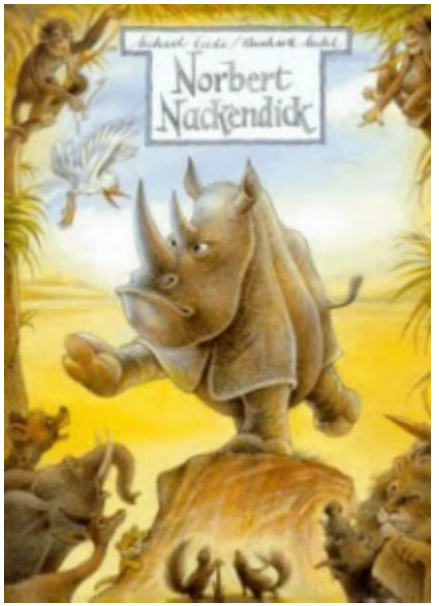
WxPerl ohne Hürden

Eine Einleitung für Programmierer.

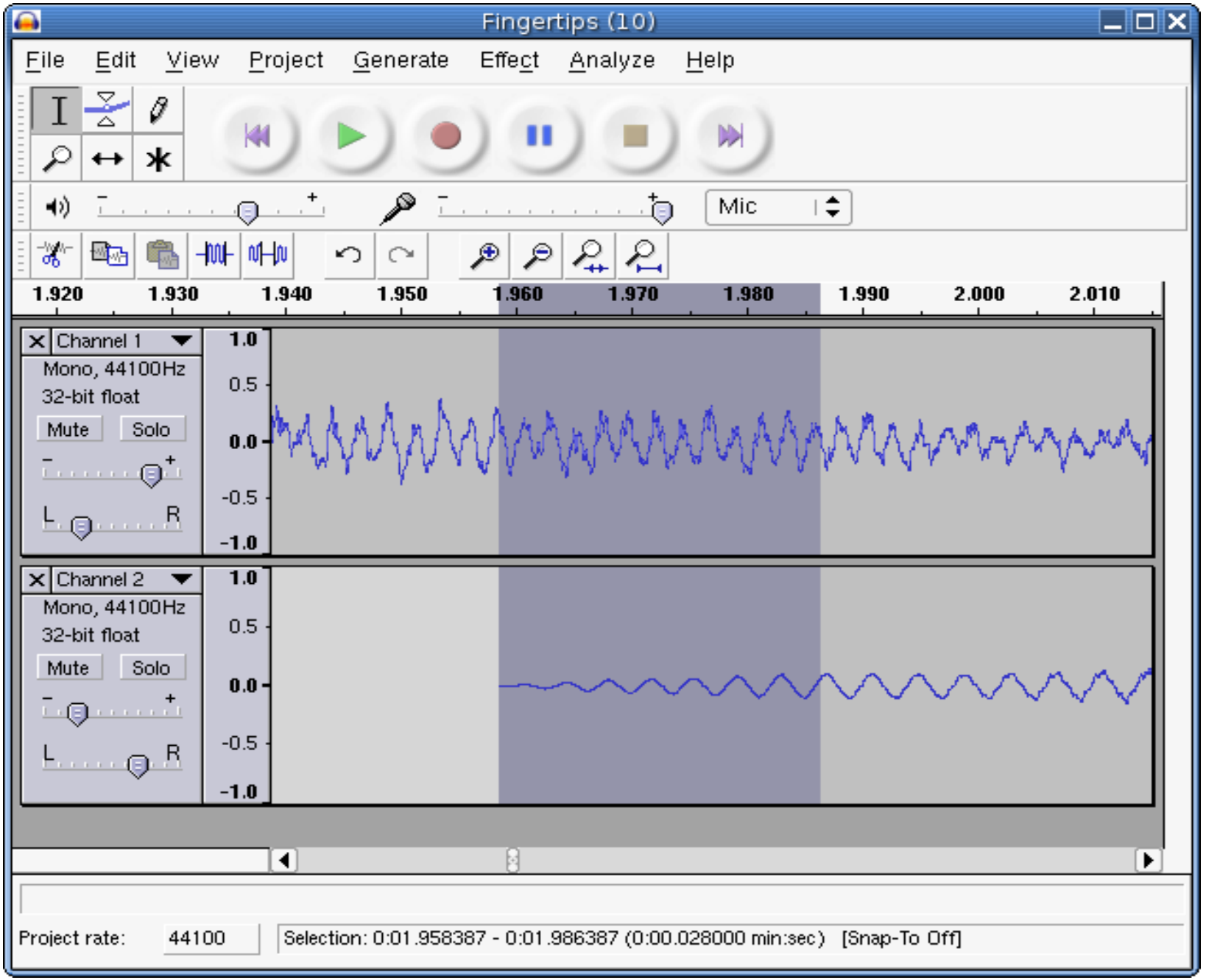


Perl für Applikationen

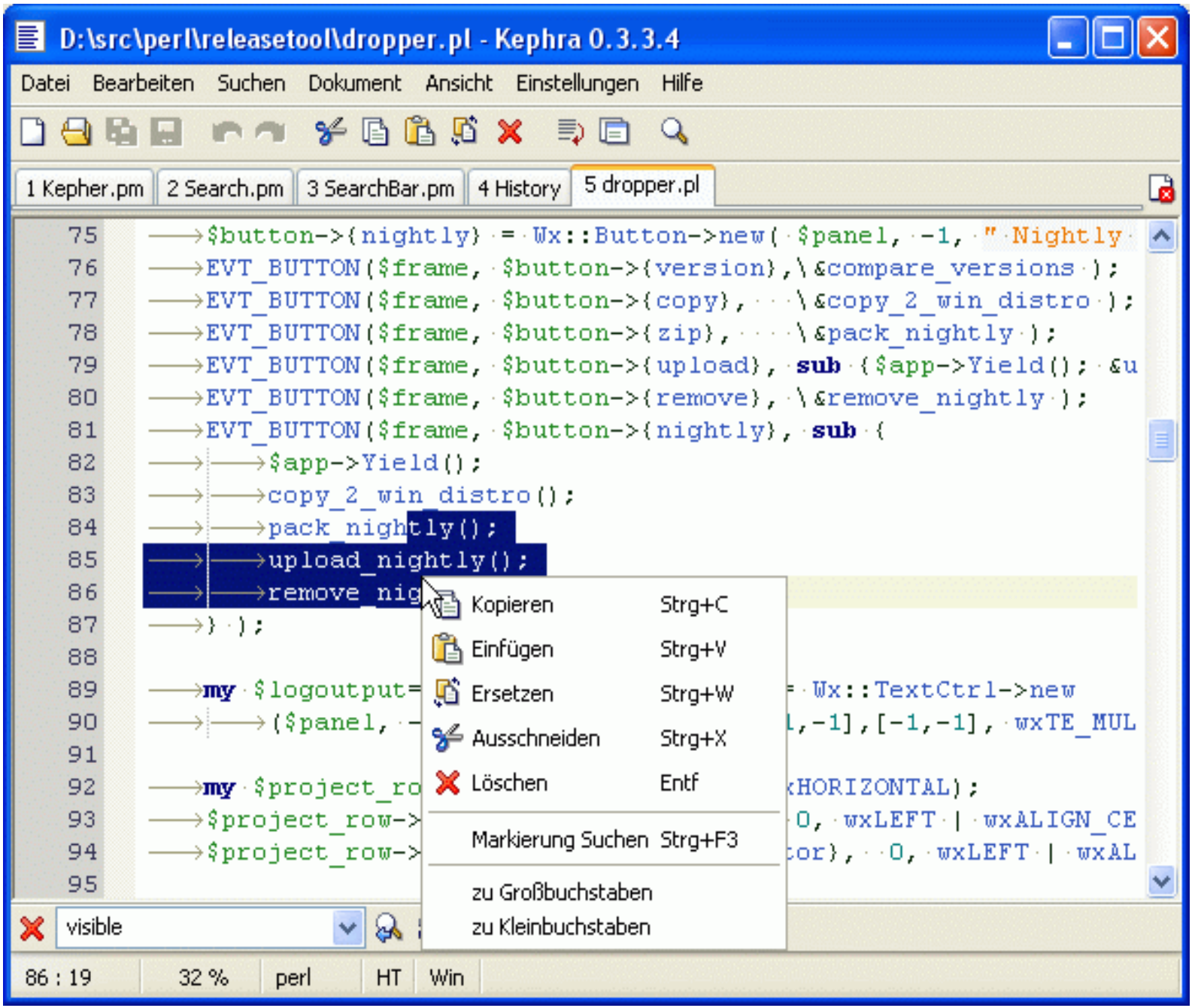
- Perl ist eure Lieblingssprache
  - (ihr wisst warum)
  - schnelle Entwicklungszeit, mächtig, vielseitig, ...

Perl für Applikationen

- Perl ist eure Lieblingssprache
  - (ihr wisst warum)
  - schnelle Entwicklungszeit, mächtig, vielseitig, ...
- auch sehr gut für Apps
  - lange keine Scriptsprache mehr
  - nicht so umständlich wie Java
  - macht viel Freude, Anerkennung der User
  - nur die Oberfläche stimmt noch nicht ganz



Audacity, ein WxWidgets Programm



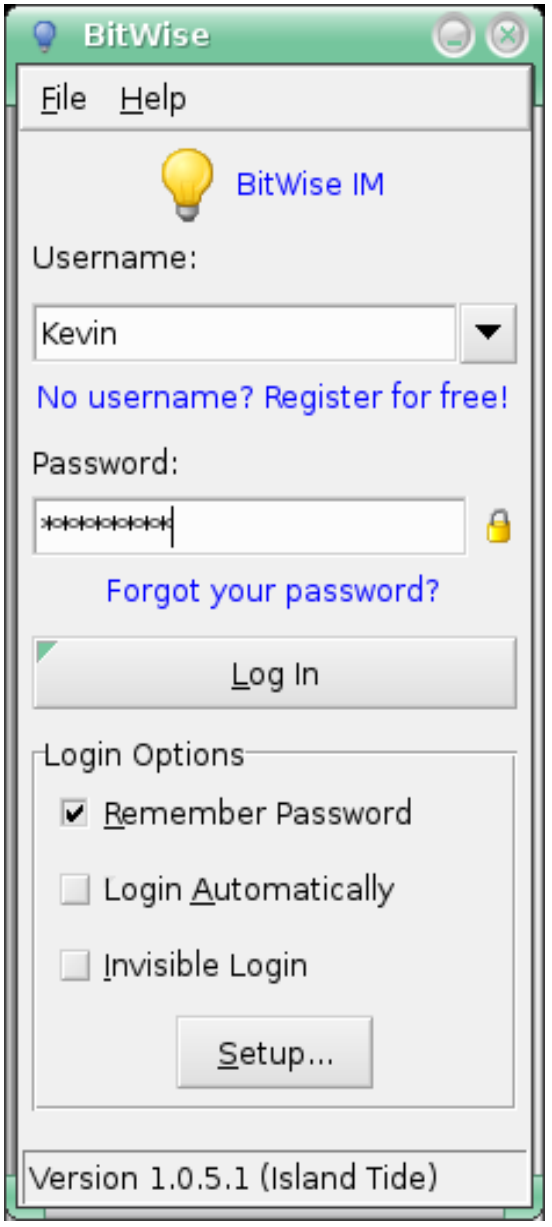
Kephra, ein WxPerl Programm

Perl für Applikationen

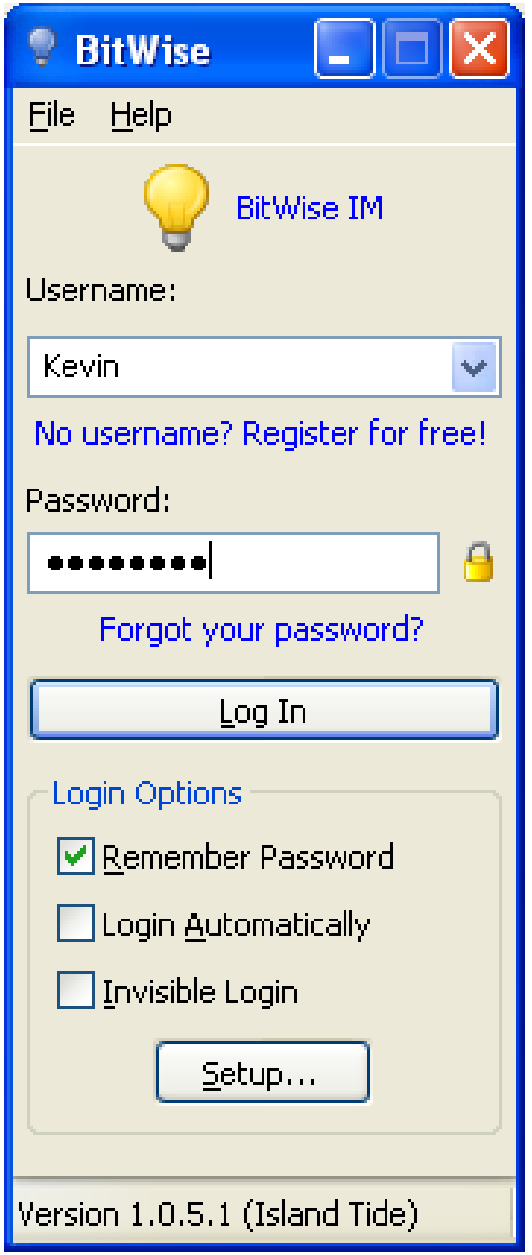
- Daß könnt ihr auch, mit 

  - nicht nur professionell aussehend
    - sondern auch OS-unabhängig

native crossplatform



native crossplatform



native crossplatform

Warum WxWidgets ?

- native Optik, Verhalten – crossplatform
  - für Mac[9|X], *nix(GTK+), Win [95..xp]
- schnell, sparsam, stabil, ausgereift (12J.)
- mächtig für GUI & mehr in konsistenter API
- verläßlich : LGPL, Docs, Gemeinschaft, Planung

Und was ist WxPerl ?

- Perl XS Wrapper für 

- ein CPAN Modul / Sourceforge Projekt
- Subset von WxWidgets denn:
  - Arrays, Regexes etc. hat Perl selber
  - Dinge wie net / socket sind im CPAN besser
  - minderwichtiges fehlt

Wo ist der Haken ?

- etwas umständlicher zu programmieren
- grosser Kern, weniger Zusatzmodule
- WxPerl ist 1 Mann – Projekt (Mattia Barbon)
- wird grad bekannter, stabilisiert sich noch

kleiner Vergleich mit Tk

- für große C++ Apps
- funktionsnahe API, MFC
- DB, Parser, Zip ...
- 8 / 1,6MB core/contrib
- gr. Kern & mehr funkt.
- wenige contribs
- Wrapper um OS call
- OS spez. feature

- von TCL Macher
- simple API
- nur GUI
- 0,8 / 1,1 MB
- kl. Kern
- viele kl. Zusatzmodule
- Emulation
- überall gleich

Wann WxPerl ?

- wenn Optik entscheidend ist
  - kommerzielle Projekte, Benutzerapps
- wenn Tk Funktionalität nicht reicht (threads, events)
- bei grösseren Projekten (Strukturen nutzen)
- wenn genug RAM da ist
  - (Kephra braucht min. 25MB)

Fangen wir an ?

Was brauch ich ?

