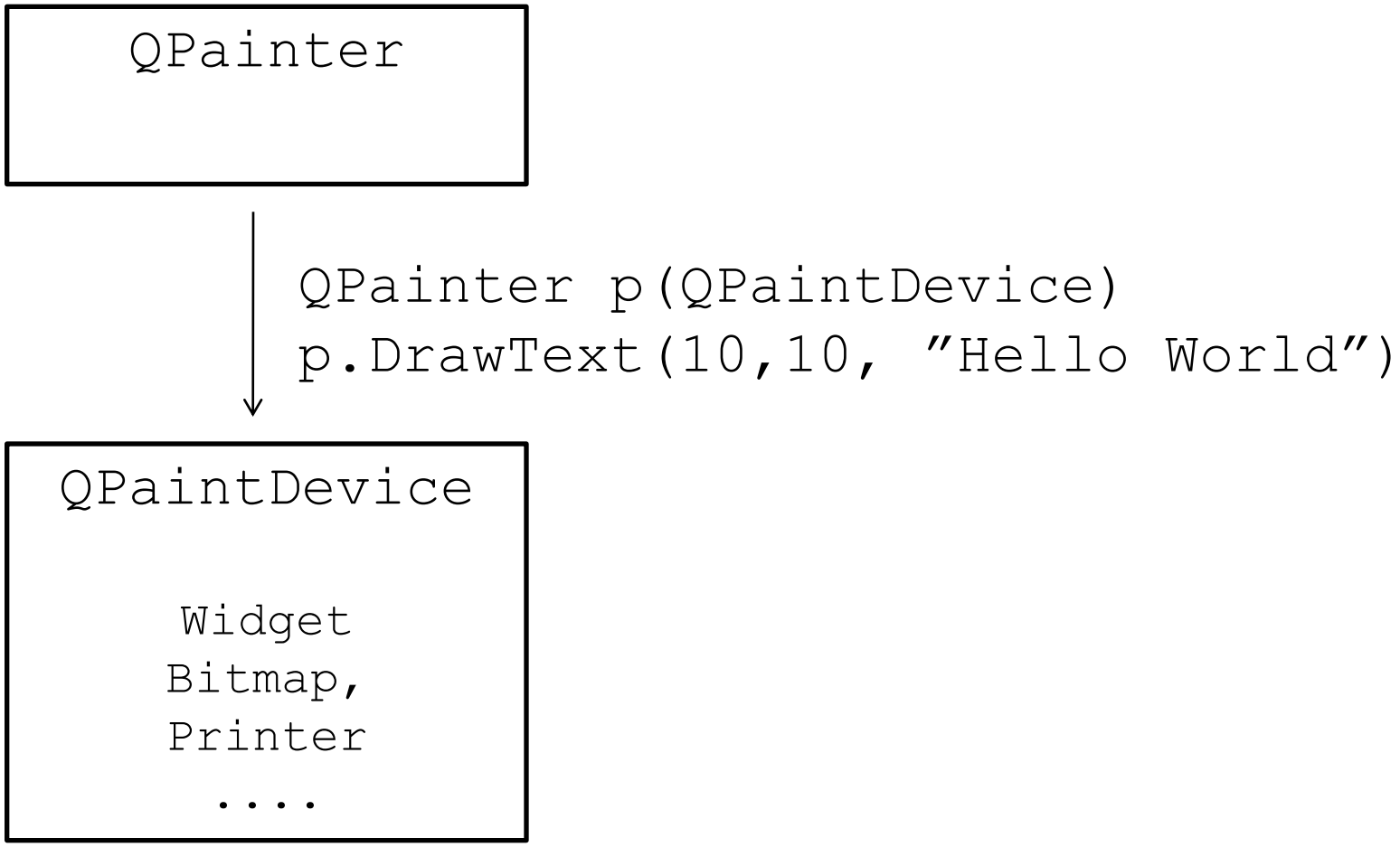
Grafik – rita med Qt

- I Qt heter grundritsystemet ”Arthur”
- Består av följande huvudklasser
  - QPainter
    - Klass som används för att rita med
  - QPaintDevice
    - Enhet som man kan rita på
  - QPaintEngine
    - Gränssnitt som används för att rita
    - ANVÄNDS NORMALT INTE AV ANVÄNDARPROGRAM !

