

# Plattformunabhängige Software mit wxWindows



Vortrag zum  
**2. Leipziger Linux-Workshop**

**Frank-Michael Schleif**  
**[schleif@gaos.org](mailto:schleif@gaos.org)**

## Freiheiten mit Open Source

- Freiheit der kostenfreien Nutzung
- Freiheit die Quellen anzupassen
- Freiheit mit Unabhängigkeit von konkretem Hersteller
- Freiheit der Wahl der Zielplattform (Dank wxWindows)



...?

### wxWindows API

| wxMSW   | wxGTK      | wxX11 | wxMotif       | wxMac             |         | wxOS2 |
|---------|------------|-------|---------------|-------------------|---------|-------|
| WIN32   | GTK+       | Xlib  | Motif/Lesstif | Classic or Carbon | Carbon  | PM    |
| Windows | Unix/Linux |       |               | MacOS 9           | MacOS X | OS/2  |

- **Was ist wxWindows?**
- **Natürlichkeit und Portabilität**
- **Die wxWindows API**
- **wxWindows Gestern – Heute - Morgen**
- **wxWindows Programmierung**
- **wxWindows Toolunterstützung**
- **Zusammenfassung**

**Das populärste, plattformübergreifende  
Open-Source Werkzeug zur Entwicklung von  
Benutzerschnittstellen mit C++**



# Was ist wxWindows ?

---

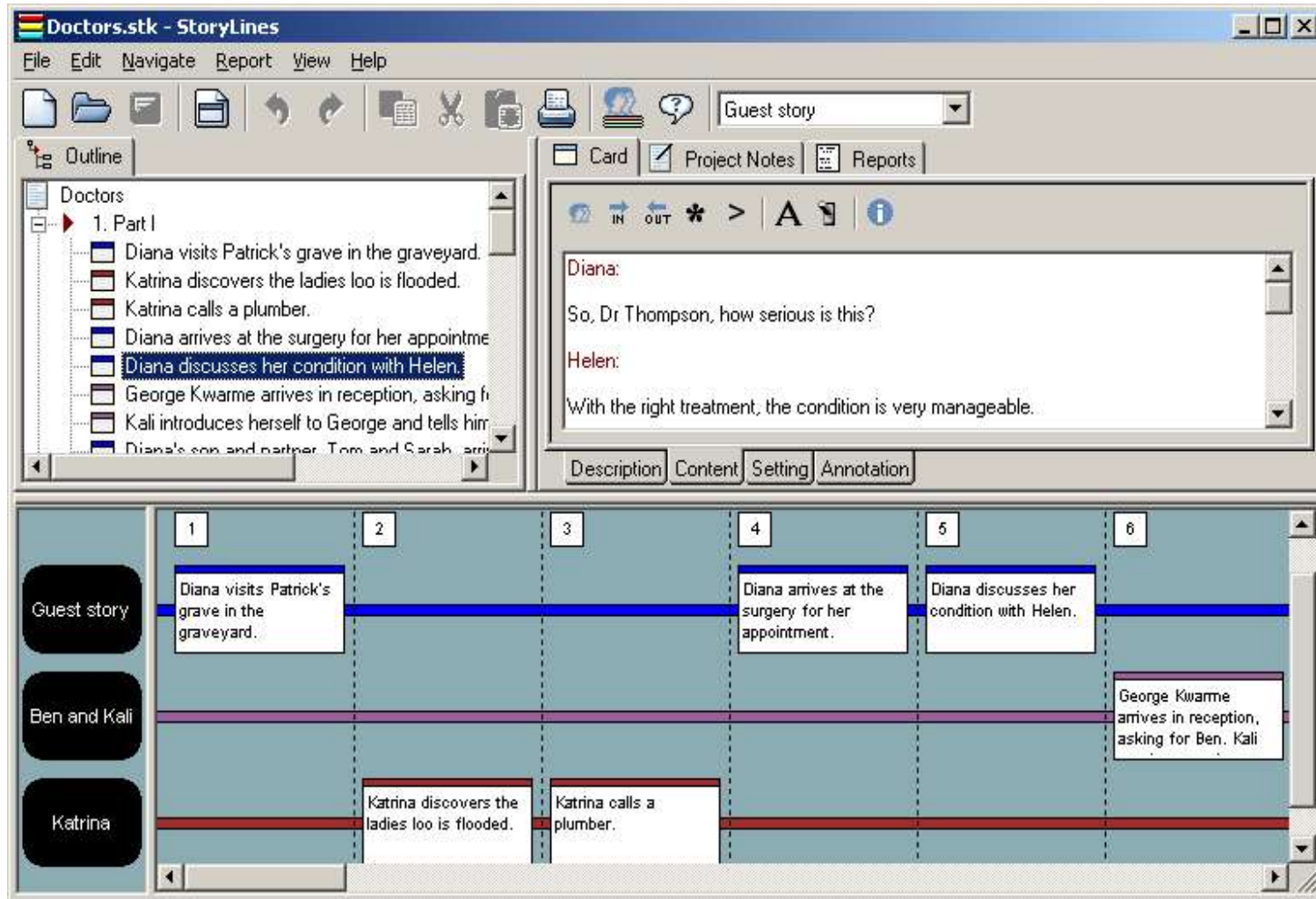
**wxWindows hilft beim Erstellen von plattformunab. Anwendungen durch:**

- Unterstützung verschiedener Plattformen (Unix, Mac, WinX, OS2...)
- Unterstützung von Mehrsprachigkeit
- Native Benutzerschnittstellen (Look & Feel)
- Schnell und effizient (z.B. Codegröße bei Shared-Objekt Nutzung)
- wxWindows ist leicht zu verwenden und einfach zu schreiben
- Flexibles Lizenzmodell frei oder kommerziell
- Langlebig (über 10 Jahre Entwicklungszeit) und robust