- natürlich Wx als ppm oder über CPAN-shell
  (jetzt auch per Alien::Wx)
- offizielle WxWidget Doku (incl. Perldelta)
- Demos & Beispiele

(extra download auf wxperl.sf.net der sich lohnt)

- WxGlade(Perl) oder DialogBlocks (XRC
  generieren oder zum testen der feature)
- wxperl.de (da wollten wir mal was aufbauen)

Grundzüge der API

Grundzüge der API

- C++ : strikte OOP, fast alles ist ein Objekt
  - Ausnahmen : Infoboxen, etc
  - WxPerl - eigene Ausnahmen : $frame->GetWH

Grundzüge der API

- C++ : strikte OOP, fast alles ist ein Objekt
  - Ausnahmen : Infoboxen, etc
  - WxPerl - eigene Ausnahmen : $frame->GetWH

- positionale Parameter,
  - wenige Schemata, von mehr bis weniger wichtig
  - nicht gebrauchtes weglassen, Standartwerte

Grundzüge der API

- C++ : strikte OOP, fast alles ist ein Objekt
  - Ausnahmen : Infoboxen, etc
  - WxPerl - eigene Ausnahmen : $frame->GetWH
- positionale Parameter,
  - wenige Schemata, von mehr bis weniger wichtig
  - nicht gebrauchtes weglassen, Standartwerte
- Konstanten definieren Eigenschaften
  - addierbar : wxLEFT | wxBOTTOM

Auf den Weg

Mutterklasse jeder App

- nennt sich Wx::App
- jede WxApp wird von ihr abgeleitet
  - our @ISA = 'Wx::App';
  - use base 'Wx::App';

Mutterklasse jeder App

- nennt sich Wx::App
- jede WxApp wird von ihr abgeleitet
  - our @ISA = 'Wx::App';
  - use base 'Wx::App';
- ist in use Wx; enthalten

Mutterklasse jeder App

- nennt sich Wx::App
- jede WxApp wird von ihr abgeleitet
  - our @ISA = 'Wx::App';
  - use base 'Wx::App';
- ist in use Wx; enthalten
- Methode OnInit {} unbedingt überschreiben
  - Vielleicht auch OnExit {}
  - OnInit {1;} # Rückgabewerte nicht vergessen

Mutterklasse jeder App

- nennt sich Wx::App
- jede WxApp wird von ihr abgeleitet
  - our @ISA = 'Wx::App';
  - use base 'Wx::App';
- ist in use Wx; enthalten
- Methode OnInit {} unbedingt überschreiben
  - Vielleicht auch OnExit {}
  - OnInit {1;} # Rückgabewerte nicht vergessen
- Aufruf : AppName->new->MainLoop;

das erste Fenster

- Methode OnInit {} des App package
- Wx::Window besser noch Wx::Frame->new()
- new($parent, $id, $title,\@pos,\@size, $style, $name)

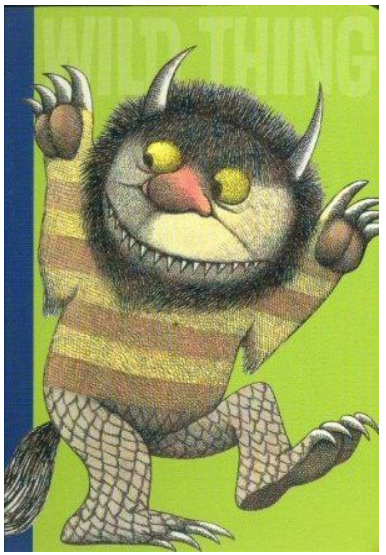
das erste Fenster

- Methode OnInit {} des App package
- Wx::Window besser noch Wx::Frame->new()
- new($parent, $id, $title,\@pos,\@size, $style, $name)
- $parent = undef; # besitzt kein eltern fenster
- $id = -1; # default, hier wie autoincrement
- $title = 'Peace Universe';
- $pos = $size = [-1,-1];# wxDefaultPosition wxDefaultSize
- $style = wxDEFAULT_FRAME_STYLE;

das erste Fenster

```
use strict;
use warnings;

MeinProgramm->new->MainLoop;

package MeinProgramm;
use base qw( Wx::App );
use Wx;

sub OnInit {
    Wx::Frame->new( undef, -1,
        'Titel', [-1,-1], [-1, -1], wxDEFAULT_FRAME_STYLE)
}
```

das erste Fenster

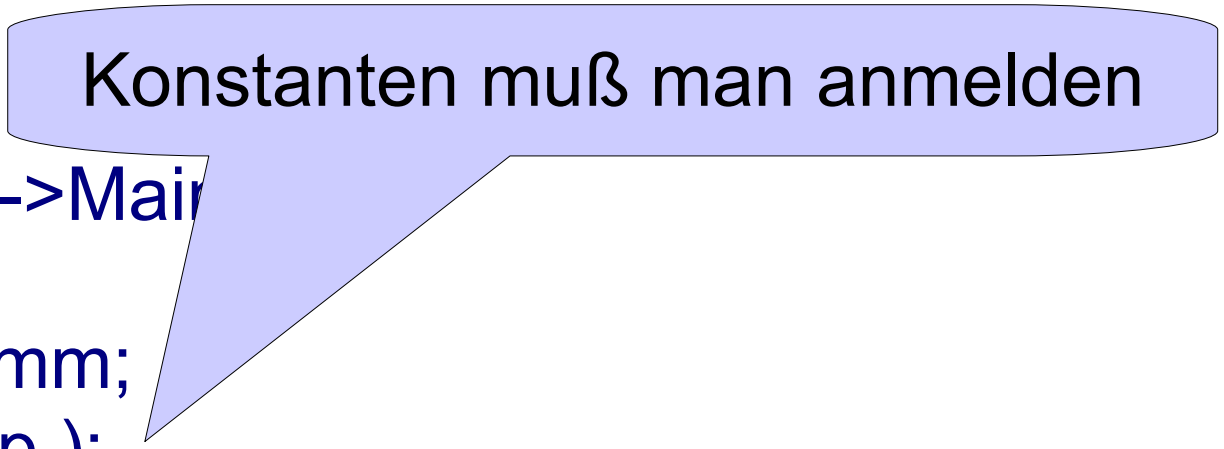



```
MeinProgramm->new->MainLoop;

package MeinProgramm;
use base qw( Wx::App );
use Wx;

sub OnInit {
    Wx::Frame->new( undef, -1,
        'Titel', [-1,-1], [-1, -1], wxDEFAULT_FRAME_STYLE)
}
```

>Bareword "wxDEFAULT_FRAME_STYLE" not allowed while "strict subs" in use at ...

das erste Fenster

```
use strict;
use warnings;

MeinProgramm->new->Mai

package MeinProgramm;
use base qw( Wx::App );
use Wx qw( wxDEFAULT_FRAME_STYLE );

sub OnInit {
    Wx::Frame->new( undef, -1,
        'Titel', [-1,-1], [-1, -1], wxDEFAULT_FRAME_STYLE)
}
```



das erste Fenster

```
use strict;
use warnings;

MeinProgramm->new->Ma

package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );          # :all wäre kürzer

sub OnInit {
    Wx::Frame->new( undef, -1,
        'Titel', [-1,-1], [-1, -1], wxDEFAULT_FRAME_STYLE)
}
```

melden wir mal alle an,
damit das nicht nochmal passiert

das erste Fenster

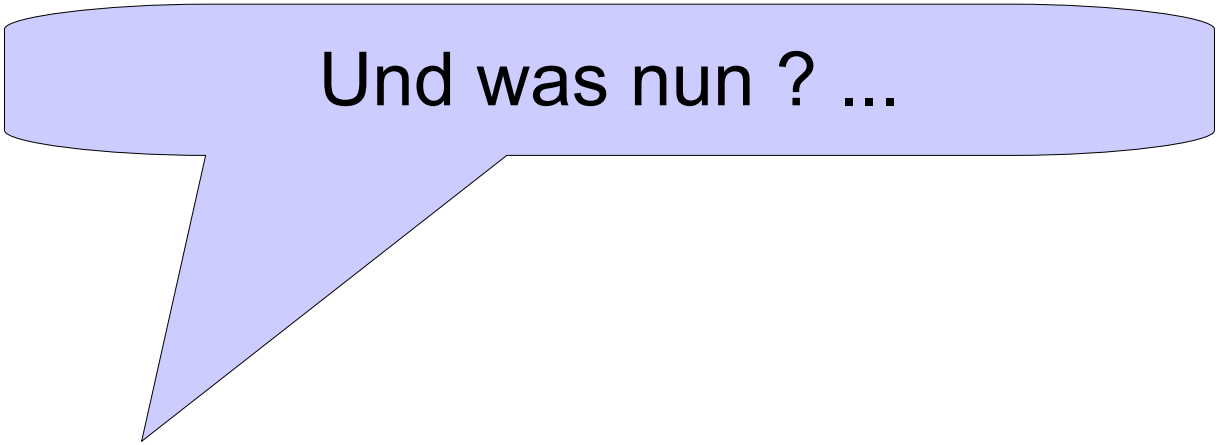
```
use strict;
use warnings;

MeinProgramm->new->MainLoop;

package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );

sub OnInit {
    Wx::Frame->new( undef, -1,
        'Titel', [-1,-1], [-1, -1], wxDEFAULT_FRAME_STYLE)
}
```

alles was default ist,
kann man auch weglassen

das erste Fenster

```
use strict;
use warnings;

MeinProgramm->new->MainLoop;

package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );

sub OnInit {
    Wx::Frame->new( undef, -1, 'Titel' )
}
```

das erste Fenster



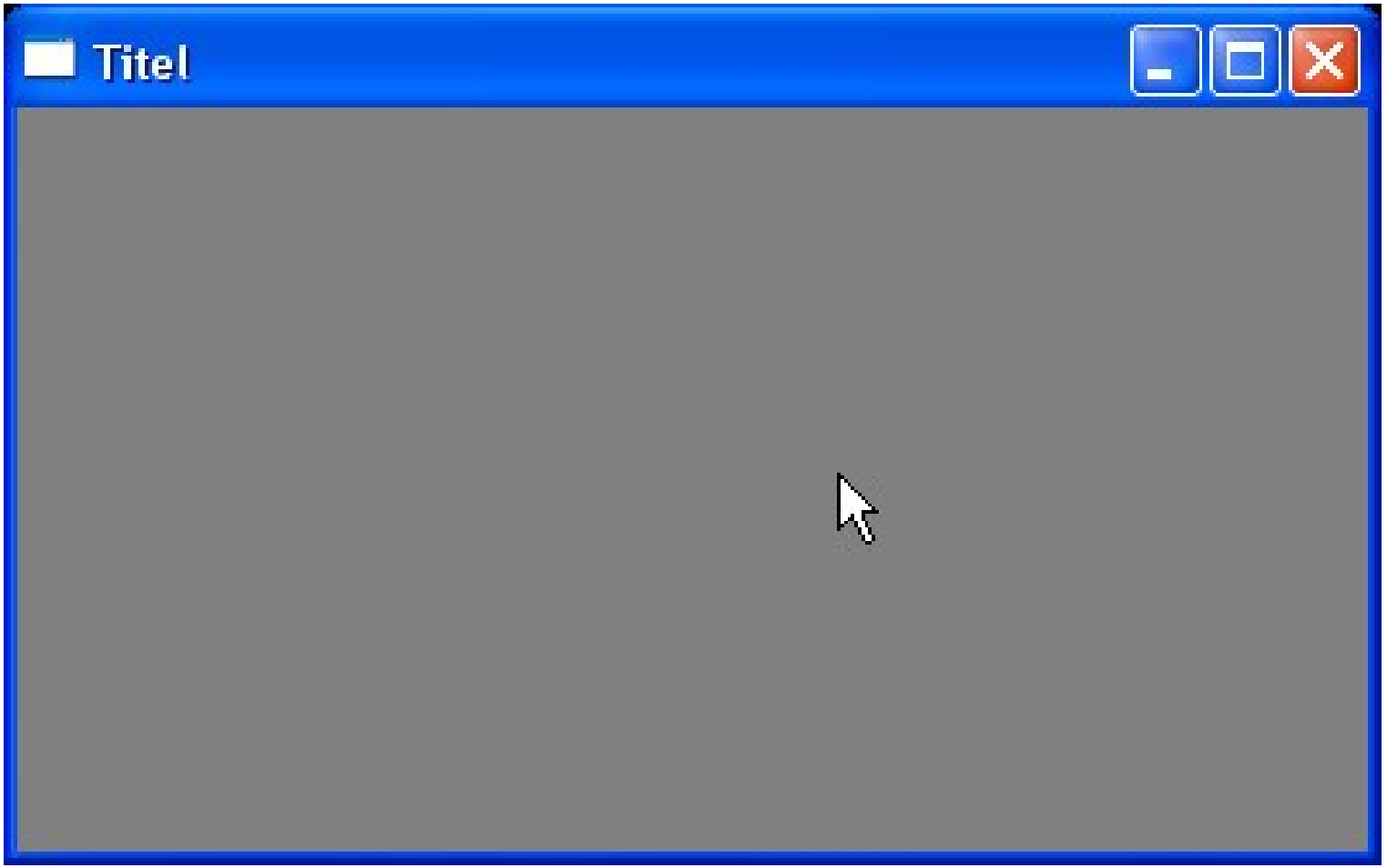
>

das erste Fenster

```
use strict;
use warnings;

MeinProgramm->new->MainLoop;

package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );

sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    $fenster->Show(1);
}
```

Show must go on ...

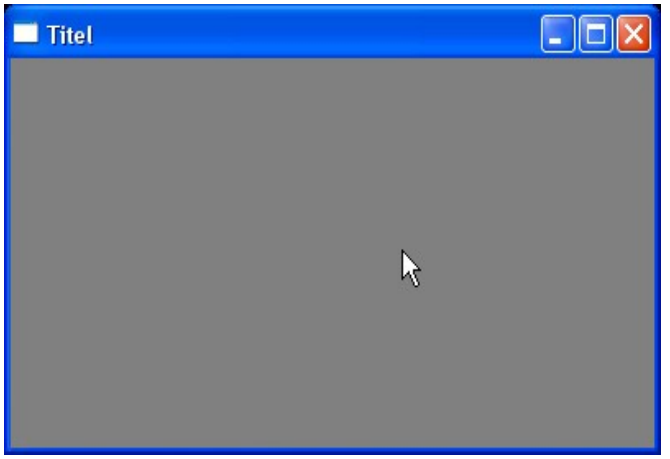
das erste Fenster



das erste Fenster

lassen wir ab jetzt weg
ich brauch den Platz

```
use strict;
use warnings;

MeinProgramm->new->MainLoop;

package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );

sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    $fenster->Show(1);
}
```



Zeit für Jim Knopf

Zeit für Jim Knopf

- Wx::Button->new
  - ($parent, $id, $label, \@pos, \@size, $style)
  - gleiche Masche wie beim Frame

Zeit für Jim Knopf

- Wx::Button->new
  - ($parent, $id, $label, \@pos, \@size, $style)
  - gleiche Masche wie beim Frame

- Wx::BitmapButton->new
  - ($parent, $id, $bitmap, \@pos, \@size, $style)

Zeit für Jim Knopf

- Wx::Button->new
  - ($parent, $id, $label, \@pos, \@size, $style)
  - gleiche Masche wie beim Frame

- Wx::BitmapButton->new
  - ($parent, $id, $bitmap, \@pos, \@size, $style)
  - gleiches Schema, 1 Unterschied

Zeit für Jim Knopf

- Wx::BitmapButton->new
  - ($parent, $id, $bitmap, \@pos, \@size, $style)
  - $parent = $frame;
  - $id = -1;
  - $bitmap = Wx::Bitmap->new
    ('file_close.xpm', wxBITMAP_TYPE_XPM);
  - Pos, size, $style = ... not yet

Zeit für Jim Knopf

```
package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );

sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');

    Wx::BitmapButton->new
        ( $fenster, -1, Wx::Bitmap->new
            ( 'file_close.xpm', wxBITMAP_TYPE_XPM) );

    $fenster->Show(1);
}
```

Zeit für Jim Knopf


```
package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );

sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');

    Wx::BitmapButton->new
        ( $fenster, -1, Wx::Bitmap->new
            ( 'file_close.xpm', wxBITMAP_TYPE_XPM) );

    $fenster->Show(1);
}
```

Zeit für Jim Knopf

```
package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );

sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');

    Wx::BitmapButton->new
        ( $frame, -1, Wx::Bitm          ew
            ( 'file_close.xpm           MAP_TYPE_XPM) );
}
```

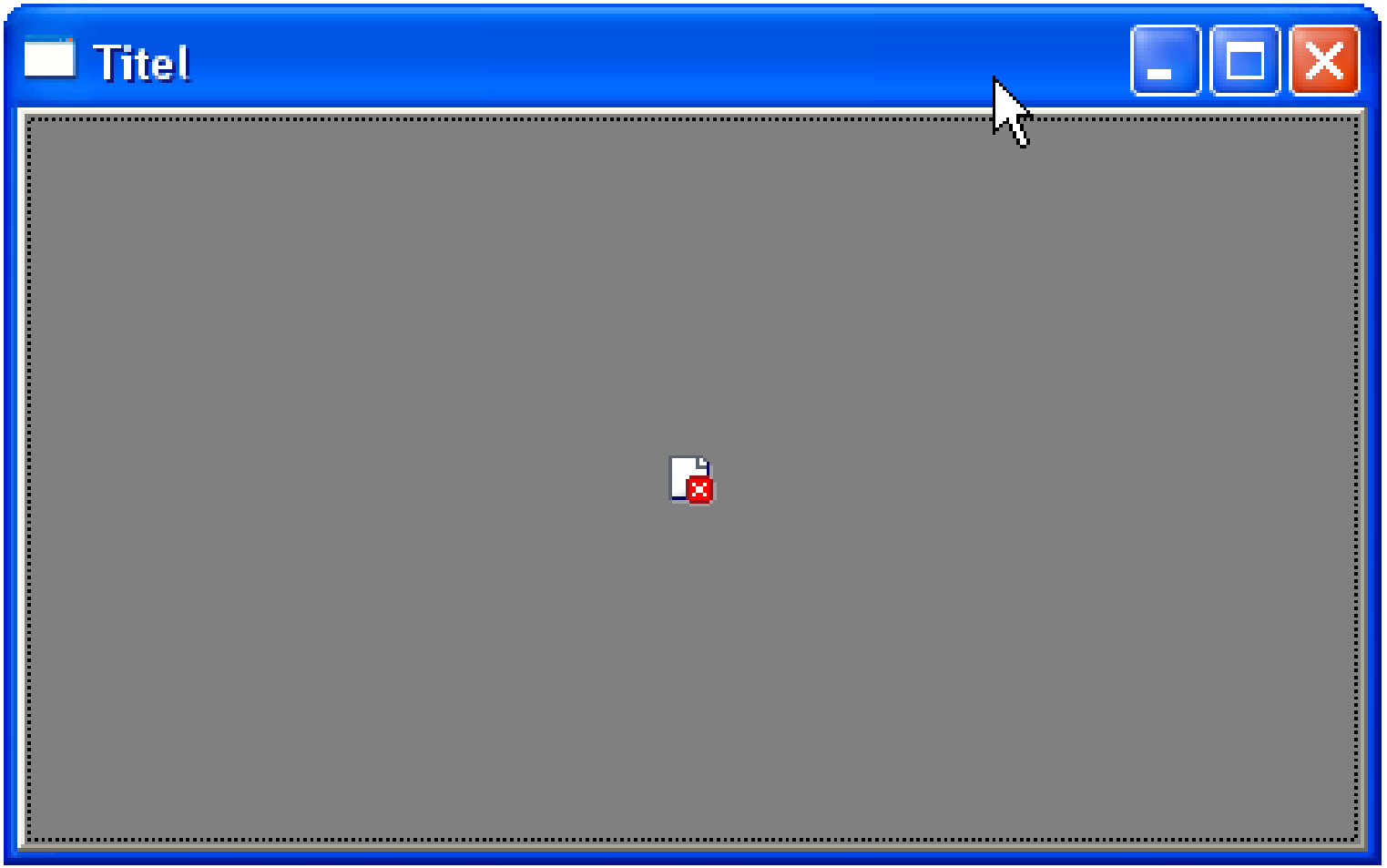
mannooo...
was ist jetzt schon wieder falsch?

