Grafisches Programmieren mit Tcl/Tk

Workshop

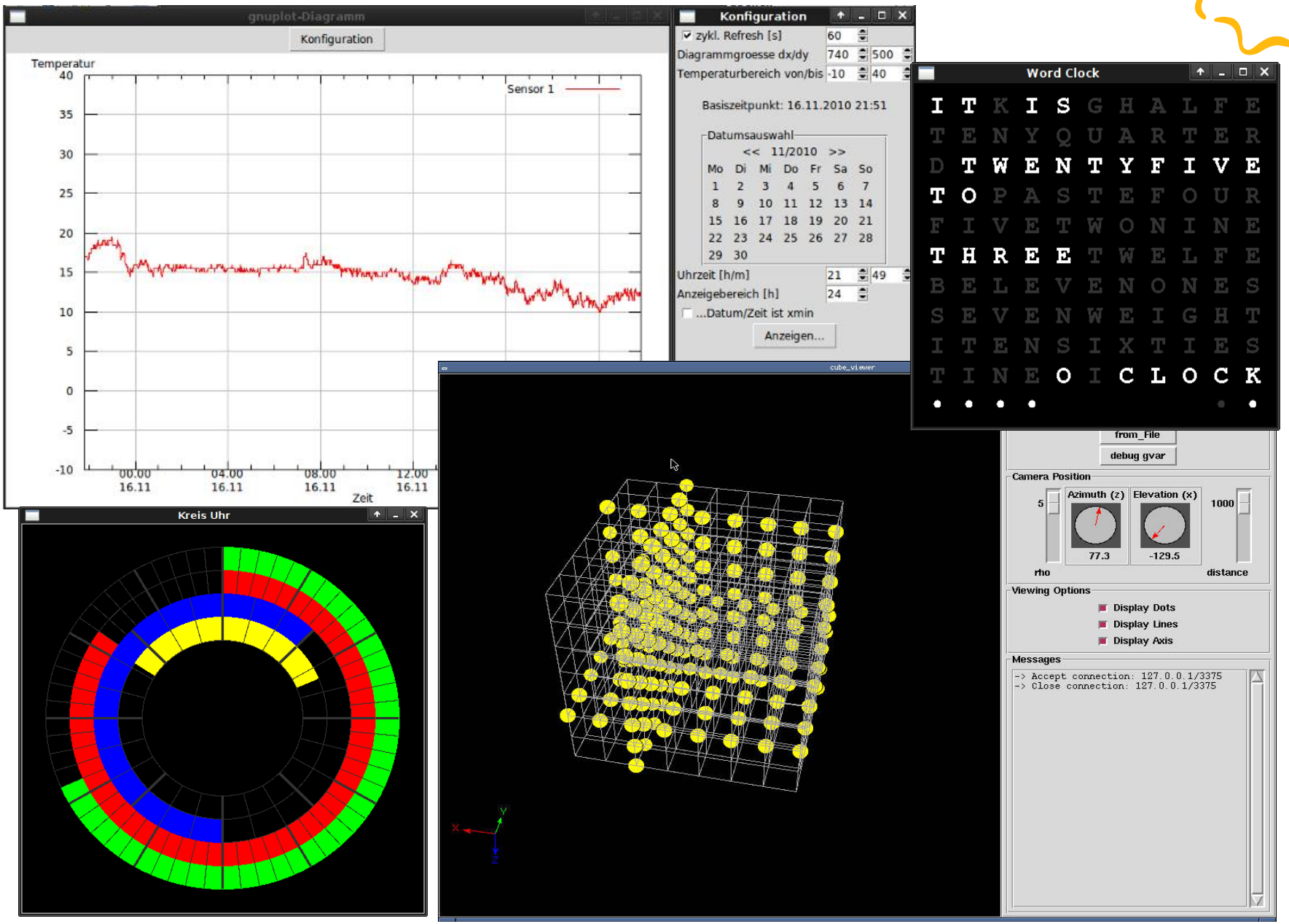
Uwe Berger; 2011

Uwe Berger

- Beruf: Softwareentwickler (PPS-Systeme)
- Linux seit ca. 1995
- Beginn meiner Tcl-Geschichte: 1995/96 (zufällig das Tcl/Tk-Buch in die Hand bekommen...)
- Freizeit: Hard- und Softwarespielereien
- Brandenburger Linux User Group e.V. (BraLUG)
- Brandenburger Linux Infotag (BLIT)
  → 8.BLIT am 05.11.2011 in Potsdam