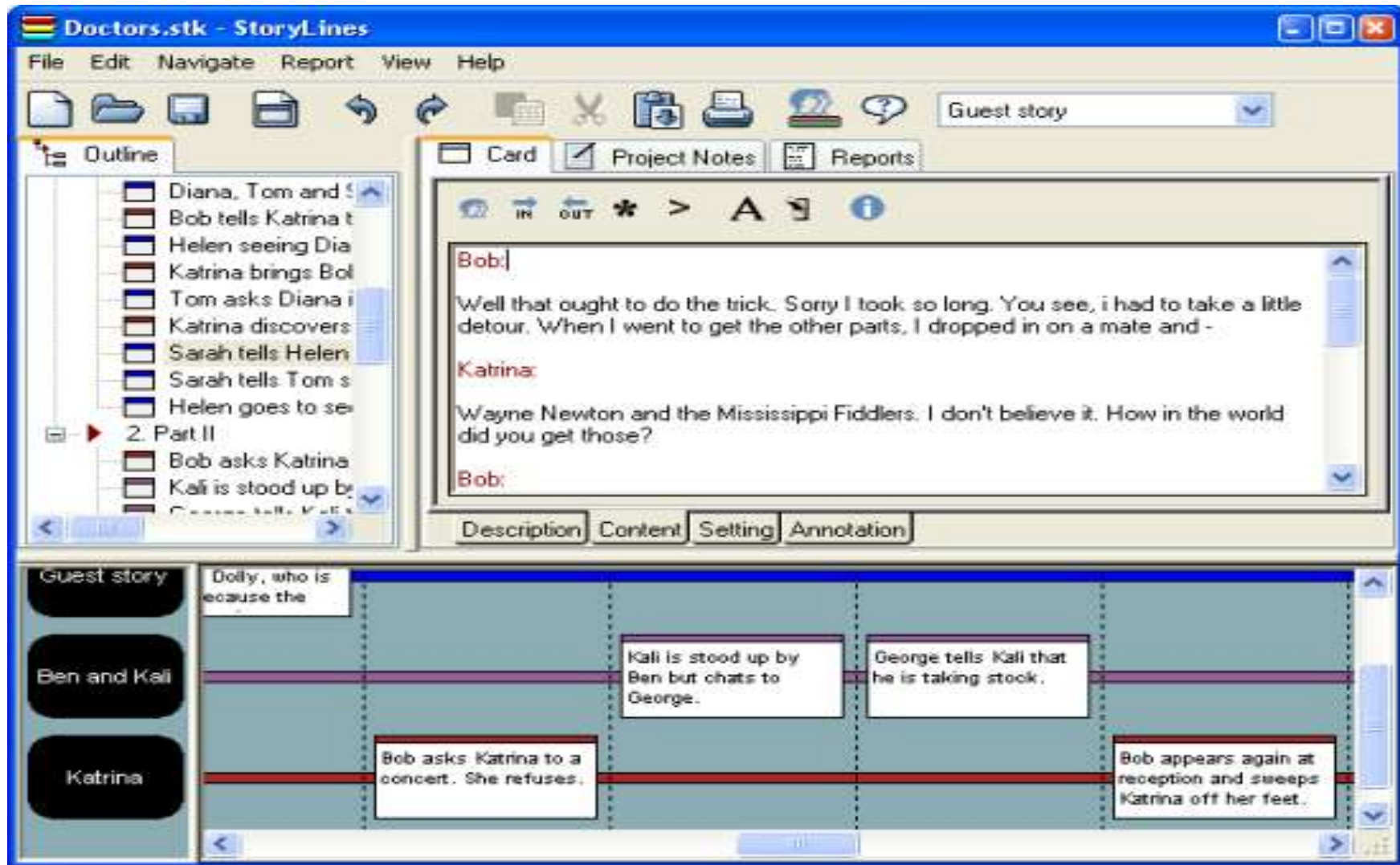
**wxWindows besteht aus:**

- C++ API\*, Menge von Bibliotheken (1/Plattform)
- 1700 - Seiten Dokumentation, über 70 Beispiele
- Help-Viewer und andere Tools
- Große Entwicklergemeinschaft

