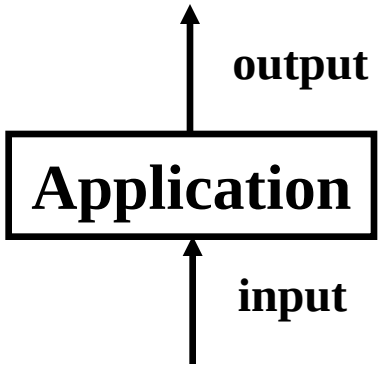
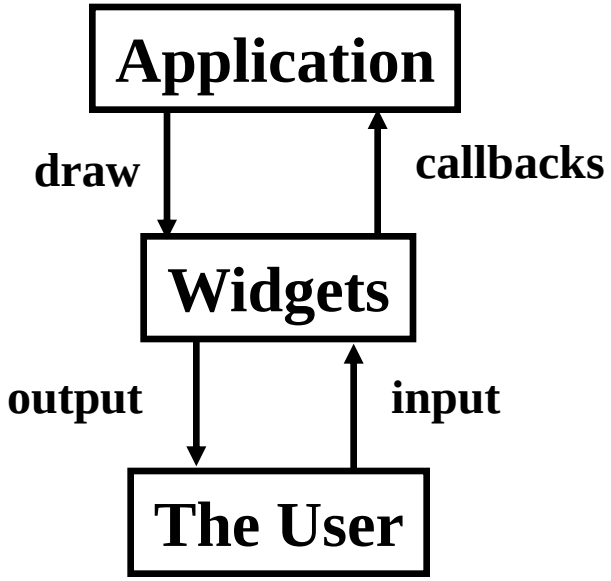
GUI-programmering med WxWidgets

www.wxwidgets.org

Paradigmer

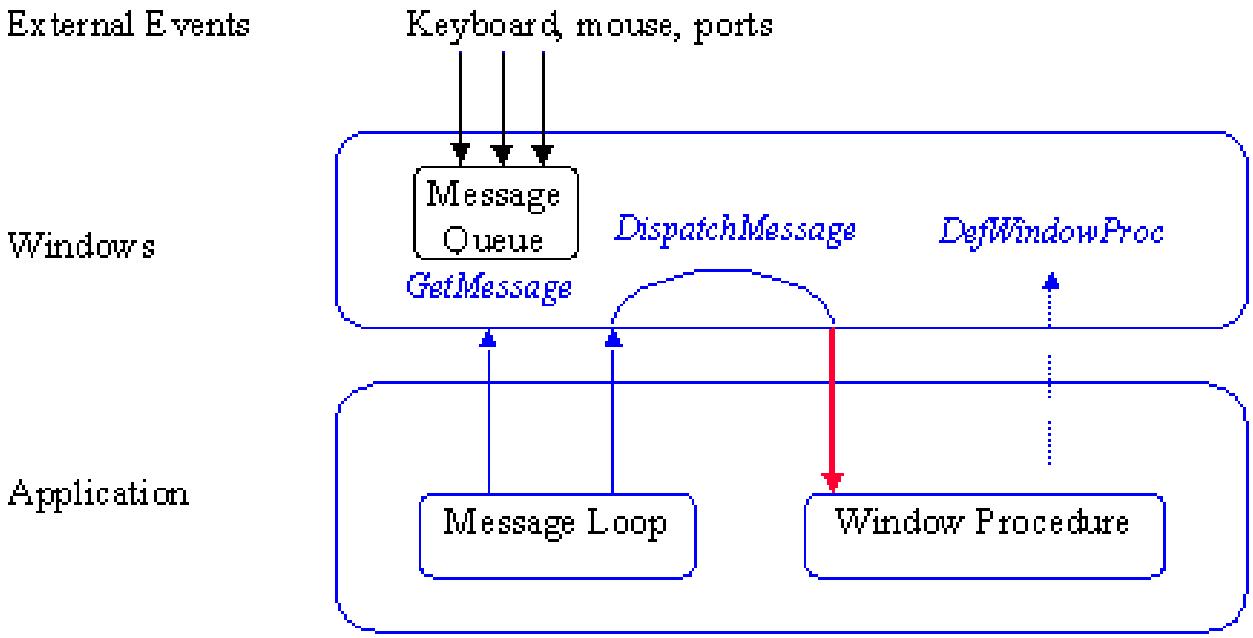




Traditionell command-line

GUI-baserad

Event/Message loop



Händelseloopen – pseudokod

```
int main() {
    return WinMain();
}

WinMain() {
  while (1) {  // loop forever, waiting for an event
    if (event_exists) {     //there is an event, figure out what to do
      if (event == keydown_a) display('user pressed the A key');
      else if (event == window_resize) display('window resized');
      else if (event == repaint) display('need to repaint window');
      else if (event == keydown_escape) exit_program();
    }
  }
}
```