Motivation

- Problem:
  - Programmierung von grafischen Oberflächen mit den herkömmlichen Methoden ist in der Regel nicht trivial
  - man muss sich dabei meist auf ein Toolkit festlegen (GTK, QT etc.)
- Lösung:
  - Tk im Verbund mit Tcl → eine mächtiges Toolkit zur Programmierung von grafischen Oberflächen, die sogar plattformunabhängig sind!
- Was geht
  - wirklich (fast) alles, was man sich nur wünschen kann...!


gnuplot-Diagramm
Konfiguration
Temperatur
Sensor 1
Zeit
Konfiguration
zykl. Refresh [s]
Diagrammgroesse dx/dy
Temperaturbereich von/bis
Basiszeitpunkt: 16.11.2010 21:51
Datumsauswahl
Uhrzeit [h/m]
Anzeigebereich [h]
...Datum/Zeit ist xmin
Anzeigen...
Word Clock
ITKISGHALFE
TENYQUARTER
DTWENTYFIVE
TOPASTEFOUR
FIVETWONINE
THREETWELFE
BELEVENONES
SEVENWEIGHT
ITENSIXTIES
TINEOICLOCK
Kreis Uhr
cube_viewer
from_File
debug gvar
Camera Position
Azimuth (z)
Elevation (x)
77.3
-129.5
rho
distance
Viewing Options
Display Dots
Display Lines
Display Axis
Messages
-> Accept connection: 127.0.0.1/3375
-> Close connection: 127.0.0.1/3375

Warum seid ihr zu diesem Workshop gekommen?

Was wollt ihr lernen/hören?

Inhalt

- Tcl/Tk – einführende Bemerkungen
- Ein paar Tcl-Grundlagen
- Grafische Oberflächen mit Tcl/Tk
  - Tk-Grundlagen
  - Widgets
  - Geometriemanager
  - Event Handler
  - Windowmanager
  - Fonts
  - Canvas
  - komplexere Beispiele

Tcl/Tk – Was ist das?

Tcl

- Tcl - Tool command language
- eine Scriptsprache wie bash, Perl, php, etc.
- eine Interpretersprache (tclsh, wish)
- auf allen bedeutenden OS-Plattformen verfügbar
- John Ousterhout, ca. 1988
- http://www.tcl.tk/about/history.html
- Simulation von Schaltkreisen
- sollte neben Java auch „Websprache“ werden
- Open Source

Tk

- Tk - Toolkit
- Programmierung von grafischen Benutzeroberflächen
- wurde ursprünglich für Tcl entwickelt (ebenfalls J.Ousterhout; 1988)
- Plattformübergreifend verfügbar (soweit grafische Oberfläche vorhanden)
- Anbindung auch für Perl, Python, Ruby und einige weitere vorhanden
- Open Source

Tcl-Grundlagen

Informationsquellen

- "Tcl und Tk. Entwicklung grafischer Benutzerschnittstellen für X Window System"; Ousterhout; Addison-Wesley
- "Effektiv Tcl/Tk programmieren"; Harrison, McLennan; Addison-Wesley
- http://wiki.tcl.tk
- http://www.activestate.com
- http://www.tcl.tk
- Tcl-Newsgroup (engl.): comp.lang.tcl

Tcl Syntax-Prinzipien

- durchgängig Polnische Notation:
  → Befehlswort Parameter1 Parameter2 …
  → Kommentar ist ein Befehl: # ...
- "Alles ist ein String"
  - alle Datentypen werden zuerst als String behandelt
  - Funktionsrückgabewerte sind immer Strings
  - Programmcode kann zur Laufzeit zusammengesetzt und evaluiert werden
- Zeichenketten stehen in doppelten Anführungsstrichen

Tcl Syntax-Prinzipien

- geschweifte Klammern schützen den Inhalt vor (der ersten) Interpretation (wirken von außen nach innen)
- Code in eckigen Klammern wird zuerst ausgeführt (wirken von innen nach außen) und wird durch den Rückgabe(-String) ersetzt
- ein Befehl wird von einem Zeilenende oder Semikolon begrenzt
- Backslash (\) am Zeilenende → Befehl geht auf nächster Zeile weiter

