Maria)*

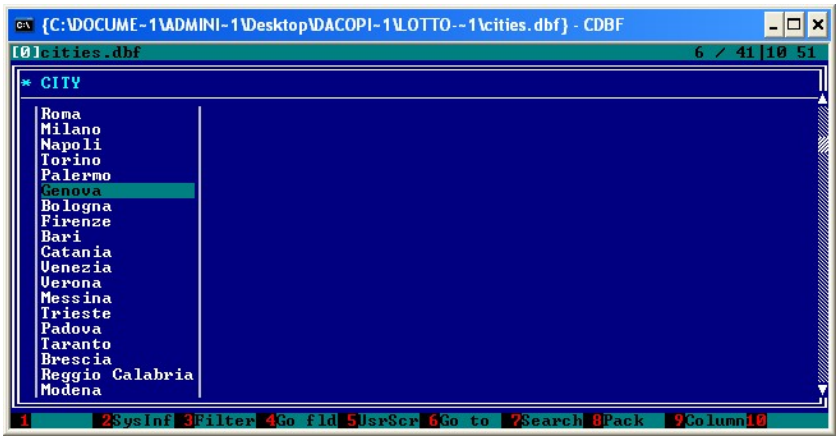

```
   // SOURCE=QComboBox.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oCombo, oLabel

   oWnd := QMainWindow()
   oWnd:resize( 320, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   oCombo := QComboBox( oWnd )
   oCombo:move( 20, 20 )
   oCombo:resize( 100, 25 )
   oCombo:addItem( "Francia" )
   oCombo:addItem( "Italia" )
   oCombo:addItem( "U.S.A." )
   oCombo:addItem( "Germania" )
   oCombo:addItem( "Belgio" )
   oCombo:addItem( "Spagna" )
   oCombo:addItem( "Portogallo" )
   oCombo:addItem( "Islanda" )
   oCombo:Connect( "currentIndexChanged(int)", { ||
       oLabel:setText( oCombo:currentText() ) } )

   oLabel := QLabel( oWnd )
```

```
    oLabel:move( 200, 150 )

    oWnd:show()
    QApplication():exec()

    RETURN

    // END SOURCE
```

30.3 Populating a ComboBox from a DBF

The following example creates a combobox, with many items. The items are appended from a DBF database. *(by Giovanni Di Maria)*





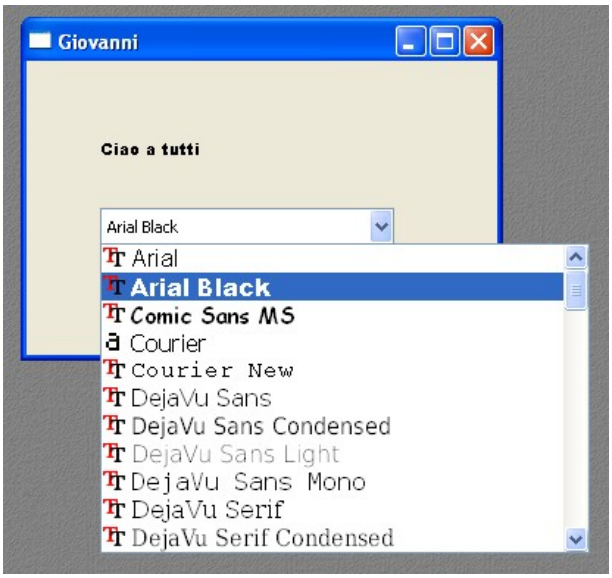

```
   // SOURCE=QComboBox.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oCombo
```

```
oWnd := QMainWindow()
oWnd:resize( 320, 200 )
oWnd:setWindowTitle( "Giovanni" )

oCombo := QComboBox( oWnd )
oCombo:move( 50, 50 )
oCombo:resize( 200, 25 )

USE ../res/cities

DO WHILE .NOT. Eof()
   oCombo:addItem( cities -> city )
   SKIP
ENDDO

USE

oWnd:show()
QApplication():exec()

RETURN

// END SOURCE
```

31 QFontComboBox

31.1 ComboBox with Fonts

The following example creates a combobox, with a set of system fonts. The text is changed with the selected font. *(by Giovanni Di Maria)*



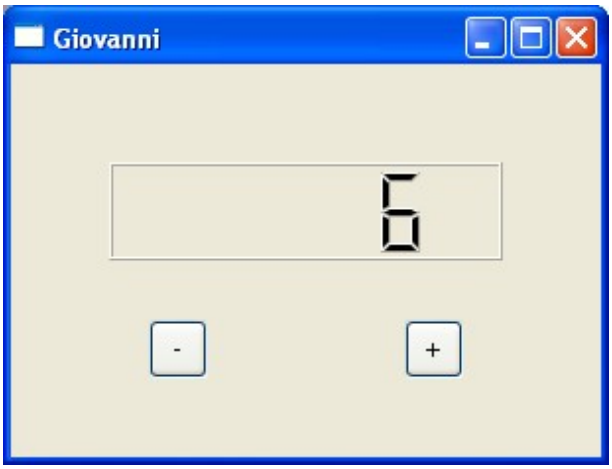

```
   // SOURCE=QFontComboBox.prg

#include "hbqtgui.ch"

   STATIC oText
   STATIC oCombo

PROCEDURE Main()

   LOCAL oWnd

   oWnd := QMainWindow()
   oWnd:resize( 320, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   oText := Qlabel( oWnd )
   oText:setText( "Ciao a tutti" )
   oText:resize( 200, 80 )
   oText:move( 50, 20 )

   oCombo := QFontComboBox( oWnd )
   oCombo:move( 50, 100 )
```

```
   oCombo:resize( 200, 25 )
   oCombo:Connect( "currentFontChanged(QFont)", { ||
      change_text() } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE change_text()

   oText:setFont( oCombo:currentFont() )

   RETURN

   // END SOURCE
```

32 QLCDNumber

32.1 LCD Display

The following example creates a nice LCD display, to view numbers. It's properties can be changed. *(by Giovanni Di Maria)*

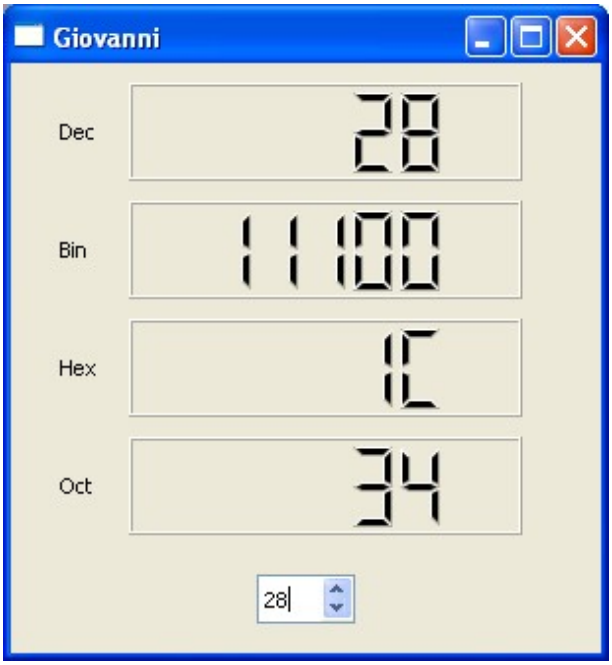



```
   // SOURCE=QLCDNumber.prg

#include "hbqtgui.ch"

   STATIC oLcd

PROCEDURE Main()

   LOCAL oWnd, oButtonMinus, oButtonPlus

   oWnd := QMainWindow()
   oWnd:resize( 300, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   oLcd := QLCDNumber( oWnd )
   oLcd:move( 50, 50 )
   oLcd:resize( 200, 50 )

   oButtonMinus := QPushButton( oWnd )
   oButtonMinus:resize( 30, 30 )
   oButtonMinus:move( 70, 130 )
   oButtonMinus:setText( "-" )
   oButtonMinus:Connect( "clicked()", { || decrementa()
      } )

   oButtonPlus := QPushButton( oWnd )
```

```
   oButtonPlus:resize( 30, 30 )
   oButtonPlus:move( 200, 130 )
   oButtonPlus:setText( "+" )
   oButtonPlus:Connect( "clicked()", { || incrementa() }
       )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE incrementa()

   LOCAL x

   x = oLcd:value()
   x ++
   oLcd:display( x )

   RETURN

PROCEDURE decrementa()

   LOCAL x

   x = oLcd:value()
   x --
   oLcd:display( x )

   RETURN

   // END SOURCE
```

32.2 LCD with number in decimal, binary, hexdecimal and octal base

The following example creates four LCD displays, that display a number in four bases: decimal, binary, hexdecimal and octal. The number is changed by a Spin Box. *(by Giovanni Di Maria)*



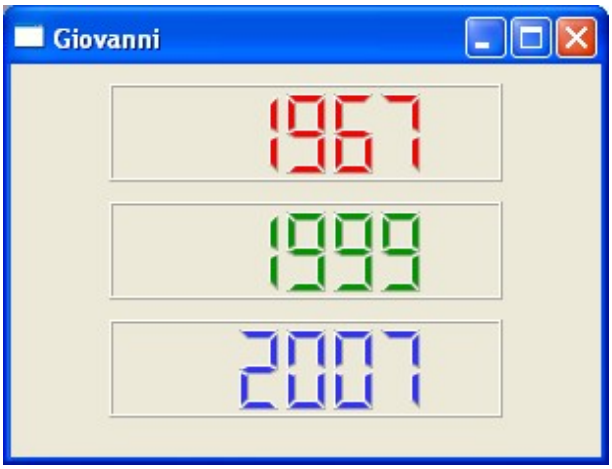

```
   // SOURCE=QLCDNumber.prg

#include "hbqtgui.ch"

   STATIC oLcd1, oLcd2, oLcd3, oLcd4

PROCEDURE Main()

   LOCAL oWnd, oSpinBox
   LOCAL oLabel1, oLabel2, oLabel3, oLabel4

   oWnd := QMainWindow()
   oWnd:resize( 300, 300 )
   oWnd:setWindowTitle( "Giovanni" )

   oLcd1 := QLCDNumber( oWnd )
   oLcd1:move( 60, 10 )
```

```
   oLcd1:resize( 200, 50 )
   oLcd1:SetMode( 1 )

   oLcd2 := QLCDNumber( oWnd )
   oLcd2:move( 60, 70 )
   oLcd2:resize( 200, 50 )
   oLcd2:SetMode( 3 )

   oLcd3 := QLCDNumber( oWnd )
   oLcd3:move( 60, 130 )
   oLcd3:resize( 200, 50 )
   oLcd3:SetMode( 0 )

   oLcd4 := QLCDNumber( oWnd )
   oLcd4:move( 60, 190 )
   oLcd4:resize( 200, 50 )
   oLcd4:SetMode( 2 )

   oLabel1 := QLabel( oWnd )
   oLabel1:setText( "Dec" )
   oLabel1:move( 25, 20 )

   oLabel2 := QLabel( oWnd )
   oLabel2:setText( "Bin" )
   oLabel2:move( 25, 80 )

   oLabel3 := QLabel( oWnd )
   oLabel3:setText( "Hex" )
   oLabel3:move( 25, 140 )

   oLabel4 := QLabel( oWnd )
   oLabel4:setText( "Oct" )
   oLabel4:move( 25, 200 )

   oSpinBox := QSpinBox( oWnd )
   oSpinBox:move( 125, 260 )
   oSpinBox:resize( 50, 25 )
   oSpinBox:Connect( "valueChanged(int)", { |x|disp( x )
       } )
   oSpinBox:setMinimum( 0 )
   oSpinBox:setMaximum( 31 )
   oSpinBox:setSingleStep( 1 )
   oSpinBox:setValue( 0 )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE DISP( x )

   oLcd1:display( x )
   oLcd2:display( x )
   oLcd3:display( x )
```

```
    oLcd4:display( x )

    RETURN

    // END SOURCE
```

32.3 Colored LCD Display

The following example creates three nice colored LCD displays. *(by Giovanni Di Maria)*



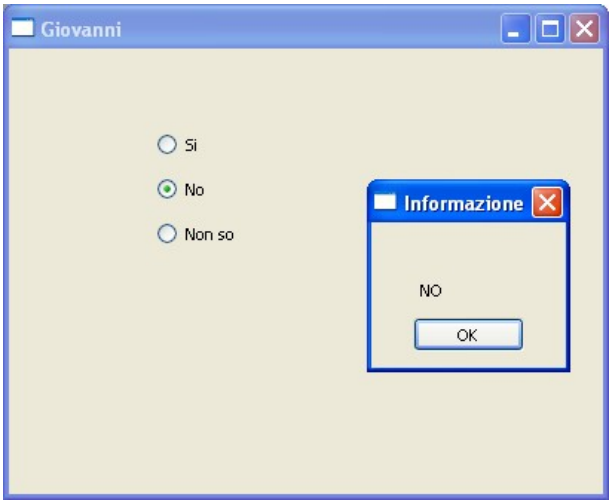

```
   // SOURCE=QLCDNumber.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oLcd1, oLcd2, oLcd3, oPalette

   oWnd := QMainWindow()
   oWnd:resize( 300, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   oPalette := QPalette()

   oPalette:SetColor( QPalette_WindowText, QColor(
      255,0,0 ) )
   oLcd1 := QLCDNumber( oWnd )
   oLcd1:move( 50, 10 )
   oLcd1:resize( 200, 50 )
   oLcd1:display( 1967 )
   oLcd1:setPalette( oPalette )

   oPalette:SetColor( QPalette_WindowText, QColor(
      0,150,0 ) )
   oLcd2 := QLCDNumber( oWnd )
   oLcd2:move( 50, 70 )
```

```
    oLcd2:resize( 200, 50 )
    oLcd2:display( 1999 )
    oLcd2:setPalette( oPalette )

    oPalette:SetColor( QPalette_WindowText, QColor(
       50,50,255 ) )
    oLcd3 := QLCDNumber( oWnd )
    oLcd3:move( 50, 130 )
    oLcd3:resize( 200, 50 )
    oLcd3:display( 2007 )
    oLcd3:setPalette( oPalette )

    oWnd:show()
    QApplication():exec()

    RETURN

    // END SOURCE
```

33 QRadioButton

33.1 Radio Buttons

The following example shows how to use a radio button, with three options. Every option is connected to a function. *(by Giovanni Di Maria)*