Zeit für Jim Knopf

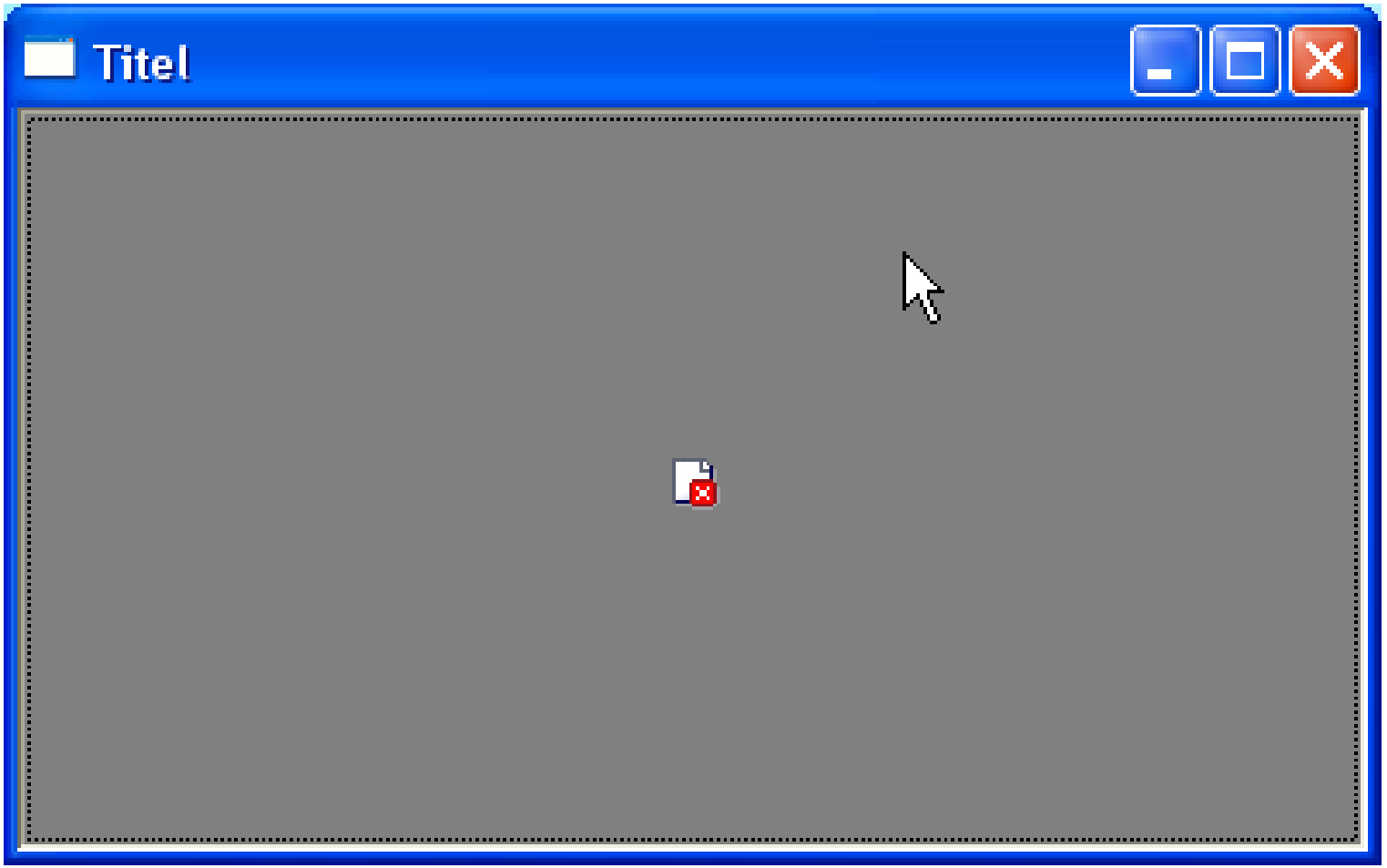
```
package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );

sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');

    Wx::InitAllImageHandlers();
    Wx::BitmapButton->new
        ( $fenster, -1, Wx::Bitmap->new
            ( 'file_close.xpm', wxBITMAP_TYPE_XPM) );
    $fenster->Show(1);
}
```

ImageHandler müssen initialisiert werden

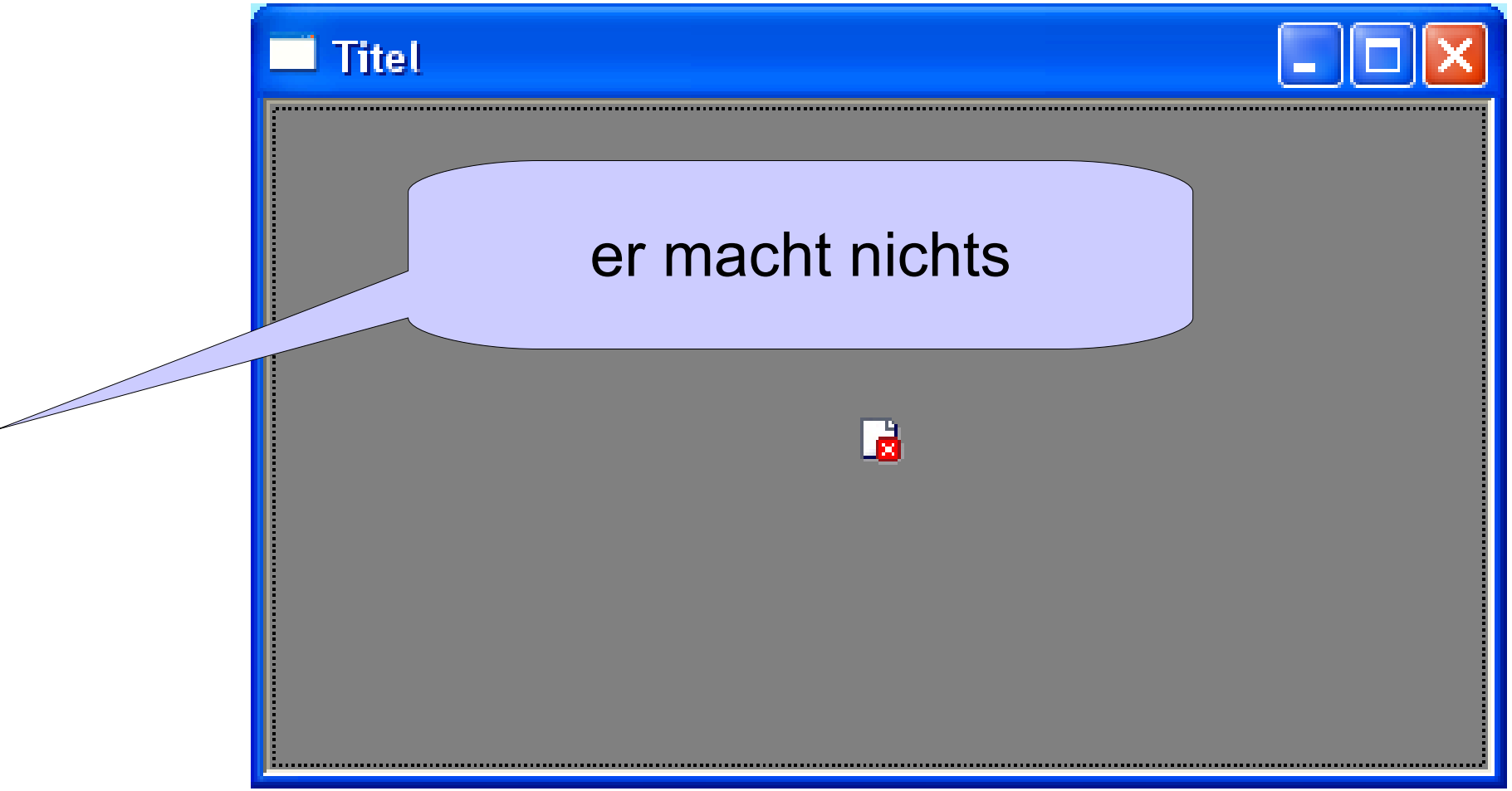
Zeit für Jim Knopf



Zeit für Jim Knopf



Zeit für Jim Knopf



Ereignisse abfangen

- er macht nichts ...
  - Events werden extra definiert
  - da Widgets bis zu 20 Events feuern können
  - je Eventdefinition ein Aufruf

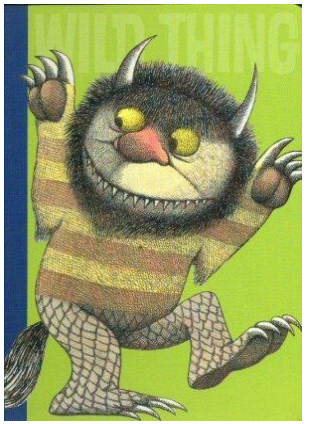
Ereignisse abfangen

- Parameter : Referenzen auf $widget, sub, ($parent)
  - EVT_BUTTON ($parent, $button, $coderef);
  - coderef als \&subname oder sub { ... }
  - sub bekommt widgetref & aktuelle Daten als Param.
  - sub { $_[0]->SetTitle('-') } # setzt Titeltext

Zeit für Jim Knopf

```
package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );

sub OnInit {
    Wx::InitAllImageHandlers();
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $jim = Wx::BitmapButton->new
        ($fenster, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );

    $fenster->Show(1);
}
```

Zeit für Jim Knopf




```
sub OnInit {
    Wx::InitAllImageHandlers();
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $jim = Wx::BitmapButton->new
        ($fenster, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );
    $fenster->Show(1);
}
```

> Undefined subroutine &MeinProgramm::EVT_BUTTON called at ...

Zeit für Jim Knopf

```
sub OnInit {
    Wx::InitAllImageHandlers();
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $jim = Wx::BitmapButton->new
        ($fenster, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );
    $fenster->Show(1);
}
```

Zeit für Jim Knopf

```
sub OnInit {
    Wx::InitAllIm
    my $fenster
    my $jim = W
        ($fenster, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );
    $fenster->Show(1);
}
```

beides uppercase ???

Zeit für Jim Knopf

```
sub OnInit {
    Wx::InitAllIm
    my $fenster
    my $jim = Wx::Bit                tton->new
        ($fenster, -1,  ...Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );
    $fenster->Show(1);
}
```

Ja,
auch das muss man anmelden

Zeit für Jim Knopf

```
package MeinProgramm;
use base qw( Wx::App );
use Wx qw( :everything );


sub OnInit {
    Wx::InitAllImageHandlers();
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $jim = Wx::BitmapButton->new
        ($fenster, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );

    $fenster->Show(1);
}
```

Nicht dort enthalten ?

Zeit für Jim Knopf

```
package MeinProgra
use base qw( Wx::App );
use Wx qw( :everything );
use Wx::Event qw( EVT_BUTTON );

sub OnInit {
    Wx::InitAllImageHandlers();
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $jim = Wx::BitmapButton->new
        ($fenster, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );

    $fenster->Show(1);
}
```