Grafik i Qt


QPainter
QPainter p(QPaintDevice)
p.DrawText(10,10, ”Hello World”)
QPaintDevice
Widget
Bitmap,
Printer
....

Exempel: Rita på QMainWindow

- Skapar en ny class som ärver Qwidget
- Implementerar funktionen

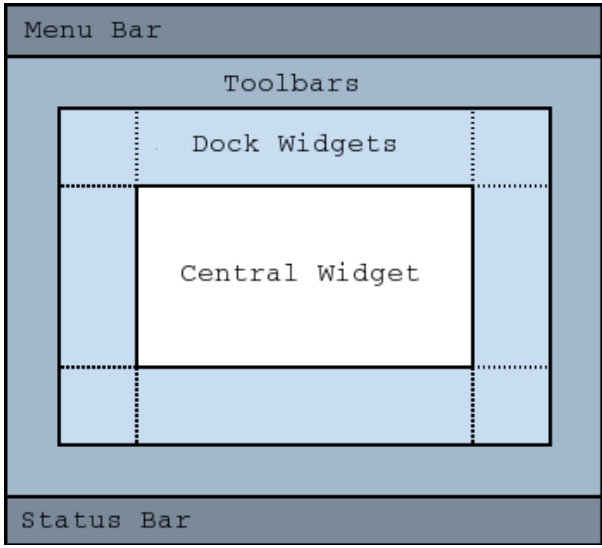
```
virtual void paintEvent(QPaintEvent *e)
```

- Sätter den nya widgeten som ”central widget i vår ärvda QMainWindow

```
setCentralWidget(new paintWidget);
```

- Omdefinierar ritfunktionen

```
void paintWidget::paintEvent(QPaintEvent
   *event) {
   QPainter painter(this);
   paintF(&painter);
   }
```



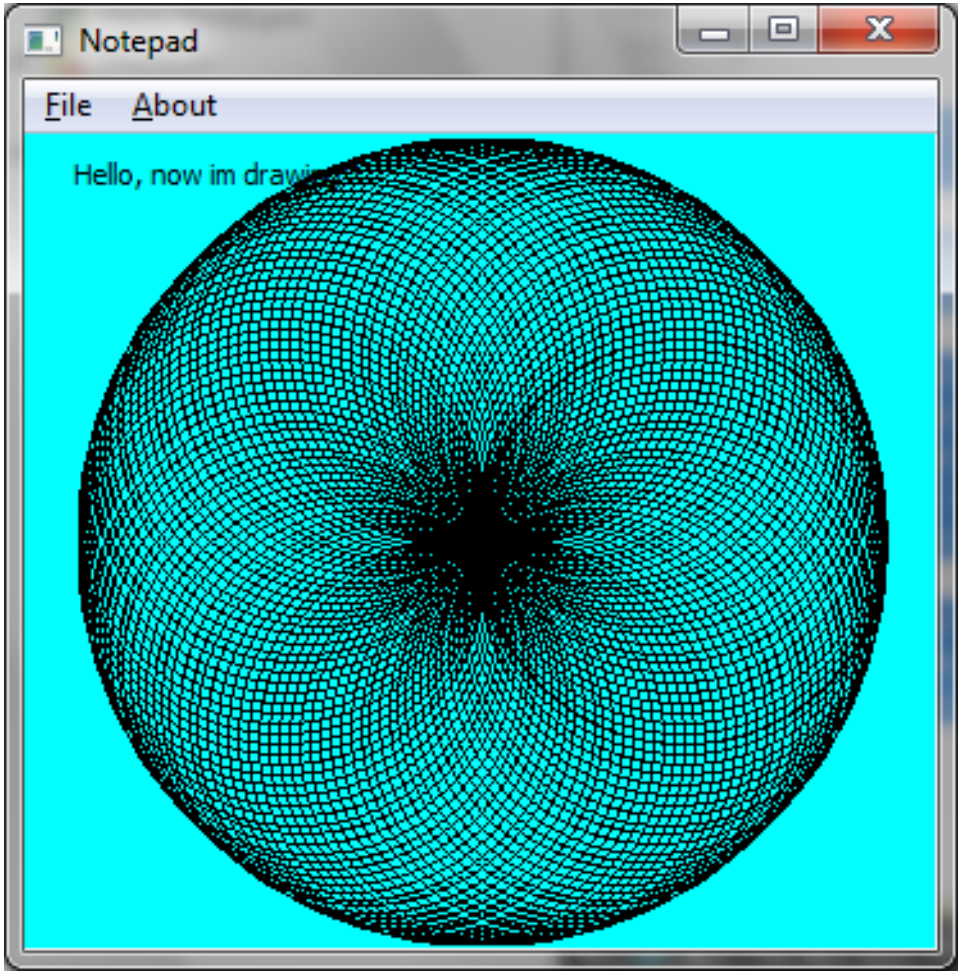
Ritfunktionen

```
void paintF(QPainter  *painter) {
#define PI 3.1415

    int w=painter->device()->width();
    int h=painter->device()->height();
    painter->setBackground(QBrush(Qt::cyan));
    painter->eraseRect(0,0, w-1, h-1);
    painter->drawText(20,20, "Hello, now im drawing!!");
    QPoint center(w/2, h/2);
    qreal radius = w<h?w/4:h/4;

    for (float angle=0; angle<2*PI; angle+=2*PI*3/360) {
      QPoint pos(cos(angle)*radius, sin(angle)*radius);
      painter->drawEllipse(center+pos, radius, radius);
    }
}
```