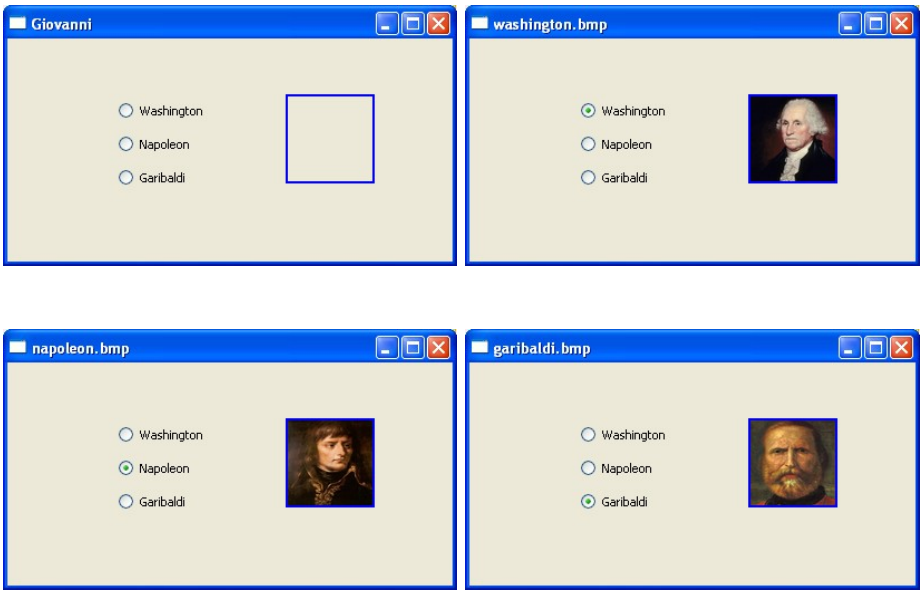



```
   // SOURCE=QRadioButton.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oButton1, oButton2, oButton3

   oWnd := QMainWindow()
   oWnd:resize( 400, 300 )
   oWnd:setWindowTitle( "Giovanni" )

   oButton1 := QRadioButton( oWnd )
   oButton1:move( 100, 50 )
   oButton1:setText( "Si" )
   oButton1:Connect( "clicked()", { || message( "SI" ) }
        )

   oButton2 := QRadioButton( oWnd )
   oButton2:move( 100, 80 )
   oButton2:setText( "No" )
   oButton2:Connect( "clicked()", { || message( "NO" ) }
        )
```

```
   oButton3 := QRadioButton( oWnd )
   oButton3:move( 100, 110 )
   oButton3:setText( "Non so" )
   oButton3:Connect( "clicked()", { || message( "NON SO"
       ) } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE message( cMsg )

   LOCAL oBox

   oBox := QMessageBox()
   oBox:setInformativeText( cMsg )
   oBox:setWindowTitle( "Informazione" )

   oBox:exec()

   RETURN

   // END SOURCE
```

33.2 Radio Buttons and images

The following example shows how to use a radio button, with three options. Every option is connected to a function, showing an image. *(by Giovanni Di Maria)*



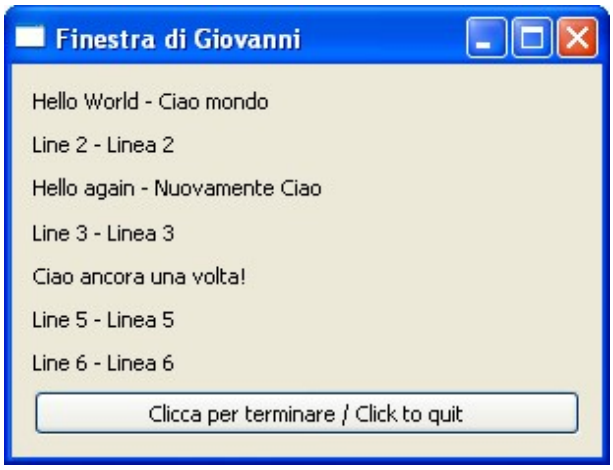

```
   // SOURCE=QRadioButton.prg

#include "hbqtgui.ch"

   STATIC oWnd
   STATIC oImg

PROCEDURE Main()

   LOCAL cF1 := "../res/washington.bmp"
   LOCAL cF2 := "../res/napoleon.bmp"
   LOCAL cF3 := "../res/garibaldi.bmp"
   LOCAL oButton1, oButton2, oButton3

   oWnd := QMainWindow()
   oWnd:resize( 400, 200 )
   oWnd:setWindowTitle( "Giovanni" )

   oButton1 := QRadioButton( oWnd )
   oButton1:move( 100, 50 )
   oButton1:setText( "Washington" )
```

```
   oButton1:Connect( "clicked()", { || message( cF1 ) }
      )

   oButton2 := QRadioButton( oWnd )
   oButton2:move( 100, 80 )
   oButton2:setText( "Napoleon" )
   oButton2:Connect( "clicked()", { || message( cF2 ) }
      )

   oButton3 := QRadioButton( oWnd )
   oButton3:move( 100, 110 )
   oButton3:setText( "Garibaldi" )
   oButton3:Connect( "clicked()", { || message( cF3 ) }
      )

   oImg := QLabel( oWnd )
   oImg:move( 250, 50 )
   oImg:resize( 80, 80 )
   oImg:setStyleSheet( "border: 2px solid #0000ff;" )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE message( cMsg )

   oWnd:setWindowTitle( cMsg )
   oImg:SetPixmap( QPixmap( cMsg ) )

   RETURN

   // END SOURCE
```

34 QVBoxLayout

34.1 Vertical Layout

The following example uses the vertical layouts, to place controls without to specify any position. The placing and position is automatic. *(by Massimo Belgrano and Marco Braida)*



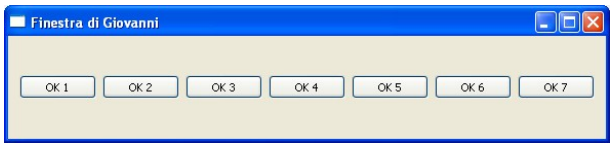

```
   // SOURCE=QVBoxLayout.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oW, oLay
   LOCAL oT0, oT1, oT2, oT3, oT4, oT5, oT6, oB0

   oW := QWidget()
   oW:setWindowTitle( "Finestra di Giovanni" )
   oW:resize( 300, 200 )

   oT0 := Qlabel()
   oT0:setText( "Hello World - Ciao mondo" )

   oT1 := Qlabel()
   oT1:setText( "Line 2 - Linea 2" )

   oT2 := Qlabel()
   oT2:setText( "Hello again - Nuovamente Ciao" )

   oT3 := Qlabel()
   oT3:setText( "Line 3 - Linea 3" )
```

```
   oT4 := Qlabel()
   oT4:setText( "Ciao ancora una volta!" )

   oT5 := Qlabel()
   oT5:setText( "Line 5 - Linea 5" )

   oT6 := Qlabel()
   oT6:setText( "Line 6 - Linea 6" )

   oB0 := QPushButton()
   oB0:setText( "Clicca per terminare / Click to quit" )
   oB0:Connect( "clicked()", { || QApplication():quit()
      } )

   oLay := QVBoxLayout( oW )
   oLay:addWidget( oT0 )
   oLay:addWidget( oT1 )
   oLay:addWidget( oT2 )
   oLay:addWidget( oT3 )
   oLay:addWidget( oT4 )
   oLay:addWidget( oT5 )
   oLay:addWidget( oT6 )
   oLay:addWidget( oB0 )

   oW:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

35 QHBoxLayout

35.1 Horizontal Layout

The following example uses the horizontal layouts, to place controls without to specify any position. The placing and position is automatic. *(by Giovanni Di Maria)*




```
   // SOURCE=QHBoxLayout.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oW, oLay
   LOCAL oP0, oP1, oP2, oP3, oP4, oP5, oP6

   oW := QWidget()
   oW:setWindowTitle( "Finestra di Giovanni" )
   oW:resize( 400, 100 )

   oP0 := QPushButton()
   oP0:setText( "OK 1" )

   oP1 := QPushButton()
   oP1:setText( "OK 2" )

   oP2 := QPushButton()
   oP2:setText( "OK 3" )

   oP3 := QPushButton()
   oP3:setText( "OK 4" )

   oP4 := QPushButton()
   oP4:setText( "OK 5" )

   oP5 := QPushButton()
   oP5:setText( "OK 6" )

   oP6 := QPushButton()
   oP6:setText( "OK 7" )

   oLay := QHBoxLayout( oW )
   oLay:addWidget( oP0 )
```

```
    oLay:addWidget( oP1 )
    oLay:addWidget( oP2 )
    oLay:addWidget( oP3 )
    oLay:addWidget( oP4 )
    oLay:addWidget( oP5 )
    oLay:addWidget( oP6 )

    oW:show()
    QApplication():exec()

    RETURN

    // END SOURCE
```

36 QLineEdit

36.1 Line Edit

The following example uses a line edit, to insert and view any data. *(by Giovanni Di Maria)*


```
   // SOURCE=QLineEdit.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oName, oLabel

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 150 )

   oLabel := QLabel( oWnd )
   oLabel:setText( "Name" )
   oLabel:move( 50, 50 )
   oLabel:resize( 100, 20 )

   oName := QLineEdit( oWnd )
   oName:move( 160, 50 )
   oName:resize( 150, 20 )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

36.2 Array of Line Edit

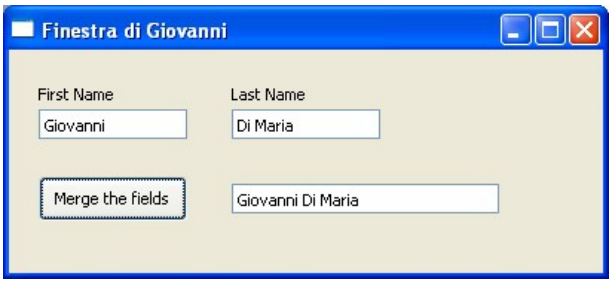
The following example uses many line edits, stored in an array, to insert and view data. *(by Giovanni Di Maria)*


```
   // SOURCE=QLineEdit.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oName[16], k

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 250 )

   for k = 1 TO 8
      oName[k] := QLineEdit( oWnd )
      oName[k]:move( 20, 22 * k )
      oName[k]:resize( 150, 20 )
   next k

   for k = 9 TO 16
      oName[k] := QLineEdit( oWnd )
      oName[k]:move( 180, 22 * ( k - 8 ) )
      oName[k]:resize( 150, 20 )
   next k

   oWnd:show()
   QApplication():exec()

   RETURN
```

```
   // END SOURCE
```

36.3 Merging two Line Edits

The following example uses two line edits, to insert and view data. By pressing the button, the values of the two Line Edits are merged and added to a third Line Edit. *(by Giovanni Di Maria)*




```
   // SOURCE=QLineEdit.prg

#include "hbqtgui.ch"

   STATIC oName, oSurname, oSum

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oLabel1, oLabel2
   LOCAL oButton

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 150 )

   oLabel1 := QLabel( oWnd )
   oLabel1:setText( "First Name" )
   oLabel1:move( 20, 20 )
   oLabel1:resize( 100, 20 )
   oName := QLineEdit( oWnd )
   oName:move( 20, 40 )
   oName:resize( 100, 20 )

   oLabel2 := QLabel( oWnd )
   oLabel2:setText( "Last Name" )
   oLabel2:move( 150, 20 )
   oLabel2:resize( 100, 20 )
   oSurname := QLineEdit( oWnd )
   oSurname:move( 150, 40 )
   oSurname:resize( 100, 20 )
```

```
   oButton := QPushButton( oWnd )
   oButton:move( 20, 85 )
   oButton:setText( "Merge the fields" )
   oButton:connect( "clicked()", { || merge() } )

   oSum := QLineEdit( oWnd )
   oSum:move( 150, 90 )
   oSum:resize( 180, 20 )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE merge()

   LOCAL st1, st2, st3

   st1 := oName:text()
   st2 := oSurname:text()
   st3 := st1 + Space( 1 ) + st2
   oSum:setText( st3 )

   RETURN

   // END SOURCE
```

36.4 Resetting Line Edits

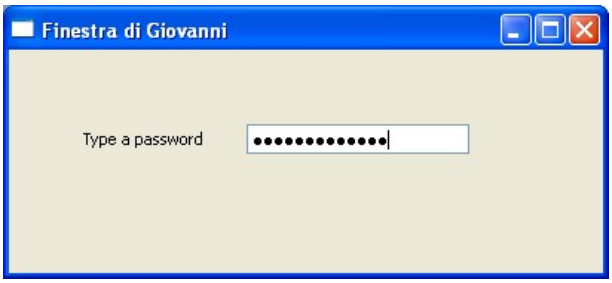
The following example shows how to reset and clear line edits. *(by Giovanni Di Maria)*


```
   // SOURCE=QLineEdit.prg

#include "hbqtgui.ch"

   STATIC oF1, oF2, oF3, oF4, oF5, oF6, oF7, oF8

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oButton

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 270, 300 )

   oF1 := QLineEdit( oWnd )
   oF1:move( 10, 10 )
   oF1:resize( 200, 20 )

   oF2 := QLineEdit( oWnd )
   oF2:move( 10, 40 )
   oF2:resize( 200, 20 )

   oF3 := QLineEdit( oWnd )
   oF3:move( 10, 70 )
   oF3:resize( 250, 20 )
```

```
   oF4 := QLineEdit( oWnd )
   oF4:move( 10, 100 )
   oF4:resize( 60, 20 )

   oF5 := QLineEdit( oWnd )
   oF5:move( 10, 130 )
   oF5:resize( 100, 20 )

   oF6 := QLineEdit( oWnd )
   oF6:move( 10, 160 )
   oF6:resize( 50, 20 )

   oF7 := QLineEdit( oWnd )
   oF7:move( 10, 190 )
   oF7:resize( 200, 20 )

   oF8 := QLineEdit( oWnd )
   oF8:move( 10, 220 )
   oF8:resize( 250, 20 )

   oButton := QPushButton( oWnd )
   oButton:move( 100, 250 )
   oButton:resize( 85, 40 )
   oButton:setText( "Clear Fields" )
   oButton:connect( "clicked()", { || clear_all() } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE clear_all()

   oF1:clear()      // Clear the field
   oF2:setText( "" )  // Clear the field
   oF3:clear()
   oF4:clear()
   oF5:clear()
   oF6:clear()
   oF7:clear()
   oF8:clear()

   RETURN

   // END SOURCE
```

36.5 Password

The following example shows how to insert a password into a line edit. The asterisks will be shown instead of the characters actually entered. *(by Giovanni Di Maria)*




```
   // SOURCE=QLineEdit.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oPasswd, oLabel

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 150 )

   oLabel := QLabel( oWnd )
   oLabel:setText( "Type a password" )
   oLabel:move( 50, 50 )
   oLabel:resize( 100, 20 )

   oPasswd := QLineEdit( oWnd )
   oPasswd:move( 160, 50 )
   oPasswd:resize( 150, 20 )
   oPasswd:setEchoMode( QLineEdit_Password )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

37 QTextEdit

37.1 Rich Text Editor

The following example shows a Rich Text editor and sets his text to bold, italic and size 20, color blue. *(by Giovanni Di Maria)*


```
   // SOURCE=QTextEdit.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oEditor, oFont