Tcl Syntax-Prinzipien

- für den Zugriff auf Variableninhalte muss dem Variablennamen ein $ voran gestellt werden
- Variablen nur lokal gültig und Gültigkeitsbereich muss explizit erweitert werden (global, upvar, uplevel)

Starten eines Tcl-Script

- Interaktiv in der Tcl-Shell (tclsh, wish) selbst; z.B.:
```
% source tcl_script.tcl
```
- Script als Parameter der Tcl-Shell:
```
tclsh tcl_script.tcl
wish tk_script.tcl
```
- Als ausführbares Script mit Angabe des Interpreters:
```
#!/usr/bin/tclsh
package require Tk
...
```

Tcl – „coole“ Sachen...

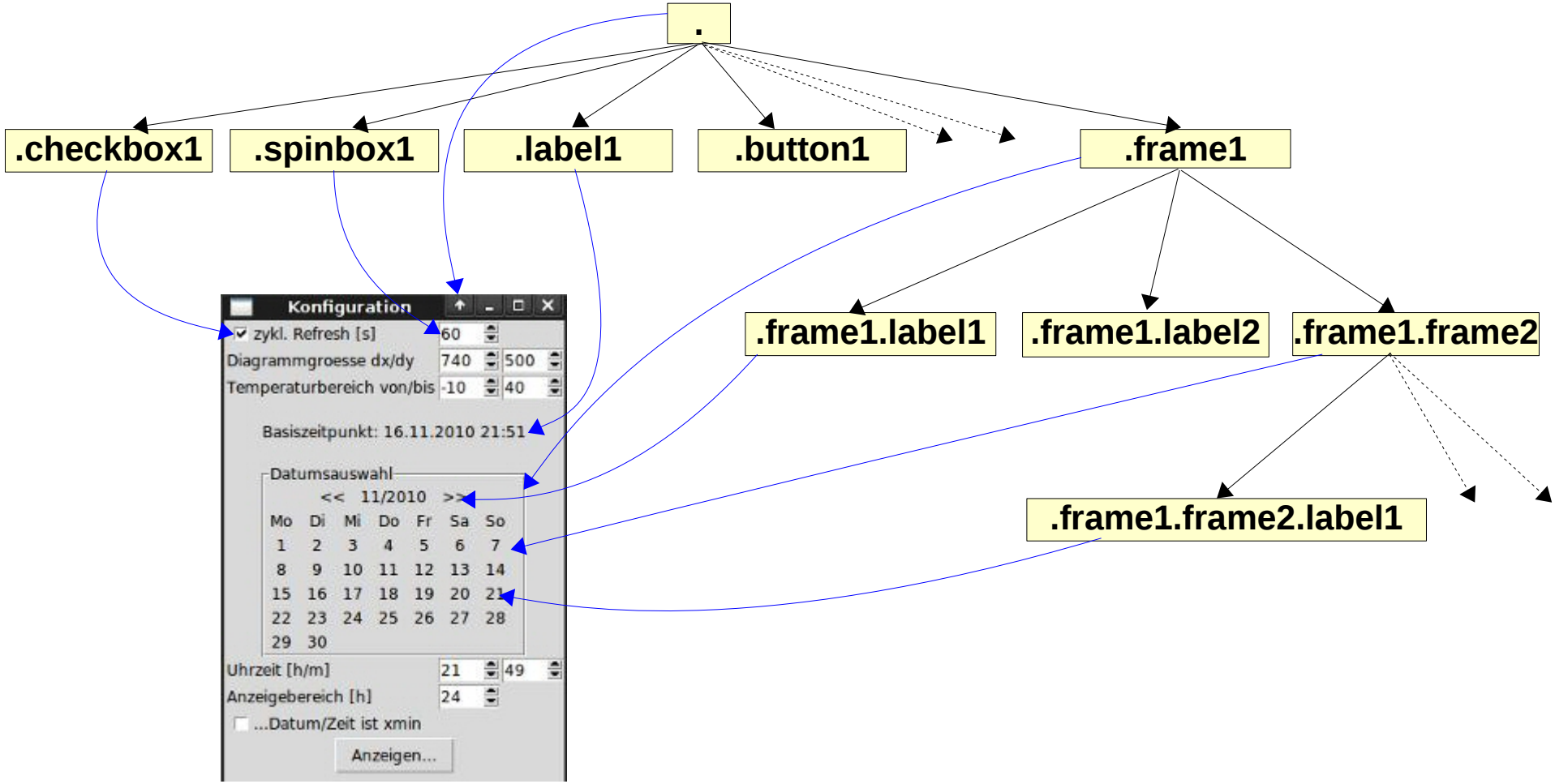
- ereignisgesteuerte Mechanismen für Socket- und Datei-Schnittstellen
- Seriell-, Parallel- und Netzwerk-Schnittstellen werden als Dateien behandelt
- Zeit- und Benutzerdefinierte Ereignisse
- einfache Client-/Server-Programmierung
- sehr gute Unterstützung bei der Verarbeitung von Listen und Arrays
- ... man findet immer wieder neue interessante Dinge ...

