Perl/Tk: A Quick Swim in Shallow Water

Eli Billauer

eli@billauer.co.il

Mini-Lecture's outline

- What is Perl/Tk?
- Perl and GUI
- Perl/Tk vs. Perl/Glade
- Porting to Win32
- An example – just to get the taste

What is Perl/Tk?

- A module set for Perl 5
- Allows scripting GUI
- Perl-like attitude (good defaults, flowing)
- Limited features
- Basic and practical look.

Perl and GUI

- Perl is intuitive and flowing. So should GUI be
- Perl is wasting resources. GUI does it anyway
- Easy handling of GUI data
- No memory allocations
- No pointers
- No casting
- No crashes
- No BS

Perl/Tk vs. Perl/Glade

Perl/Glade:

- Hands-on graphical design
- Fancy GUI is possible
- Encourages to deal with details (such as tooltips)

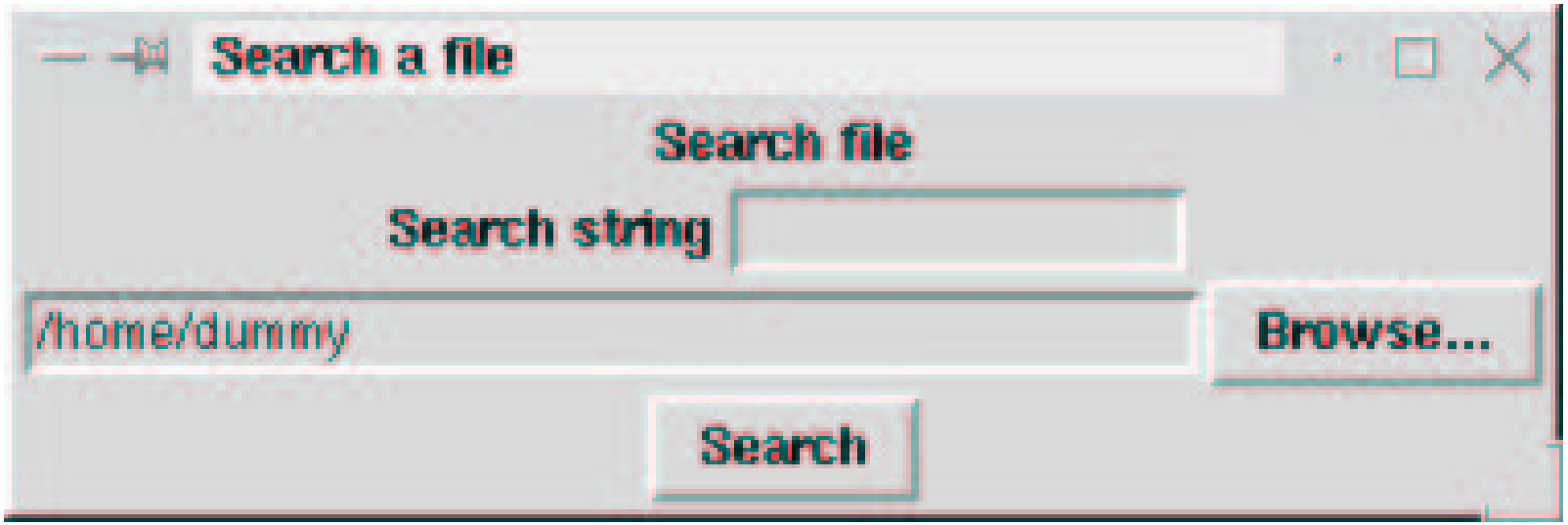
Perl/Tk:

- Short, clear and concise code
- The entire development suite may be one script
- Easier to start doing something with the windows
- No messy API
- Easy callbacks