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 300 )

   oFont := Qfont()
   oFont:setBold( .T. )
   oFont:setItalic( .T. )
   oFont:setPointSize( 20 )

   oEditor := QTextEdit( oWnd )
   oEditor:resize( 300, 200 )
   oEditor:move( 50, 50 )
   oEditor:setTextColor( QColor( 0,0,200 ) )
   oEditor:setCurrentFont( oFont )

   oWnd:show()
   QApplication():exec()
```

```
   RETURN

   // END SOURCE
```

38 QEvent

38.1 Events List

Here is a list of all events. These defines are in the hbqtgui.ch. Please look at this file, for more informations. *(by Giovanni Di Maria)*

```
#define QEvent_None                                0
#define QEvent_Timer                               1
#define QEvent_MouseButtonPress                    2
#define QEvent_MouseButtonRelease                  3
#define QEvent_MouseButtonDblClick                 4
#define QEvent_MouseMove                           5
#define QEvent_KeyPress                            6
#define QEvent_KeyRelease                          7
#define QEvent_FocusIn                             8
#define QEvent_FocusOut                            9
#define QEvent_Enter                               10
#define QEvent_Leave                               11
#define QEvent_Paint                               12
#define QEvent_Move                                13
#define QEvent_Resize                              14
#define QEvent_Show                                17
#define QEvent_Hide                                18
#define QEvent_Close                               19
#define QEvent_ParentChange                        21
#define QEvent_WindowActivate                      24
#define QEvent_WindowDeactivate                    25
#define QEvent_ShowToParent                        26
#define QEvent_HideToParent                        27
#define QEvent_Wheel                               31
#define QEvent_WindowTitleChange                   33
#define QEvent_WindowIconChange                    34
#define QEvent_ApplicationWindowIconChange         35
#define QEvent_ApplicationFontChange               36
#define QEvent_ApplicationLayoutDirectionChange    37
#define QEvent_ApplicationPaletteChange            38
#define QEvent_PaletteChange                       39
#define QEvent_Clipboard                           40
#define QEvent_MetaCall                            43
#define QEvent_SockAct                             50
#define QEvent_ShortcutOverride                    51
#define QEvent_DeferredDelete                      52
#define QEvent_DragEnter                           60
#define QEvent_DragLeave                           62
#define QEvent_DragMove                            61
#define QEvent_Drop                                63
#define QEvent_ChildAdded                          68
#define QEvent_ChildPolished                       69
#define QEvent_ChildInserted                       70
#define QEvent_ChildRemoved                        71
```

```
#define QEvent_PolishRequest                    74
#define QEvent_Polish                           75
#define QEvent_LayoutRequest                    76
#define QEvent_UpdateRequest                    77
#define QEvent_UpdateLater                      78
#define QEvent_ContextMenu                      82
#define QEvent_InputMethod                      83
#define QEvent_AccessibilityPrepare             86
#define QEvent_TabletMove                       87
#define QEvent_LocaleChange                     88
#define QEvent_LanguageChange                   89
#define QEvent_LayoutDirectionChange            90
#define QEvent_TabletPress                      92
#define QEvent_TabletRelease                    93
#define QEvent_OkRequest                        94
#define QEvent_IconDrag                         96
#define QEvent_FontChange                       97
#define QEvent_EnabledChange                    98
#define QEvent_ActivationChange                 99
#define QEvent_StyleChange                      100
#define QEvent_IconTextChange                   101
#define QEvent_ModifiedChange                   102
#define QEvent_WindowBlocked                    103
#define QEvent_WindowUnblocked                  104
#define QEvent_WindowStateChange                105
#define QEvent_MouseTrackingChange              109
#define QEvent_ToolTip                          110
#define QEvent_WhatsThis                        111
#define QEvent_StatusTip                        112
#define QEvent_ActionChanged                    113
#define QEvent_ActionAdded                      114
#define QEvent_ActionRemoved                    115
#define QEvent_FileOpen                         116
#define QEvent_Shortcut                         117
#define QEvent_WhatsThisClicked                 118
#define QEvent_AccessibilityHelp                119
#define QEvent_ToolBarChange                    120
#define QEvent_ApplicationActivate              121
#define QEvent_ApplicationActivated             121
#define QEvent_ApplicationDeactivate            122
#define QEvent_QueryWhatsThis                   123
#define QEvent_EnterWhatsThisMode               124
#define QEvent_LeaveWhatsThisMode               125
#define QEvent_ZOrderChange                     126
#define QEvent_HoverEnter                       127
#define QEvent_HoverLeave                       128
#define QEvent_HoverMove                        129
#define QEvent_AccessibilityDescription         130
#define QEvent_ParentAboutToChange              131
#define QEvent_WinEventAct                      132
#define QEvent_EnterEditFocus                   150
#define QEvent_LeaveEditFocus                   151
#define QEvent_MenubarUpdated                   153
#define QEvent_GraphicsSceneMouseMove           155
```

```
#define QEvent_GraphicsSceneMousePress              156
#define QEvent_GraphicsSceneMouseRelease            157
#define QEvent_GraphicsSceneMouseDoubleClick        158
#define QEvent_GraphicsSceneContextMenu             159
#define QEvent_GraphicsSceneHoverEnter              160
#define QEvent_GraphicsSceneHoverMove               161
#define QEvent_GraphicsSceneHoverLeave              162
#define QEvent_GraphicsSceneHelp                    163
#define QEvent_GraphicsSceneDragEnter               164
#define QEvent_GraphicsSceneDragMove                165
#define QEvent_GraphicsSceneDragLeave               166
#define QEvent_GraphicsSceneDrop                    167
#define QEvent_GraphicsSceneWheel                   168
#define QEvent_KeyboardLayoutChange                 169
#define QEvent_DynamicPropertyChange                170
#define QEvent_TabletEnterProximity                 171
#define QEvent_TabletLeaveProximity                 172
#define QEvent_NonClientAreaMouseMove               173
#define QEvent_NonClientAreaMouseButtonPress        174
#define QEvent_NonClientAreaMouseButtonRelease      175
#define QEvent_NonClientAreaMouseButtonDblClick     176
#define QEvent_MacSizeChange                        177
#define QEvent_ContentsRectChange                   178
#define QEvent_GraphicsSceneResize                  181
#define QEvent_GraphicsSceneMove                    182
#define QEvent_CursorChange                         183
#define QEvent_ToolTipChange                        184
#define QEvent_GrabMouse                            186
#define QEvent_UngrabMouse                          187
#define QEvent_GrabKeyboard                         188
#define QEvent_UngrabKeyboard                       189
```

38.2 QEvent_Close

The following example shows how to intercept the Close Event. In this example, the close window button is ignored and you can exit from the window with the user button ("Quit") only . *(by Giovanni Di Maria)*


```
   // SOURCE=QEvent_Close.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd, oButton1

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 300, 200 )
   oWnd:connect( QEvent_Close , { |x| x:ignore()  } )

   oButton1 := QPushButton( oWnd )
   oButton1:setText( "Quit" )
   oButton1:move( 50, 50 )
   oButton1:Connect( "clicked()", { || QApplication():
      quit() } )

   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

38.3 QEvent_KeyPress

The following example shows how to capture a key pressed. The even can capture also Shift, Alt and Ctr keys. *(by Giovanni Di Maria)*


```
   // SOURCE=QEvent_KeyPress.prg

#include "hbqtgui.ch"

   STATIC oWnd
   STATIC oLabel2

PROCEDURE Main()

   LOCAL oLabel

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 240, 200 )
   oWnd:connect( QEvent_KeyPress , { |k| keypressed( k )
        } )

   oLabel := QLabel( oWnd )
   oLabel:setText( "Please press a key..." )
   oLabel:move( 10, 10 )

   oLabel2 := QLabel( oWnd )
```

```
   oLabel2:move( 10, 100 )
   oLabel2:resize( 220, 50 )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE keypressed( x )

   LOCAL string

   string = "You have pressed the key: " + "<br><br>"
   string = string + "VALUE= " + Str( x:key() ) + "<br>"
   string = string + "KEY= " + Chr( x:key() )
   oLabel2:setText( string )

   RETURN

   // END SOURCE
```

38.4 QEvent_KeyPress

The following example shows how to move a window by pressing the arrow keys. *(by Giovanni Di Maria)*


```
   // SOURCE=QEvent_KeyPress.prg

#include "hbqtgui.ch"

   STATIC oWnd

PROCEDURE Main()

   LOCAL oLabel

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 240, 200 )
   oWnd:connect( QEvent_KeyPress , { |k| keypressed( k )
       } )

   oLabel := QLabel( oWnd )
   oLabel:setText( "Press Arrow Keys to move the window
      ..." )
   oLabel:resize( 220, 20 )
   oLabel:move( 10, 10 )
```

```
   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE keypressed( kk )

   DO CASE
   CASE kk:key() = Qt_Key_Up
      oWnd:move( oWnd:x() , oWnd:y() - 4 )
   CASE kk:key() = Qt_Key_Down
      oWnd:move( oWnd:x() , oWnd:y() + 4 )
   CASE kk:key() = Qt_Key_Left
      oWnd:move( oWnd:x() - 4 , oWnd:y() )
   CASE kk:key() = Qt_Key_Right
      oWnd:move( oWnd:x() + 4 , oWnd:y() )
   ENDCASE

   RETURN

   // END SOURCE
```

38.5 QEvent_Move

The following example shows how to intercept the event of moving the window. *(by Giovanni Di Maria)*


```
   // SOURCE=QEvent_Move.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 180 )
   oWnd:connect( QEvent_Move , { || Message()  } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE Message()

   LOCAL oMB

   oMB := QMessageBox()
   oMB:setInformativeText( "You have moved the window" )
   oMB:setWindowTitle( "Harbour-QT" )
   oMB:exec()
   oMB := NIL

   RETURN

   // END SOURCE
```

39 QPainter

39.1 Drawing a face

The following example shows how to draw a face with primitives, like circle and line. This source uses the class QPainter. *(by Giovanni Di Maria)*


```
   // SOURCE=QPainter.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oPixmap
   LOCAL oPainter
   LOCAL oLabel

   oWnd := QWidget()
   oWnd:resize( 300, 200 )
   oWnd:setWindowTitle( "Finestra di Giovanni" )

   oPixmap := QPixmap( 100, 100 )
   oPixmap:fill( QColor( 255,255,150 ) )

   oPainter := QPainter( oPixmap )
   oPainter:drawEllipse( QPointF( 50,50 ), 40, 40 ) //
      Face
   oPainter:drawEllipse( QPointF( 40,30 ), 5, 5 )   //
      Eye
   oPainter:drawEllipse( QPointF( 40,30 ), 1, 1 )   //
      Eye center
   oPainter:drawEllipse( QPointF( 60,30 ), 5, 5 )   //
      Eye
```

```
   oPainter:drawEllipse( QPointF( 60,30 ), 1, 1 )   //
      Eye center
   oPainter:drawEllipse( QPointF( 50,50 ), 5, 15 )  //
      Nose
   oPainter:drawLine( 30, 70, 70, 75 )             //
      Mouth
   oPainter:end()

   oLabel := QLabel( oWnd )
   oLabel:setPixmap( oPixmap )
   oLabel:move( 100, 20 )
   oWnd:show()
   QApplication():exec()

   RETURN

   // END SOURCE
```

39.2 Moving a face

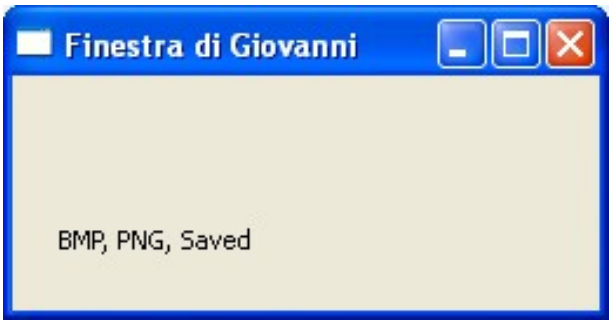
The following example shows how to draw and move a face with buttons. This source uses the class QPainter. *(by Giovanni Di Maria)*

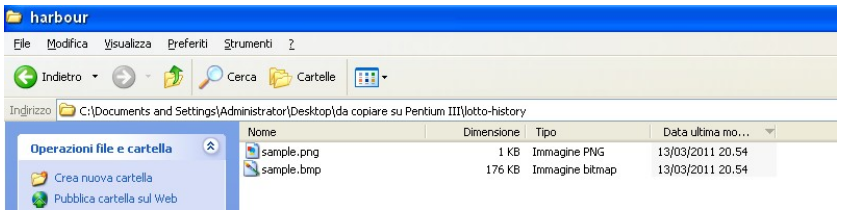

```
   // SOURCE=QPainter.prg

#include "hbqtgui.ch"

   STATIC oLabel

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oPixmap
   LOCAL oPainter
   LOCAL oButtonLeft, oButtonRight

   oWnd := QWidget()
   oWnd:resize( 300, 200 )
   oWnd:setWindowTitle( "Finestra di Giovanni" )

   oPixmap := QPixmap( 100, 100 )
   oPixmap:fill( QColor( 255,255,150 ) )

   oPainter := QPainter( oPixmap )
   oPainter:drawEllipse( QPointF( 50,50 ), 40, 40 ) //
      Face
   oPainter:drawEllipse( QPointF( 40,30 ), 5, 5 )   //
      Eye
   oPainter:drawEllipse( QPointF( 40,30 ), 1, 1 )   //
      Eye center
```

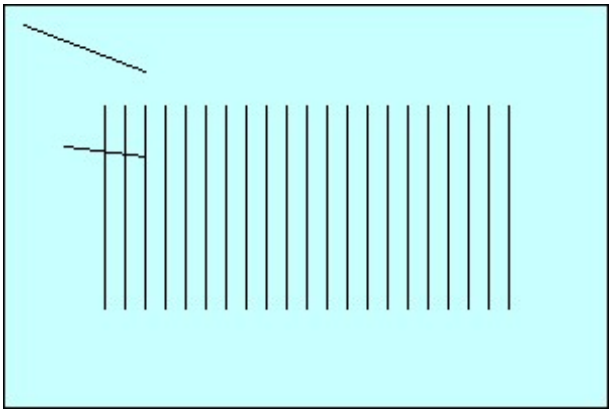
```
   oPainter:drawEllipse( QPointF( 60,30 ), 5, 5 )   //
      Eye
   oPainter:drawEllipse( QPointF( 60,30 ), 1, 1 )   //
      Eye center
   oPainter:drawEllipse( QPointF( 50,50 ), 5, 15 )  //
      Nose
   oPainter:drawLine( 30, 70, 70, 75 )             //
      Mouth
   oPainter:end()

   oLabel := QLabel( oWnd )
   oLabel:setPixmap( oPixmap )
   oLabel:move( 100, 20 )

   oButtonLeft := QPushButton( oWnd )
   oButtonLeft:setText( "Left" )
   oButtonLeft:move( 30, 160 )
   oButtonLeft:Connect( "clicked()", { || MoveLeft() } )

   oButtonRight := QPushButton( oWnd )
   oButtonRight:setText( "Right" )
   oButtonRight:move( 200, 160 )
   oButtonRight:Connect( "clicked()", { || MoveRight() }
       )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE MoveLeft()

   oLabel:move( oLabel:x() - 5, 20 )

   RETURN

PROCEDURE MoveRight()

   oLabel:move( oLabel:x() + 5, 20 )

   RETURN

   // END SOURCE
```