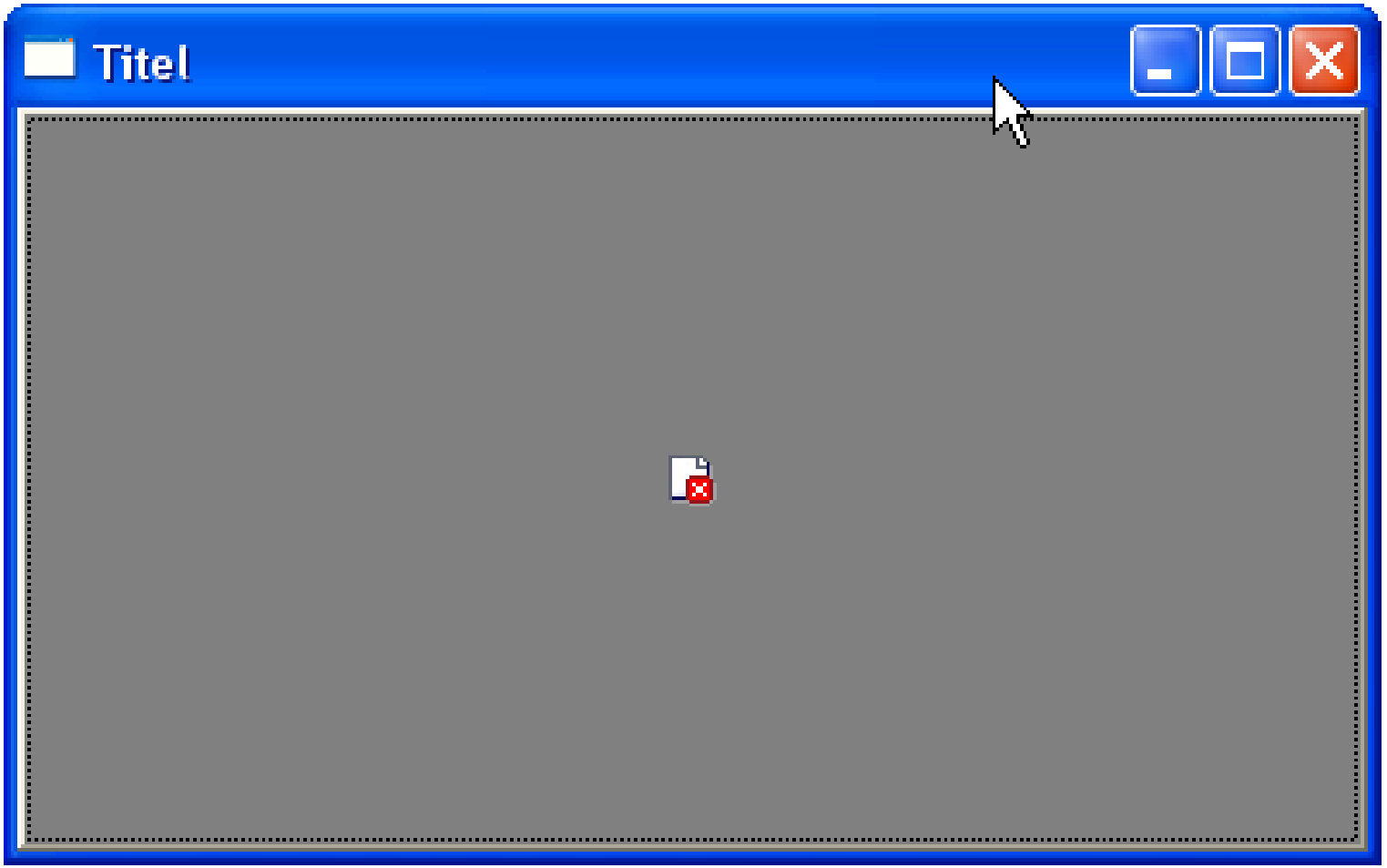
Events werden extra angemeldet

Zeit für Jim Knopf

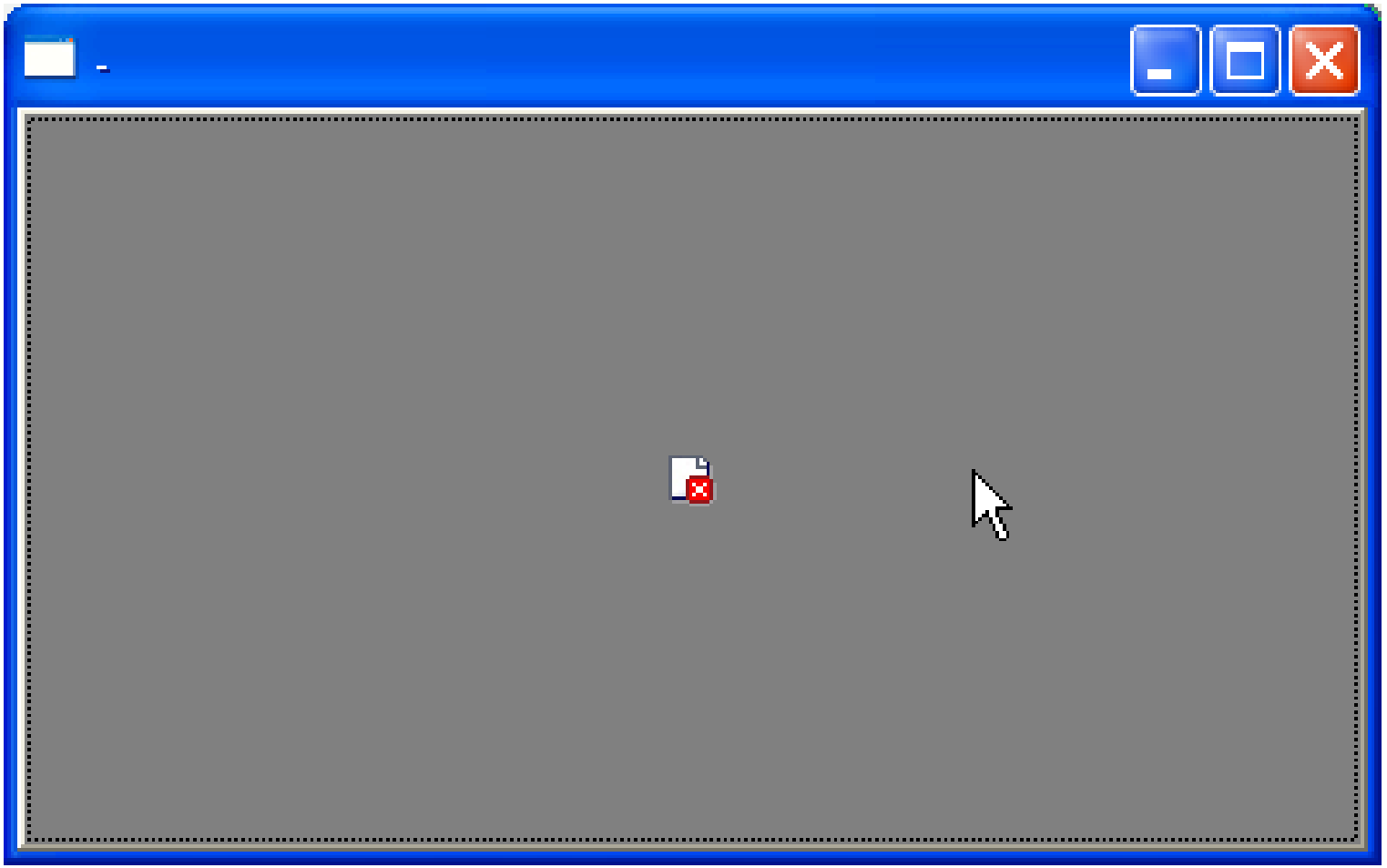
```
package MeinProgra
use base qw( Wx::App );
use Wx qw( :everything );
use Wx::Event qw( :everything );

sub OnInit {
    Wx::InitAllImageHandlers();
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $jim = Wx::BitmapButton->new
        ($fenster, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );

    $fenster->Show(1);
}
```

Das das ja nicht nochmal passiert

Zeit für Jim Knopf

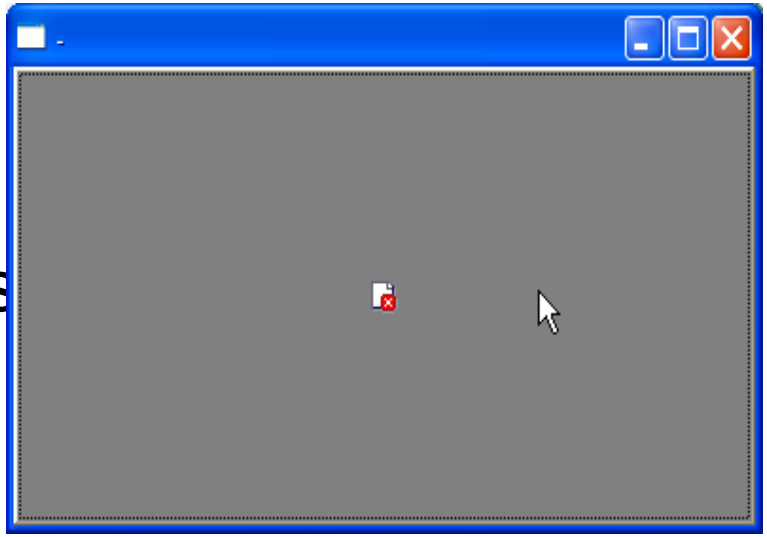


Zeit für Jim Knopf



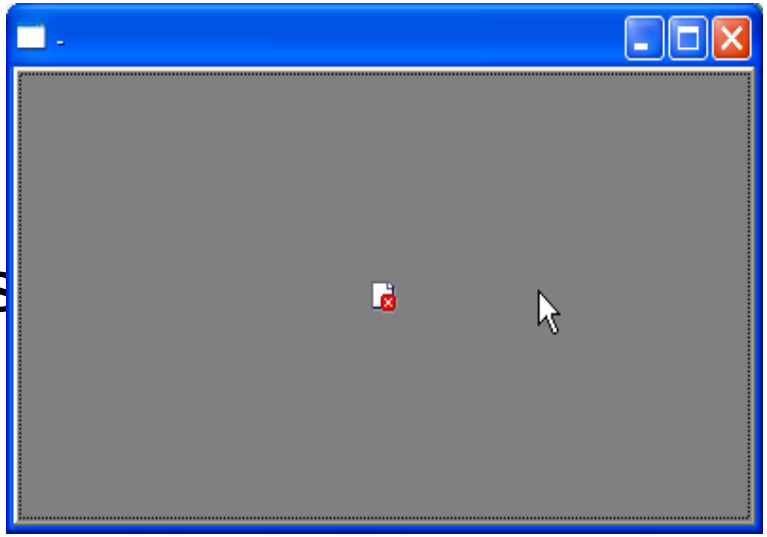
Seltsames Grau nicht ?

- ist Defaultfarbe des Fensters
  - Buttons sind transparent
  - die sich natürlich ändern läs



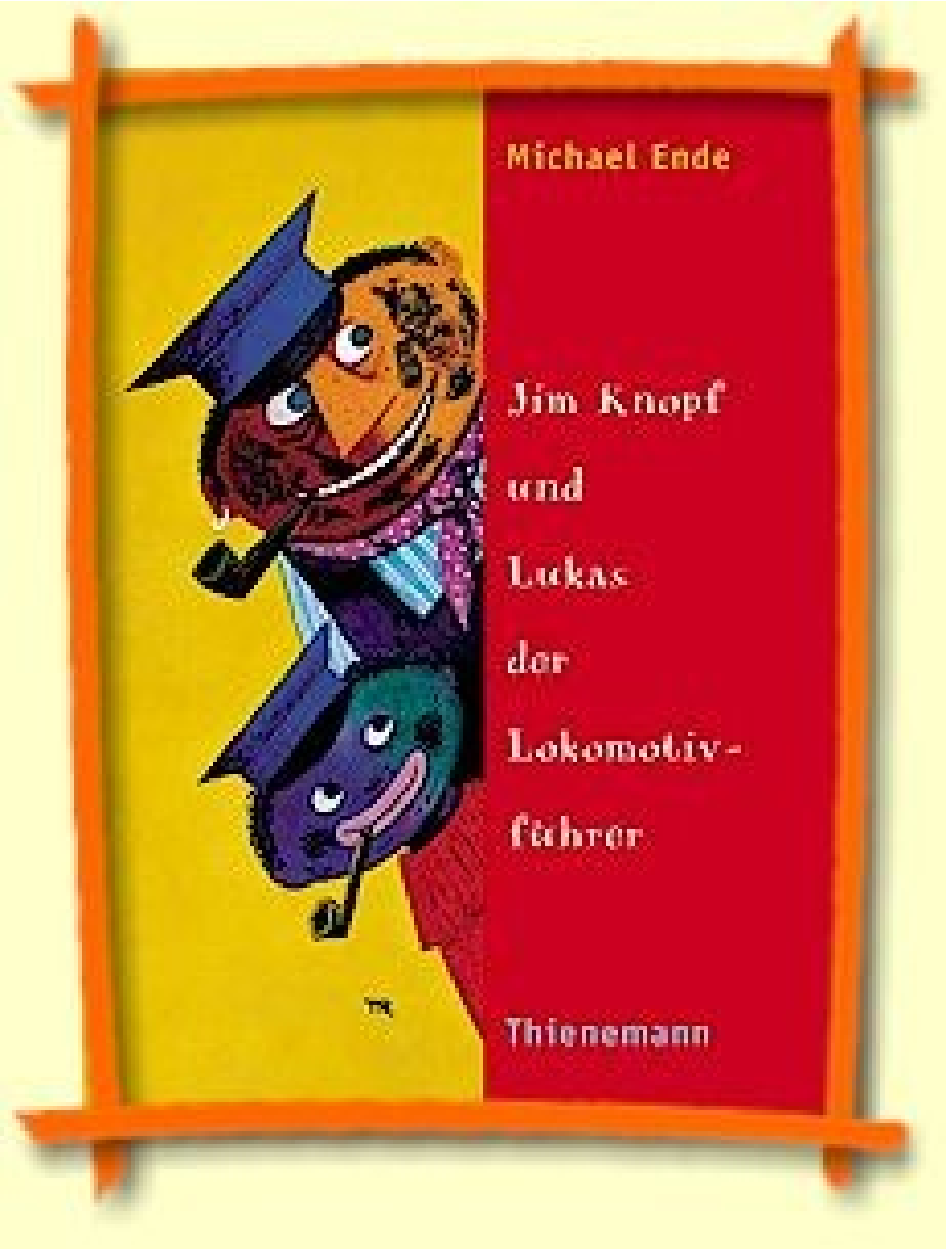
Seltsames Grau nicht ?

- ist Defaultfarbe des Fensters
  - Buttons sind transparent
  - die sich natürlich ändern läs



- aber wir brauchen ein Panel
  - Grundflächen auf die man Widget befestigt
  - Arbeitsflächen im Programm
  - Wx::Panel->new($parent, $id, \@pos, \@size, $style);

Jim braucht einen Freund

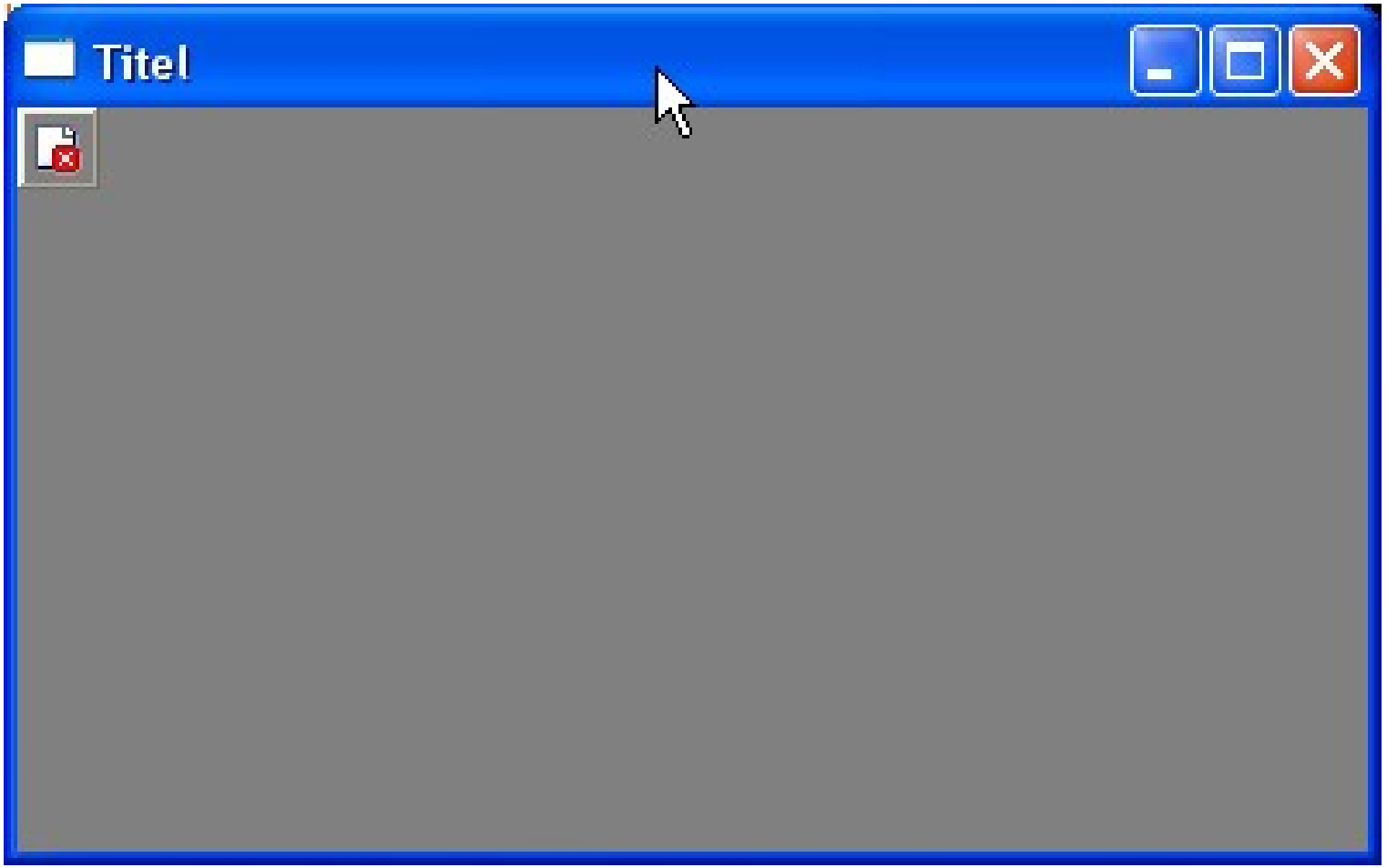


Wir fügen das Panel hinzu

```
sub OnInit {
    Wx::InitAllImageHandlers();
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $lukas = Wx::Panel->new($fenster, -1);

    my $jim = Wx::Bi                apButton->new
        ($fenster, -1          Bitmap->new
            ('file_cl                 ', wxBITMAP_TYPE_XPM) );
    EVT_BUT                   ster, $jim, sub {$_[0]->SetTitle('-')} );

}
```

Der Button nahm gesamtes Fenster ein,
weil er einziges Kindobjekt des Fensters,
wenn jetzt das Panel das direkte Kind ist ...