Huvudloopen i Win32 GUI

```
int WINAPI WinMain(HINSTANCE hinstance, HINSTANCE hprev, PSTR cmdline, int ishow)
{
    HWND hwnd;
    MSG msg;
    //initialization code goes here
    while(1) {
          // Get message(s) if there is one
          if(PeekMessage(&msg,hwnd,0,0,PM_REMOVE)) {
                if(msg.message == WM_QUIT)
                            break;
                TranslateMessage(&msg);
                DispatchMessage(&msg);   //this calls the CALLBACK function WinProc()
          } else {
                DrawScene(); //display the OpenGL/DirectX scene
         }
       }
}
```

Huvudloopen i Win32 GUI (2)

```
LRESULT CALLBACK WinProc(HWND hwnd, UINT message, WPARAM wparam, LPARAM
      lparam)
{
   PAINTSTRUCT ps;
    // Depending on the message -- we'll do different stuff
    switch(message) {
                case WM_PAINT:
                            Draw();
                            return 0;
                // ESC will quit program
                case WM_KEYDOWN:   //user pressed a key
                            if(GetAsyncKeyState(VK_ESCAPE))  //it was the escape key
                                        PostQuitMessage(0);         //quit program
                            return 0;
                            case WM_DESTROY:  //windows wants the program to die
                case WM_CLOSE:                        //or the user closed the window
                            PostQuitMessage(0);    //quit program
                             return 0;
    }
    return DefWindowProc(hwnd, message, wparam, lparam);
}
```

GUI koncept

- Widgets – grafiska element som erbjuder någon funktionalitet
  - button, toolbar, menu
- Window – container för widgets
- Child/parent – relationer mellan fönster
- Event / message – fönster ”kommunicerar” via dessa

Programmering i MS Windows

- Historia
  - WIN32 API: grundbibliotek
  - MFC: C++ -kod runt de flesta WIN32 API-funktionerna
- Många makron
- Mycket kod som ”blivit kvar”
- Icke portabelt, icke gratis
- Fungerar dock ganska bra

WxWidgets

- Cross-platform C++ GUI, varför?
  - C++ är populärt bland programmerare
  - objektorienterat
  - exekvering är snabb
  - finns stort antal färdiga komponenter
  - kan även accessera lågnivå utan några problem
  - många större mjukvaruprojekt är skrivna med C++

WxWidgets fördelar

- Det finns ett antal system för cross-platform-utveckling med C++. Fördelen med WxWidgets är
  - 100 % open source kod
  - Liknar mycket Microsoft MFC, gör det lätt att gå från det ena till det andra
  - Funnits sedan 1992, nuvarande version är rätt stabil
  - Finns för många plattformar
    - MS windows
    - De flesta versioner av Linux
    - Macintosh

WxWidgets Hello World

- Skapa en egen klass som ärver wxApp, skriv en funktion OnInit() för denna klass

```
#include <wx/wx.h>
class HelloWorldApp : public wxApp {
public:
   bool OnInit();
};

DECLARE_APP(HelloWorldApp) // Skapar en global att nå denna appl.
IMPLEMENT_APP(HelloWorldApp) // Skapar kod, antingen main() eller
   WinMain() beroende på plattform

bool HelloWorldApp::OnInit() {
   wxFrame *frame = new wxFrame((wxFrame*) NULL, -1, L"Hello World
   Title"); // Skapar ett nytt fönster
   frame->CreateStatusBar(); // lägger till status bar
   frame->SetStatusText(L"Hello World Status Bar"); // Text i status-
   bar
   frame->Show(TRUE); // Visar fönstret
   SetTopWindow(frame); // Sätt till topp-fönster
   return true; // Om returnerar false, avslutas allt genast
}
```

WxWidgets Hello World

- Skapa en egen klass som ärver wxFrame, där specifik funktionalitet för vår fönsterklass implementeras