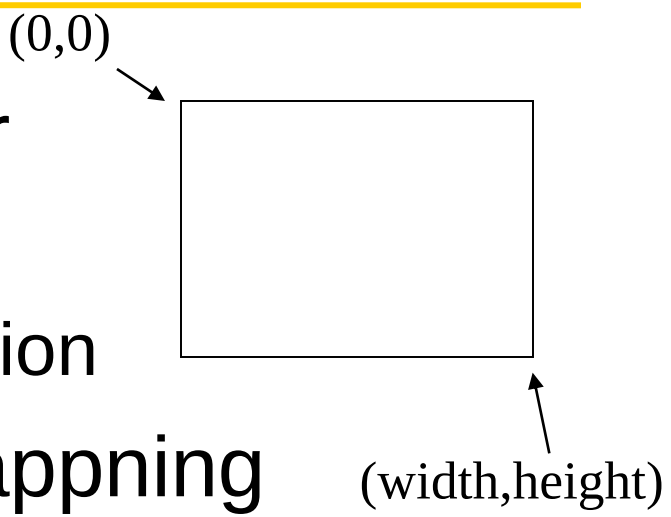
Resultat



QPaintDevice

- En Paint Device (PD) är en generaliserad modell av ett område man kan rita på
  - Linjer,text, cirklar,färger etc.etc.
- När man ritar, gör man det alltid på en PD, inte t.ex. direkt på fönstret
- En PD kan också representera
  - bitkarta, printer, ...,bara man tar i beaktande skalningen
- I QWidgets representeras denna PD av grundklassen QPaintDevice

Lite om koordinatsystem



- *Device units* – motsvarar enhetens pixel
  - t.ex. för printer via resolution
- *Logical units* – via en mappning motsvarar dessa device units
- *Mapping mode* – vilken grundenhet som används för att övergå från logiska till enhetens koordinatsystem

Lite om koordinatsystem

- I wxWindows:
  - wxMM_TWIPS – varje enhet är 1/20 av en point = 1/1440 tum
  - wxMM_POINTS – varje logiska enhet är en point = 1/72 tum
  - wxMM_METRIC – varje logiska enhet är en millimeter
  - wxMM_LOMETRIC – varje logiska enhet är en 1/10 millimeter
  - wxMM_TEXT – varje logiska enhet är en pixel, även default mappning
- Dessutom finns det *user scale*,vilken mapping-modens skala multipliceras med

Klippningsregion

- Varje PD kan också förknippas med en *clip region*, klippningsregion
- Då man ritar, kommer det man ritar endast att uppdateras inom clip region-rectangeln
- Kan t.ex. användas för att hindra text från att gå över annan information
  - Skapa en clip region
  - Skriv ut text
  - Radera clip region