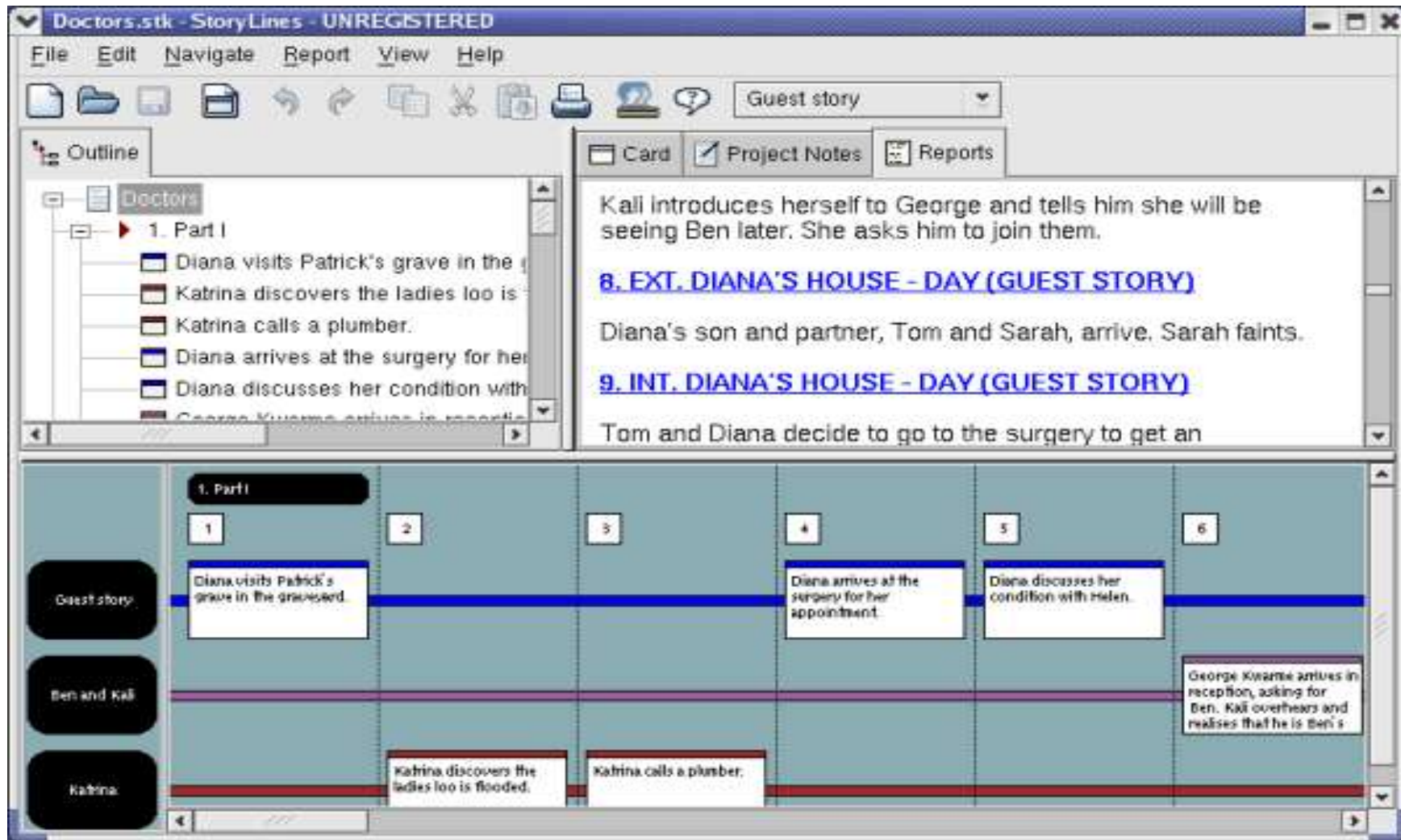
(\* ) auch Binding für Python, Perl, Basic, JavaScript, Lua, Eiffel, .NET



Eine wxWindows Anwendung unter Windows 2000

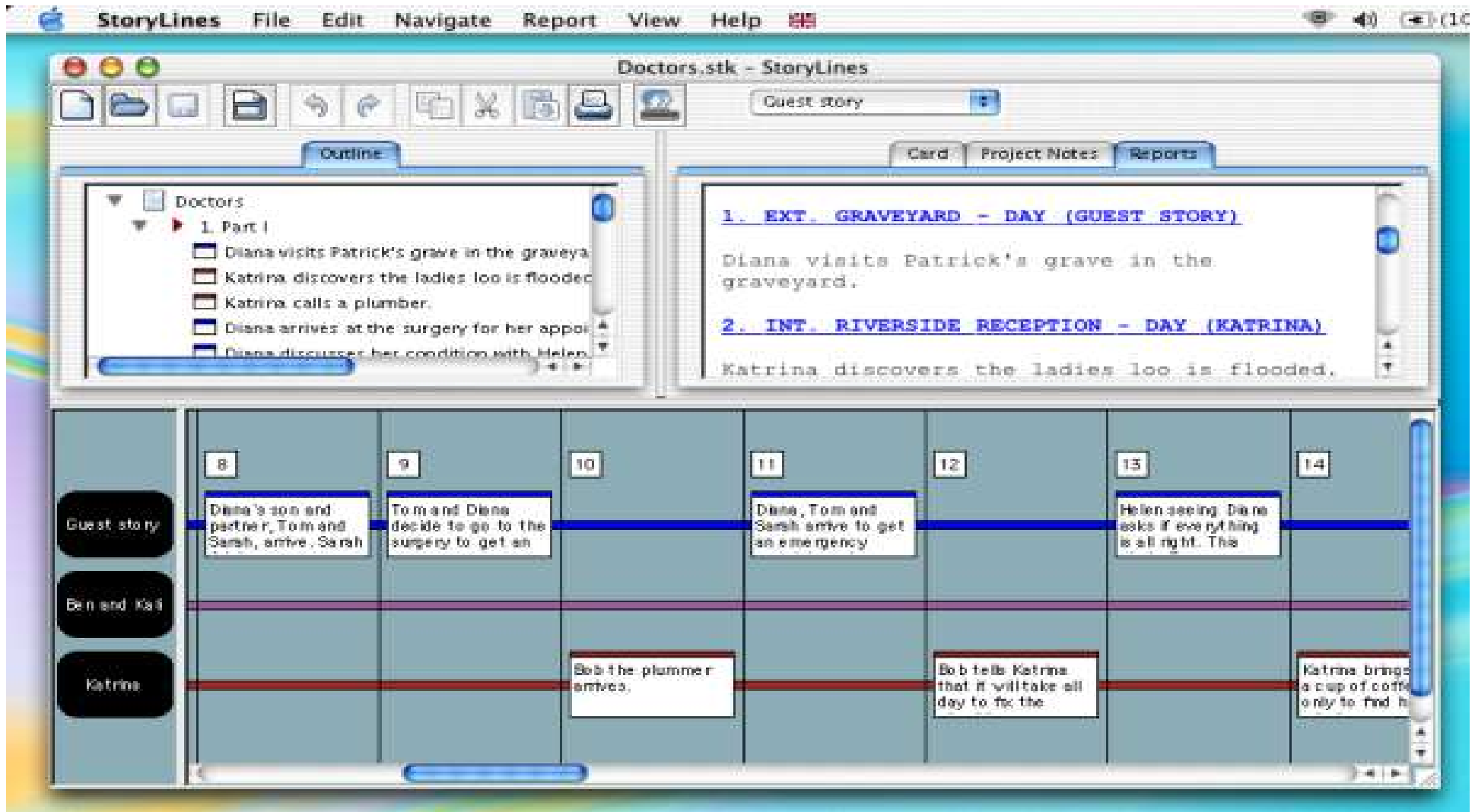


Die gleiche Anwendung unter Windows XP



... und noch einmal unter Red Hat Linux 8.0 mit GNOME

# Plattformunabhängigkeit - wxMac (OS X)



Die gleiche Anwendung unter Mac OS X

## Um das Problem des kleinsten gemeinsamen Nenners zu umgehen:

- wxWindows implementiert fehlende Funktionalität z.B. tree controls, MDI
- oder abstrahiert um die verschiedenartig implementierte Funktionalität zu umfassen; z.B. benutzen die IPC Klassen entweder DDE (Windows) oder TCP/IP (andere Plattformen)
- oder (als letzte Rettung) Einschränkung der Funktionalität auf eine einzelne Plattform z.B. wxMetafile, OLE – wenn plattformübergreifende Reimplementierung zu aufwändig