39.3 Saving a picture

The following example shows how to save a picture to BMP and PNG files. The picture is not visible on the screen. *(by Giovanni Di Maria)*







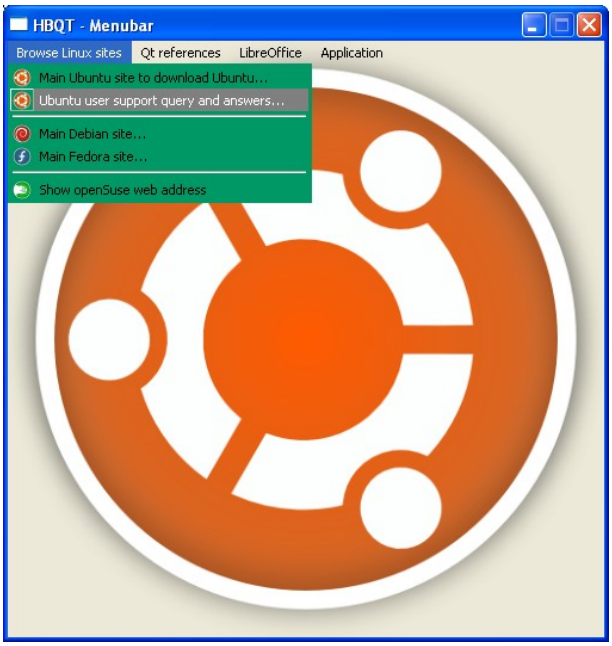
```
   // SOURCE=QPainter.prg

#include "hbqtgui.ch"

PROCEDURE Main()
```

```
LOCAL oWnd
LOCAL oPixmap
LOCAL oPainter
LOCAL k, oLabel

oWnd := QWidget()
oWnd:resize( 250, 100 )
oWnd:setWindowTitle( "Finestra di Giovanni" )

oPixmap := QPixmap( 300, 200 )
oPixmap:fill( QColor( 200,255,255 ) )

oPainter := QPainter( oPixmap )
oPainter:drawLine( 30, 70, 70, 75 )
oPainter:drawLine( 10, 10, 70, 33 )
for k = 50 TO 250 STEP 10
   oPainter:drawLine( k, 50, k, 150 )
next k

oPixmap:save( "../res/sample.bmp" )
oPixmap:save( "../res/sample.png" )

oLabel := QLabel( oWnd )
oLabel:setText( "BMP, PNG, Saved" )
oLabel:move( 20, 20 )
oLabel:resize( 200, 100 )

oWnd:show()
QApplication():exec()

RETURN

// END SOURCE
```

40 QDesktopServices

40.1 Web Browsing

The following example shows how to open a URL of a Web page. *(by Marco Braida)*




```
      // SOURCE=QDesktopServices.prg


#define K_MENU_STYLE "border-image: url(./no-image.png);
     no-repeat; color: black; selection-color: white;
    selection-background-color:gray; background-color
    :#009966"

PROCEDURE Main ()

   LOCAL oWin
   LOCAL oMenuBar, oM1, oM2, oM2a, oM3
   LOCAL oIt1_1, oIt1_2, oIt1_3, oIt1_4, oIt1_5
   LOCAL oIt2_1, oIt2_2, oIt2_3, oIt2_4
   LOCAL oIt3_1
   LOCAL oqDS
   LOCAL oQproc

   LOCAL img_path
```

```
img_path := hb_dirBase() + "../res/"

// see http://thesmithfam.org/blog/2009/09/10/qt-
    stylesheets-tutorial/

oWin := QMainWindow()

oWin:setStyleSheet( "border-image: url(../res/
    ubuntu_menu.png) no-repeat;" )

oWin:setWindowTitle( "HBQT - Menubar" )
oWin:SetFixedSize( 600, 600 )



oMenuBar := QMenuBar( oWin )     // create the menu
    bar with window as parent
oMenuBar:setStyleSheet( "border-image: url(./no-image
    .png) no-repeat;" )
oMenuBar:resize( 700, 22 )

oqDS := QDesktopServices( oWin )     // create a
    Desktop Service object

oQproc := qProcess( owin )

oM1 := QMenu( "&Browse Linux sites", oWin )   //
    build a vertical menu

oIt1_1 := oM1:addAction( img_path + "ubuntu_menu.png
    ", "Main &Ubuntu site to download Ubuntu..." )
oIt1_1:connect( "triggered(bool)", { || oqDS:openUrl(
     Qurl('http://www.ubuntu.com/' ) ) } )

oIt1_2 := oM1:addAction( img_path + "ubuntu_menu.png
    ", "Ubuntu user support &query and answers..." )
oIt1_2:connect( "triggered(bool)", { || oqDS:openUrl(
     Qurl('https://answers.launchpad.net/ubuntu/' ) )
    } )

oM1:addSeparator()

oIt1_3 := oM1:addAction( img_path + "debian_menu.png
    ", "Main &Debian site..." )
oIt1_3:connect( "triggered(bool)", { || oqDS:openUrl(
     Qurl('http://debian.org/' ) ) } )

oIt1_4 := oM1:addAction( img_path + "fedora_menu.png
    ", "Main &Fedora site..." )
oIt1_4:connect( "triggered(bool)", { || oqDS:openUrl(
     Qurl('http://fedoraproject.org/' ) ) } )

oM1:addSeparator()
```

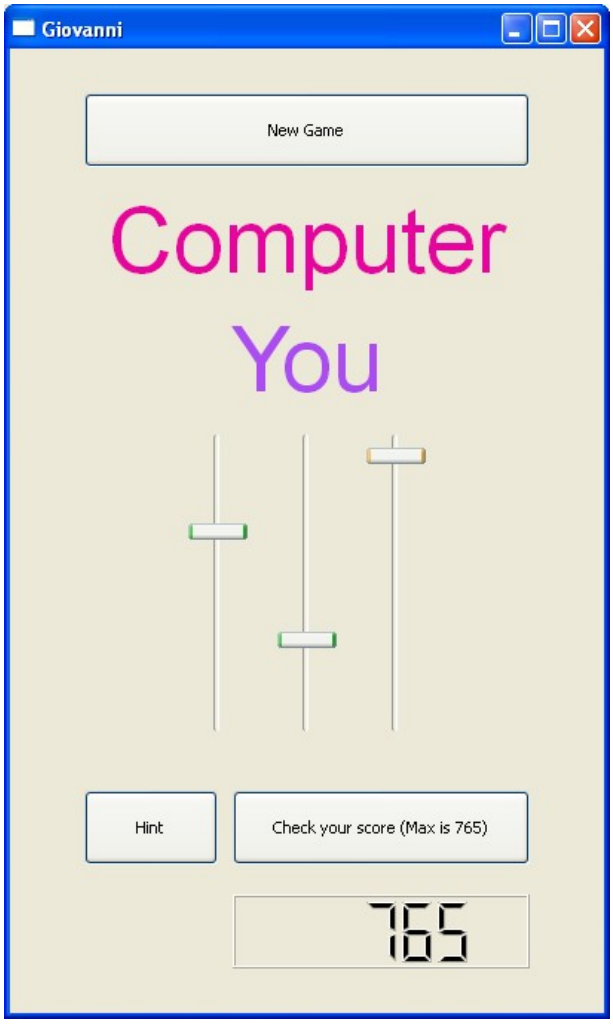
```
oIt1_5 := oM1:addAction( img_path +  "opensuse_menu.
    png", "Show &openSuse web address" )
oIt1_5:connect( "triggered(bool)", { ||showMsg( "http
    ://opensuse.org",oWin ) } )


oM2 := QMenu( "&Qt references", oWin )

oIt2_1 := oM2:addAction( img_path + "logo_qt.png", "
    Main Q&t site Qt reference doc..." )
oIt2_1:connect( "triggered(bool)", { || oqDS:openUrl(
     Qurl('http://doc.qt.nokia.com/' ) ) } )

oM2a := QMenu( "&LibreOffice", oWin )

oIt2_2 := oM2a:addAction( '' , "Open LibreOffice site
    ..." )
oIt2_2:connect( "triggered(bool)", { || oqDS:openUrl(
     Qurl('http://libreoffice.org/' ) ) } )

oIt2_3 := oM2a:addAction( '' , "&Run LibreOffice
    suite (if it is installed...)" )
oIt2_3:connect( "triggered(bool)", { || oqProc:start(
     'libreoffice' ) } )

oIt2_4 := oM2a:addAction( '' , "&Open with LO this
    app source code..." )
oIt2_4:connect( "triggered(bool)", { || oqProc:start(
     'libreoffice menubar.prg' ) } )

oM3 := QMenu( "&Application", oWin )

oIt3_1 := oM3:addAction( '' , "&Exit and Close" )
oIt3_1:connect( "triggered(bool)", { ||QApplication()
    :quit() } )


oM1:setStyleSheet( K_MENU_STYLE )
oMenuBar:addMenu( oM1 )

oM2:setStyleSheet( K_MENU_STYLE )
oMenuBar:addMenu( oM2 )

oM2a:setStyleSheet( K_MENU_STYLE )
oMenuBar:addMenu( oM2a )

oM3:setStyleSheet( K_MENU_STYLE )
oMenuBar:addMenu( oM3 )

oWin:show()
QApplication():exec()

RETURN
```

```
STATIC FUNCTION showMsg ( cMsg, oW )

   LOCAL oMBox

   oMBox := QMessageBox( oW )      // Create a message
      object window
   oMBox:setWindowTitle( "Message for you..." )
      // set the window title
   oMbox:setIcon( 1 )       // set the message type icon
   oMBox:setInformativeText( cMsg )     // set the
      message
   oMBox:exec()        // Execute the object in modal
      mode
   oMbox := nil

   RETURN NIL

   // END SOURCE
```

41 Games

41.1 Game of Colors

This is a game where the player must find the exact color randomly chosen by computer. The color is chosen by three slider of the RGB colors (red, green, blue). Player can get an hint from the computer. The maximum score is 765. Have fun. *(by Giovanni Di Maria)*




```
    // SOURCE=game-color.prg

#include "hbqtgui.ch"

   STATIC oPalette
```

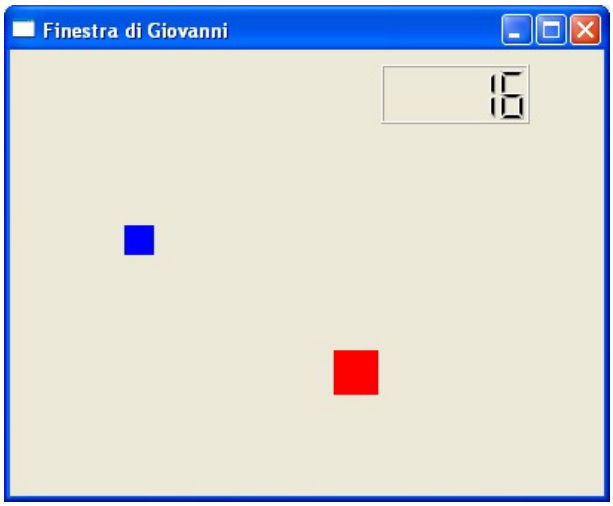
```
   STATIC oRed, oGreen, oBlu
   STATIC oTextMan, oTextPC
   STATIC nScore
   STATIC nRndRed, nRndGreen, nRndBlu
   STATIC oLcd

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oButtonStart, oButtonCheck, oButtonHint

   oWnd := QMainWindow()
   oWnd:resize( 400, 650 )
   oWnd:setWindowTitle( "Giovanni" )

   oButtonStart := QPushButton( oWnd )
   oButtonStart:setText( "New Game" )
   oButtonStart:move( 50, 30 )
   oButtonStart:resize( 300, 50 )
   oButtonStart:Connect( "clicked()", { || newgame() } )

   oTextPC := QLabel( oWnd )
   oTextPC:setText( "Computer" )
   oTextPC:move( 50, 80 )
   oTextPC:resize( 300, 100 )
   oTextPC:setFont( QFont( "Arial",46 ) )
   oTextPC:setAlignment( Qt_AlignVCenter +
      Qt_AlignHCenter )

   oTextMan := QLabel( oWnd )
   oTextMan:setText( "You" )
   oTextMan:move( 50, 160 )
   oTextMan:resize( 300, 100 )
   oTextMan:setFont( QFont( "Arial",46 ) )
   oTextMan:setAlignment( Qt_AlignVCenter +
      Qt_AlignHCenter )

   oRed := QSlider( oWnd )
   oRed:resize( 40, 200 )
   oRed:move( 120, 260 )
   oRed:setMinimum( 0 )
   oRed:setMaximum( 255 )
   oRed:setSingleStep( 1 )
   oRed:setPageStep( 10 )
   oRed:setValue( 0 )
   oRed:Connect( "valueChanged(int)", { || change_colors
      () } )

   oGreen := QSlider( oWnd )
   oGreen:resize( 40, 200 )
   oGreen:move( 180, 260 )
   oGreen:setMinimum( 0 )
   oGreen:setMaximum( 255 )
   oGreen:setSingleStep( 1 )
```

```
   oGreen:setPageStep( 10 )
   oGreen:setValue( 0 )
   oGreen:Connect( "valueChanged(int)", { ||
      change_colors() } )

   oBlu := QSlider( oWnd )
   oBlu:resize( 40, 200 )
   oBlu:move( 240, 260 )
   oBlu:setMinimum( 0 )
   oBlu:setMaximum( 255 )
   oBlu:setSingleStep( 1 )
   oBlu:setPageStep( 10 )
   oBlu:setValue( 0 )
   oBlu:Connect( "valueChanged(int)", { || change_colors
      () } )

   oButtonCheck := QPushButton( oWnd )
   oButtonCheck:setText( "Check your score (Max is 765)"
       )
   oButtonCheck:move( 150, 500 )
   oButtonCheck:resize( 200, 50 )
   oButtonCheck:Connect( "clicked()", { || score() } )

   oButtonHint := QPushButton( oWnd )
   oButtonHint:setText( "Hint" )
   oButtonHint:move( 50, 500 )
   oButtonHint:resize( 90, 50 )
   oButtonHint:Connect( "clicked()", { || hint() } )

   oLcd := QLCDNumber( oWnd )
   oLcd:move( 150, 570 )
   oLcd:resize( 200, 50 )

   newgame()
   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE change_colors()

   oPalette := QPalette()
   oPalette:SetColor( QPalette_WindowText, QColor( oRed:
      value() ,oGreen:value() , oBlu:value() ) )
   oTextMan:setPalette( oPalette )

   RETURN

PROCEDURE newgame()

   oRed:setValue( 0 )
   oGreen:setValue( 0 )
   oBlu:setValue( 0 )
   oLcd:display( 0 )
```

```
   nRndRed = HB_RandomInt( 0, 255 )
   nRndGreen = HB_RandomInt( 0, 255 )
   nRndBlu = HB_RandomInt( 0, 255 )

   oPalette := QPalette()
   oPalette:SetColor( QPalette_WindowText, QColor(
      nRndRed , nRndGreen , nRndBlu ) )
   oTextPC:setPalette( oPalette )

   RETURN

PROCEDURE score()

   nScore = Abs( nRndRed - oRed:value() ) + Abs(
      nRndGreen - oGreen:value() ) + Abs( nRndBlu - oBlu
      :value() )
   nScore = 765 - nScore
   oLcd:display( nScore )

   RETURN

PROCEDURE hint()

   oRed:setValue( nRndRed )
   oGreen:setValue( nRndGreen )
   oBlu:setValue( nRndBlu )

   RETURN

   // END SOURCE
```