Zeit für Jim Knopf

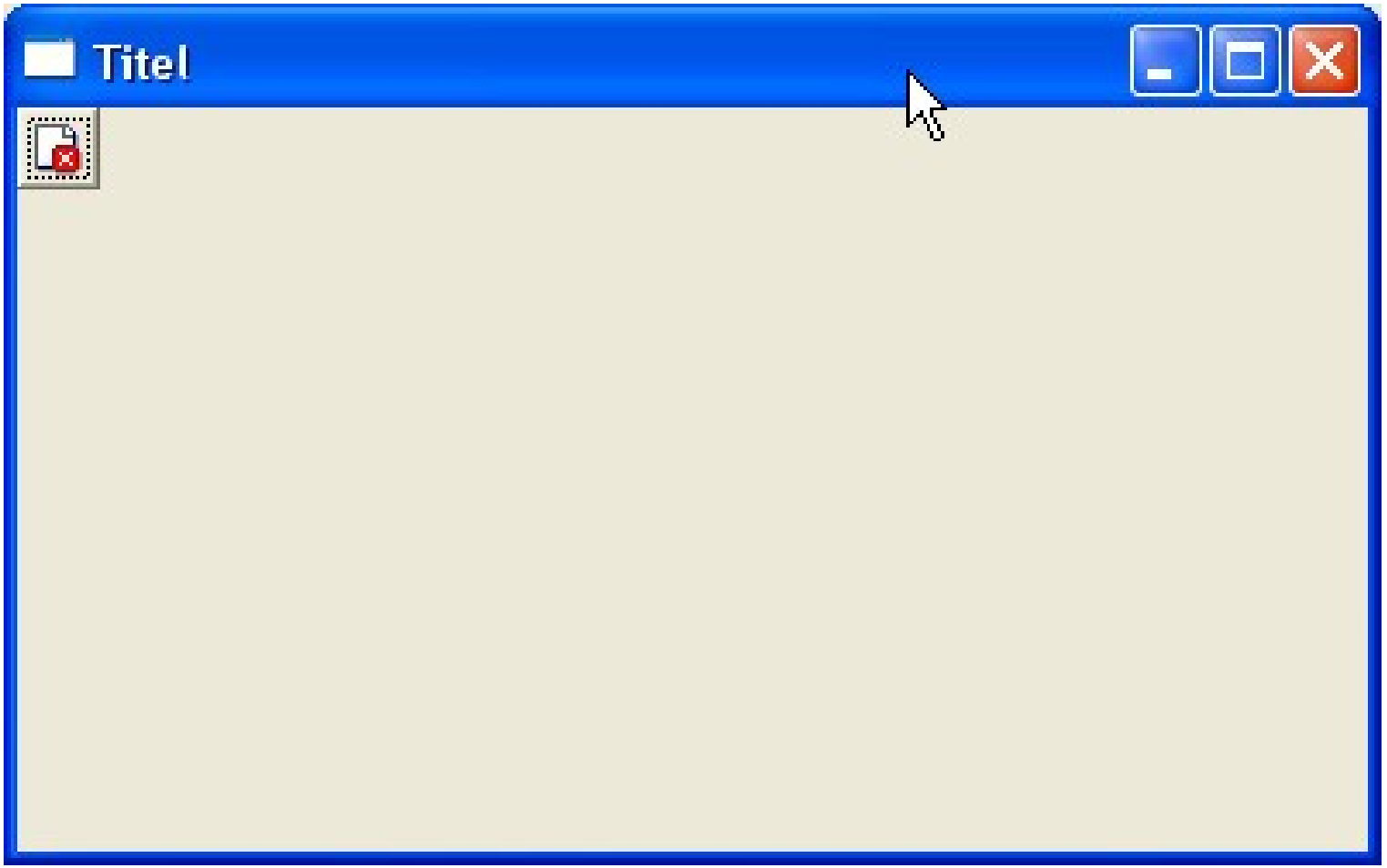


Wo ist das Panel ?

Widgets die auf einem Panel erscheinen sollen, müssen Kinder des Panels sein.

```
sub C
    Wx::Init
    my $fenste      ame->new( undef, -1, 'Titel');
    my $lukas       Panel->new($fenster, -1);

    my $jim = Wx::BitmapButton->new
        ($lukas, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );

    $fenster->Show(1);
}
```

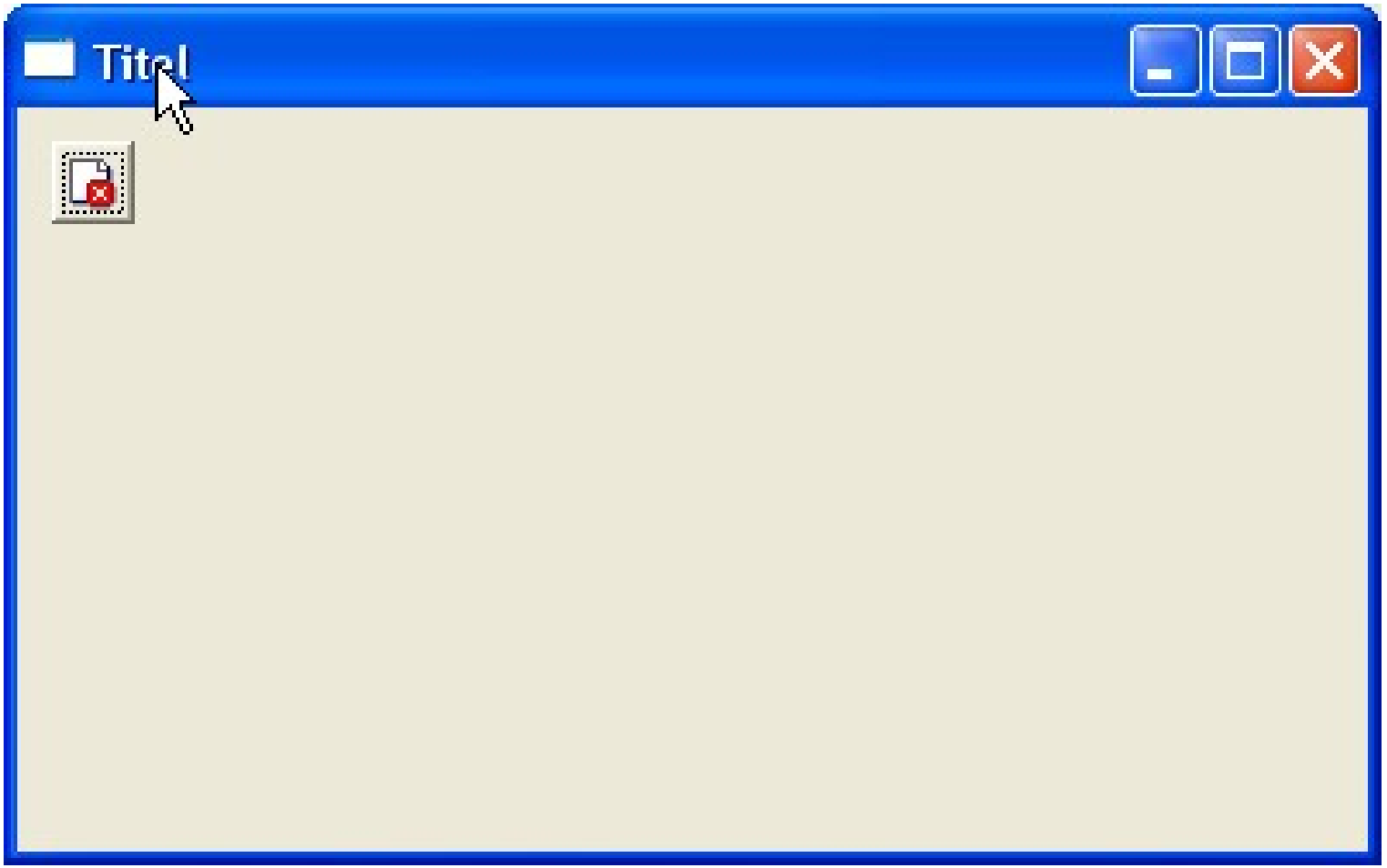
Da ist das Panel ! :)



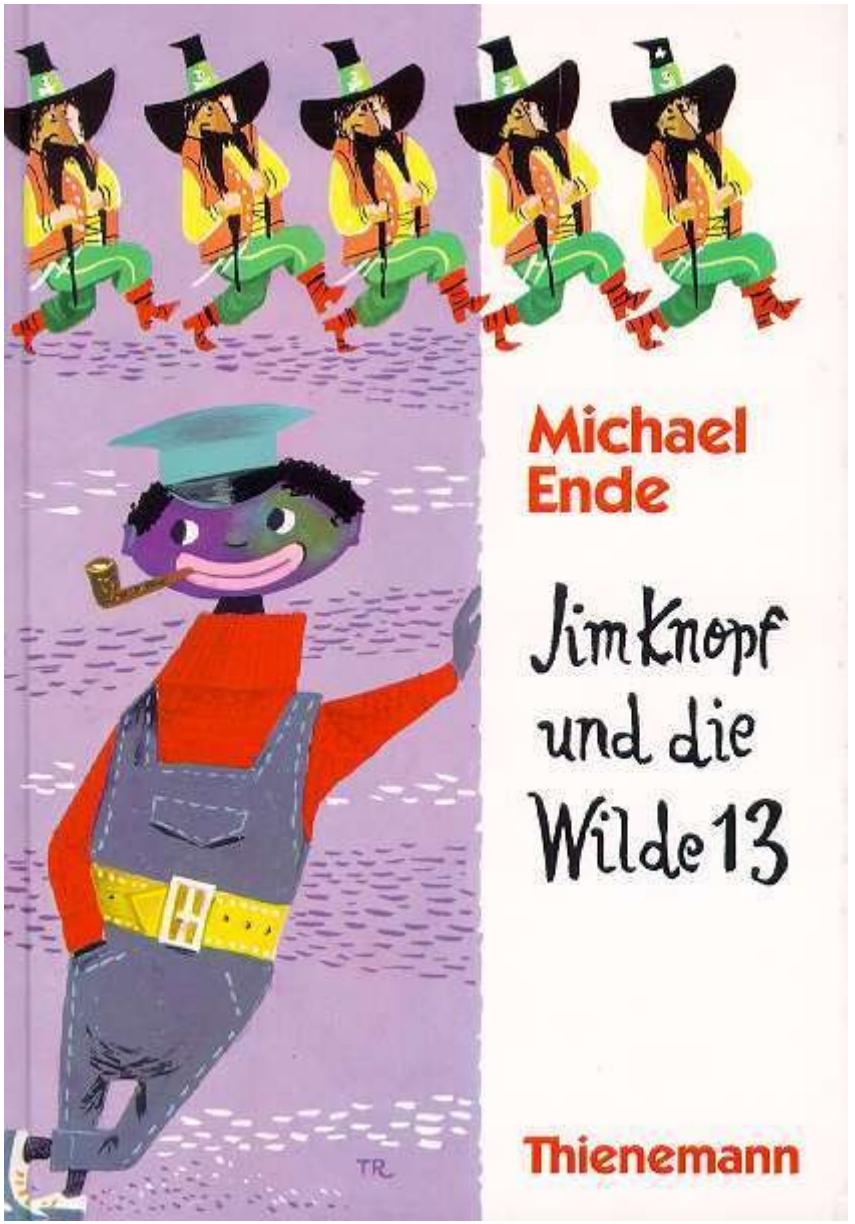
Positionieren des Knopfes

```
sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $lukas = Wx::P
                                                    Position des Knopfes hinzugefügt

    Wx::InitAllImageHandler
    my $bmp = Wx::Bitmap->
            ('file_close.xpm', wx  MAP_TYPE_XPM);
    my $jim = Wx::BitmapButton->new
        ($lukas, -1, $bmp , [20, 20] );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );

    $fenster->Show(1);
}
```

Positionieren des Knopfes



noch mehr Action



noch mehr Action

- Sizer (ordene Widgets auf Panel/Frame an)
  - kein pack Befehl sondern Objekt
  - BoxSizer : Faden, vertikal oder horizontal
  - StaticBoxSizer : mit Rahmen
  - GridSizer : Gitter mit gleichgroßen Zellen
  - FlexGridSizer : Gitter mit anpaßbaren Zeilen/Spalten
  - GridBagSizer : Table mit colspan und rowspan

Jim will einen BoxSizer

- $wilde13 = Wx::BoxSizer->new(wxVERTICAL);
  - oder wxHORIZONTAL
  - $wilde13->Add($widget, $autosize, $flags, $margin);
  - $autosize (0 | 1) : darf ich Rest des Platzes haben ?
  - $flags (wxTOP, wxLEFT, wxALL) : Ausrichtung
  - $margin : vorgelagerter Rand in px, ->AddSpace();
  - $widget->SetSizer($w13);
  - $widget->SetAutoLayout(1); # oder $widget-

Elegantes Positionieren

```
sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $lukas = Wx::Panel->new($fenster, -1);
    my $wilde13 = Wx::BoxSizer->new( wxHORIZONTAL );

    Wx::InitAllImageHandlers();
    my $jim = Wx::BitmapButton->new
        ($lukas, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );
    $wilde13->Add($jim, 0, wxLEFT, 10);
    $fenster->SetSizer( $wilde13 );
    $fenster->SetAutoLayout(1);
    $fenster->Show(1);
}
```

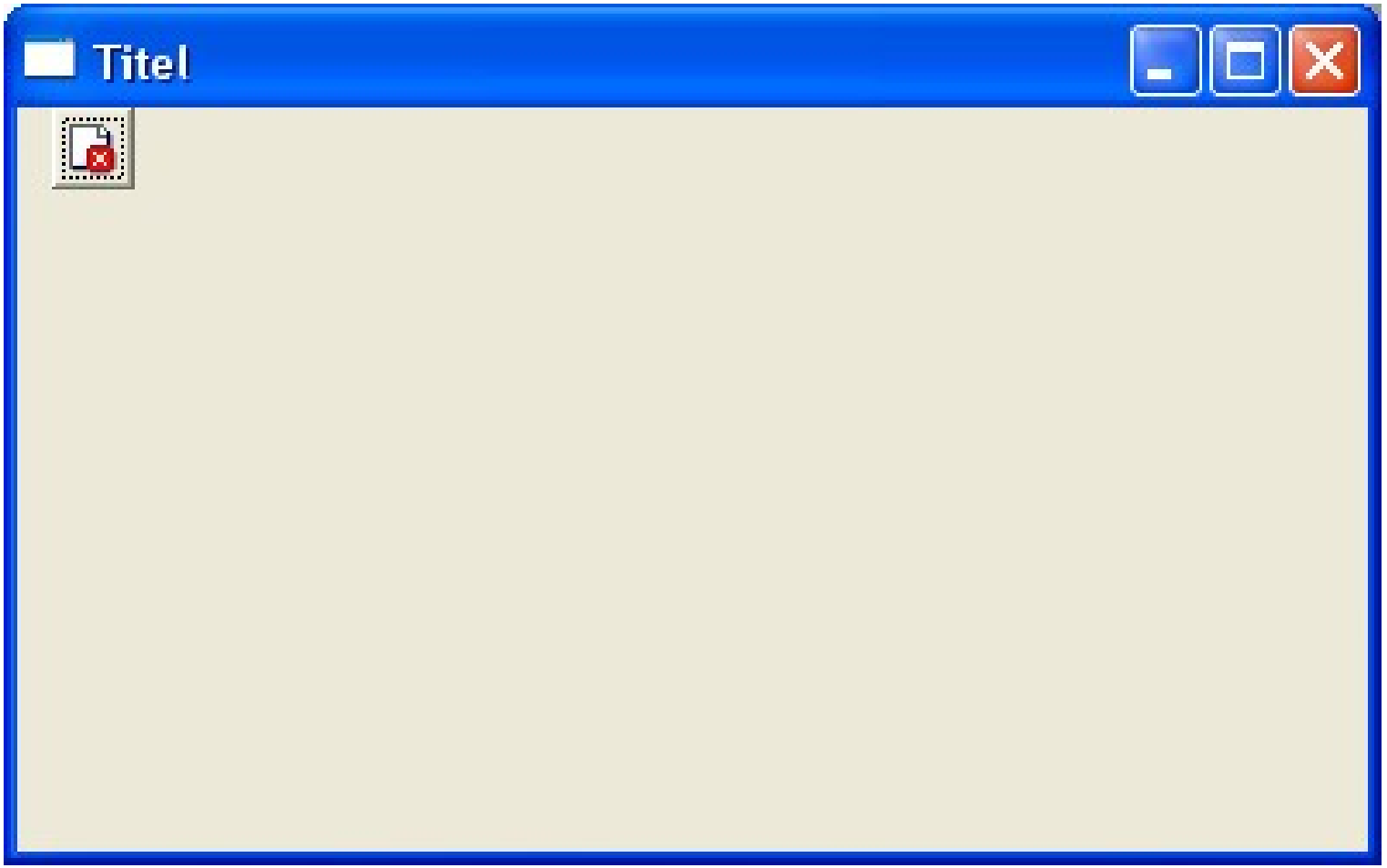
Elegantes Positionieren

Elegantes Positionieren

```
sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $lukas = Wx::Panel->new($fenster, -1);
    my $wilde13 = Wx::BoxSizer->new( wxHORIZONTAL );

    Wx::InitAllImage
    my $jim = Wx::Bitm                    on->new
        ($lukas, -1, W                    map->new
            ('file_clo                    pm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );
    $wilde13->Add($jim, 0, wxLEFT, 10);
    $lukas->SetSizer( $wilde13 );
    $lukas->SetAutoLayout(1);
    $fenster->Show(1);
}
```