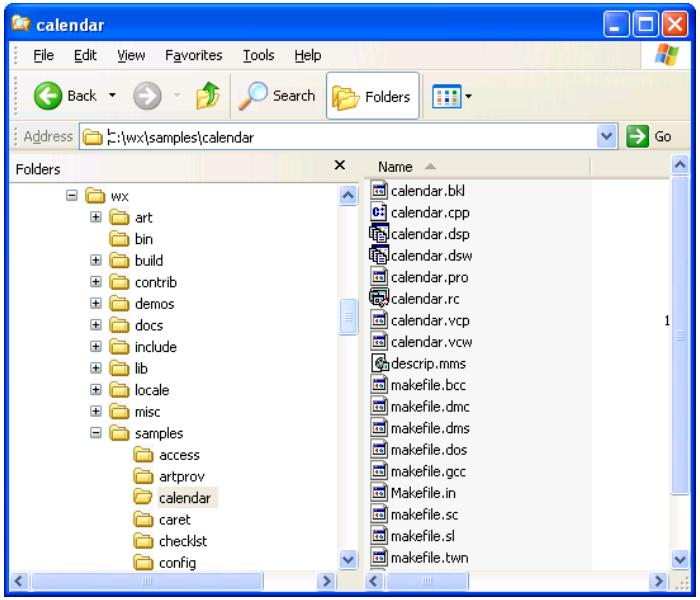
```
class wxMyFrame : public wxFrame {
public:

 wxMyFrame(const wxChar *title, int xpos,
 int ypos, int width, int height);
};

wxMyFrame::wxMyFrame(const wxChar *title, int xpos, int ypos, int
    width, int height)
: wxFrame((wxFrame *) __null, -1, title, wxPoint(xpos, ypos),
    wxSize(width, height))
{}
// I detta fall i praktiken ingenting, vi använder
// direkt grundklassen. OBS, vi ser till att vi använder oss av
// grundklassens konstruktor.
```

Makefile för Linux/Unix

```
// wxhello.mak: Usage:  % make –f wxhello.mak
CXX = $(shell wx-config --cxx)

PROGRAM = wxhello #Obs, detta ändras åtminstone vid nytt prog.

OBJECTS = $(PROGRAM).o

# implementation

.SUFFIXES:      .o .cpp

.cpp.o :
        $(CXX) -c `wx-config --cxxflags` -o $@ $<

all:    $(PROGRAM)

$(PROGRAM):     $(OBJECTS)
        $(CXX) -o $(PROGRAM) $(OBJECTS) `wx-config --libs`

clean:
        rm -f *.o $(PROGRAM)

------------------
Notera användingen av `wx-config` (notera riktningen på `-tecknet
Direkt även
% g++ `wx-config --cxxflags --libs` wxhello.cpp –o wxhello
```

Byggande i Windows

- Enklast: Tag från ”samples”-katalogen en kopia, och börja rader / lägga till filer till denna / skriv ny implementation



Vår wxHello World applikation

Addition av egenskaper

- Text editor i fönstret:
  - Skapa en egen klass som ärver wxFrame, där specifik funktionalitet för vår fönsterklass implementeras