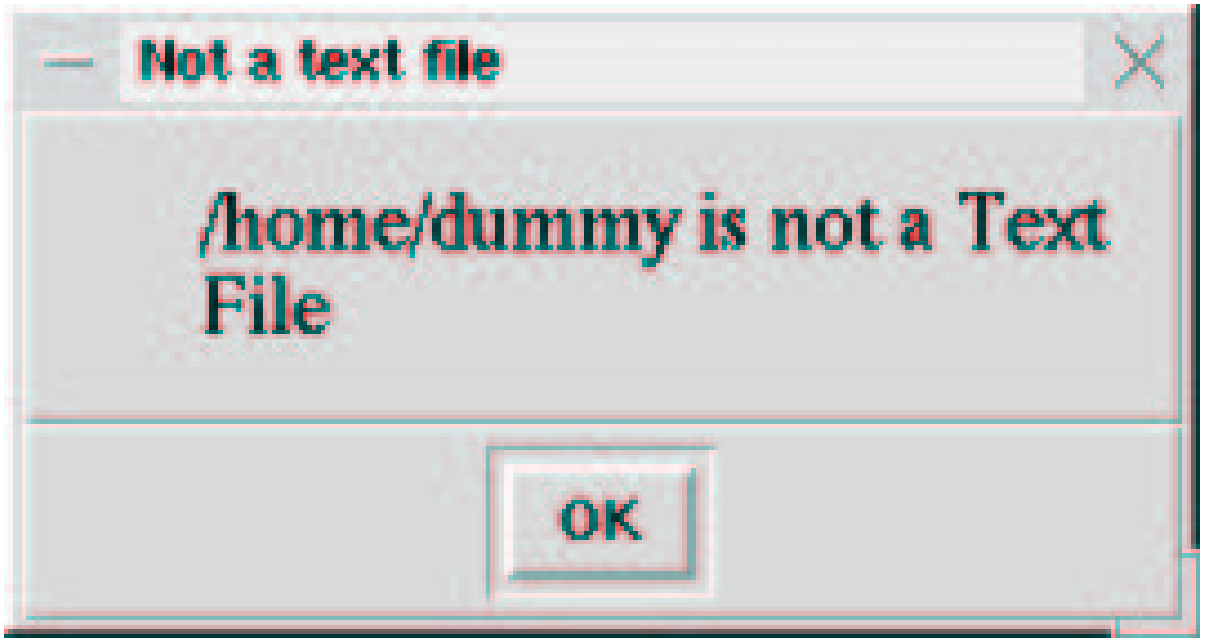
Porting to Win32

- My Perl under Windows supports Tk
- The GUI looks and feels X-windowish
- Works nice
- The perl2exe program allows exe file generation