Sizer wird aufs Panel gespannt

Elegantes Positionieren



Elegantes Positionieren

```
sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $wilde13 = Wx::BoxSizer->new( wxHORIZONTAL );

    Wx::InitAllImageHandlers();
    my $jim = Wx::BitmapButton->new
        ($lukas, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );
    $wilde13->Add($jim, 0, wxLEFT | wxTOP, 10);
    $lukas->SetSizer( $wilde13 );
    $lukas->SetAutoLayout(1);
    $fenster->Show(1);
}
```

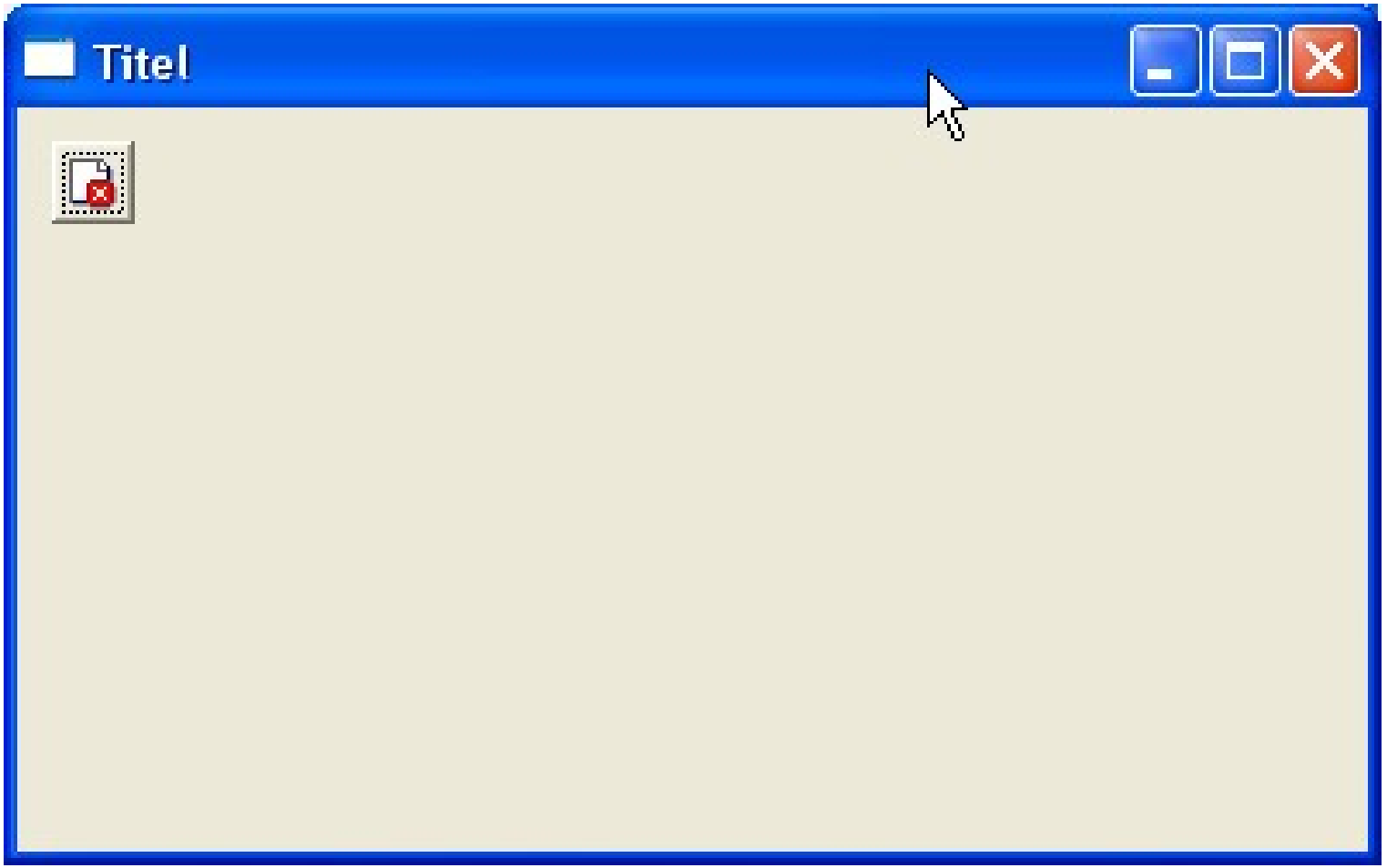
wir erzwingen damit den Rand nach oben

Elegantes Positionieren

```
sub OnInit {
    my $fenster = Wx::Frame->new( undef, -1, 'Titel');
    my $wilde13 = Wx::BoxSizer->new( wxHORIZONTAL );

    Wx::InitAllImageHandlers();
    my $jim = Wx::BitmapButton-
        ($lukas, -1, Wx::Bitmap->r
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );
    $wilde13->Add($jim, 0, wxALL, 10);
    $lukas->SetSizer( $wilde13 );
    $lukas->SetAutoLayout(1);
    $fenster->Show(1);
}
```

wir erzwingen damit den Rand nach allen Seiten

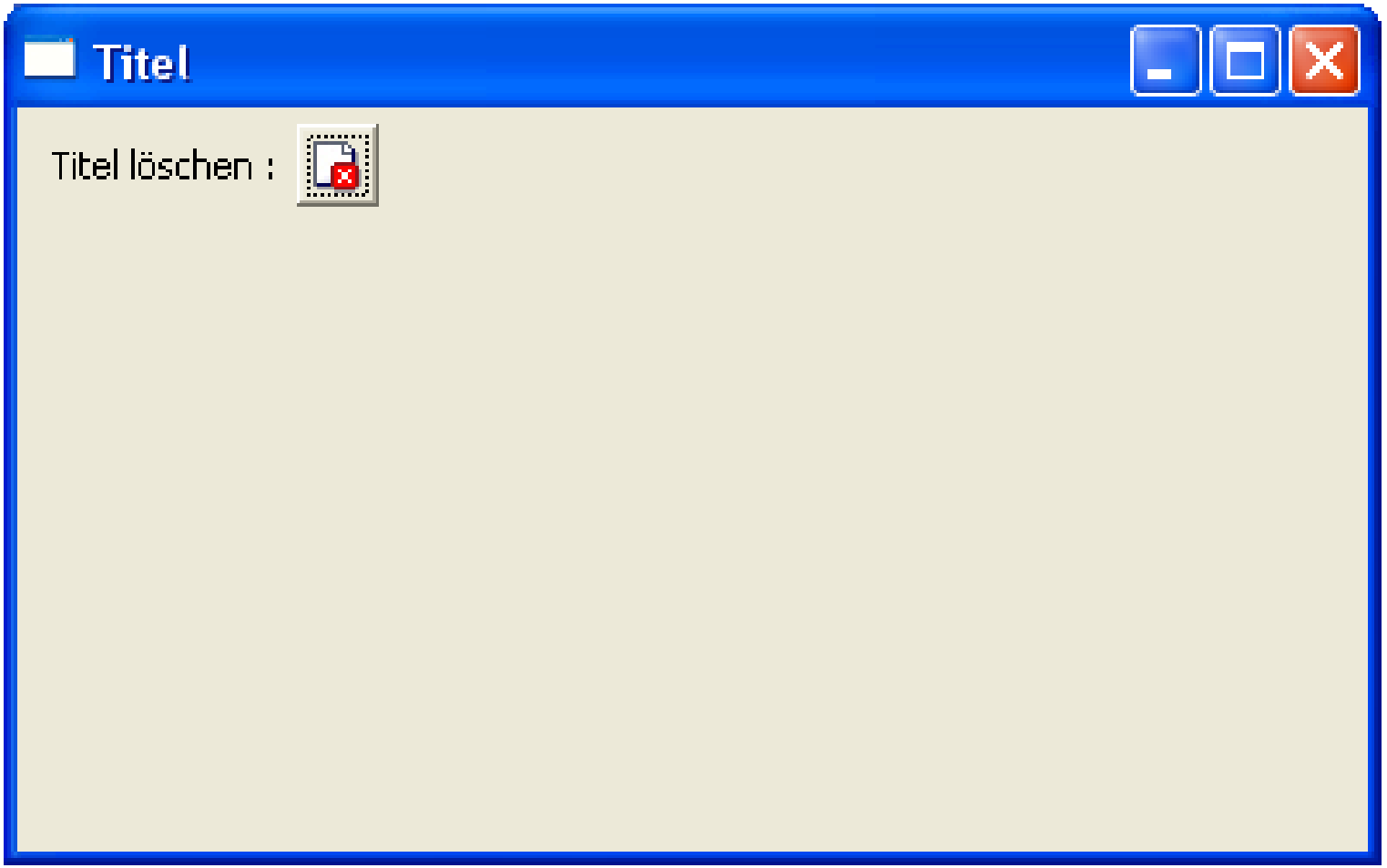
Elegantes Positionieren



Textlabel zufügen

```
sub OnInit {
    ...
    my $t = Wx::StaticText->new($lukas, -1, 'Titel löschen' );
    Wx::InitAllImageHandlers();
    my $jim = Wx::BitmapButton->new
        ($lukas, -1, Wx::Bitmap->new
            ('file_close.xpm', wxBITMAP_TYPE_XPM) );
    EVT_BUTTON( $fenster, $jim, sub {$_[0]->SetTitle('-')} );

    my $wilde13 = Wx::BoxSizer->new( wxHORIZONTAL );
    $wilde13->Add($t,    0, wxLEFT | wxTOP, 10);
    $wilde13->Add($jim, 0, wxLEFT | wxTOP,   5);
    $lukas->SetSizer( $wilde13 );
    ...
}
```

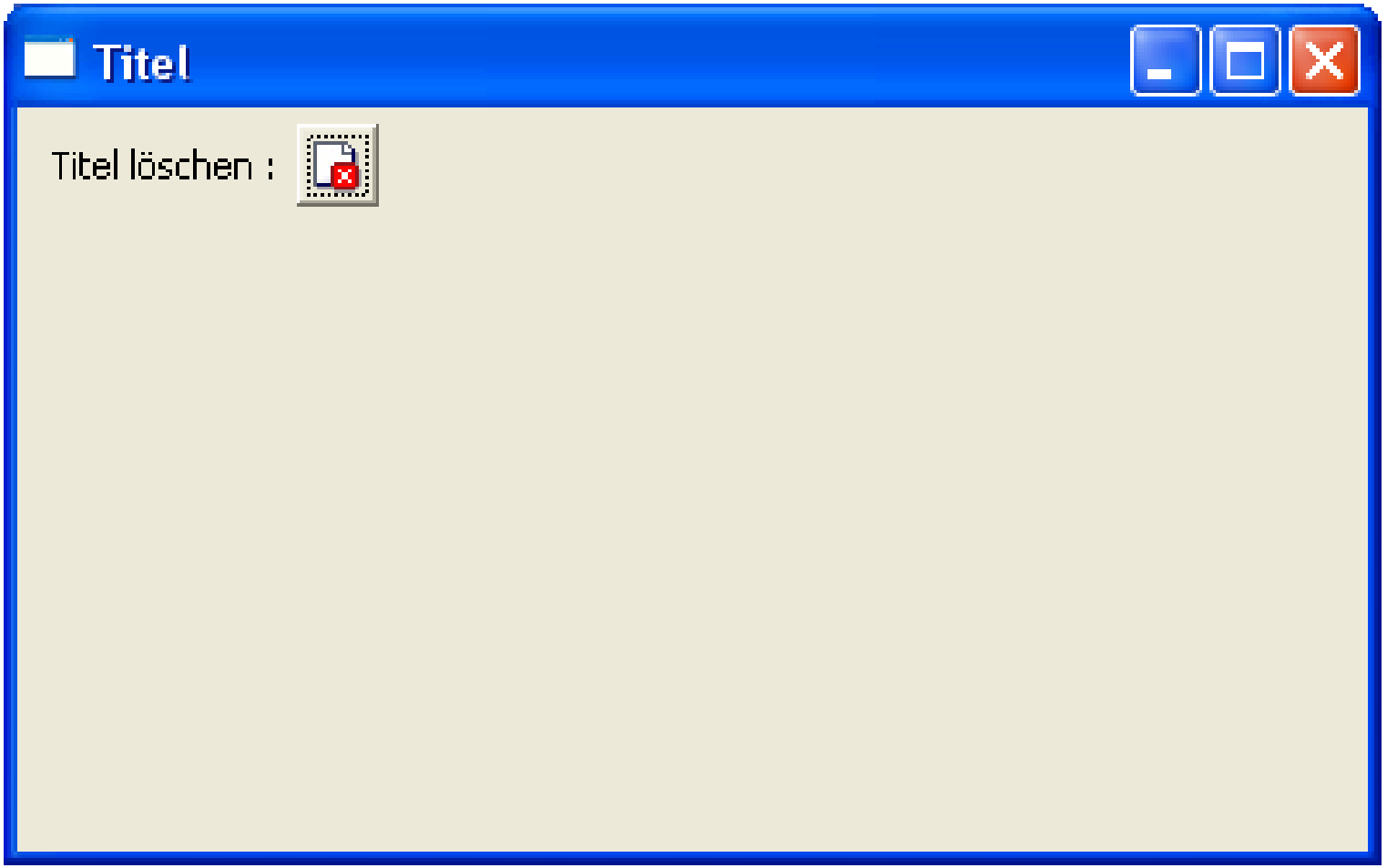
Textlabel zufügen



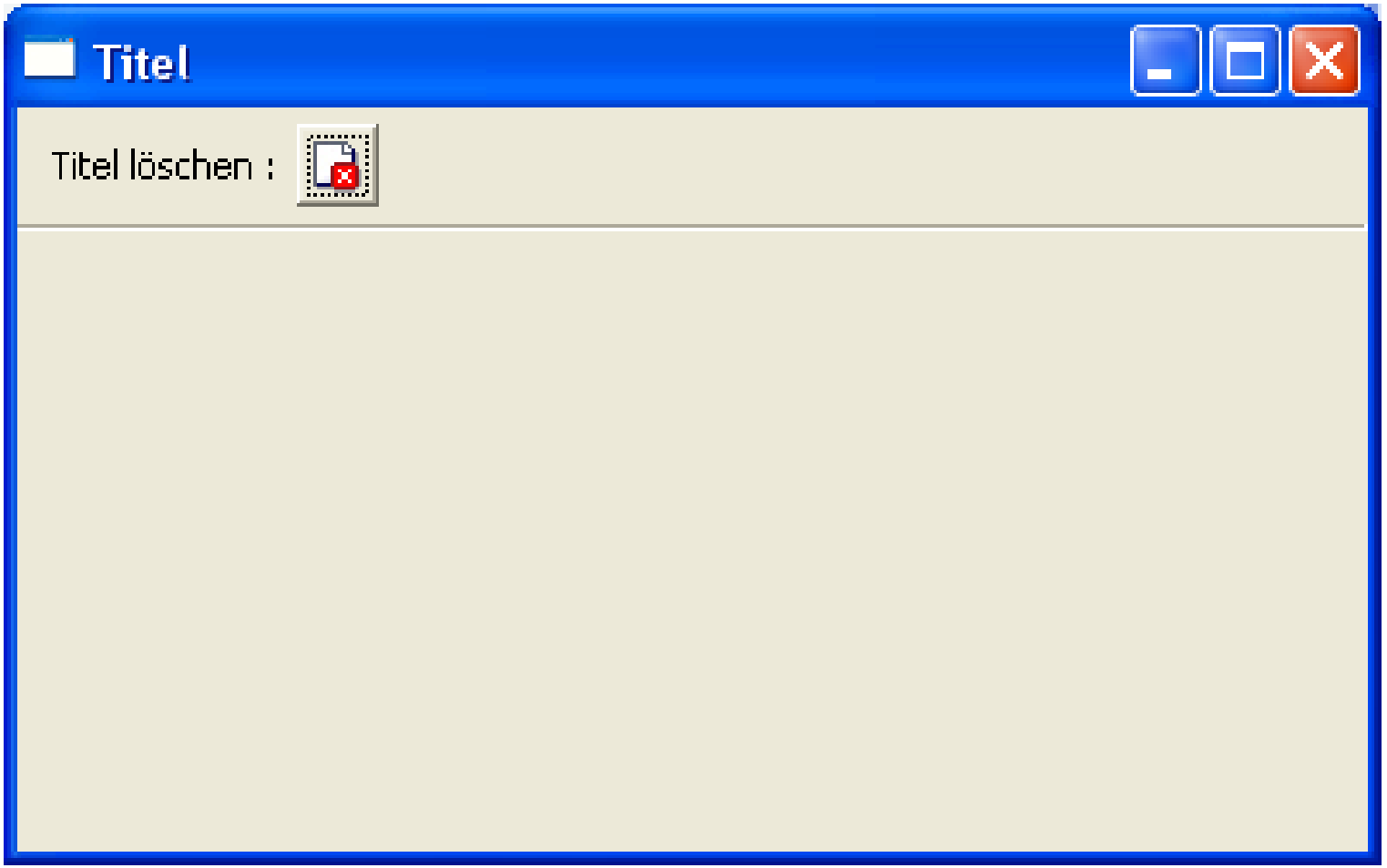
BoxSizer schachteln

```
sub OnInit {
    ...
    my $linie = Wx::StaticLine->new
        ($lukas, -1, [-1, -1], [-1, 2], wxLI_HORIZONTAL);

    my $wilde13 = Wx::BoxSizer->new( wxHORIZONTAL );
    $wilde13->Add($t,    0, wxLEFT | wxTOP, 10);
    $wilde13->Add($jim, 0, wxLEFT | wxTOP,  5);

    my $banden = Wx::BoxSizer->new( wxVERTICAL );
    $banden->Add($wilde13, 0, wxTOP, 0);
    $banden->Add($linie,       0, wxTOP, 5);
    $lukas->SetSizer( $banden );
    ...
}
```