41.2 Game of the Dog and the Cat (with boxes)

This is a nice game where the player is a dog (the red square). The dog must take the cat (the blue square). Every time that the dog takes the cat, the score increments its value by 1. The dog is moved by arrow keys. Have fun. *(by Giovanni Di Maria)*




```
    // SOURCE=game-dog-cat.prg

#include "hbqtgui.ch"

   STATIC oWnd
   STATIC oDog, oCat
   STATIC oScore

PROCEDURE Main()

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 300 )
   oWnd:connect( QEvent_KeyPress , { |k| keypressed( k )
       } )

   oCat := QLabel( oWnd )
   oCat:resize( 20, 20 )
   oCat:setStyleSheet( "background-color: #0000FF" )

   oDog := QLabel( oWnd )
   oDog:resize( 30, 30 )
   oDog:move( 150, 250 )
```

```
   oDog:setStyleSheet( "background-color: #FF0000" )

   oScore := QLCDNumber( oWnd )
   oScore:resize( 100, 40 )
   oScore:move( 250, 10 )
   oScore:display( 0 )

   place_cat()

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE keypressed( x )

   LOCAL nStep := 4
   LOCAL nXdiff, nYdiff

   //--------------------Keys----------------
   DO CASE
   CASE x:key() = Qt_Key_Left
      oDog:move( oDog:x() - nStep , oDog:y() )
   CASE x:key() = Qt_Key_Right
      oDog:move( oDog:x() + nStep , oDog:y() )
   CASE x:key() = Qt_Key_Up
      oDog:move( oDog:x() , oDog:y() - nStep )
   CASE x:key() = Qt_Key_Down
      oDog:move( oDog:x() , oDog:y() + nStep )
   ENDCASE

   //----------------Check for collision----------------
   nXdiff = Abs ( oDog:x() - oCat:x() )
   nYdiff = Abs ( oDog:y() - oCat:y() )
   IF nXdiff < 10 .AND. nYdiff < 10
      collision()
   ENDIF

   RETURN

PROCEDURE place_cat()

   LOCAL nXCat, nYCat

   nXCat = HB_RandomInt( 0, 380 )
   nYCat = HB_RandomInt( 60, 280 )
   oCat:move( nXCat, nYCat )

   RETURN

PROCEDURE collision()

   place_cat()
   oScore:display( oScore:value() + 1 )
```

```
   RETURN

   // END SOURCE
```

41.3 Game of the Dog and the Cat (with images)

This is a nice game where the player is a dog. The dog must take the cat. Every time that the dog takes the cat, the score increments its value by 1. The dog and the cat are real bitmap images. The dog is moved by arrow keys. Have fun. *(by Giovanni Di Maria)*


```
    // SOURCE=game-dog-cat.prg

#include "hbqtgui.ch"

   STATIC oWnd
   STATIC oDog, oCat
   STATIC oScore

PROCEDURE Main()

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 300 )
   oWnd:connect( QEvent_KeyPress , { |k| keypressed( k )
       } )

   oCat := QLabel( oWnd )
   oCat:resize( 30, 30 )
   oCat:SetPixmap( QPixmap( "../res/cat.bmp" ) )

   oDog := QLabel( oWnd )
   oDog:resize( 50, 50 )
   oDog:move( 150, 250 )
```

```
   oDog:SetPixmap( QPixmap( "../res/dog.bmp" ) )

   oScore := QLCDNumber( oWnd )
   oScore:resize( 100, 40 )
   oScore:move( 250, 10 )
   oScore:display( 0 )

   place_cat()

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE keypressed( x )

   LOCAL nStep := 4
   LOCAL nXdiff, nYdiff

   //--------------------Keys----------------
   DO CASE
   CASE x:key() = Qt_Key_Left
      oDog:move( oDog:x() - nStep , oDog:y() )
   CASE x:key() = Qt_Key_Right
      oDog:move( oDog:x() + nStep , oDog:y() )
   CASE x:key() = Qt_Key_Up
      oDog:move( oDog:x() , oDog:y() - nStep )
   CASE x:key() = Qt_Key_Down
      oDog:move( oDog:x() , oDog:y() + nStep )
   ENDCASE

   //----------------Check for collision----------------
   nXdiff = Abs ( oDog:x() - oCat:x() )
   nYdiff = Abs ( oDog:y() - oCat:y() )
   IF nXdiff < 15 .AND. nYdiff < 15
      collision()
   ENDIF

   RETURN

PROCEDURE place_cat()

   LOCAL nXCat, nYCat

   nXCat = HB_RandomInt( 0, 370 )
   nYCat = HB_RandomInt( 60, 270 )
   oCat:move( nXCat, nYCat )

   RETURN

PROCEDURE collision()

   place_cat()
   oScore:display( oScore:value() + 1 )
```

```
   RETURN

   // END SOURCE
```

41.4 Game of the Dog and the Cat (with images and sound)

This is a nice game where the player is a dog. The dog must take the cat. Every time that the dog takes the cat, the score increments its value by 1 and the cat meows. The dog and the cat are real bitmap images. The dog is moved by arrow keys. Have fun. *(by Giovanni Di Maria)*

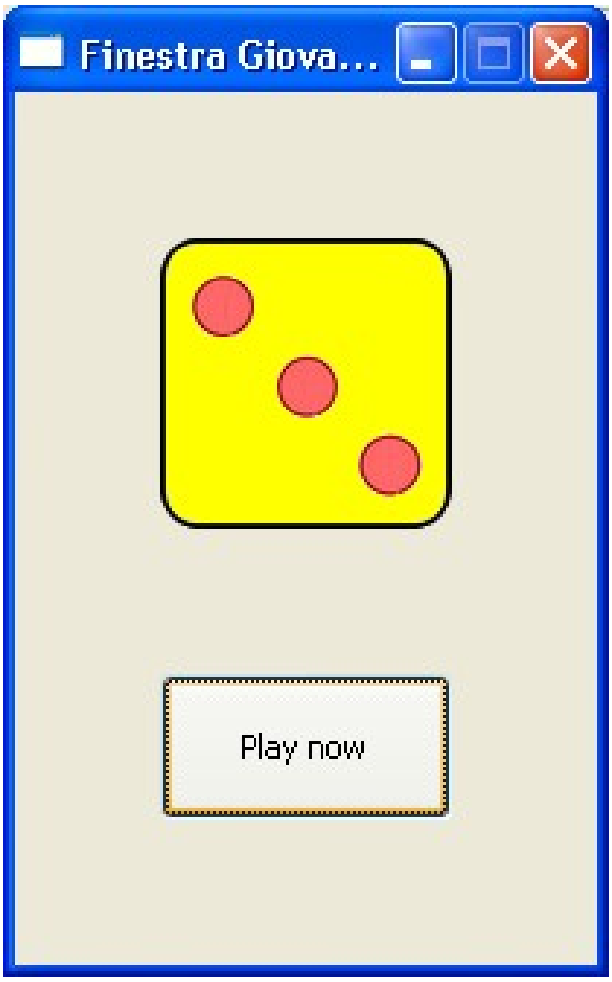

```
     // SOURCE=game-dog-cat.prg

#include "hbqtgui.ch"
   STATIC oWnd
   STATIC oDog, oCat
   STATIC oScore

PROCEDURE Main()

   LOCAL oLabel

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 300 )
   oWnd:connect( QEvent_KeyPress , { |k| keypressed( k )
       } )

   oCat := QLabel( oWnd )
   oCat:resize( 30, 30 )
   oCat:SetPixmap( QPixmap( "../res/cat.bmp" ) )

   oDog := QLabel( oWnd )
```

```
   oDog:resize( 50, 50 )
   oDog:move( 150, 250 )
   oDog:SetPixmap( QPixmap( "../res/dog.bmp" ) )

   oLabel := QLabel( oWnd )
   oLabel:setText( "<b>Game with SOUND</b>" )
   oLabel:move( 40, 20 )

   oScore := QLCDNumber( oWnd )
   oScore:resize( 100, 40 )
   oScore:move( 250, 10 )
   oScore:display( 0 )

   place_cat()
   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE keypressed( x )

   LOCAL nStep := 4
   LOCAL nXdiff, nYdiff

   //--------------------Keys----------------
   DO CASE
   CASE x:key() = Qt_Key_Left
      oDog:move( oDog:x() - nStep , oDog:y() )
   CASE x:key() = Qt_Key_Right
      oDog:move( oDog:x() + nStep , oDog:y() )
   CASE x:key() = Qt_Key_Up
      oDog:move( oDog:x() , oDog:y() - nStep )
   CASE x:key() = Qt_Key_Down
      oDog:move( oDog:x() , oDog:y() + nStep )
   ENDCASE

   //----------------Check for collision----------------
   nXdiff = Abs ( oDog:x() - oCat:x() )
   nYdiff = Abs ( oDog:y() - oCat:y() )
   IF nXdiff < 15 .AND. nYdiff < 15
      collision()
   ENDIF

   RETURN

PROCEDURE place_cat()

   LOCAL nXCat, nYCat

   nXCat = HB_RandomInt( 0, 370 )
   nYCat = HB_RandomInt( 60, 270 )
   oCat:move( nXCat, nYCat )

   RETURN
```

```
PROCEDURE collision()

   LOCAL oSound

   place_cat()
   oScore:display( oScore:value() + 1 )
   oSound := QSound()
   oSound:play( "../res/cat.wav" )

   RETURN

   // END SOURCE
```

41.5 Dice Game (1 die)

This is a nice game where the player play with one die. *(by Giovanni Di Maria)*

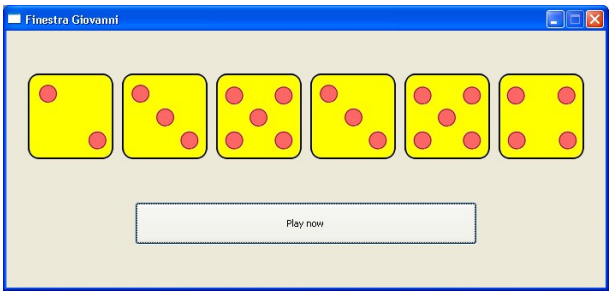



```
    // SOURCE=dice.prg
#include "hbqtgui.ch"
   STATIC oWnd
PROCEDURE Main()
   LOCAL oDice
```

```
   LOCAL oButton

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 200, 300 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oButton := QPushButton( oWnd )
   oButton:resize( 100, 50 )
   oButton:move( 50, 200 )
   oButton:setText( "Play now" )
   oButton:Connect( "clicked()", { ||dice( oDice )  } )

   oDice := QLabel( oWnd )
   oDice:move( 50, 50 )
   oDice:resize( 100, 100 )

   dice( oDice )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE dice( oD )

   LOCAL cFileName[6]
   LOCAL nRandom

   cFileName[1] = "../res/dice1.png"
   cFileName[2] = "../res/dice2.png"
   cFileName[3] = "../res/dice3.png"
   cFileName[4] = "../res/dice4.png"
   cFileName[5] = "../res/dice5.png"
   cFileName[6] = "../res/dice6.png"

   nRandom = HB_RandomInt( 1, 6 )
   oD:SetPixmap( QPixmap( cFileName[nRandom] ) )

   RETURN

   // END SOURCE
```

41.6 Dice Game (6 dice)

This is a nice game where the player play with six dice. The dice are stored in a vector. *(by Giovanni Di Maria)*



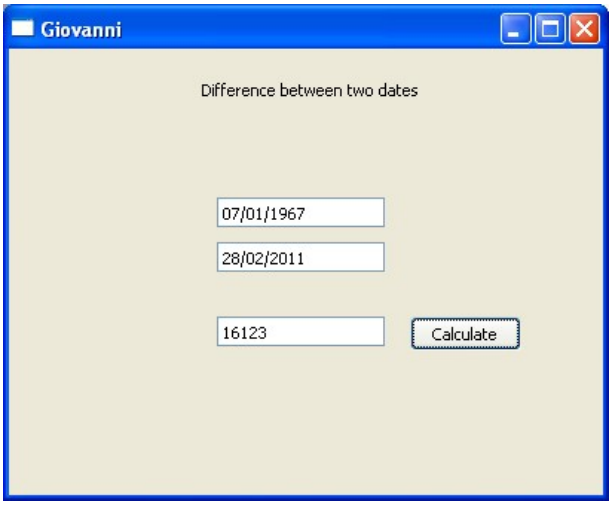

```
    // SOURCE=dice.prg

#include "hbqtgui.ch"

   STATIC oWnd

PROCEDURE Main()

   LOCAL oDice[6], k
   LOCAL oButton

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 700, 300 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oButton := QPushButton( oWnd )
   oButton:resize( 400, 50 )
   oButton:move( 150, 200 )
   oButton:setText( "Play now" )
   oButton:Connect( "clicked()", { ||dice( oDice )  } )

   FOR k = 1 TO 6
      oDice[k] := QLabel( oWnd )
      oDice[k]:move( k * 110 - 85, 50 )
      oDice[k]:resize( 100, 100 )
   NEXT k

   dice( oDice )

   oWnd:show()
   QApplication():exec()
```

```
   RETURN

PROCEDURE dice( oD )

   LOCAL k
   LOCAL cFileName[6]
   LOCAL nRandom

   cFileName[1] = "../res/dice1.png"
   cFileName[2] = "../res/dice2.png"
   cFileName[3] = "../res/dice3.png"
   cFileName[4] = "../res/dice4.png"
   cFileName[5] = "../res/dice5.png"
   cFileName[6] = "../res/dice6.png"

   FOR k = 1 TO 6
      nRandom = HB_RandomInt( 1, 6 )
      oD[k]:SetPixmap( QPixmap( cFileName[nRandom] ) )
   NEXT k

   RETURN

   // END SOURCE
```

42 Sample Applications

42.1 Difference between two dates

The following example calculates the difference betwee two dates. The user must type two dates and then press the Calculate button. The program shows the difference, in days. *(by Giovanni Di Maria)*