## Für bessere Portabilität nutzt wxWindows nur einen Teil von C++:

- Keine Exceptions
- Keine Templates (statt dessen pseudo-template container Klassen).  
Man kann aber STL und wxWindows code mischen
- Eigene Streambibliothek

### Gründe:

- Kompiler-, Bibliotheksfehler oder nicht implementierte Merkmale

Mangelnde Unterstützung auf manchen System (z.B. exceptions unter Windows CE)

**Der Programmierer wird vielfältig unterstützt, um portable Software zu entwickeln, ohne dabei auf zu viel Funktionalität verzichten zu müssen:**

- Sizer-basiertes Layout für portable und größenverstellbare Fenster
- Internationalisierung (Meldungskataloge, Unicode, Encoding-Konvertierungen)
- XPM Unterstützung auf allen Plattformen (inline und Laden zur Laufzeit)
- wxImage Klasse zum Handling verschiedenster populärer Bildformate
- Maschinenunabhängige Stream-Klasse
- Unterstützung für fast alle gängigen Compiler
- Unterstützung für verschiedene Hilfeformate
- Plattform-spezifische Funktionalität, z.B. Dateitype-Einstellungen auf dem Mac

# wxWindows API - > 5000 Funktionen in >300 Klassen

## Verwaltete Fenster:

- wxDialog
- wxFrame
- wxMDIParentFrame
- wxMDIChildFrame
- wxMiniFrame
- wxTipWindow
- wxWizard

## Kontainer-Fenster:

- wxNotebook
- wxPanel
- wxSashWindow
- wxScrolledWindow
- wxSplitterWindow
- wxStatusBar
- wxToolBar
- wxMenuBar
- wxMenu

## Erweiterte Fenster:

- wxCalendarCtrl
- wxCheckListBox
- wxDirCtrl
- wxGrid
- wxListCtrl
- wxTreeCtrl

## Gemeinsame Dialoge:

- wxColourDialog
- wxDirDialog
- wxFileDialog
- wxFindReplaceDialog
- wxFontDialog
- wxPageSetupDialog
- wxPrintDialog
- wxMessageDialog
- wxTextEntryDialog

## Einfache Fenster:

- wxBitmapButton
- wxButton
- wxCheckBox
- wxChoice
- wxComboBox
- wxGauge
- wxListBox
- wxRadioButton
- wxRadioBox
- wxScrollBar
- wxSlider
- wxSpinCtrl
- wxStaticBitmap
- wxStaticBox
- wxStaticLine
- wxStaticText
- wxTextCtrl
- wxWindow
- wxControl

## Gerätekontexte:

- wxWindowDC
- wxClientDC
- wxPaintDC
- wxScreenDC
- wxPrinterDC
- wxPostScriptDC
- wxMetafileDC
- wxMemoryDC

## Datentransfer:

- wxDataObject
- wxTextDataObject
- wxFileDataObject
- wxBitmapDataObject
- wxCustomDataObject
- wxClipboard
- wxDropTarget
- wxFileDropTarget
- wxTextDropTarget
- wxDropSource

# wxWindows API - > 5000 Funktionen in >300 Klassen

## Kontainer/Daten Klassen:

- wxDateTime
- wxDateSpan
- wxTimeSpan
- wxHashMap
- wxHashTable
- wxList
- wxLongLong
- wxNode
- wxObject
- wxPoint
- wxRect
- wxRegex
- wxString
- wxStringList
- wxCmdLineParser
- wxVariant

## Thread Klassen:

- wxThread
- wxMutex
- wxMutexLocker
- wxCriticalSection
- wxCriticalSectionLocker
- wxCondition
- wxSemaphore

## Datei Klassen:

- wxFileName
- wxDir
- wxDirTraverser
- wxFile
- wxFFile
- wxTempFile
- wxTextFile

## Verschiedene andere Klassen:

- wxApp
- wxCaret
- wxCmdLineParser
- wxConfig
- wxDllLoader
- wxProcess
- wxTimer
- wxStopWatch
- wxMimeTypeManager
- wxSystemSettings
- wxSystemOptions
- wxAcceleratorTable
- wxAutomationObject
- wxFontMapper
- wxEncodingConverter
- wxQuantize
- wxSingleInstanceChecker
- wxStringTokenizer

- Statisches (Eventtable) oder dynamisches (Connection-Konzept) event routing
- RTTI - eigenes Runtime-Type-Interface, Macro-basiert
- Sizer basiertes Fensterlayout
- wxHTML Rendering und Druck
- Document/view Framework und Command-Processor (für undo/redo)
- Print/Preview Framework
- wxODBC classes (für portablen Datenbankzugriff)
- XML-basiertes laden von UI Ressourcen
- Netzwerk Klassen (sockets, dialup, ftp, ...)
- High-level Interprozesskommunikation (über DDE oder TCP/IP)
- Debugging (logging, tracing, assertion, ...)
- Unterstützung für Tooltips and kontextsensitive Hilfe
- Anwendungskonfiguration mittels (Dateien, Registry, .ini) – je nach Plattform
- Mehrsprachenunterstützung: Meldungskataloge, Unicode