Grafisches Programmieren mit Tcl/Tk

Tk-Grundlagen

- eine grafische Oberfläche besteht aus
  - (mindestens) einem Fenster
  - einzelnen Grundelementen (Widgets)
  - einem Event-Handler
- grafische Elemente werden in Containern (Fenster, Frame etc.) hierarchisch angeordnet
- die Art und Weise der Anordnung der Elemente im Container bestimmt der Geometriemanager
- eine grafische Oberfläche wartet auf Benutzeraktionen bzw. andere externe Ereignisse → Event-Handler

Tk-Grundlagen - Objekthierarchie



Widgets

- die grafischen Grundelemente
- die wichtigsten Standard-Widgets:
  - Frame, Labelframe
  - Button, Checkbox, Radiobutton, Spinbox,
  - Label
  - Textbox, Entry, Listbox
  - Scrollbar, Slider
  - Canvas
  - Menue
  - Standarddialoge
  - Toplevel

Widgets

- Syntax (allgemein):
  `widgettyp .wname [Optionen]`
- Syntax-Beispiel (z.B. Button)
  `button .button1 -text "Ende" -command exit`
- Rückgabewert ist der Name des Widgets
  → Empfehlung: lange Namen in Variablen ablegen
- es wird ein neuer Befehl für jedes definierte Widget kreiert (Stichwort: durchgängige Polnische Notation...)
- erst mit dem Aufruf des Geometriemanagers zu jedem Widget, wird dieses angezeigt

Widgets

- Widget-Eigenschaften unterteilen sich in:
  - Standard-Eigenschaften, die jedes Widget hat (z.B. Farbe, Größe, Beschriftung etc.)
  - Widget-spezifische Eigenschaften (...)
- unbedingt dazu immer ein Manual griffbereit haben
- Widgets-Eigenschaften sind zur Laufzeit änderbar:
  `.wname configure [Optionen]`

Widgets

- Beispiele:
  - diverse Widgets: beispiele/widgets/widgets.tcl
  - Scrollbars: beispiele/widgets/scrollbar.tcl
  - Paned-Window: beispiele/widgets/panedwin.tcl
  - Menüs: beispiele/widgets/menue.tcl
  - Toplevel: beispiele/widgets/toplevel.tcl
  - ein kleiner „Editor“: beispiele/widgets/list_text_scroll.tcl
  - Standarddialoge: beispiele/widgets/standarddialoge.tcl
  - „kein Taschenrechner“ ;-): beispiele/widgets/calc.tcl

Widgets

- welche Widgets vermisst man im Tk-Standard?
  - Combobox
  - Tree
  - „Tabbed Folder“
  - Einbinden von diversen Bildformaten (JPEG, PNG etc.)
- es existieren zahlreiche Zusatzpakete, die keine Wünsche offen lassen, z.B.:
  - BWidget
  - Iwidgets
  - Tklib
  - tix

Geometriemanager

- ordnen die Widgets nach festgelegten Mustern/Algorithmen im Container an und stellen sie dar
- ordnen die Widgets neu, wenn sich die Größe des Containers ändert
- folgende Geometriemanager gibt es:
  - pack → „der Verpacker“
  - grid → „das Gitternetz“
  - place → „der Platzierer“
- die unterschiedlichen Geometriemanager können unter gewissen Voraussetzungen parallel angewendet werden