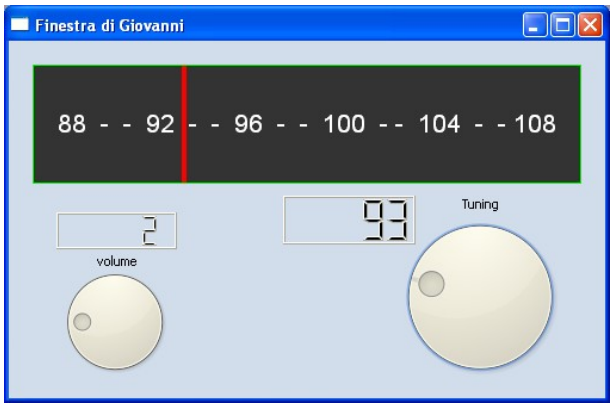



```
    // SOURCE=difference-dates.prg

#include "hbqtgui.ch"

   STATIC oEdit1, oEdit2, oEdit3

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oText
   LOCAL oButtonCalculate

   SET DATE ITALIAN

   oWnd := QMainWindow()
   oWnd:resize( 400, 300 )
   oWnd:setWindowTitle( "Giovanni" )

   oText := QLabel( oWnd )
   oText:setText( "Difference between two dates" )
   oText:move( 130, 20 )
   oText:resize( 171, 16 )

   oEdit1 := QLineEdit( oWnd )
```

```
   oEdit1:resize( 113, 20 )
   oEdit1:move( 140, 100 )

   oEdit2 := QLineEdit( oWnd )
   oEdit2:resize( 113, 20 )
   oEdit2:move( 140, 130 )

   oEdit3 := QLineEdit( oWnd )
   oEdit3:resize( 113, 20 )
   oEdit3:move( 140, 180 )

   oButtonCalculate := QPushButton( oWnd )
   oButtonCalculate:resize( 75, 23 )
   oButtonCalculate:move( 270, 180 )
   oButtonCalculate:setText( "Calculate" )
   oButtonCalculate:Connect( "clicked()", { || calculate
      () } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE calculate()

   LOCAL nDifference

   nDifference = Abs( CToD( oEdit1:text() ) - CToD(
      oEdit2:text() ) )
   oEdit3:setText( AllTrim( Str(nDifference ) ) )

   RETURN

   // END SOURCE
```

42.2 Radio Simulation (Visual only)

The following example simulates a simple radio receiver. It's visual only, of course. I use labels to draw the window tuning and the cursor of sintony. *(by Giovanni Di Maria)*



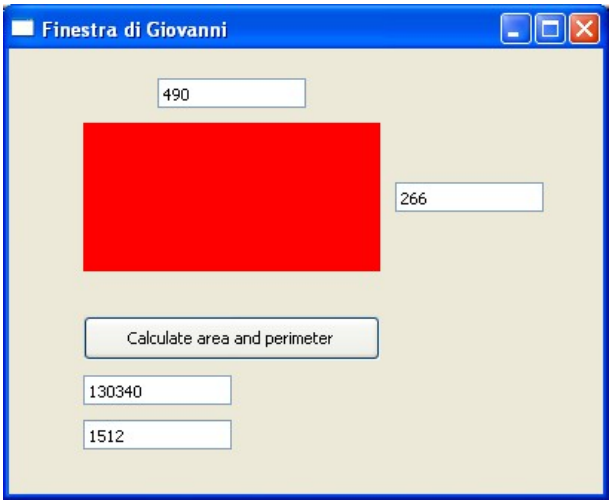

```
    // SOURCE=radio.prg

#include "hbqtgui.ch"

   STATIC oDisplayTuning, oWheelTuning, oCursor

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oTuning
   LOCAL oWheelVolume
   LOCAL oPalette
   LOCAL oLabelTuning, oLabelVolume
   LOCAL oDisplayVolume

   oPalette := QPalette()
   oPalette:SetColor( QPalette_Window, QColor(
      210,221,236 ) )

   oWnd := QMainWindow()
   oWnd:resize( 500, 300 )
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:setPalette( oPalette )


   // ---------------TUNING PANEL--------------
   oTuning := QLabel( oWnd )
```

```
   oTuning:resize( 460, 100 )
   oTuning:move( 20, 20 )
   oTuning:setStyleSheet( "background-color: #333333;
      color: #FFFFFF; border: 1px solid #00FF00;" )
   oTuning:setText( "88  -  -  92  -  -  96  -  -  100
      - -  104  -  - 108" )
   oTuning:setAlignment( Qt_AlignHCenter +
      Qt_AlignVCenter )
   oTuning:setFont( QFont( "Arial",16 ) )

   // ---------------TUNING--------------
   oWheelTuning := QDial( oWnd )
   oWheelTuning:move( 320, 140 )
   oWheelTuning:resize( 150, 150 )
   oWheelTuning:setMinimum( 88 )
   oWheelTuning:setMaximum( 108 )
   oWheelTuning:setSingleStep( 1 )
   oWheelTuning:setWrapping( .F. )
   oWheelTuning:Connect( "valueChanged(int)", { ||
      tuning_change() } )

   oLabelTuning := QLabel( oWnd )
   oLabelTuning:resize( 46, 13 )
   oLabelTuning:move( 380, 130 )
   oLabelTuning:setPalette( oPalette )
   oLabelTuning:setText( "Tuning" )

   oDisplayTuning := QLCDNumber( oWnd )
   oDisplayTuning:resize( 111, 41 )
   oDisplayTuning:move( 230, 130 )
   oDisplayTuning:display( 88 )

   // ---------------VOLUME--------------
   oWheelVolume := QDial( oWnd )
   oWheelVolume:move( 39, 186 )
   oWheelVolume:resize( 100, 100 )
   oWheelVolume:setMinimum( 0 )
   oWheelVolume:setMaximum( 10 )
   oWheelVolume:setSingleStep( 1 )
   oWheelVolume:setWrapping( .F. )
   oWheelVolume:Connect( "valueChanged(int)", { ||
      oDisplayVolume:display( oWheelVolume:value() ) } )

   oLabelVolume := QLabel( oWnd )
   oLabelVolume:resize( 46, 13 )
   oLabelVolume:move( 74, 178 )
   oLabelVolume:setPalette( oPalette )
   oLabelVolume:setText( "volume" )

   oDisplayVolume := QLCDNumber( oWnd )
   oDisplayVolume:resize( 101, 31 )
   oDisplayVolume:move( 40, 144 )
```

```
   // ---------------CURSOR--------------
   oCursor := QLabel( oWnd )
   oCursor:resize( 4, 98 )
   oCursor:move( 45, 21 )
   oCursor:setStyleSheet( "background-color: #FF0000" )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE tuning_change()

   LOCAL oPosition

   oPosition := oWheelTuning:value() * 20 - 1715
   oDisplayTuning:display( oWheelTuning:value() )
   oCursor:move( oPosition, 21 )

   RETURN

   // END SOURCE
```

42.3 Area and perimeter of a rectangle

The following example calculates the area and the perimeter of a rectangle. Input data are typed by user. *(by Giovanni Di Maria)*




```
    // SOURCE=rectangle.prg

#include "hbqtgui.ch"

   STATIC oWidth, oHeigth
   STATIC oArea, oPerimeter

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oRectangle
   LOCAL oButton

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 300 )

   oRectangle := QLabel( oWnd )
   oRectangle:resize( 200, 100 )
   oRectangle:move( 50, 50 )
   oRectangle:setStyleSheet( "background-color : red;" )

   oWidth := QLineEdit( oWnd )
   oWidth:move( 100, 20 )
   oWidth:resize( 100, 20 )
```

```
   oHeigth := QLineEdit( oWnd )
   oHeigth:move( 260, 90 )
   oHeigth:resize( 100, 20 )

   oButton := QPushButton( oWnd )
   oButton:move( 50, 180 )
   oButton:resize( 200, 30 )
   oButton:setText( "Calculate area and perimeter" )
   oButton:connect( "clicked()", { || formulas() } )

   oArea := QLineEdit( oWnd )
   oArea:move( 50, 220 )
   oArea:resize( 100, 20 )

   oPerimeter := QLineEdit( oWnd )
   oPerimeter:move( 50, 250 )
   oPerimeter:resize( 100, 20 )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE formulas()

   LOCAL nArea, nPerimeter

   nArea := Val( oWidth:text() ) * Val( oHeigth:text() )
   nPerimeter := ( Val( oWidth:text() ) + Val( oHeigth:
      text() ) ) * 2
   oArea:setText( AllTrim( Str(nArea ) ) )
   oPerimeter:setText( AllTrim( Str(nPerimeter ) ) )

   RETURN

   // END SOURCE
```

42.4 Alarm

The following example shows how to create a simple programmable alarm. *(by Giovanni Di Maria)*

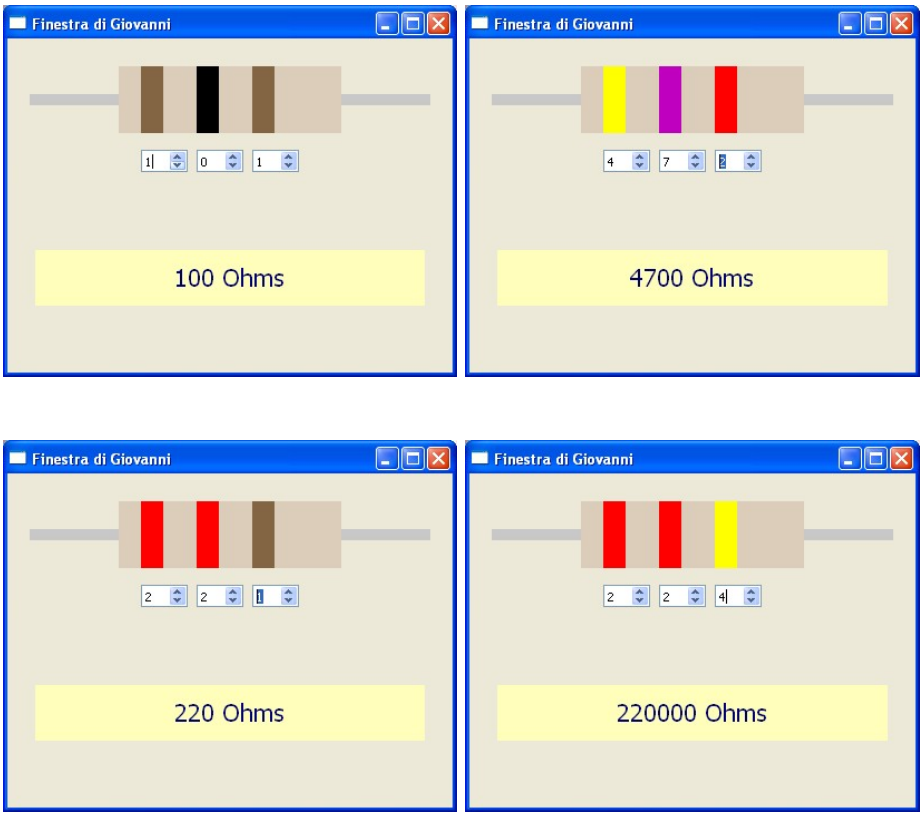

```
    // SOURCE=alarm.prg

#include "hbqtgui.ch"

   STATIC oText
   STATIC oSetting

PROCEDURE Main()
```

```
   LOCAL oWnd
   LOCAL oClock
   LOCAL oText2

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 250, 150 )

   oText := Qlabel( oWnd )
   oText:setText( "clocking..." )
   oText:move( 100, 100 )

   oSetting := QTimeEdit( oWnd )
   oSetting:move( 100, 50 )

   oText2 := Qlabel( oWnd )
   oText2:setText( "Alarm at" )
   oText2:move( 40, 50 )

   oClock := QTimer()
   oClock:Connect( "timeout()", { || tick() } )
   oClock:start( 1000 )

   oWnd:show()
   QApplication():exec()
   oClock:stop()

   RETURN

PROCEDURE tick()

   LOCAL oSetTime, nSetHou, nSetMin, nSetSec
   LOCAL cCurTime, nCurHou, nCurMin, nCurSec
   LOCAL oBox

   oText:setText( Time() )

   oSetTime = oSetting:Time()
   nSetHou = oSetTime:hour()
   nSetMin = oSetTime:minute()
   nSetSec = oSetTime:second()

   cCurTime = Time()
   nCurHou = Val( Left( cCurTime,2 ) )
   nCurMin = Val( SubStr( cCurTime,4,2 ) )
   nCurSec = Val( Right( cCurTime,2 ) )

   IF nSetHou = nCurHou .AND. nSetMin = nCurMin .AND.
       nSetSec = nCurSec
      oBox := QMessageBox()
      oBox:setWindowTitle( "Alarm" )
      oBox:setInformativeText( "It's " + cCurTime + " o'
          clock" )
```

```
      oBox:exec()
   ENDIF

   RETURN

   // END SOURCE
```

42.5 Resistors

The following example shows a program to calculate the value of a resistor. It show also the colors of its bars. *(by Giovanni Di Maria)*




```
    // SOURCE=resistor.prg

#include "hbqtgui.ch"

   STATIC oBand1, oBand2, oBand3
   STATIC oChoose1, oChoose2, oChoose3
   STATIC oValue

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oTerminals, oBody

   oWnd := QMainWindow()
```

```
    oWnd:setWindowTitle( "Finestra di Giovanni" )
    oWnd:resize( 400, 300 )

    //----- Draw Terminals-------------
    oTerminals := QLabel( oWnd )
    oTerminals:resize( 360, 10 )
    oTerminals:move( 20, 50 )
    oTerminals:setStyleSheet( "background-color : #C8C8C8
        ;" )

    //----- Draw body of resistance -------------
    oBody := QLabel( oWnd )
    oBody:resize( 200, 60 )
    oBody:move( 100, 25 )
    oBody:setStyleSheet( "background-color : #DCCCBA;" )

    //----- Draw BAND 1-------------
    oBand1 := QLabel( oWnd )
    oBand1:resize( 20, 60 )
    oBand1:move( 120, 25 )
    oBand1:setStyleSheet( "background-color : #000000;" )

    oChoose1 := QSpinBox( oWnd )
    oChoose1:move( 120, 100 )
    oChoose1:resize( 42, 20 )
    oChoose1:setMinimum( 0 )
    oChoose1:setMaximum( 9 )
    oChoose1:Connect( "valueChanged(int)", { ||resistor()
        } )

    //----- Draw BAND 2-------------
    oBand2 := QLabel( oWnd )
    oBand2:resize( 20, 60 )
    oBand2:move( 170, 25 )
    oBand2:setStyleSheet( "background-color : #000000;" )

    oChoose2 := QSpinBox( oWnd )
    oChoose2:move( 170, 100 )
    oChoose2:resize( 42, 20 )
    oChoose2:setMinimum( 0 )
    oChoose2:setMaximum( 9 )
    oChoose2:Connect( "valueChanged(int)", { ||resistor()
        } )

    //----- Draw BAND 3-------------
    oBand3 := QLabel( oWnd )
    oBand3:resize( 20, 60 )
    oBand3:move( 220, 25 )
    oBand3:setStyleSheet( "background-color : #000000;" )

    oChoose3 := QSpinBox( oWnd )
    oChoose3:move( 220, 100 )
    oChoose3:resize( 42, 20 )
    oChoose3:setMinimum( 0 )
```

```
   oChoose3:setMaximum( 9 )
   oChoose3:Connect( "valueChanged(int)", { ||resistor()
       } )

   //----- Resistor value-------------
   oValue := QLabel( oWnd )
   oValue:resize( 350, 50 )
   oValue:move( 25, 190 )
   oValue:setStyleSheet( "background-color : #FFFFBB;" )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE resistor()

   LOCAL cColors[10]
   LOCAL nC1, nC2, nC3
   LOCAL v

   cColors[1]  := "#000000"
   cColors[2]  := "#836543"
   cColors[3]  := "#FF0000"
   cColors[4]  := "#FF9146"
   cColors[5]  := "#FFFF00"
   cColors[6]  := "#00FF00"
   cColors[7]  := "#0000FF"
   cColors[8]  := "#BF00BF"
   cColors[9]  := "#C0C0C0"
   cColors[10] := "#FFFFFF"

   nC1 = oChoose1:value()
   nC2 = oChoose2:value()
   nC3 = oChoose3:value()

   oBand1:setStyleSheet( "background-color : " + cColors
       [nC1+1] + ";" )
   oBand2:setStyleSheet( "background-color : " + cColors
       [nC2+1] + ";" )
   oBand3:setStyleSheet( "background-color : " + cColors
       [nC3+1] + ";" )

   v = Val( AllTrim( Str(nC1 ) ) + AllTrim( Str(nC2 ) )
       )
   v = Int( v * 10 ^ nC3 )
   v = AllTrim( Str( v ) ) + " Ohms"
   v = "<font color=#000077 size=6>" + v + "</font>"
   v = "<center>" + v + "</center>"

   oValue:setText( v )

   RETURN
```

```
    // END SOURCE
```