Trennlinie zufügen



Trennlinie zufügen

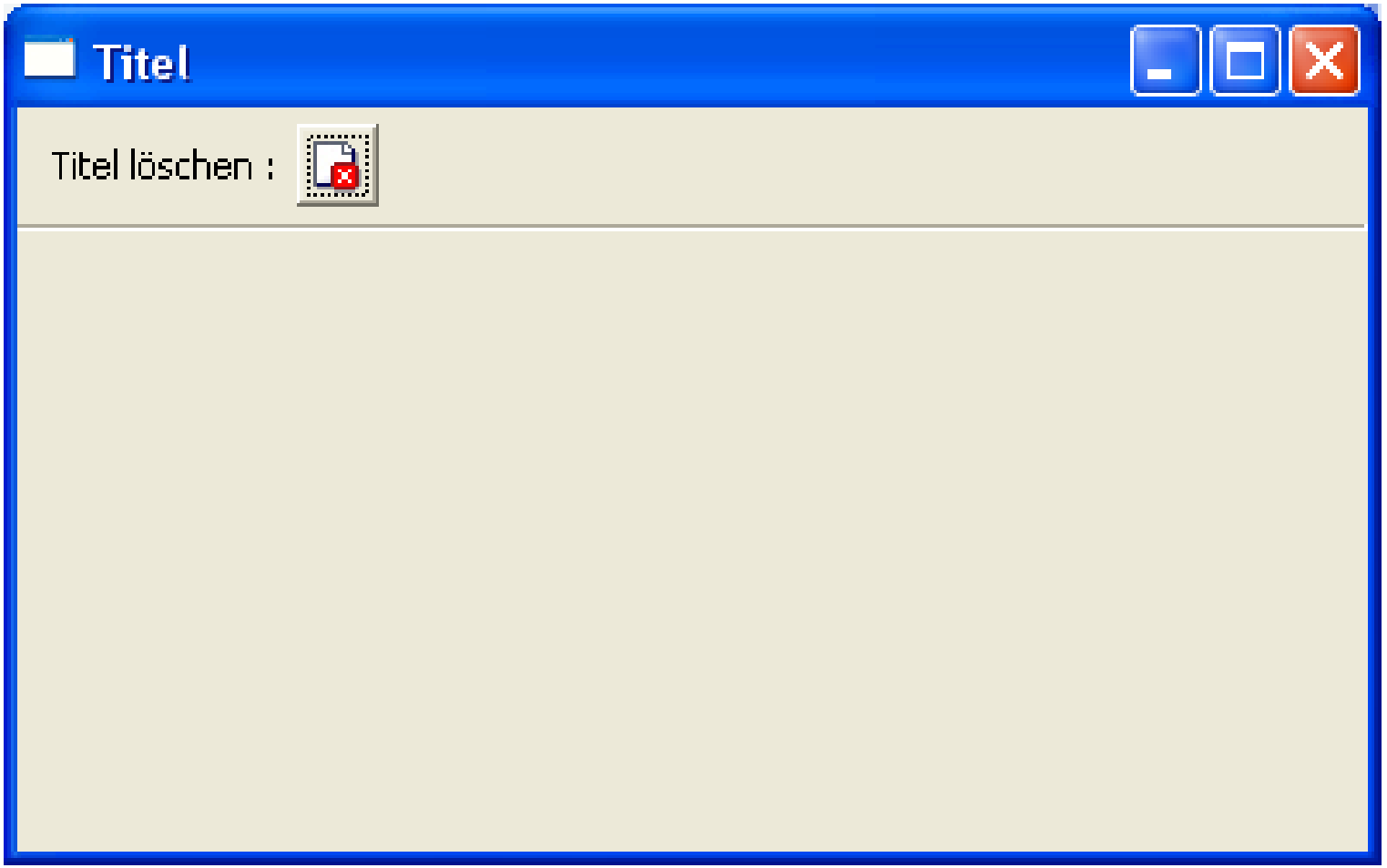


Trennlinie zufügen

```
sub OnInit {
    my $wilde13 = Wx::BoxSizer->ne                RIZONTAL );
    $wilde13->Add($t,   0, wxLEFT | w             0);
    $wilde13->Add($jim, 0, wxLEFT | wx            5);

    my $banden = Wx::BoxSizer->new( wxV           RTICAL );
    $banden->Add($wilde13, 0, wxTOP, 0);
    $banden->Add($linie,       0, wxTOP | wxGROW, 5);
    $lukas->SetSizer( $banden );
    ...
}
```

nimm dir die ganze Breite,
(Platz in der Dimension,
die nicht durch Sizer definiert wird)

Trennlinie zufügen



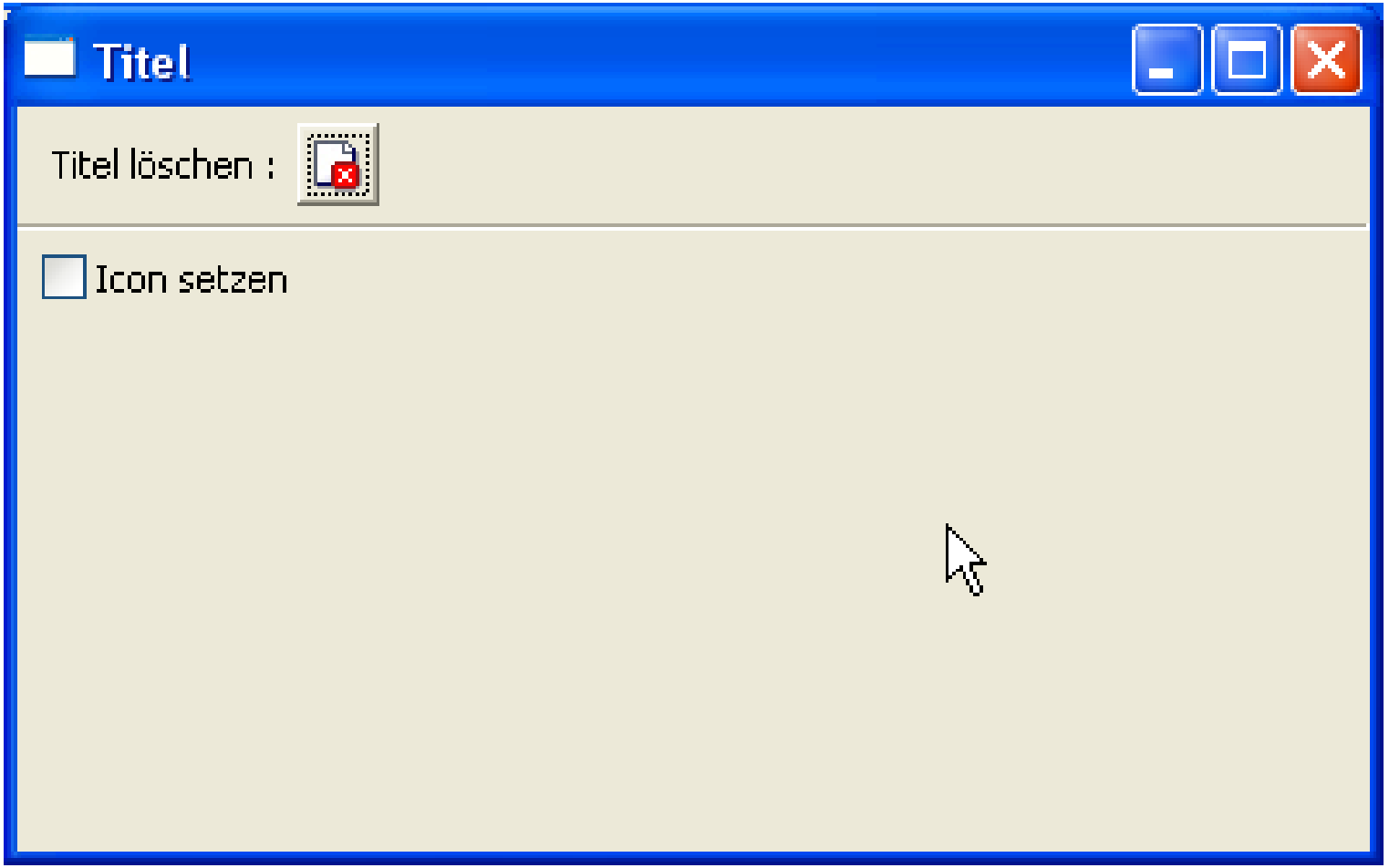
CheckBox zufügen

- Wx::CheckBox->new(
  - $parent = $lukas
  - $id = -1
  - $label = 'Icon setzen'
  - \@pos ...
  - \@size ...
  - $style ...
  - $validator ...
  - $name ... lassen wir weg

CheckBox zufügen

```
sub OnInit {
    ...
    my $hplotz = Wx::CheckBox->new
        ($lukas, -1, 'Icon setzen');

    my $banden = Wx::BoxSizer->new( wxVERTICAL );
    $banden->Add($wilde13, 0, wxTOP, 0);
    $banden->Add($linie,       0, wxTOP | wxGROW, 5);
    $banden->Add($hplotz,    0, wxTOP | wxLEFT,    7);
    $lukas->SetSizer( $banden );
    ...
}
```

CheckBox zufügen

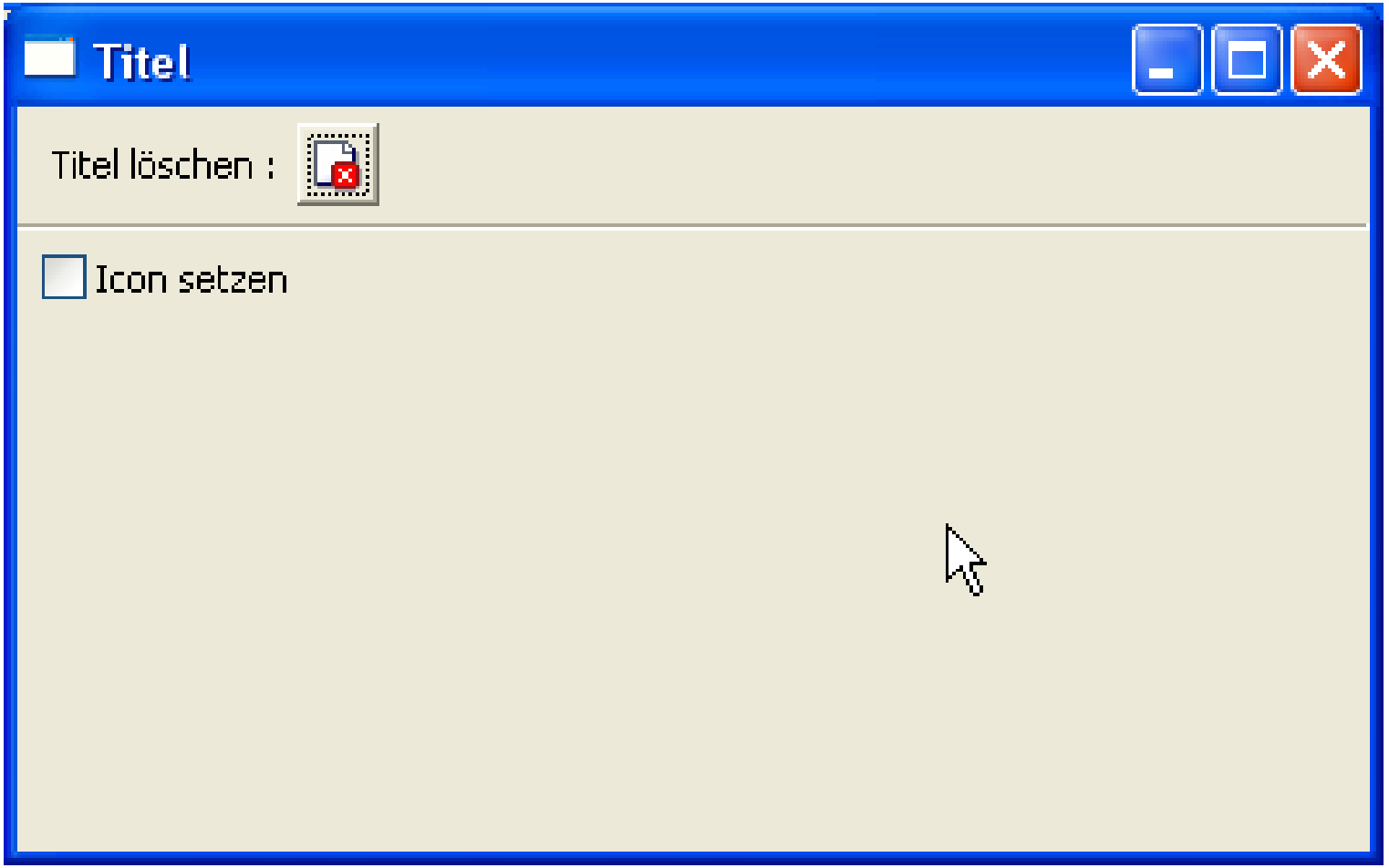


Parameter einer Evtroutine

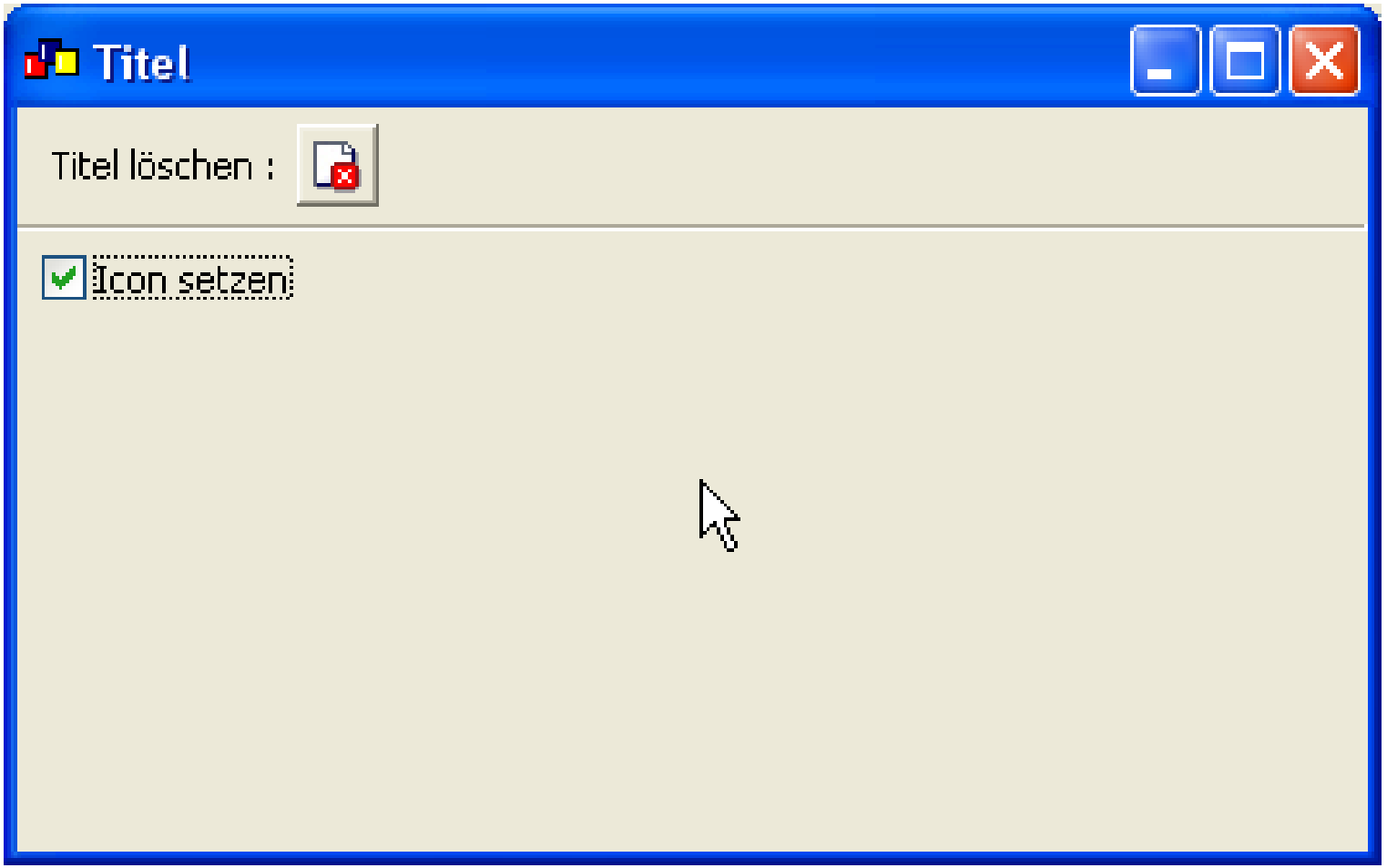
```
sub OnInit { ...
    my $hplotz = Wx::CheckBox->new
        ($lukas, -1, 'Icon setzen');
    EVT_CHECKBOX( $fenster, $hplotz, sub{
        $_[0]->SetIcon( Wx::Icon->new(
            'wxwin.ico', wxBITMAP_TYPE_ICO ) );
    } );

    my $banden = Wx::BoxSizer-                    xVERTICAL );
    $lukas->SetSizer( $banden );
     ...
}
```