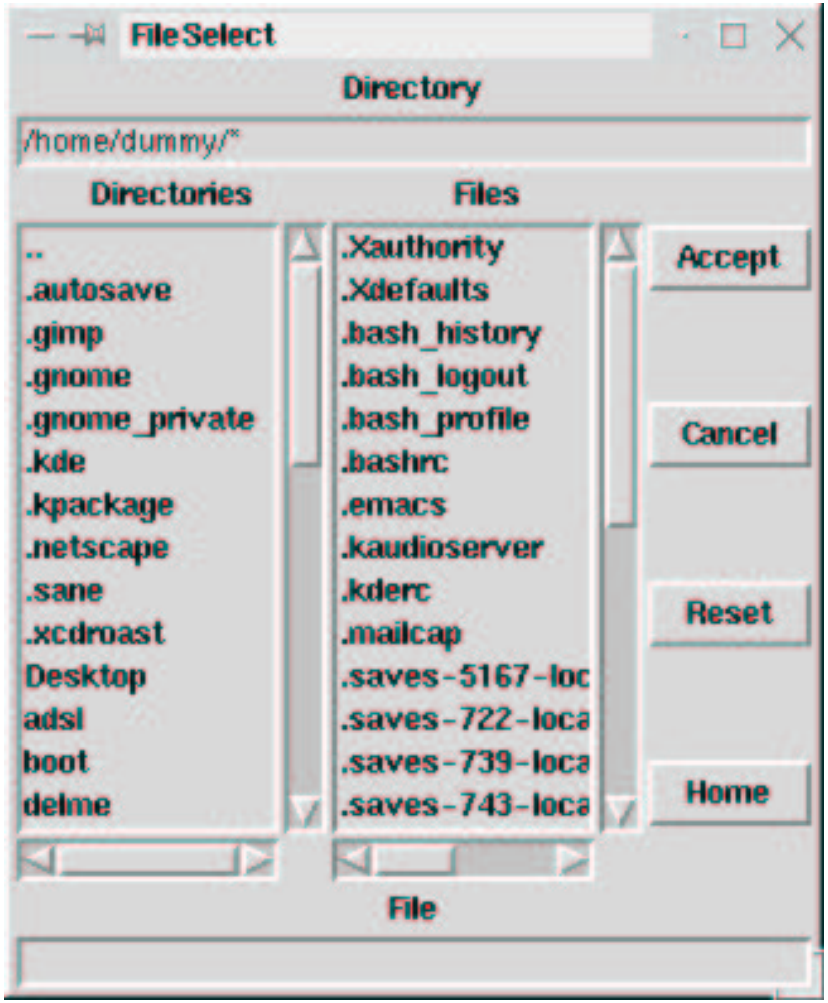
The main window



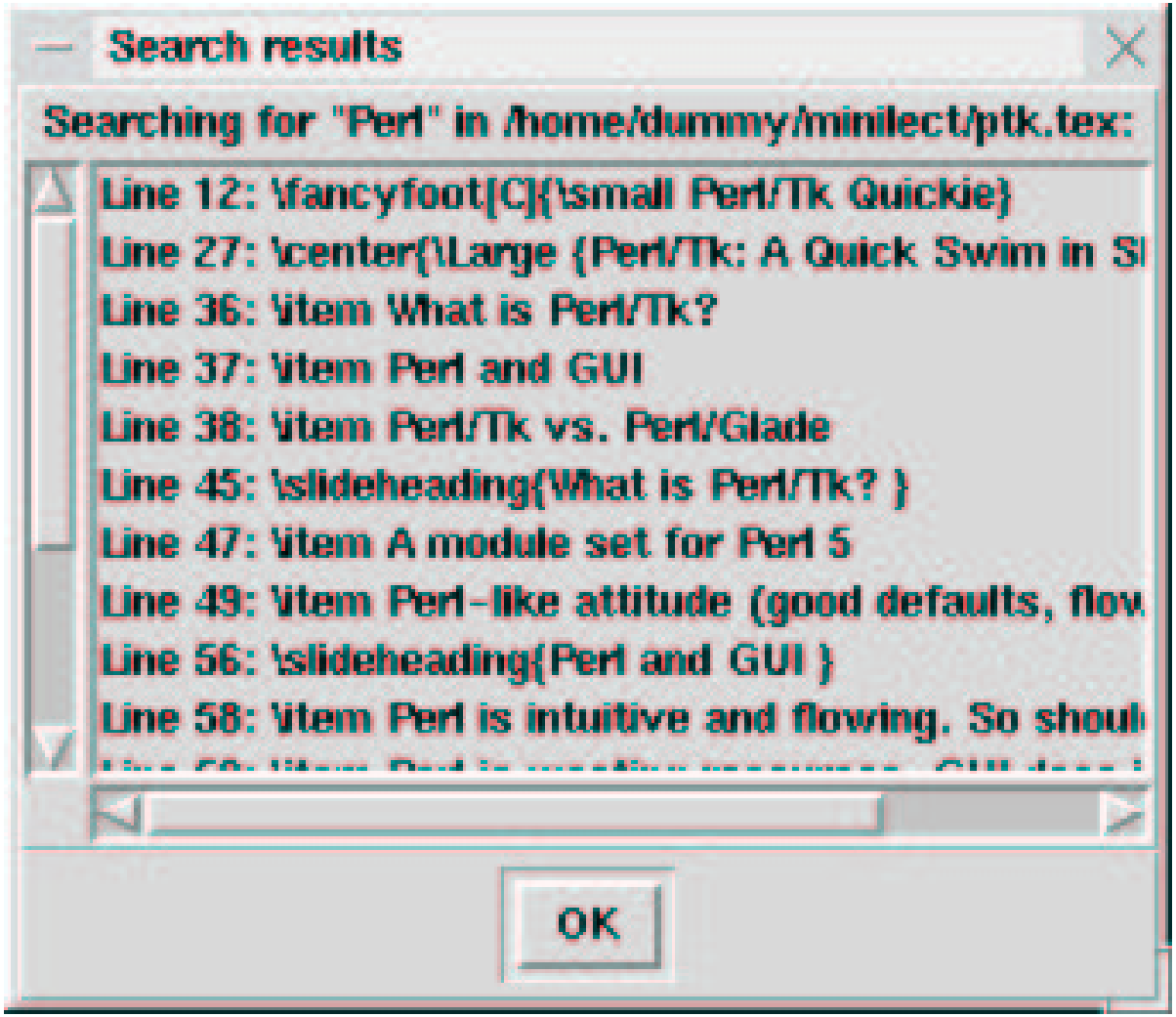
When no good file is chosen...