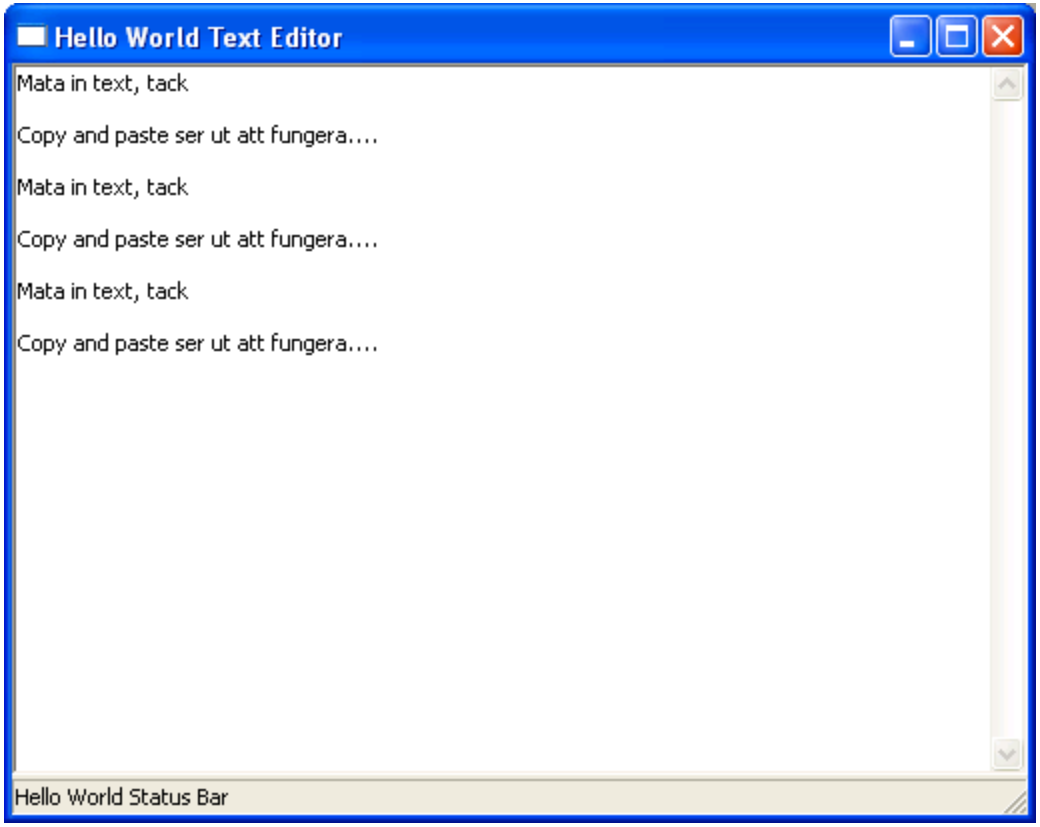
```
class MyFrame : public wxFrame {
public:

 MyFrame(wxWindow *parent, const wxString& title);
private:
    wxTextCtrl *m_pTextCtrl;  // Pekare till text-editor-objektet
};

MyFrame::MyFrame(wxWindow *parent, const wxString& title)
: wxFrame(parent, -1, title)
{
    m_pTextCtrl = new wxTextCtrl(this, -1, wxString("Mata in text,
    tack"),wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);

}
```

Addition av egenskaper

- Måste nu också ändra **`OnInit`**-funktionen tidigare, så att den istället instantierar ett objekt av klassen **`MyFrame`**

```
bool HelloWorldApp::OnInit() {
   MyFrame *frame = new MyFrame(NULL, "Hello World Title"); //
   Skapar ett nytt fönster
   frame->CreateStatusBar(); // lägger till status bar
   frame->SetStatusText("Hello World Status Bar"); // Text i
   status-bar
   frame->Show(TRUE); // Visar fönstret
   SetTopWindow(frame); // Sätt till topp-fönster
   return true; // Om returnerar false, avslutas allt genast
}
```

Addition av egenskaper



Menyer....

- Bara ett text-editor-fönster är något oanvändbart, borde ha fil-access, sköts genom att:
  - 1. Addera menyer
  - 2. Koppla menyerna till funktionalitet

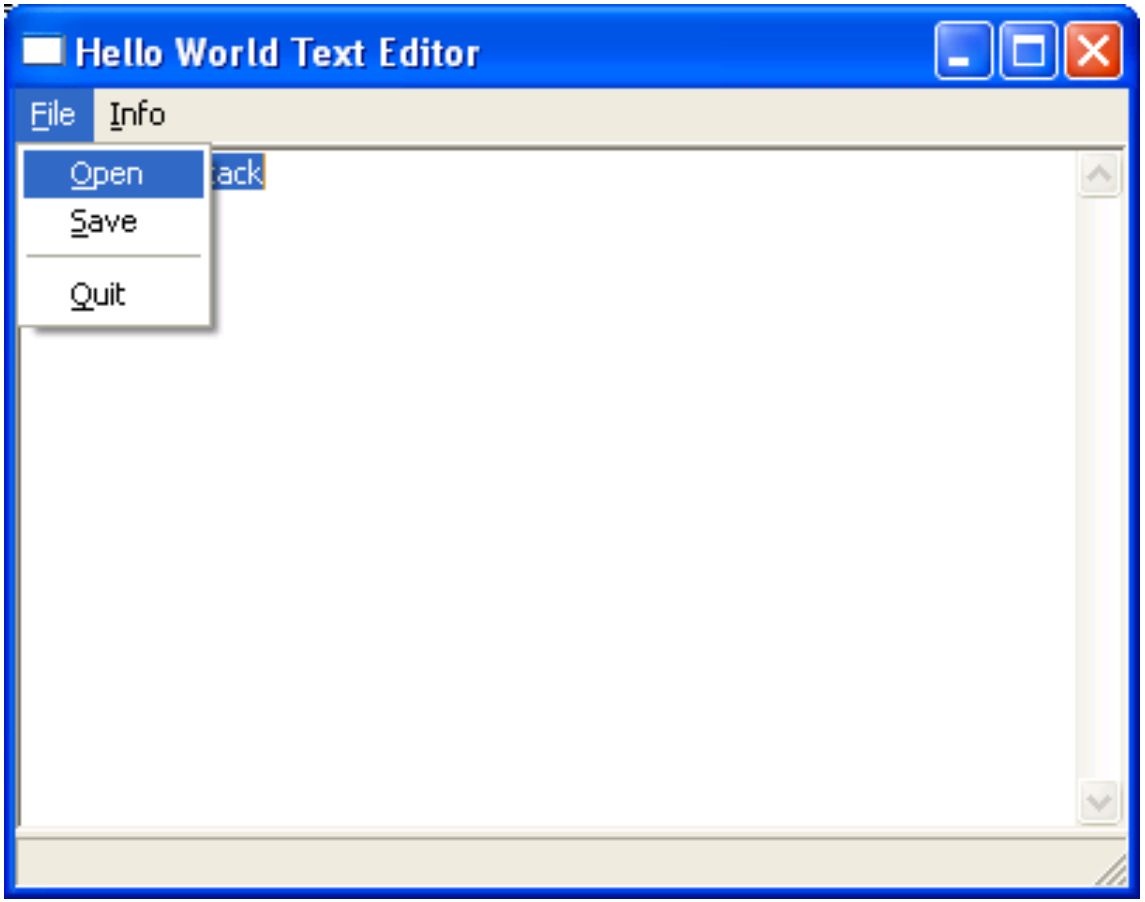
Adderar meny

```
private:
   wxTextCtrl *m_pTextCtrl;
   wxMenuBar *m_pMenuBar;
   wxMenu *m_pFileMenu;
   wxMenu *m_pInfoMenu;
   enum
   {
        MENU_FILE_OPEN,
        MENU_FILE_SAVE,
        MENU_FILE_QUIT,
        MENU_INFO_ABOUT
   };
```