# Projekte um wxWindows herum

## Sprachbindungen:

- wxPython
- wxPerl
- wxBasic
- wxLua
- wxEiffel
- wxJavaScript
- wx.NET (am Anfang)

## Anderes:

- wxMozilla
- wxIE (ActiveX/Internet Explorer)
- Klassen für Sprachprüfung (Joseph Blough)
- IDEs: wxWorkshop, wxHatch, wxGlade, Boa Constructor
- TWAIN/SANE Klassen (Derry Bryson)

# wxWindows Gestern – Heute - Morgen

---

- 1992: 1tes Release, für XView and MFC
- 1993-1995: Motif und Xt Port, wxPython
- 1998-2002: wxGTK Port; wxMac 2.0 Port begonnen, wxX11 Port beg., OS/2 Port
- 2003: wxWindows 2.4.0

## **Stetige Weiterentwicklung und Verbesserung:**

- Zunehmender Einsatz des Unicode-enabled wxGTK Ports in Asien
- wxPython Buch in Vorbereitung, Buch zu wxWindows (Julian Smart)
- Winelib Unterstützung
- Migrationswerkzeuge zum Umstieg von MFC auf wxWindows

## **Wunschliste:**

- Verbesserte Unterstützung des C++ Standards (templates, namespaces, ...)
- Bessere KDE/GNOME Integration
- Komponentenunterstützung
- Ports zu Symbian, PalmOS, Windows CE

# wxWindows Programmierung - Werkzeuge

## Unter Linux:

- Der GCC
- GDB (console, emacs, ddd)
- optional, eine IDE wie z.B. Anjuta
- optional, cross-compiler für Windows



## Unter Windows:

- Eines der folgenden Tools VC++, BC++, MinGW, Cygwin, Watcom C++



## On Mac:

- Für MacOS 8/9, Metrowerks CodeWarrior
- Für MacOS X, CodeWarrior oder die gcc-basierten Apple Developer Tools
- Optional ProjectBuilder IDE mit Apple-Tools



# wxWindows Programmierung - Minibeispiel

```
#include "wx/wx.h"
#include "mondrian.xpm"

// Definition einer neuen wxWindows Anwendungsklasse
class MyApp : public wxApp {
public:
    // Initialisierung
    virtual bool OnInit();
};

// Eigener Frametyp: wird das Hauptfenster
class MyFrame : public wxFrame {
public:
    // ctor(s)
    MyFrame(const wxString& title, const wxPoint& pos,
            const wxSize& size,
            long style = wxDEFAULT_FRAME_STYLE);

    // Event Handler
    void OnQuit(wxCommandEvent& event);
    void OnAbout(wxCommandEvent& event);
private:
    // Deklaration der Eventtable
    DECLARE_EVENT_TABLE()
};
```

```
// Erzeugung eines neuen Anwendungsobjekts
IMPLEMENT_APP(MyApp)

// entspricht dem Einsprungpunkt main
bool MyApp::OnInit()
{
    // Erzeugt das Hauptfenster
    MyFrame *frame =
        new MyFrame(_T("Demo"),
                    wxPoint(50, 50), wxSize(450, 340));

    frame->Show(TRUE);
    return TRUE;
}
```

```
MyFrame::MyFrame(const wxString& title, const wxPoint&
    pos, const wxSize& size, long style)
    : wxFrame(NULL, -1, title, pos, size, style)
{
    SetIcon(wxIcon(mondrian));

    wxMenu *menuFile = new wxMenu;

    wxMenu *helpMenu = new wxMenu;
    helpMenu->Append(Minimal_About, _T("&About...\tF1"),
        _T("Show about dialog"));

    menuFile->Append(Minimal_Quit, _T("E&xit\tAlt-X"),
        _T("Quit this program"));

    // Menüs zur Menübar hinzufügen...
    wxMenuBar *menuBar = new wxMenuBar();
    menuBar->Append(menuFile, _T("&File"));
    menuBar->Append(helpMenu, _T("&Help"));

    // ... die Menübar zum Hauptfenster hinzufügen
    SetMenuBar(menuBar);
}
```

```
BEGIN_EVENT_TABLE(MyFrame, wxFrame)
    EVT_MENU(Minimal_Quit, MyFrame::OnQuit)
    EVT_MENU(Minimal_About, MyFrame::OnAbout)
END_EVENT_TABLE()
```

```
void MyFrame::OnQuit(wxCommandEvent& event)
{
    // schließen des Hauptfensters
    Close(TRUE);
}
```

```
void MyFrame::OnAbout(wxCommandEvent&
    event)
{
    wxMessageBox(_T("Welcome to the demo"), _T(
        "About"), wxOK | wxICON_INFORMATION,
        this);
}
```

Zeichnen in Fenstern, Bitmaps auf Druckern usw, geschieht über Geräte-Kontexte (DCs):  
wxClientDC, wxPaintDC, wxMemoryDC, wxScreenDC, wxPrinterDC, wxPostScriptDC...

Zeichnen eines Rechtecks (200x200px) – schwarz mit roter Außenlinie:

```
BEGIN_EVENT_TABLE(MyWindow, wxWindow)
    EVT_PAINT(MyWindow::OnPaint)
END_EVENT_TABLE()

void MyWindow::OnPaint(wxPaintEvent& event)
{
    wxPaintDC dc(this);

    wxPen pen("RED", 1, wxSOLID);
    wxBrush brush("BLACK", wxSOLID);

    dc.SetPen(pen);
    dc.SetBrush(brush);

    dc.DrawRectangle(0, 0, 200, 200);
}
```

## Problem:

- Wie tauscht man Daten zwischen Controls und Datenstrukturen aus?
- Wie validiert man Eingabedaten?