42.6 Aquarius

The following example shows how to create a virtual aquarius with fishes. The fishes move from side to side.The fishes are in .PNG format image, with transparency. *(by Giovanni Di Maria)*

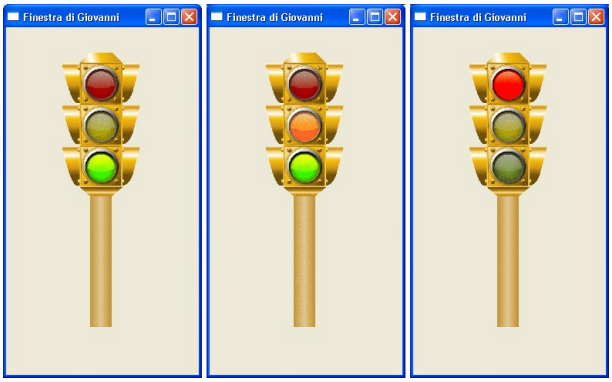

```
    // SOURCE=aquarius.prg

#include "hbqtgui.ch"


   STATIC oFish1, oFish2, oFish3, oFish4, oFish5

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oSea
   LOCAL oClock

   oWnd := QmainWindow()
   oWnd:SetFixedSize( 600, 400 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oClock := QTimer()
   oClock:Connect( "timeout()", { || moveFish() } )
   oClock:start( 50 )

   oSea := QLabel( oWnd )
   oSea:move( 0, 0 )
   oSea:resize( 600, 400 )
   oSea:SetPixmap( QPixmap( "../res/aquarium1.bmp" ) )
```

```
   oFish1 := QLabel( oWnd )
   oFish1:move( 300, 110 )
   oFish1:resize( 180, 180 )
   oFish1:SetPixmap( QPixmap( "../res/fish1.png" ) )

   oFish2 := QLabel( oWnd )
   oFish2:move( 200, 40 )
   oFish2:resize( 150, 150 )
   oFish2:SetPixmap( QPixmap( "../res/fish2.png" ) )

   oFish3 := QLabel( oWnd )
   oFish3:move( 100, 320 )
   oFish3:resize( 100, 50 )
   oFish3:SetPixmap( QPixmap( "../res/fish3.png" ) )

   oFish4 := QLabel( oWnd )
   oFish4:move( 300, 250 )
   oFish4:resize( 150, 70 )
   oFish4:SetPixmap( QPixmap( "../res/fish4.png" ) )

   oFish5 := QLabel( oWnd )
   oFish5:move( 50, 10 )
   oFish5:resize( 150, 70 )
   oFish5:SetPixmap( QPixmap( "../res/fish5.png" ) )

   oWnd:show()
   QApplication():exec()
   oClock:stop()

   RETURN

PROCEDURE moveFish()

   oFish1:move( oFish1:x - 1, oFish1:y )
   IF oFish1:x <- 180
      oFish1:move( 600, oFish1:y )
   end IF

   oFish2:move( oFish2:x - 1.5, oFish2:y )
   IF oFish2:x <- 150
      oFish2:move( 600, oFish2:y )
   end IF

   oFish3:move( oFish3:x - 2, oFish3:y )
   IF oFish3:x <- 100
      oFish3:move( 600, oFish3:y )
   end IF

   oFish4:move( oFish4:x + 1.3 , oFish4:y )
   IF oFish4:x > 600
      oFish4:move( - 150, oFish4:y )
   end IF

   oFish5:move( oFish5:x + 1.5 , oFish5:y )
```

```
   IF oFish5:x > 600
      oFish5:move( - 150, oFish5:y )
   end IF

   RETURN

   // END SOURCE
```

42.7 Semaphore (automatic)

The following example shows how to create an automatic semaphore. It changes its color every a second. *(by Giovanni Di Maria)*




```
    // SOURCE=semaphore.prg

#include "hbqtgui.ch"

   STATIC SEQUENCE

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oImg
   LOCAL oClock
```

```
   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 250, 450 )

   oImg := QLabel( oWnd )
   oImg:move( 25, 25 )
   oImg:resize( 200, 400 )

   sequence = 1
   semaphore( oImg )

   oClock := QTimer()
   oClock:Connect( "timeout()", { || semaphore( oImg ) }
       )
   oClock:start( 1000 )

   oWnd:show()
   QApplication():exec()
   oClock:stop()

   RETURN

PROCEDURE semaphore( oI )

   DO CASE
   CASE sequence = 1
      oI:SetPixmap( QPixmap( "../res/semaforo1.png" ) )
   CASE sequence = 2
      oI:SetPixmap( QPixmap( "../res/semaforo2.png" ) )
   CASE sequence = 3
      oI:SetPixmap( QPixmap( "../res/semaforo3.png" ) )
   ENDCASE
   sequence ++
   IF sequence = 4
      sequence = 1
   ENDIF

   RETURN

   // END SOURCE
```

42.8 Semaphore (manual)

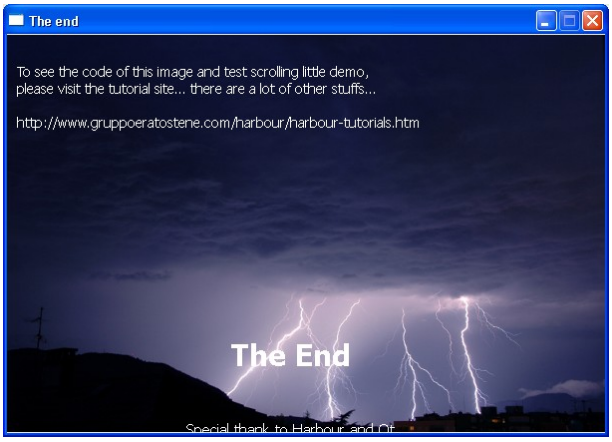
The following example shows how to create a manual semaphore. It changes its color if the button is pressed. *(by Giovanni Di Maria)*


```
    // SOURCE=semaphore.prg

#include "hbqtgui.ch"

   STATIC SEQUENCE

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oImg
   LOCAL oButton

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 250, 500 )

   oImg := QLabel( oWnd )
   oImg:move( 25, 25 )
   oImg:resize( 200, 400 )
```

```
   sequence = 1
   semaphore( oImg )

   oButton := QPushButton ( oWnd )
   oButton:move( 50, 430 )
   oButton:resize( 150, 30 )
   oButton:setText( "Press to change color" )
   oButton:Connect( "clicked()", { || semaphore( oImg )
      } )


   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE semaphore( oI )

   DO CASE
   CASE sequence = 1
      oI:SetPixmap( QPixmap( "../res/semaforo1.png" ) )
   CASE sequence = 2
      oI:SetPixmap( QPixmap( "../res/semaforo2.png" ) )
   CASE sequence = 3
      oI:SetPixmap( QPixmap( "../res/semaforo3.png" ) )
   ENDCASE
   sequence ++
   IF sequence = 4
      sequence = 1
   ENDIF

   RETURN

   // END SOURCE
```

42.9 Scrolling titles

The following example shows how to create scrolling titles, like movie titles. Them scroll up and shows some image, too. *(by Marco Braida)*




```
    // SOURCE=scrolling-titles.prg

#include "hbqtgui.ch"

   STATIC oStr1, oStr2, oStr3
   STATIC oImg1, oImg2

PROCEDURE Main()

   LOCAL oWnd
```

```
LOCAL OWinBg
LOCAL oClock
LOCAL nVTitleSpace, nSpeed, sTxt

nVTitleSpace := 300
nSpeed := 2

oWnd := QmainWindow()
oWnd:SetFixedSize( 600, 400 )
oWnd:setWindowTitle( "The end" )

oClock := QTimer()
oClock:Connect( "timeout()", { || moveTitle( nSpeed )
    } )
oClock:start( 50 )

OWinBg := QLabel( oWnd )
OWinBg:move( 0, 0 )
OWinBg:resize( 600, 400 )
OWinBg:SetPixmap( QPixmap( "../res/bra01-background.
   png" ) )

oImg1 := QLabel( oWnd )
oImg1:move( 10, nVTitleSpace )
oImg1:resize( 300, 300 )
oImg1:SetPixmap( QPixmap( "../res/bra01-arbour.png" )
    )

oImg2 := QLabel( oWnd )
oImg2:move( 310, nVTitleSpace )
oImg2:resize( 600, 300 )
oImg2:SetPixmap( QPixmap( "../res/bra01-logo_qt.png"
   ) )

oStr1 := QLabel( oWnd )
oStr1:move( 10, nVTitleSpace )
oStr1:resize( 550, 60 )
oStr1:SetText( "<center><h3>Harbour and Qt</h3></
   center><br>" )
oStr1:setStylesheet( "background: gray; border: 2px
   dotted white; border-radius: 20px; background:
   transparent; font-size: 35px; color: white;" )

sTxt := "To see the code of this image and test
   scrolling little demo,<br>"
sTxt += "please visit the tutorial site... there are
   a lot of other stuffs...<br>"
sTxt += "<br>"
sTxt += "http://www.gruppoeratostene.com/harbour/
   harbour-tutorials.htm<br>"

oStr2 := QLabel( oWnd )
oStr2:move( 10, nVTitleSpace * 2 )
oStr2:resize( 550, 140 )
```

```
   oStr2:SetText( sTxt )
   oStr2:setStylesheet( "font-size: 14px; color: white;"
       )

   oStr3 := QLabel( oWnd )
   oStr3:move( 10, nVTitleSpace * 3 )
   oStr3:resize( 550, 140 )
   oStr3:SetText( "<center> <h1>The End</h1><br><br>
       Special thank to Harbour and Qt<br>" + ;
      "www.harbour-project.org <br> qt.nokia.com" + ;
      "</center>" )
   oStr3:setStylesheet( "font-size: 15px; color: white;"
       )

   oWnd:show()
   QApplication():exec()
   oClock:stop()

   RETURN

PROCEDURE moveTitle( nSpeed )

   oImg1:move( oImg1:x , oImg1:y - nspeed )
   oImg2:move( oImg2:x , oImg2:y - nspeed )
   oStr1:move( oStr1:x , oStr1:y  - nSpeed )
   oStr2:move( oStr2:x , oStr2:y - nSpeed )

   IF ostr3:y < 120
      RETURN
   ENDIF
   oStr3:move( oStr3:x, oStr3:y - nSpeed )

   RETURN

   // END SOURCE
```

43 Sample Applications with Databases

43.1 DBF Database Append Record

The following example shows how to store data into a DBF database. Pressing the Save button, the data are saved into a DBF, thank to REPLACE command. *(by Giovanni Di Maria)*

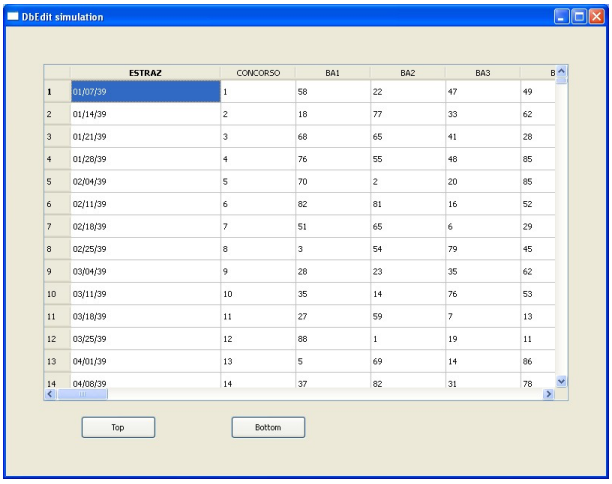

```
    // SOURCE=append.prg

#include "hbqtgui.ch"

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oName, oLabel1
   LOCAL oBorn, oLabel2
   LOCAL oSons, oLabel3
```

```
   LOCAL oButton

   SET DATE ITALIAN
   SET CENTURY ON

   oWnd := QMainWindow()
   oWnd:setWindowTitle( "Finestra di Giovanni" )
   oWnd:resize( 400, 300 )

   oLabel1 := QLabel( oWnd )
   oLabel1:setText( "Name" )
   oLabel1:move( 50, 50 )
   oLabel1:resize( 100, 20 )

   oName := QLineEdit( oWnd )
   oName:move( 160, 50 )
   oName:resize( 150, 20 )

   oLabel2 := QLabel( oWnd )
   oLabel2:setText( "Date of birth " )
   oLabel2:move( 50, 80 )
   oLabel2:resize( 100, 20 )

   oBorn := QLineEdit( oWnd )
   oBorn:move( 160, 80 )
   oBorn:resize( 90, 20 )

   oLabel3 := QLabel( oWnd )
   oLabel3:setText( "Number of sons" )
   oLabel3:move( 50, 110 )
   oLabel3:resize( 100, 20 )

   oSons := QLineEdit( oWnd )
   oSons:move( 160, 110 )
   oSons:resize( 50, 20 )

   oButton := QPushButton( oWnd )
   oButton:setText( "Save" )
   oButton:move( 150, 200 )
   oButton:Connect( "clicked()", { || store( oName,oBorn
      ,oSons ) } )

   oWnd:show()
   QApplication():exec()

   RETURN

PROCEDURE STORE( oN, oB, oS )

   USE ../res/sample
   APPEND BLANK
   REPLACE sample -> cognome       WITH oN:text()
   REPLACE sample -> nascita       WITH CToD( oB:text()
      )
```

```
    REPLACE sample -> figli          WITH Val( oS:text() )
    USE

    RETURN

    // END SOURCE
```