QPaintDevice-typer: exempel

- QWidget: Rita på skärmen (via en widget)
- QImage: Rita i ett hårdvareoberoende grafikområde
- QPixmap: Bitkarta, som inte visas på skärmen
- QPicture: Sparar och spelar upp ritkommandon (jfr. Windows Metafile)
- QPrinter: Representrar en skrivare
- Stöd för hårdvara
  - QGLFramebufferObject, QGLPixelBuffer, QCustomRasterPaintDevice

Exempel: Rita på skärmen

```
BEGIN_EVENT_TABLE(MyWindow, wxWindow)
    EVT_MOTION(MyWindow::OnMotion)
END_EVENT_TABLE()

void MyWindow::OnMotion(wxMouseEvent& event)
{
    if (event.Dragging())
    {
        wxClientDC dc(this);
        wxPen pen(*wxRED, 5); // red pen of width 5
        dc.SetPen(pen);
        dc.DrawPoint(event.GetPosition());
        dc.SetPen(wxNullPen);
    }
}
```

Rita till minnet (bitkarta)

```
void myWidget::fileBitmap() {
        QImage image(400,400, QImage::Format_RGB32);
        QPainter p(&image);
        paintF(&p);

        QString fileName = QFileDialog::getSaveFileName(this,
         "Save File", "", "Bitmap (*.bmp)");
        if (fileName != "") {
            QImageWriter ir(fileName, "BMP");
            ir.write(image);
        }
}
```

Använd hela skärmen

```
void myWidget::screenShot() {

        QPixmap screenShot =
                QPixmap::grabWindow(QApplication::desktop()-
   >winId());
        QImage image = screenShot.toImage();

        QString fileName = QFileDialog::getSaveFileName(this,
   "Save File",
        "", "Bitmap (*.bmp)");
        if (fileName != "") {
            QImageWriter ir(fileName, "BMP");
            ir.write(image);
        }


}
```

Utskrift med QPrinter

- Win, Mac har inbyggt printerstöd:
- Unix, andra: Varierande, men ofta idag via CUPS (som ofta använder *ghostscript* Postscript -> PDF, raster för printer)
- QPrinter kan användas som QPaintDevice
- Tillgång till en QPrinter fås normalt via att kalla på en färdig printdialog QPrintDialog, varefter vi får del valda QPrinter-enheten

Utskrift av en sida

```
void myWidget::print() {
    QPrintDialog printD;
    if (printD.exec() == QDialog::Accepted)
   {
        statusBar()-
   >showMessage(QString("Printer name: ") +
   printD.printer() >printerName());
        QPainter painter;
        painter.begin(printD.printer());
        paintF(&painter);
        painter.end();
    }
}
```

Utskrift till postscript

```
void MyFrame::OnMenuFilePrint(wxCommandEvent &e) {

   wxPostScriptDC *dc = new wxPostScriptDC(L"fil.ps",true,
   this);
   if (dc.Ok() {
        dc->StartDoc(L"wxhello");
        dc->StartPage();
        Paint(*dc);
        dc->EndPage();
        dc->EndDoc();
   } else {
        wxMessageBox(L”Could not set up PS device”,”Error”);
   }
   delete dc;
   return;
}
```

Utskrift av dokument

- Använd **`wxPrintout`** för att sköta utskrift, sköter
  - Kopior, val av sidor att printa, avbrytning av utskrift etc.
- Ärv klassen **`wxPrintout`**
- Implementera vissa funktioner
  - **`OnPrintPage(int pagenum)`**
  - **`Haspage(int pagenum)`**

Acceleratorer

- Acceleratorer används för att skapa snabbval till funktionalitet, t.ex. ctrl+O öppnar file
- Acceleratorer kan direkt läggas till

```
// Add accelerators
   openA = new QAction("&Open", this); openA->setShortcut(QKeySequence::Open);
    saveA = new QAction("&Save", this);saveA->setShortcut(QKeySequence::Save);
    exitA = new QAction("E&xit", this); exitA->setShortcut(QKeySequence::Quit);
    aboutA = new QAction("&About", this);
    printA = new QAction("&Print", this); printA-
   >setShortcut(QKeySequence::Print);
    bmA = new QAction("&Bitmap", this);
```