Browse window



Result window



The Perl script: Opening

```
#!/usr/bin/perl5

use strict;
use Tk;
require Tk::FileSelect;
require Tk::Dialog;

my $fname = $ENV{'HOME'};
```

The Perl script: First steps...

```
my $main = new MainWindow(-title => 'Search a file');
$main->Label(-text => 'Search file')->pack;

my $whatframe=$main->Frame;
$whatframe->pack;
$whatframe->Label(-text => 'Search string')
                          ->pack(-side => 'left');
my $what = $whatframe->Entry(-width => 15);
$what->pack(-side => 'right');
```

The Perl script: More!

```
my $fileframe=$main->Frame;
$fileframe->pack;
my $filename = $fileframe->Entry(-width => 40,
                                 -textvariable => \$fname);
$filename->pack(-side => 'left');
$fileframe->Button(-text => 'Browse...',
      -command => sub{$fname=do_browse($fname);}
  )->pack(-side => 'right');
$main->Button(-text => 'Search',
      -command => sub{do_search($fname, $what->get)}
     )->pack;
MainLoop;
```

The Perl script: Browse Window

```
sub do_browse {
  my ($dir) = @_;

  my $fs=$main->FileSelect;
  my $new=$fs->Show;
  if (-T $new) {
    $dir=$new;
  } else {
    bad_file($new);
  }
  return $dir;
}
```

The Perl script: “Bad File” dialog

```
sub bad_file {
  my ($sel) = @_;
  $sel="Your selection" unless $sel;
  my $dialog=$main->Dialog(-title => 'Not a text file',
   -text => "$sel is not a Text File");
  $dialog->Show;
}
```

The Perl script: Doing the work

```
sub do_search {
  my ($f, $w) = @_;

  unless (-T $f) {
    bad_file($f);
    return;
  }
```

The Perl script: The action itself

```
open (FILE,$f);
my $line;
my $tmp;
my @found=();

for ($line=1; !eof(FILE); $line++) {
  $tmp=<FILE>;
  chomp $tmp;
  push @found, "Line $line: $tmp" if $tmp=~/$w/;
}
```

The Perl script: Showing it all

```
  my $dialog=$main->DialogBox(-title => 'Search results');
  my $frame=$dialog->add('Frame');
  $frame->pack(-fill => 'both', -expand => 1);
  $frame->Label(-text => "Searching for \"$w\" in $f:")->pack;

  my $listbox = $frame->Scrolled('Listbox');
  $listbox->pack(-fill => 'both', -expand => 1);
  $listbox->insert('end',@found);

  $dialog->Show;
  close FILE;
}
```

Conclusion

- It's as messy as any Perl script
- It's less messy than C or C++ code
- It's written and read by humans
- The results are OK