## Lösung:

- Überschreibe `wxWindow::TransferDataToWindow`, `wxWindow::TransferDataFromWindow`
- Diese Funktionen werden von `OnInitDialog` und `OnOK` Handlern aufgerufen

```
bool MyWindow::TransferDataFromWindow() {
    wxTextCtrl* textCtrl =
        wxDynamicCast(FindWindow(ID_TEXTCTRL), wxTextCtrl);
    wxString str = textCtrl->GetValue();
    if (!OnlyContainsAlpha(str)) {
        wxMessageBox("Enter only alphabetic characters.",
                    "Problem", wxICON_EXCLAMATION|wxOK);
        return FALSE;    }
    m_text = str;
    return TRUE;
}
```

```
// Implementing the transfer explicitly
bool MyWindow::TransferDataToWindow() {
    wxTextCtrl* textCtrl =
        wxDynamicCast(FindWindow
                      (ID_TEXTCTRL), wxTextCtrl);
    textCtrl->SetValue(m_text);
    return TRUE;
}
```

- Oder Verwendung von Validatoren: `wxTextValidator`, `wxGenericValidator`, ...

```
// Using a validator
FindWindow(ID_TEXTCTRL)->SetValidator(wxTextValidator(wxFILTER_ALPHA, & m_text));
```

- Bzw. Erzeugung eigener Validatoren
- Das Äquivalent der MFC's DDX (dynamic data exchange)

**Sizer bestimmen die Größe und Position der controls in einem Dialog, mittels einer Hierarchie, die *separat* von der Fensterhierarchie existiert**

## Vorteile von Sizern:

- ästhetische Dialoge und Panels
- Portable Dialoge
- Text passt in Dialoge, auch nach Übersetzung
- Dialoge sind skalierbar bzgl. ihrer Größe

## Arten von Sizerobjekten:

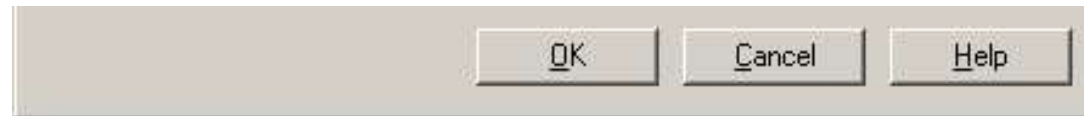
- wxBoxSizer
- wxStaticBoxSizer
- wxNotebookSizer
- wxGridSizer

## Eigenschaften der Sizer:

- Minimale Fenstergröße (spez. Im Fensterkonstrukt)
- Rahmengröße
- Alignment (links, rechts, oben, unten, zentriert)
- Stretchfaktor (wie soll der Platz aufgeteilt werden)

## Arten der Sizer-Erzeugung:

- von Hand (Code oder XRC) – etwas mühselig
- per wxDesigner
- XRCed (wxPython)
- wxGlade (wxPython)



```
wxBoxSizer *item13 = new wxBoxSizer( wxHORIZONTAL );
item13->Add( 0, 0, 1, wxALIGN_CENTRE|wxALL, 5 );
wxButton *item14 = new wxButton( parent, wxID_OK, _("&OK"), wxDefaultPosition, wxDefaultSize, 0 );
item13->Add( item14, 0, wxALIGN_CENTRE|wxALL, 5 );
wxButton *item15=new wxButton(parent,wxID_CANCEL, _("&Cancel"), wxDefaultPosition, wxDefaultSize, 0 );
item13->Add( item15, 0, wxALIGN_CENTRE|wxALL, 5 );
wxButton *item16 = new wxButton( parent, wxID_HELP, _("&Help"), wxDefaultPosition, wxDefaultSize, 0 );
item13->Add( item16, 0, wxALIGN_CENTRE|wxALL, 5 );
```

## Man braucht:

- Projektdatei (hhp)
- Content Datei (hhc)
- Keyword Datei (hhk)
- HTML Seiten

## Unter Verwendung von:

- Tex2RTF
- HelpBlocks
- andere HTML Help-Editoren
- notepad/emacs/vi...

## Verfügbar Hilfecontroller:

- wxWinHelpController
  - wxCHMHelpController
  - wxHtmlHelpController
  - wxExtHelpController (using a browser)
  - wxBestHelpController (CHM or HLP)
- oder einfach wxHelpController  
(default controller für Plattform)

## Verschiedene Sorten von Hilfe:

- Onlinehilfe mit Inhalten, Index, Suche
- Tooltips und kontext-sensitive Hilfe

## Unterstützte Formate:

- MS HTML Help (.chm)
  - MS Windows Help (.hlp)
  - wxWindows HTML Help (.htb)
- die .htb Datei ist ein Zip-Archiv mit .hhp, .hhc, .hhk, und HTML Dateien als Inhalt

```
// Normalerweise ein Mitglied der app Klasse  
wxHelpController m_helpController;
```

```
// Initialisiere den controller mit einer Datei  
m_helpController.Initialise("mymanual");
```

```
// Stellt den Inhalt dar  
m_helpController.DisplayContents();
```

```
// Zeige einen bestimmte Abschnitt (keyword oder Datei)  
m_helpController.DisplaySection("intro.htm");
```

```
// Suche nach einem bestimmten keyword  
m_helpController.KeywordSearch("Introduction");
```

# wxWindows Programmierung - Tooltip

```
// Switching tooltips on
wxToolTip::Enable(TRUE);

// Set the delay in milliseconds
wxToolTip::SetDelay(4000);

// Specifying a tooltip with a toolbar tip
toolBar.AddTool(wxID_OPEN, "Open", bmpOpen,
               "Open file");

// Adding a tooltip to a control
FindWindow(wxID_OK)->SetToolTip("Confirms selection");
```

```
// Spezifiziert eine einfache Hilfe
wxHelpProvider::Set(new wxSimpleHelpProvider);

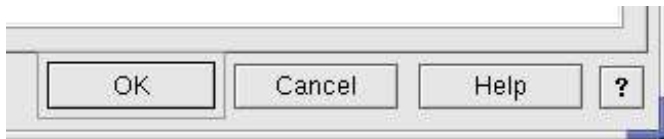
// Hilfe per Hilfe-Controller wo immer möglich
wxHelpProvider::Set(
    new wxHelpControllerHelpProvider(&
        m_helpController));

// Hilfe zu einem Control hinzufügen
FindWindow(wxID_OK)->SetHelpText("Confirms
    selection");

// Aufruf des context-sensitive Hilfemodus
// sendet wxEVT_HELP event zum Fenster
wxContextHelp contextHelp(myWindow);
```

Hilfe ist für jedes Fenster verfügbar.