Jämförelse MFC och wxWidgets

MFC and wxWidgets Macros

MFC Version	wxWidgets Version
BEGIN_MESSAGE_MAP	BEGIN_EVENT_TABLE
END_MESSAGE_MAP	END_EVENT_TABLE
DECLARE_DYNAMIC	DECLARE_CLASS
DECLARE_DYNCREATE	DECLARE_DYMAMIC_CLASS
IMPLEMENT_DYNAMIC	IMPLEMENT_CLASS
IMPLEMENT_DYNCREATE	IMPLEMENT_DYNAMIC_CLASS
IsKindOf(RUNTIME_CLASS(CWindow))	IsKindOf(CLASSINFO(wxWindow))

Jämförelse MFC och wxWidgets

Miscellaneous Classes	
MFC Version	**wxWidgets Version**
CWinApp	wxApp
CObject	wxObject
CCmdTarget	wxEvtHandler
CCommandLineInfo	wxCmdLineParser
CMenu	wxMenu, wMenuBar, wxMenuItem
CWaitCursor	wxBusyCursor
CDataExchange	wxValidator
Window Classes	
MFC Version	**wxWidgets Version**
CFrameWnd	wxFrame
CMDIFrameWnd	wxMDIParentFrame
CMDIChildWnd	wxMDIChildFrame
CSplitterWnd	wxSplitterWindow
CToolBar	wxToolBar
CStatusBar	wxStatusBar
CReBar	wxCoolBar, but see contrib/src/fl and wxDockIt
CPropertyPage	wxPanel
CPropertySheet	wxNotebook, wxPropertySheetDialog

Dialoger

- Byggande av dialog kan ske genom att manuellt lägga till dialogobjekt (aka widgets)
  - buttons
  - radio-buttons
  - text-inputs
  - listboxar / combo-boxar
  - statisk text
  - statiska texter
  - m.m.

Börjar med att ärva wxDialog

```
#include ”wx/wx.h”
class myApp:public wxApp {
   bool OnInit();
}
class myDialog:public wxDialog {
   wxButton *m_pOkButton;
   wxButton *m_pCancelButton;
   wxTextCtrl *m_pText;

   enum {ID_OKBUTTON,ID_CANCELBUTTON, ID_TEXT};

public:
   myDialog();
   void OnOK(wxCommandEvent &e);
   void OnCancel(wxCommandEvent &e);
   void OnTextMax(wxCommandEvent &e);
   DECLARE_EVENT_TABLE()
};
```

Implementation

```
/* Implementation */
MPLEMENT_APP(myApp)

BEGIN_EVENT_TABLE(myDialog, wxDialog)
EVT_BUTTON(ID_OKBUTTON,myDialog::OnOK)
EVT_BUTTON(ID_CANCELBUTTON,myDialog::OnCancel)
EVT_TEXT_MAXLEN(ID_TEXT, myDialog::OnTextMax)
END_EVENT_TABLE()

bool myApp::OnInit() {
    myDialog *d = new myDialog();
    d->ShowModal();
    return false; //Do nothing....
}

myDialog::myDialog()
:wxDialog(NULL, -1, "Test dialog", wxDefaultPosition, wxSize(250,150)) {
    m_pOkButton = new wxButton(this, ID_OKBUTTON, "&OK", wxPoint(20,70));
    m_pCancelButton = new wxButton(this, ID_CANCELBUTTON, "&Cancel",
    wxPoint(120,70));
    m_pText = new wxTextCtrl(this, ID_TEXT, "Empty", wxPoint(20,20),
    wxSize(80,20));
    m_pText->SetMaxLength(20);
}
```

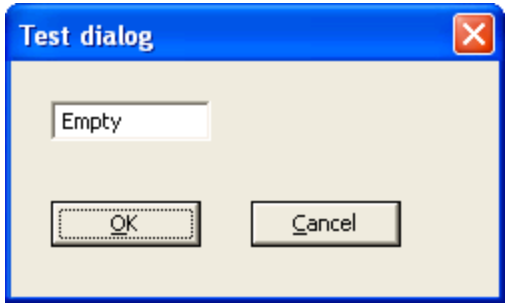
Implementation....

```
void myDialog::OnOK(wxCommandEvent &e) {
   EndModal(ID_OKBUTTON);
}

void myDialog::OnCancel(wxCommandEvent &e) {
   EndModal(ID_CANCELBUTTON);
}

void myDialog::OnTextMax(wxCommandEvent &e) {
   wxMessageBox("Text maxlen is 20", "Error", wxOK);
}
```