Geometriemanager: pack

- die Widgets werden nach Richtungen in Spalten oder Zeilen angeordnet
- die Plazierung erfolgt immer von einem Rand des Containers aus
- als Richtungsangaben sind Himmelsrichtungen bzw. Seitenbezeichnungen zulässig
- bei Größenänderung des Containers erfolgt eine Neupositionierung der Widgets nach den festgelegten Regeln
- Syntax: `pack .wname1 [.wname] [Optionen]`

Geometriemanager: pack

- Optionen:
  - -anchor → Ausrichtung der Widgets
  - -expand → Ausdehnung des Widgets
  - -after → nach einem anderen Widget
  - -before → vor einem anderen Widget
  - -fill → füllt vorhandenen Platz in vorgeg. Richtung
  - -ipadx → internes Padding in x-Richtung
  - -ipady → internes Padding in y-Richtung
  - -padx → externes Padding in x-Richtung
  - -pady → externes Padding in y-Richtung
  - -side → Seitenposition des Widgets
- Beispiele: beispiele/geometriemanager/pack*.tcl

Geometriemanager: grid

- gezielte Anordnung der Widgets in einem gedachten Raster (Zeilen/Spalten) innerhalb des Containers
- das größte Widget in der jeweiligen Spalte bzw. Zeile bestimmt deren Breite bzw. Höhe
- hervorragend geeignet für typische Dialog-Layouts (Kombination Label/Eingabefeld...) und Tabellen
- bei Größenänderung des Containers erfolgt eine Neupositionierung der Widgets nach den festgelegten Regeln
- Syntax: `grid Widget1 [Widget2] [Optionen]`

Geometriemanager: grid

- Optionen:
  - -ipadx → internes Padding in x-Richtung
  - -ipady → internes Padding in y-Richtung
  - -padx → externes Padding in x-Richtung
  - -pady → externes Padding in y-Richtung
  - -sticky → Ausrichtung in Gitternetzzelle vorgeben
  - -row → Zeilenzahl zuweisen
  - -column → Spaltenzahl zuweisen
  - -rowspan → Zeilen zusammenfassen
  - -columnspan → Spalten zusammenfassen
- Beispiele: beispiele/geometriemanager/grid*.tcl

Geometriemanager: place

- der Vollständigkeit halber...
- definierte Anordnung der Widgets in einem Container nach festgelegten xy-Koord., Höhe/Breite
- pixelgenaues Positionieren von Widgets im Container
- Vor- oder Nachteil? Widgets können auch so positioniert werden, dass sie sich überlappen bzw. beschnitten sind
- die Positionierung wird auch bei Veränderung der Containergröße beibehalten
- Syntax: `place .wname [Optionen]`

Geometriemanager: place

- Optionen:
  - -x → absolute x-Koordinate
  - -y → absolute y-Koordinate
  - -anchor → „Null-Punkt-Ecke“ des Widgets
  - -relx → relative x-Koordinate
  - -rely → relative y-Koordinate
  - -height → Höhe
  - -width → Breite
  - -relheight → relative Höhe
  - -relwidth → realtive Breite
- Beispiele: beispiele/geometriemanager/place*.tcl