Adderar meny

```
MyFrame::MyFrame(wxWindow *parent, const wxString& title)
: wxFrame(parent, -1, title)
{
   m_pTextCtrl = new wxTextCtrl(this, -1, wxString("Mata in text,
   tack"),wxDefaultPosition, wxDefaultSize, wxTE_MULTILINE);
   m_pMenuBar = new wxMenuBar();
   // File Menu
   m_pFileMenu = new wxMenu();
        // Notera, ampersand (&) ger automatisk keyboard-shortcut
   m_pFileMenu->Append(MENU_FILE_OPEN, "&Open");
   m_pFileMenu->Append(MENU_FILE_SAVE, "&Save");
   m_pFileMenu->AppendSeparator();
   m_pFileMenu->Append(MENU_FILE_QUIT, "&Quit");
   m_pMenuBar->Append(m_pFileMenu, "&File");
   // About menu
   m_pInfoMenu = new wxMenu();
   m_pInfoMenu->Append(MENU_INFO_ABOUT, "&About");
   m_pMenuBar->Append(m_pInfoMenu, "&Info");
   SetMenuBar(m_pMenuBar); //Här sätter vi ny menyrad till fönstret
}
```

Adderar meny



Koppla aktioner till menyn

- Lägger till klass-definition för **`MyFrame`** följande metoder, som ju beskriver vad vi vill göra

```
class MyFrame : wxFrame {
....
public:
void OnMenuFileOpen(wxCommandEvent &event);
void OnMenuFileSave(wxCommandEvent &event);
void OnMenuFileQuit(wxCommandEvent &event);
void OnMenuInfoAbout(wxCommandEvent &event);
....
protected:
DECLARE_EVENT_TABLE() //Klassen har en event table
....
}
```

Koppla aktioner till menyn

- I implementationen används följande makron för att bygga en länkning mellan händelser och motsvarande metoder
  - t.ex. skapas händelsen **`MENU_OPEN_FILE`**, skall man kalla på **`OnMenuFileOpen()`**-metoden

```
BEGIN_EVENT_TABLE(MyFrame, wxFrame)
EVT_MENU(MENU_FILE_OPEN, MyFrame::OnMenuFileOpen)
EVT_MENU(MENU_FILE_SAVE, MyFrame::OnMenuFileSave)
EVT_MENU(MENU_FILE_QUIT, MyFrame::OnMenuFileQuit)
EVT_MENU(MENU_INFO_ABOUT, MyFrame::OnMenuInfoAbout)
END_EVENT_TABLE()
```