Resultat:



Dialoger och resurser

- Att manuellt implementera grafiska dialoger är kodintensiva och svåra att editera efteråt
  - GUI-system brukar implementera ”resurser”, dvs beskrivningen av grafiska element sker skillt från själva koden
    - Placering, statisk text,storlek ID
    - wxWidgets: XRC
      - XML-baserad beskrivning av resurser

Vår dialog via XRC-beskrivning

```
<?xml version="1.0"?>
<resource version="2.3.0.1">
 <object class="wxDialog" name="myDialog2">
   <size>250,150</size>
   <title>Test dialog</title>
    <object class="wxButton" name="ID_OKBUTTON">
        <pos>20,70</pos>
        <label>_OK</label>
    </object>
    <object class="wxButton" name="ID_CANCELBUTTON">
        <pos>120,70</pos>
        <label>_Cancel</label>
    </object>
    <object class="wxTextCtrl" name="ID_TEXT">
        <pos>20,20</pos>
        <size>80,20</size>
        <value>Empty</value>
    </object>
 </object>
</resource>
```

XRC-koncept

- Include the appropriate headers: normally "wx/xrc/xmlres.h" will suffice;
- If you are going to use XRS files, install wxFileSystem ZIP handler first with wxFileSystem::AddHandler(new wxZipFSHandler);
- call wxXmlResource::Get()->InitAllHandlers() from your wxApp::OnInit function, and then call wxXmlResource::Get()->Load("myfile.xrc") to load the resource file;
- to create a dialog from a resource, create it using the default constructor, and then load it using for example wxXmlResource::Get()->LoadDialog(&dlg, this, "dlg1");
- set up event tables as usual but use the XRCID(str) macro to translate from XRC string names to a suitable integer identifier, for example EVT_MENU(XRCID("quit"), MyFrame::OnQuit).

Modifieringar i koden

```
class myDialog2:public wxDialog {
public: // Ingen konstruktor
    void OnOK(wxCommandEvent &e);
    void OnCancel(wxCommandEvent &e);
    void OnTextMax(wxCommandEvent &e);
    DECLARE_EVENT_TABLE()
};

BEGIN_EVENT_TABLE(myDialog2, wxDialog)
EVT_BUTTON(XRCID("ID_OKBUTTON"),myDialog2::OnOK) //ID via macro XRCID, mappas till
    ”name” i XRC-filen
EVT_BUTTON(XRCID("ID_CANCELBUTTON"),myDialog2::OnCancel)
EVT_TEXT_MAXLEN(XRCID("ID_TEXT"), myDialog2::OnTextMax)
END_EVENT_TABLE()

bool myApp::OnInit() {
    wxXmlResource::Get()->InitAllHandlers();
    wxXmlResource::Get()->Load("dialog.xrc");

    myDialog2 d;
    wxXmlResource::Get()->LoadDialog(&d, NULL, "myDialog2");
    XRCCTRL(d,"ID_TEXT",wxTextCtrl)->SetMaxLength(20); // Hämtar referens till
                    kontrollen via macro XRCCTRL
    d.ShowModal();

    return false; //Do nothing....
}
```

Resurser i XRC/XRS/CPP-format

- Resurser kan finnas i XRC (=XML)-format tillsammans med den exekverbara koden
- Resurser kan också kompileras till XRS-format (binärt, i praktiken ZIP)

```
% wxrc dialog.xrc
```

→ **Skapar filen resource.xrs**

- Genom att kompilera resursen till cpp-format, kan filen inkluderas i själva programmet

```
% wxrc –c dialog.xrc
```

→ **Skapar filen resource.cpp – kan nu läggas till projektet, varvid resoursen är inkluderad i det exekverbara programmet**