Kontrolliert durch den aktuellen 'help provider', der das Verhalten eines Hilfefensters implementiert. Unter Windows können Dialoge wxDIALOG\_EX\_CONTEXTHELP benutzen um das ? Icon zu unterstützen.



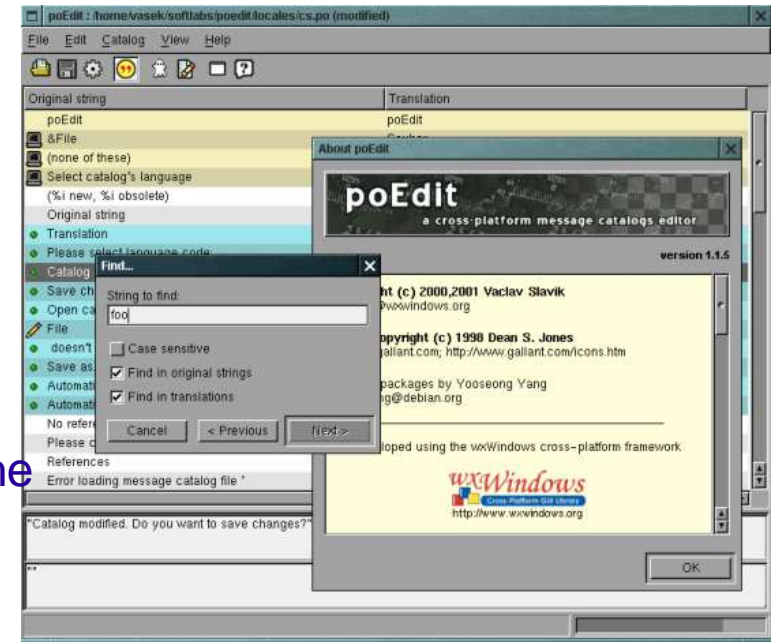
Auf anderen Systemen, erzeugt man einen wxContextHelpButton.

Man kann auch einen **What's this?** Hilfemenu-Eintrag, und ein toolbar Tool.

Diese Kommandos werden durch Erzeugen eines wxContextHelp-Objektes verarbeitet.

## Hinzufügen von I18N:

- 'gettext' package – für Windows, unter: [home.a-city.de/franco.bez/gettext/gettext\\_win32\\_en.html](http://home.a-city.de/franco.bez/gettext/gettext_win32_en.html)
- wrap zu übersetzende Strings mit `_(“”) Makro im Code`
- Nutze z.B. `poEdit`, um Strings zu extrahieren und Messagekatalog aufzubauen eine `(.po Datei)` pro Sprache
- compile `.po` in `.mo` mittels `msgfmt`, und füge es in Speziell bezeichnete Verzeichnisse
- füge das `wxLocale` Objekt zur App-Klasse hinzu --> `init` + hinzufügen des Katalogs



```
// Fügt wxstd.mo Katalog hinzu und macht dieses wxLocale Objekt aktiv  
m_locale.Init(wxLANGUAGE_FRENCH);
```

```
// Fügt Katalog für eigene Anwendung hinzu  
m_locale.AddCatalog(wxT("myapp"));
```

## Initialisiere drag and drop

1. **Vorbereitung:** Ein Datenobjekt mit Dragdaten muss erzeugt und initialisiert werden z.B.:

```
wxTextDataObject my_data("This text will be dragged.");
```

2. **Drag start:** Um das Dragging zu starten (z.B. auf Mausklick) ruft man wxDropSource::DoDragDrop auf:

```
wxDropSource dropSource( window );  
dropSource.SetData( my_data );  
wxDragResult result = dropSource.DoDragDrop( TRUE );
```

3. **Dragging:** Der Aufruf von DoDragDrop() blockiert das Programm bis der Nutzer den Mausknopf los lässt.

DoDragDrop() returns an effect code:

```
switch (result)  
{  
    case wxDragCopy: /* copy the data */ break;  
    case wxDragMove: /* move the data */ break;  
    default: /* do nothing */ break;  
}
```

## Implementierung des Dropziels

```
class DnDText : public wxDropTarget {
public:
    DnDText(wxListBox *pOwner) { m_pOwner = pOwner; SetDataObject(new wxTextDataObject); }

    virtual wxDragResult OnData(wxCoord x, wxCoord y, wxDragResult def)
    {
        if ( !GetData() )
            return wxDragNone;

        wxTextDataObject *dobj = (wxTextDataObject *)m_dataObject;
        m_pOwner->Append(dobj->GetText());
        return def ;
    }

private:
    wxListBox *m_pOwner;
};
...
m_ctrlText = new wxListBox(this, -1, pos, size, 0, NULL);
m_ctrlText->SetDropTarget(new DnDText(m_ctrlText));
...
```