Koppla aktioner till menyn

- Sedan måste det finnas någon implementation av meny-funktionaliteten

```
// wxhello.cpp
....
void MyFrame::OnMenuFileOpen(wxCommandEvent &event)
   {SetStatusText("File open aktiverad");}
void MyFrame::OnMenuFileSave(wxCommandEvent &event)
   {SetStatusText("File save aktiverad");}
void MyFrame::OnMenuFileQuit(wxCommandEvent &event)
   {Close(false);}
void MyFrame::OnMenuInfoAbout(wxCommandEvent &event)
   {SetStatusText("About aktiverad");}
....
```

File/open funktionalitet

- 1. Låt användaren öppna en fil
  - Det finns färdiga komponenter för att välja filer: wxFileDialog
- 2. Öppna filen, och läs in innehållet till text-editorn
  - Igen färdiga komponenter, såsom wxFile

wxFileDialog

- Konstruktor:

```
wxFileDialog(wxWindow *parent, const wxString& message = “Choose a
   file”,
   const wxString& defaultDir = “”, const wxString& defaultFile = “”,
   const wxString& wildcard = “*.*”, long style = 0,
   const wxPoint& pos = wxDefaultPosition);
```

- I koden:

```
void MyFrame::OnMenuFileOpen(wxCommandEvent &event)  {
   wxFileDialog *dlg = new wxFileDialog(this, "Open a text file", "",
   "", "All files(*.*)|*.*|Text Files(*.txt)|*.txt", wxOPEN,
   wxDefaultPosition);
   if ( dlg->ShowModal() == wxID_OK) {
         m_pTextCtrl->LoadFile(dlg->GetPath());
         SetStatusText(dlg->GetFilename(), 0);
   }
   dlg->Destroy();
}
```

wxFileDialog (2)

```
void MyFrame::OnMenuFileSave(wxCommandEvent &event) {
   wxFileDialog *dlg = new wxFileDialog(this, "Save a text
   file",
        "", "", "All files(*.*)|*.*|Text Files(*.txt)|*.txt",
        wxSAVE, wxDefaultPosition);
   if ( dlg->ShowModal() == wxID_OK ) {
        m_pTextCtrl->SaveFile(dlg->GetPath());
        SetStatusText(dlg->GetFilename(), 0);
   }
   dlg->Destroy();
}
```

wxMessageBox

- Denna funktion visar ett enkelt meddelande på skärmen
  - Använder här för ”About”-boxen

```
void MyFrame::OnMenuInfoAbout(wxCommandEvent &event) {
   wxMessageBox(”Minimal Hello World Text Editor\nC J
   Bjorkqvist,2005”, ”About Hello World”);
}
```

Rita direkt på klient-området

- När fönster måste ritas på nytt, får fönster en EVT_PAINT-händelse
  - Modifierar EVENT_TABLE, och skriver en funktion som ritar skärmen

```
BEGIN_EVENT_TABLE(MyFrame, wxFrame)
EVT_MENU(MENU_FILE_OPEN, MyFrame::OnMenuFileOpen)
EVT_MENU(MENU_FILE_SAVE, MyFrame::OnMenuFileSave)
EVT_MENU(MENU_FILE_QUIT, MyFrame::OnMenuFileQuit)
EVT_MENU(MENU_INFO_ABOUT, MyFrame::OnMenuInfoAbout)
EVT_MENU(MENU_INFO_ABOUT, MyFrame::OnMenuInfoAbout)
EVT_PAINT(MyFrame::OnPaint)
END_EVENT_TABLE()
```

Rita direkt på klient-området

```
void MyFrame::OnPaint(wxPaintEvent &event) {
   wxPaintDC dc(this);
   dc.SetBackground(*wxGREEN_BRUSH);
   dc.Clear();
}
```

Det är fritt fram att rita:

```
#define PI 3.141592654
void MyFrame::OnPaint(wxPaintEvent &event) {
    wxPaintDC dc(this);
    dc.SetBackground(*wxGREEN_BRUSH);
    dc.Clear();

    wxCoord m,n;
    float rad;

    dc.GetSize(&m, &n);
    for (rad=0; rad< 2*PI; rad+=(float)(2.0*PI/100)) {
          wxCoord x,y;
          x = cos(rad)*m/5+m/2;
          y = sin(rad)*n/5+n/2;
          dc.DrawCircle(x,y, m<n?m/10:n/10);
    }
    wxFont f = dc.GetFont();
    f.SetPointSize(37);
    dc.SetFont(f);
    dc.DrawText("Hello World", 30, 30);
}
```