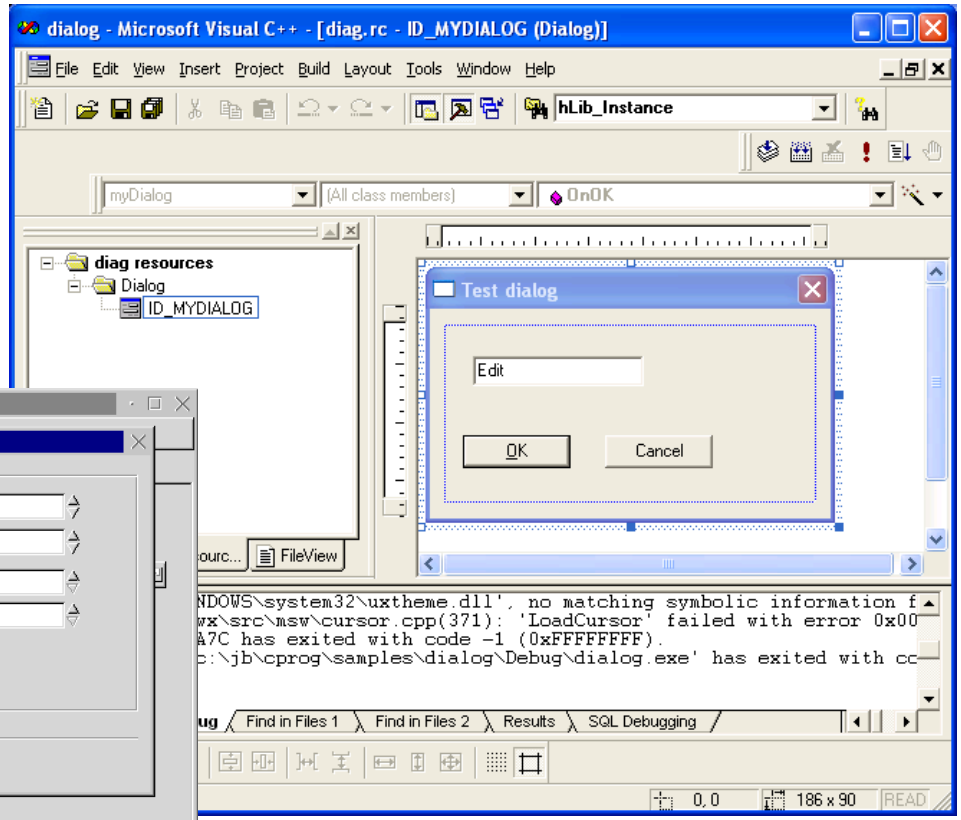
Jfr MFC resurser

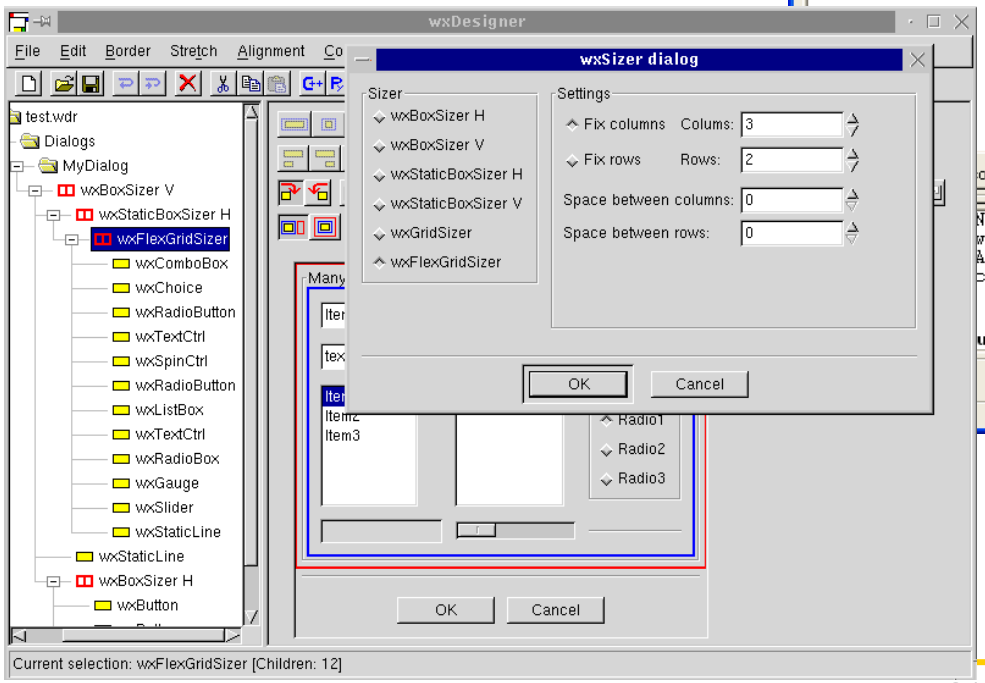
```
// Dialog dialog.rc – resource script

ID_MYDIALOG DIALOG DISCARDABLE  0, 0, 186, 90
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Test dialog"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "&OK",ID_OKBUTTON,15,54,50,14
    PUSHBUTTON      ”&Cancel",ID_CANCELBUTTON,81,54,50,14
    EDITTEXT        ID_TEXT,20,20,79,13,ES_AUTOHSCROLL |
   ES_WANTRETURN
END
```