## Einige Hinweise zur portablen Programmierung mit wxWindows:

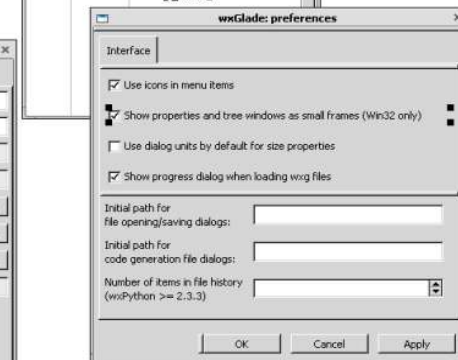
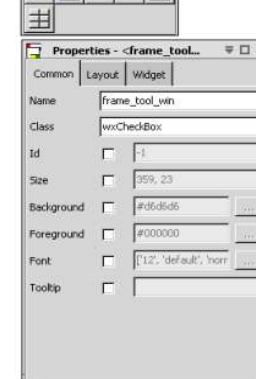
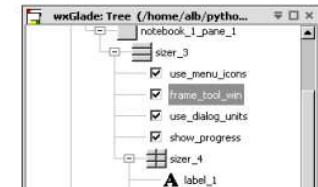
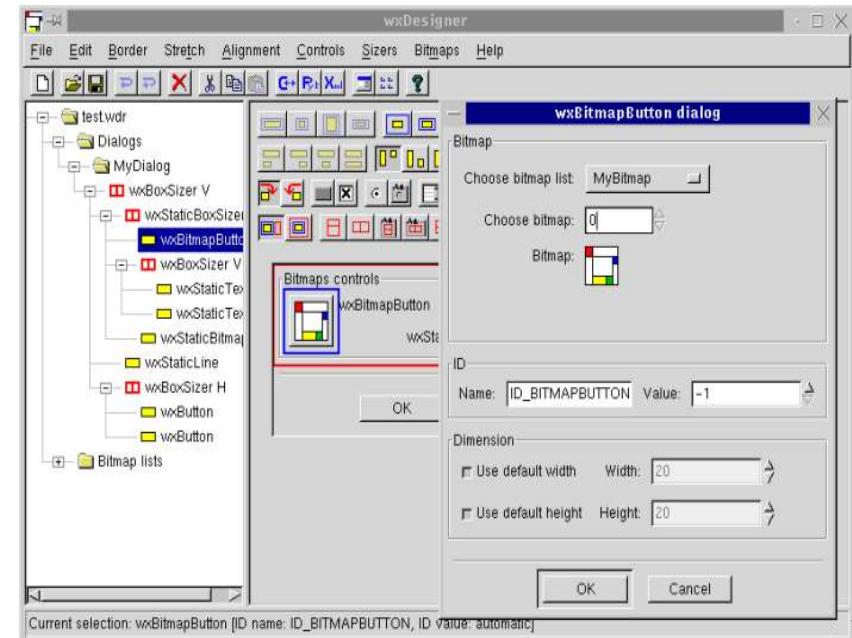
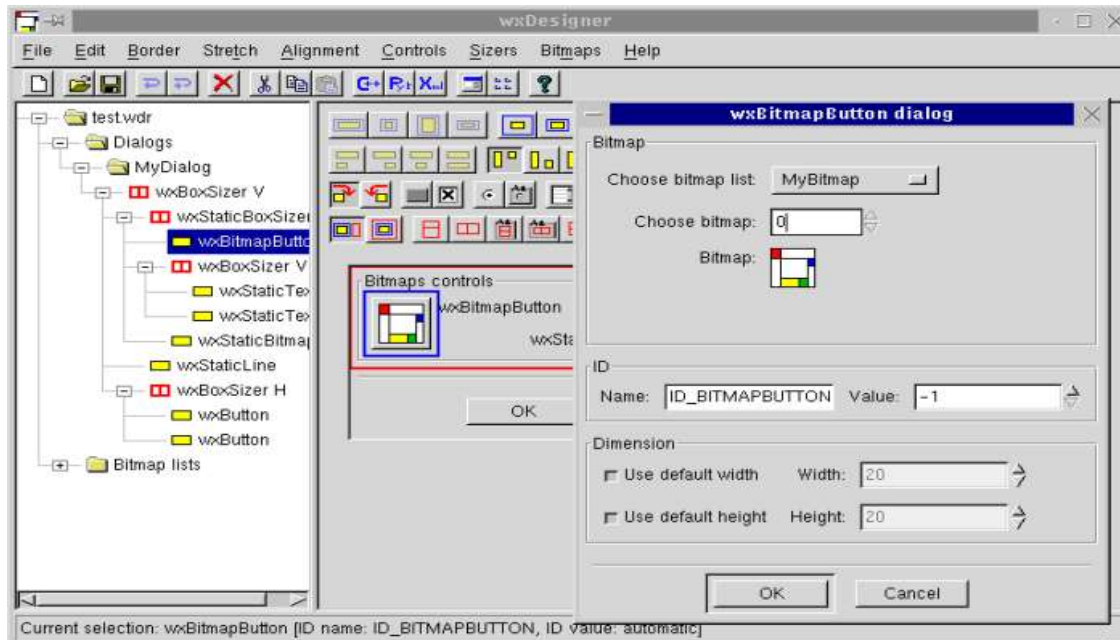
- Teste die Anwendung regelmäßig auf den verschiedenen Ziel-Plattformen
- Lerne, welches Anwendungsverhalten die Benutzer auf der jeweiligen Plattform erwarten
- Hin und wieder wird bedingte Compilierung mittels Präprozessordirektiven nötig
- Verwende Sizer und einen der verfügbaren Dialogeditoren
- Man wird dennoch ganz sicher einen Debugger brauchen :-)
- Schau ins wxWindows Wiki, die Doku, die FAQ-Seiten und die Beispiele

# Toolunterstützung für wxWindows

wxDesigner von Robert Roebling

wxGlade von Alberto Griggio

wxHatch von Chris Elliott



## Weiter Tools:

- **poEdit:** Nachrichtenkatalog-Editor, von Vaclav Slavik
- **convertrc:** RC zu XRC Ressourcenkonverter
- **wxrc:** XRC Ress. Konv. zu C++ oder kompr. Format
- **Tex2RTF:** Dokumentationswerkzeug
- **HelpView:** für das Anzeigen von Doks. oder als Hilfeviewer
- **Anjuta, KDevelop:** - Entwicklungsumgebungen die wxWindows-Projekte unterstützen

# Zusammenfassung

---

- Natives Look & Feel
- Umfangreiche API
- Ideal zur Migration von MFC Code
- Has increasing 'mindshare'

**Helft dem Projekt oder schreibt einfach  
exzellente Multiplattform-Anwendungen!**

<http://www.wxwindows.org>