Zustand erfahr ich durch $_[1]->IsChecked,
da : @_ = [$parent, $wxCommandEvent]

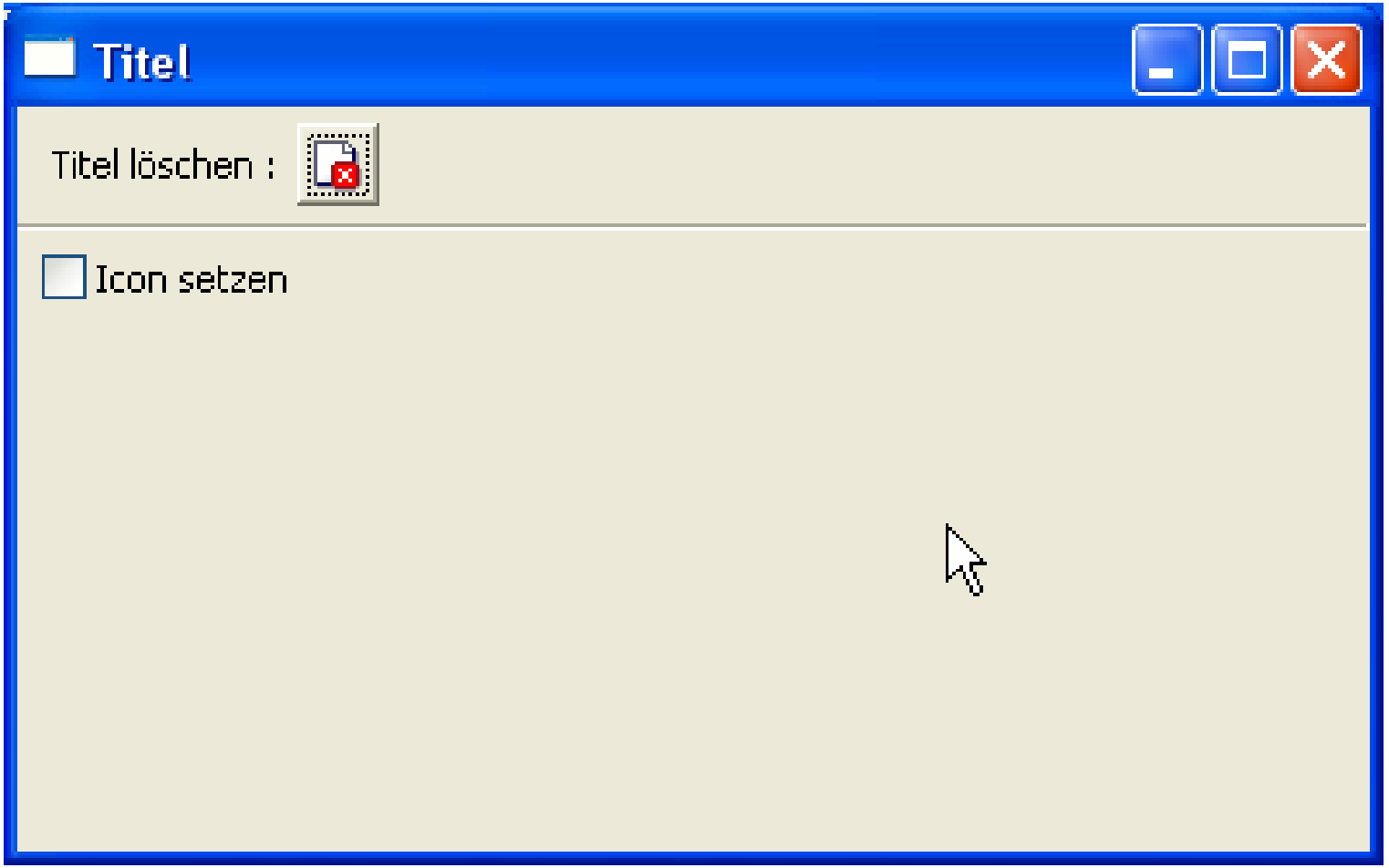
Parameter einer Evtroutine



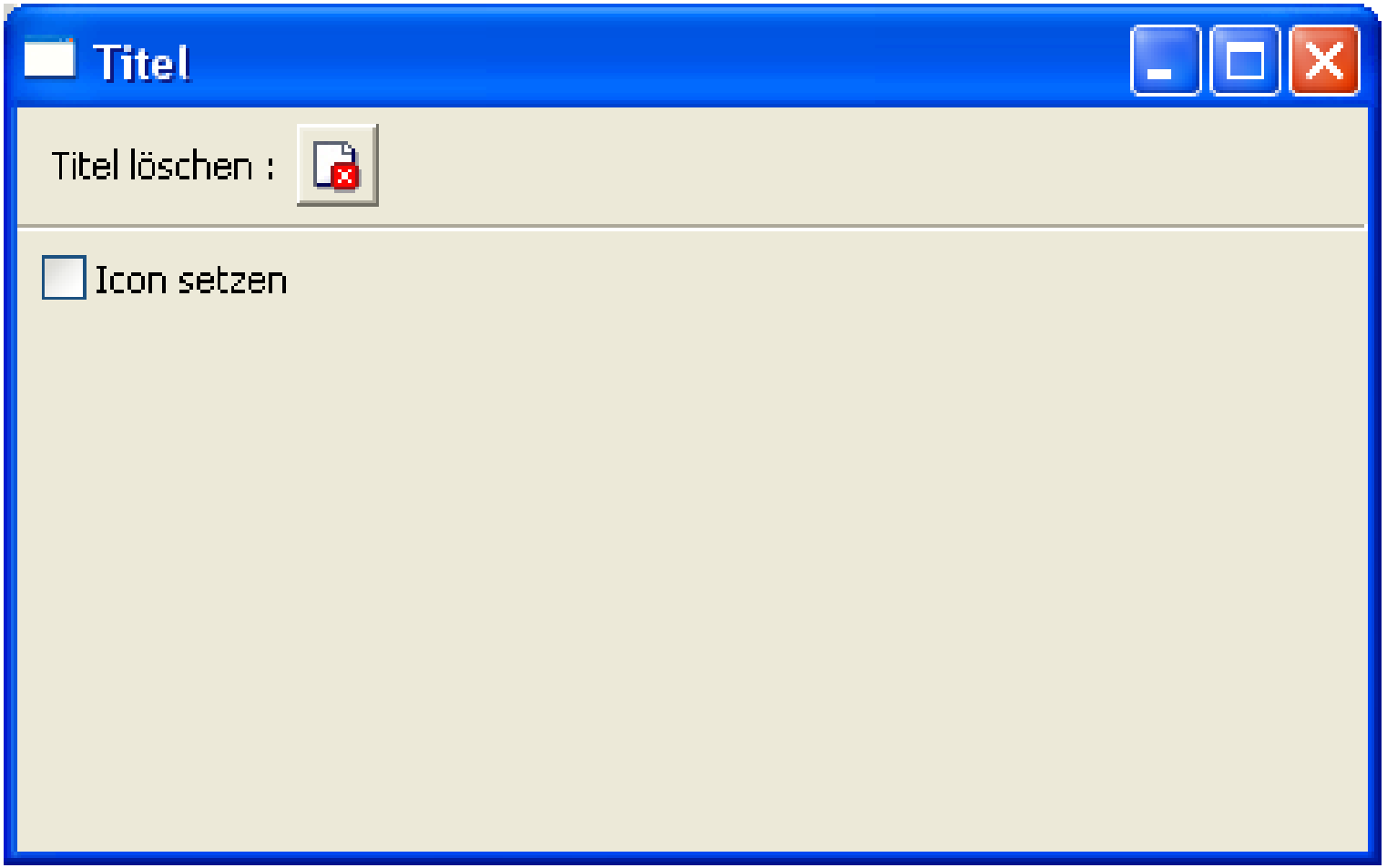
Parameter einer Evtroutine



Focus stört



Focus stört



- Wx::Window::SetFocus($linie);

Statuszeile

- Nur für Frames, nicht Windows
  - $bar = $frame->CreateStatusBar(1);
  - $bar = Wx::StatusBar($parent, $id, $style, $name);
  - $frame->SetStatusBar($bar);

Statuszeile

- Nur für Frames, nicht Windows
  - $bar = $frame->CreateStatusBar(1);
  - $bar = Wx::StatusBar($parent, $id, $style, $name);
  - $frame->SetStatusBar($bar);
  - $frame->GetStatusBar->SetFieldsCount(6);
  - $bar->SetStatusWidths(90, 66, 60, 40, 70, -1);
  - $win->SetStatusBarPane(5);
  - $win->GetStatusBar->Show(0|1);
  - $win->SetStatusText($feldnr, $text);

Menüzeile

- MenuBar -> Menu -> MenuItem

Menüzeile

- MenuBar -> Menu -> MenuItem

- $menuzeile = Wx::MenuBar->new();
  - $menuzeile->Append($menu, $title);
  - $menuzeile->Insert($pos, $menu, $title);
  - $menuzeile->Replace($pos, $menu, $title);
  - $menuzeile->Remove($pos);
  - alles was man mit Items auch tun kann ($id angeben)
  - $fenster->SetMenuBar($menuzeile);

Menüs

- $menu = Wx::Menu->new($title);
  - Methoden : Append, Prepend, Insert, Remove
  - kein Replace, alles was mit Items geht, suchen
  - $menu->Append($menuitem);
  - $menu->Append($id, $label, $menu, $helpstr);
  - $menu->Append($id, $label, $helpstr, $kind);
  - $kind : `wxITEM_SEPARATOR`, `wxITEM_NORMAL`, `wxITEM_CHECK`, `wxITEM_RADIO`
  - oder : $menu->AppendSeparator; # AppendCheckItem
  - gleiches gilt für Prepend & Insert auch
  - Remove nur mit $item oder $id

MenuEvent

- EVT_MENU_OPEN ($parent_win, \&$coderef);
  - \&coderef bekommt in @_ : $win, $menu_event
- EVT_RIGHT_DOWN ($widget, \&$coderef);
  - $_[0]->PopupMenu($menu, $_[1]->GetX, $_[1]->GetY)
  - PopupMenu ist Wx::Window Methode
  - Mutterklasse aller Widgets
  - MouseEvent kennt Koordinaten und weiteres

Menültem

- $menu = Wx::MenuItem->new(...);
  - $parent_menu , $id : hier ein Muss, autoinc > 5000
  - $label, : z.B. „Find\tCtrl+F“, !! autogenertes Keybind.
  - $helpstr, $kind : wie bekannt
  - $submenu : undef wenn kein submenu

- Noch etwas zu den ItemID's
  - Standartfunktionen haben benannte ID's
  - kein Muss, aber deren Zahlenbereich ausweichen

MenüItem

- $menu = Wx::MenuItem->new(...);

- $menu_item->SetBitmap($bmp); # is extra Schritt
- $menu_item->Enable(0 | 1);
- $menu_item->Check(0 | 1); # wenn Check o Radiobtn
- $menu->($item_id, 0 | 1); # oder auch mit $menubar

- IsChecked, IsCheckable, IsEnabled, IsMenu, IsSeperat.
- GetLabel („Find“) vs GetText („Find\tCtrl+F“)
- EVT_MENU($parent_win, $item_id, $coderef);

Haupttoolbar

- $tb =$frame->CreateToolBar($style, $id, $name);
  - klebt immer unterm Menü im Frame
- $tb->AddTool($ID, $label, $bmp, $hint, $kind);
- o. ($ID, $label, $bmp, $bmp2, $kind, $hint, $help, $data)
  - gleichen $kind const. wie bei Item
  - dito explizite Formen wie ->AddCheckTool(
  - $tb->InsertControl($pos, $widget);
    - für andere widget wie combobox (! button)

Toolbars

- $tb = Wx::ToolBar->new
  - ($parent, $id, $pos, $size, $style, $name);
  - mit Sizern positionierbar
  - $tb->SetRows(1);
  - $tb->SetToolBitmapSize(16, 15);
- $tb->ToggleTool($id, 0 | 1); # nur `wxITEM_CHECK`
- $tb->EnableTool($id, 0 | 1);
- EVT_TOOL($parent_win, $item_id, $coderef);
  - EVT_MENU ist das selbe

Dialog Boxen

- Wx::MessageBox($msg, $topic, $style, $parent, $x, $y)
  - wxYES | wxNO | wxYES_NO | wxCANCEL
  - wxICON_QUESTION | wxICON_INFORMATION
  - wxSTAY_ON_TOP
  - Konstanten sind auch Ergebnistyp

- $filename = Wx::FileSelector
  - ($title, $path, $file, $ext,$wildcards, $flags, $parent, $x, $y)
  - $wildcards = BMP files (*.bmp)|*.bmp|GIF files (*.gif)|*.gif

Dialoge

- statt Funktion geht auch Dialogobjekt
- Wx::FileDialog->new
  - ($parent, $topic, $dir, $file , $filterstring, $flags, @pos)
  - gleiche flags natürlich
- $answer = $dialog->ShowModal
- @files = $dialog->GetPaths

Dialoge

- oder eigene Dialoge bauen (Wx::Dialog ableiten)
- ist Wx::Window (kein menu- tool o. Statusbar)
- dafür mit eingebautem Panel u. Modalfunktion
- womit wir wieder fast am Anfang wären

Michael Ende

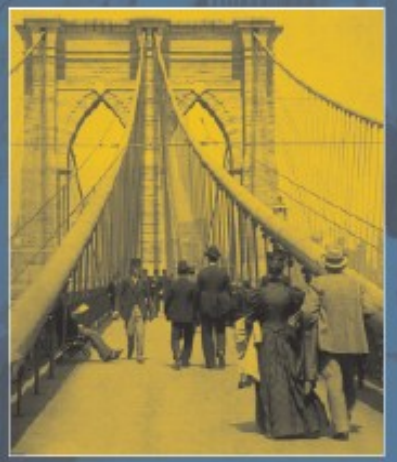
Die unendliche Geschichte

Thienemann

"This book is the best way for beginning developers to learn wxWidgets programming in C++. It is a must-have for programmers thinking of using wxWidgets and those already using it."
—Mitch Kapor, founder of Lotus Software and the Open Source Applications Foundation

BRUCE PERENS' OPEN SOURCE SERIES

CROSS-PLATFORM GUI PROGRAMMING WITH WXWIDGETS



- Build advanced cross-platform applications that support native look-and-feel on Windows, Linux, Unix, Mac OS X, and even Pocket PC
- Master wxWidgets from start to finish—even if you've never built GUI applications before
- Leverage advanced wxWidgets capabilities: networking, multithreading, streaming, and more
- CD-ROM: Library of development tools, source code, and sample applications

Foreword by Mitch Kapor, founder, Lotus Development and Open Source Application Foundation

JULIAN SMART AND KEVIN HOCK
WITH STEFAN CSOMOR

Stolperfallen

- Konstanten anmelden : use Wx qw(.....);
  - während Aufbau : use Wx qw(:all);
  - Events getrennt : use Wx::Event qw (...);

- Rückgabewerte können kritisch sein
  - die du setzt und bekommst
  - zur Sicherheit 1; am Ende (wie bei Modulen)
  - kenne die Hierarchien
  - Events skippen

Danke