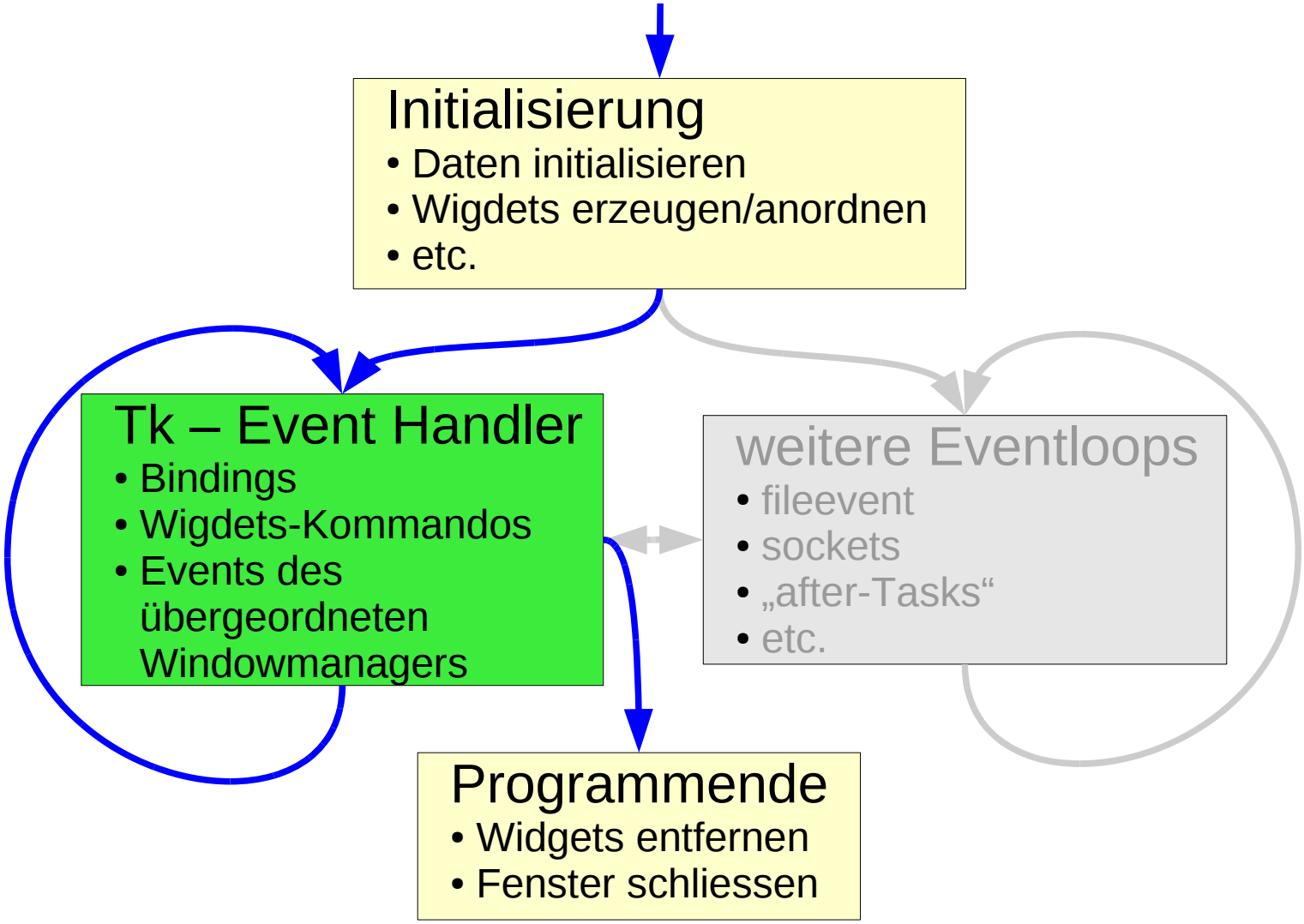
Geometriemanager

- die Eigenschaften der Widgets innerhalb ihres Geometriemanagers können abgefragt und verändert werden
- wichtigste Methoden:
  - configure → Umkonfiguration eines Widgets
  - forget → Entzug der Kontrolle über ein Widgets
  - info → Liste der Eigenschaften eines Widgets
  - slaves → Liste der Widgets im Container
  - … plus einiger spezifischer Methoden → siehe Manual
- Syntax z.B.: `pack configure .wname [Optionen]`

Geometriemanager

- weitere Beispiele:
  - beispiele/geometriemanager/configure.tcl
  - beispiele/geometriemanager/combi.tcl

Tk-Grundlagen – Event Handler



Event Handler: Option -command

- Buttons, Checkboxen, Radiobuttons etc. kann ein Kommando bzw. Script übergeben werden, welches bei Betätigung evaluiert wird
- Syntax (z.B. bei button):
```
button .b -text "Beenden" -command exit
```
- Beispiel: beispiele/event_handler/command.tcl

Event Handler: Bindings

- Ereignissen zu einem Widgets, können Kommandos zugeordnet werden
- Ereignisse können u.a. sein (siehe auch Manual):
  - Mauszeiger im Bereich des Widgets (und/oder Maustaste)
  - Geometrieänderungen
  - Tastenbetätigungen
- Syntax: `bind .wname <event> {script}`
- Beispiel:
  - beispiele/event_handler/bind.tcl
  - beispiele/event_handler/bind_win.tcl