I praktiken används ofta resurs-editorer

Visual studio





wxDesigner

Validatorer

- I exemplet med dialog-boxen, ville vi mata in text
- Denna text används normalt i något sammanhang, t.ex. återspeglar en variabel i programmet
  - Detta innebär att då man visar dialogen
    - Vid initialiseringen skall kopiera över från programmet till dialogboxen
    - Vid avslutning skall kopiera över från dialogboxen till programmet
    - Om vi vill begränsa vad en användare matar in → måste även validera input / output

Bakgrund

```
wxString gStr(”Start value”); //Initialvärde

myDialog::myDialog()
:wxDialog(NULL, -1, "Test dialog", wxDefaultPosition, wxSize(250,150))
    {
    m_pOkButton = new wxButton(this, ID_OKBUTTON, "&OK",
    wxPoint(20,70));
    m_pCancelButton = new wxButton(this, ID_CANCELBUTTON, "&Cancel",
    wxPoint(120,70));
    m_pText = new wxTextCtrl(this, ID_TEXT, "Empty", wxPoint(20,20),
    wxSize(80,20));
    m_pText->SetMaxLength(20);
    m_pText->SetValue(gStr);
}

void myDialog::OnOK(wxCommandEvent &e) {
    gStr = m_pText->GetValue(); // Kopierar tillbaka värdet
    EndModal(ID_OKBUTTON);
}
```

Automatiserat genom validator

```
Konstruktor:
wxTextValidator(long style = wxFILTER_NONE, wxString* valPtr = NULL)

myDialog::myDialog()
:wxDialog(NULL, -1, "Test dialog", wxDefaultPosition, wxSize(250,150))
    {
    m_pOkButton = new wxButton(this, ID_OKBUTTON, _T("&OK"),
    wxPoint(20,70));
    m_pCancelButton = new wxButton(this, ID_CANCELBUTTON, _T("&Cancel"),
    wxPoint(120,70));
    m_pText = new wxTextCtrl(this, ID_TEXT, "Empty", wxPoint(20,20),
    wxSize(80,20),0, wxTextValidator(wxFILTER_ALPHA, &gStr));
    m_pText->SetMaxLength(20);
}

void myDialog::OnOK(wxCommandEvent &e) {
    wxDialog::OnOK(e); // Using implicitly functions Validate() and
    TransferDataFromWindow()
    //EndModal(ID_OKBUTTON);
}
```

Alternativ

- Endast kopiering
  - `wxGenericValidator(...)`
    - **`wxGenericValidator`**`(`**`bool*`** *`valPtr`*`)`
    - **`wxGenericValidator`**`(`**`wxString*`** *`valPtr`*`)`
    - **`wxGenericValidator`**`(`**`int*`** *`valPtr`*`)`
    - **`wxGenericValidator`**`(`**`wxArrayInt*`** *`valPtr`*`)`
- Egen validator
  - Måste implementera
    - `TransferFromWindow(), TransferToWindow(), Validate()`

Debugging

Debugging

- Undvik fel
  - Använd `ASSERT (wxAssert(condition))`
  - använd `wxString` alltid då möjligt
- Identifiera fel
  - Använd en debugger
  - Använd loggningsfacilitieter (::wxLogDebug)