43.2 DBEdit simulation

The following example shows how to simulate the DbEdit function to browse a DBF. The fields and records cannot be changed but only viewed. *(by Giovanni Di Maria)*



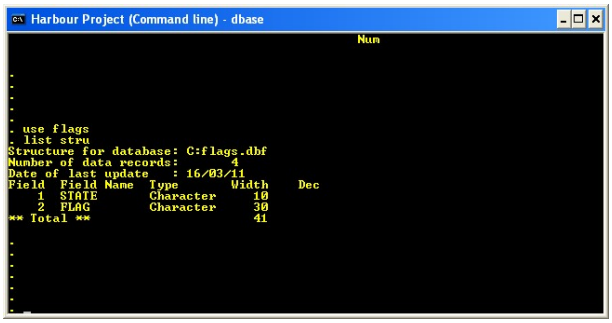

```
    // SOURCE=dbedit.prg

#include "hbqtgui.ch"

   STATIC oTable

PROCEDURE Main()

   //------------Declaration--------------
   LOCAL oWnd
   LOCAL oCell, valore
   LOCAL nNumField, nNumRecord, oLabels
   LOCAL nX, nY
   LOCAL button_top, button_bottom

   oWnd := QMainWindow()
   oWnd:resize( 800, 600 )
   oWnd:setWindowTitle( "DbEdit simulation" )

   //------------Open DBF and count field and record
      ------------
   USE ../res/sample
   nNumRecord = RecCount()
   nNumField = FCount()
```

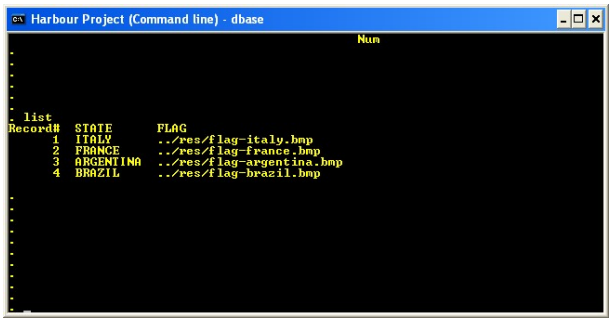
```
//------------Table--------------
oTable := QTableWidget( oWnd )
oTable:move( 50, 50 )
oTable:resize( 700, 450 )
oTable:setRowCount( nNumRecord )
oTable:setColumnCount( nNumField )
oTable:setColumnWidth( 0, 200 )

//------------Fill table--------------
for nX = 1 TO nNumRecord
   for nY = 1 TO  nNumField
      oCell := QTableWidgetItem()
      valore = FieldGet( nY )
      DO CASE
      CASE ValType( valore ) = "C"
         oCell:setText( valore )
      CASE ValType( valore ) = "N"
         oCell:setText( AllTrim( Str(valore ) ) )
      CASE ValType( valore ) = "D"
         oCell:setText( Dtoc( valore ) )
      CASE ValType( valore ) = "L"
         oCell:setText( iif( valore = .T. ,"Yes","No"
             )  )
      end CASE
      oTable:setItem( nX - 1, nY - 1, oCell )
   next nY
   SKIP
next nX

//------------Label Table--------------
oLabels := QStringList()
for nX = 1 TO 50
   oLabels:append( field( nX ) )
next nX
oTable:setHorizontalHeaderLabels( oLabels )
USE

//------------Button--------------
button_top := QPushButton( oWnd )
button_top :move( 100, 520 )
button_top:setText( "Top" )
button_top:Connect( "clicked()", { || top() } )

button_bottom := QPushButton( oWnd )
button_bottom:move( 300, 520 )
button_bottom:setText( "Bottom" )
button_bottom:Connect( "clicked()", { || bottom() } )

//------------Exec--------------
oWnd:show()
QApplication():exec()

RETURN
```

```
PROCEDURE top()

   oTable:scrollToTop()
   oTable:setCurrentCell( 0, 0 )
   oTable:setFocus()

   RETURN

PROCEDURE bottom()

   oTable:scrollToBottom()
   oTable:setCurrentCell( oTable:rowCount() - 1, 0 )
   oTable:setFocus()

   RETURN

   // END SOURCE
```

43.3 Flags and Database

The following example shows flags on a form. The flags are stored in a DBF archive. User can browse data by the "prev" and "next" buttons. *(by Giovanni Di Maria)*





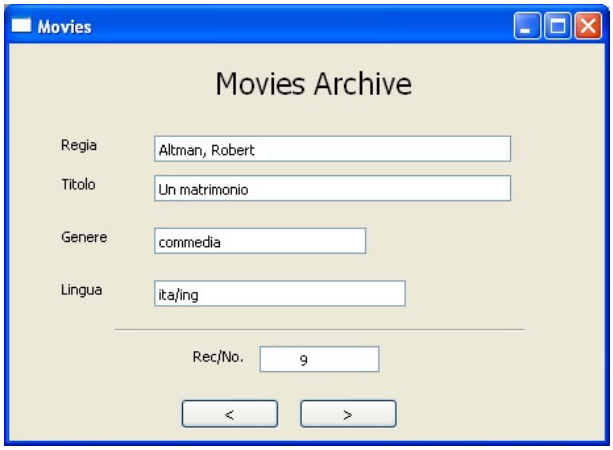

```
    // SOURCE=flags.prg

#include "hbqtgui.ch"

   STATIC oImg, oState

PROCEDURE Main()

   LOCAL oWnd
   LOCAL oPrev, oNext
```

```
   oWnd := QmainWindow()
   oWnd:SetFixedSize( 400, 300 )
   oWnd:setWindowTitle( "Finestra Giovanni" )

   oState := QLineEdit( oWnd )
   oState:move( 100, 20 )
   oState:resize( 200, 20 )
   oState:setReadOnly( .T. )

   oImg := QLabel( oWnd )
   oImg:move( 50, 50 )
   oImg:resize( 300, 200 )
   oImg:setStyleSheet( "border: 2px solid #0000ff;" )

   oPrev := QPushButton( oWnd )
   oPrev:move( 50, 260 )
   oPrev:setText( "Prev" )
   oPrev:Connect( "clicked()", { || go_prev() } )

   oNext := QPushButton( oWnd )
   oNext:move( 250, 260 )
   oNext:setText( "Next" )
   oNext:Connect( "clicked()", { || go_next() } )

   USE ../res/flags

   show_flag()
   oWnd:show()
   QApplication():exec()

   USE

   RETURN

PROCEDURE show_flag()

   oState:setText( flags -> state )
   oImg:SetPixmap( QPixmap( AllTrim(flags -> flag ) ) )

   RETURN

PROCEDURE go_prev()

   skip - 1
   show_flag()

   RETURN

PROCEDURE go_next()

   SKIP
   show_flag()
```

```
    RETURN

    // END SOURCE
```

43.4 Movies Database and UI Creator

The following example shows how to use a .UI design, created by QT Creator. The data are stored in a DBF archive. User can browse data by the "prev" and "next" buttons. The application was build in 10 minutes. Please read the UI section below. *(by Giovanni Di Maria)*




```
. use movies
. list stru
Structure for database: C:movies.dbf
Number of data records:     119
Date of last update    : 20/03/11
Field  Field Name  Type       Width    Dec
    1  REGIA       Character     25
    2  TITOLO      Character     25
    3  GENERE      Character     20
    4  LINGUA      Character     16
    5  SOTT        Character     22
** Total **                    109
```

```
    // SOURCE=ui-movies-archive.prg
   // HBP=movies.ui

#include "hbqtgui.ch"

   STATIC oWnd

PROCEDURE Main()

   oWnd := hbqtui_movies()
   oWnd:setWindowTitle( "Movies" )
   oWnd:q_pushButton:Connect( "clicked()", { || prev() }
        )
```

```
   oWnd:q_pushButton_2:Connect( "clicked()", { || next()
        } )

   USE ../res/movies
   show_record()

   oWnd:show()
   QApplication():exec()
   USE

   RETURN

PROCEDURE show_record()

   oWnd:q_lineEdit:setText( movies -> regia )
   oWnd:q_lineEdit_2:setText( movies -> titolo )
   oWnd:q_lineEdit_3:setText( movies -> genere )
   oWnd:q_lineEdit_4:setText( movies -> lingua )
   oWnd:q_lineEdit_5:setText( Str( RecNo() ) )

   RETURN

PROCEDURE prev()

   skip - 1
   show_record()

   RETURN

PROCEDURE next()

   SKIP
   show_record()

   RETURN

   // END SOURCE
```

The file movies.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Form</class>
 <widget class="QWidget" name="Form">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>452</width>
```

```
    <height>302</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Form</string>
  </property>
  <widget class="QLabel" name="label">
   <property name="geometry">
    <rect>
     <x>140</x>
     <y>10</y>
     <width>181</width>
     <height>41</height>
    </rect>
   </property>
   <property name="font">
    <font>
     <pointsize>17</pointsize>
    </font>
   </property>
   <property name="text">
    <string>Movies Archive</string>
   </property>
   <property name="alignment">
    <set>Qt::AlignCenter</set>
   </property>
  </widget>
  <widget class="QLabel" name="label_2">
   <property name="geometry">
    <rect>
     <x>40</x>
     <y>70</y>
     <width>46</width>
     <height>13</height>
    </rect>
   </property>
   <property name="text">
    <string>Regia</string>
   </property>
  </widget>
  <widget class="QLabel" name="label_3">
   <property name="geometry">
    <rect>
     <x>40</x>
     <y>100</y>
     <width>46</width>
     <height>13</height>
    </rect>
   </property>
   <property name="text">
    <string>Titolo</string>
   </property>
  </widget>
  <widget class="QLabel" name="label_4">
```

```
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>140</y>
      <width>46</width>
      <height>13</height>
     </rect>
    </property>
    <property name="text">
     <string>Genere</string>
    </property>
   </widget>
   <widget class="QLabel" name="label_5">
    <property name="geometry">
     <rect>
      <x>40</x>
      <y>180</y>
      <width>46</width>
      <height>13</height>
     </rect>
    </property>
    <property name="text">
     <string>Lingua</string>
    </property>
   </widget>
   <widget class="QLabel" name="label_6">
    <property name="geometry">
     <rect>
      <x>140</x>
      <y>230</y>
      <width>61</width>
      <height>16</height>
     </rect>
    </property>
    <property name="text">
     <string>Rec/No.</string>
    </property>
   </widget>
   <widget class="QLineEdit" name="lineEdit">
    <property name="geometry">
     <rect>
      <x>110</x>
      <y>70</y>
      <width>271</width>
      <height>20</height>
     </rect>
    </property>
   </widget>
   <widget class="QLineEdit" name="lineEdit_2">
    <property name="geometry">
     <rect>
      <x>110</x>
      <y>100</y>
      <width>271</width>
```

```
      <height>20</height>
     </rect>
    </property>
   </widget>
   <widget class="QLineEdit" name="lineEdit_3">
    <property name="geometry">
     <rect>
      <x>110</x>
      <y>140</y>
      <width>161</width>
      <height>20</height>
     </rect>
    </property>
   </widget>
   <widget class="QLineEdit" name="lineEdit_4">
    <property name="geometry">
     <rect>
      <x>110</x>
      <y>180</y>
      <width>191</width>
      <height>20</height>
     </rect>
    </property>
   </widget>
   <widget class="QLineEdit" name="lineEdit_5">
    <property name="geometry">
     <rect>
      <x>190</x>
      <y>230</y>
      <width>91</width>
      <height>20</height>
     </rect>
    </property>
   </widget>
   <widget class="QPushButton" name="pushButton">
    <property name="geometry">
     <rect>
      <x>130</x>
      <y>270</y>
      <width>75</width>
      <height>23</height>
     </rect>
    </property>
    <property name="text">
     <string>&lt;</string>
    </property>
   </widget>
   <widget class="QPushButton" name="pushButton_2">
    <property name="geometry">
     <rect>
      <x>220</x>
      <y>270</y>
      <width>75</width>
      <height>23</height>
```

```
     </rect>
    </property>
    <property name="text">
     <string>&gt;</string>
    </property>
   </widget>
   <widget class="Line" name="line">
    <property name="geometry">
     <rect>
      <x>80</x>
      <y>210</y>
      <width>311</width>
      <height>16</height>
     </rect>
    </property>
    <property name="orientation">
     <enum>Qt::Horizontal</enum>
    </property>
   </widget>
  </widget>
  <resources/>
  <connections/>
 </ui>
```

44 Photos

Figure 1: My books

Giovanni Di Maria

A Appendices

A.1 Appendix - Contributors

- Giovanni Di Maria (he is still the main developer)
- Marco Braida
- Massimo Belgrano
- Mario Wan Stadnik (Qatan)
- Apolinar

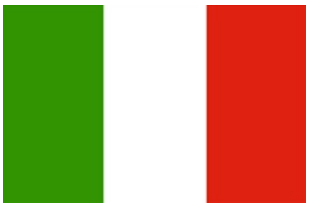
A.2 Appendix - What users think

A collection of phrases and thanks by people who use this tutorial.

- Sounds good! (*Viktor Szakáts*)
- Thanks again for your nice tutorial. (*Viktor Szakáts*)
- It's very nice job, thanks for this tutorial. (*Viktor Szakáts*)
- As said earlier it is a noble initiative and will stir a lot of enthusiam for Harbour users. (*Pritpal Bedi*)
- Thanks for your great initiative providing that important tutorial. It is encouraging me to try HBQT and I found it is not as difficult as I first thought. Thanks for your help. Regards. (*Qatan*)
- Thank you Giovanni for the tutorial. (*Guy Roussin*)
- Complimenti Giovanni il tuo lavoro del quale riporto per comodit il link diretto. E' utilissimo... grazie per la condivisione... (*Marco Braida*)
- Thanks a lot Giovanni! (*Maurilio Longo*)
- That said, I really appreciate your tutorial! (*Maurilio Longo*)
- I think your tutorial's link should be loaded on harbour's web page. (*Angel Pais*)
- It's a fine document. (*Alex Strickland*)
- Many thanks for your great work. (*Shum*)
- Congratulations for nice contribuition. (*Itamar M. Lins Jr. Lins*)
- Dear Giovanni: Thank you for the tutorial of hbqt. It is very useful for me. Best regards. (*Daniel Tello Argentina*)
- Imo this tutorial is good also as official harbour doc. (*Massimo Belgrano*)
- It's great service for whole Harbour community. Well done and keep it up. Also a great example for how to contribute. (*Viktor Szakáts*)

B Notes

B.1 Notes




Caltanissetta

***

The End