Windowmanager allgemein

- ein Windowmanager ist nicht X, sondern nur ein Programm welches unter X läuft!
- Aufgaben eines Windowmanagers sind u.a:
  - Anordnen, Vergrößern, Minimieren, Verschieben, Schließen von Fenstern
  - Fensterdekoration (Titelleiste, Umrandung)
  - diverse Effekte
- Windowmanager z.B.: Compiz, Xfwm, Openbox, IceWM, Metacity, Kwin
- X stellt „nur“ die Grafikprimitiven zur Verfügung

Windowmanager & Tk

- wish: (die Tk-Windowshell) stellt das erste Fenster mit dem Namen . automatisch zur Verfügung
- es existieren Tk-Befehle zum:
  - Abfragen von Informationen über Fenster → `winfo`
  - Manipulieren des Verhaltens von Fenstern → `wm`
- auch hier empfiehlt sich ein Blick ins Manual...
- Beispiele:
  - beispiele/windowmanager/bind_geo.tcl
  - beispiele/windowmanager/helpballoon.tcl

Schriftart setzen

- die Schriftart der meisten Widgets kann mit der Option `-font` gesetzt werden
- 1.Methode:
  - es wird die „UNIX-übliche“ Darstellungsform verwendet, z.B.:
    -misc-fixed-bold-r-normal—0-0-75-75-c-0-iso8859-9
  - Wildcards sind zulässig, es wird dann die jeweils erste zutreffende Option verwendet
  - siehe (unter Linux) auch: `xfontsel, xlsfonts`
  - Beispiel: beispiele/font/font.tcl

Schriftart setzen

- 2.Methode (seit Tcl/Tk 8.0):
  - `font`-Befehl
  - Option `-font` als Liste angebbar
  - Beispiele:
    - beispiele/font/font_new.tcl
    - beispiele/font/font_create.tcl
    - beispiele/font/font_liste.tcl

Widget: Canvas

- Canvas → die Leinwand
- der Widget-Befehl canvas stellt eine Zeichenfläche zur Verfügung
  `canvas .wname [Optionen]`
- diese Zeichenfläche kann Grafikelemente aufnehmen
  `.wname create type <Koord.> [Optionen]`
- Grafikelemente: Linien, Kreissegmente, Ovale, Polygone, Vierecke, Text, Windows(?), Bitmaps, Bilder
- Beispiel: beispiele/canvas/canvas_items.tcl

Widget: Canvas

- die Eigenschaften der Grafikelemente können zur Laufzeit verändert werden, dazu muss dem Element ein Name gegeben werden (Option -tags)
- mehrere Grafikelemente mit gleichen Namen (-tags) können mit einem Befehl umkonfiguriert werden
- `.wname itemconfigure tag [Optionen]`
- Beispiele:
  - beispiele/canvas/canvas_tags.tcl
  - Word-Uhr: beispiele/canvas/word_uhr.tcl
  - Berliner Uhr: beispiele/canvas/berliner_uhr.tcl
  - Kreisuhr: beispiele/canvas/kreis_uhr.tcl

Tk & gnuplot?

- gnuplot: ein Kommandozeilentool zur grafischen Darstellung von Messreihen und mathematischen Funktionen
  → http://www.gnuplot.info
- neben diversen anderen Ausgabeformaten gibt es auch:
```
set terminal tkcanvas size 800, 600
set output 'canvas.tk'
```
- es wird eine Datei erstellt, welche die anzuzeigenden Daten als Grafik-Elemente für ein Canvas-Widget enthält
- Beispiel: beispiele/weitere_beispiele/gpl_gui.tcl

War es das?

- Nein, da fehlt noch eine ganze Menge, wie z.B.:
  - die unzähligen Optionen und Möglichkeiten jedes einzelnen Widgets
  - Manipulationen des Event Handlers
  - Mauszeiger-Manipulation
  - Fokus vorgeben, -reihenfolge konfigurieren
  - Bilder einbinden
  - die zahlreichen Zusatzpakete für Tk
  - ...
- Viel Spaß beim experimentieren!

Weitere Tk-Beispiele

- im Verzeichnis: beispiele/weitere_beispiele
  - CPU-Usage: cpu_usage.tcl
  - /proc-Browser: tk_proctree.tcl
  - Bilderrätsel: puzzle.tcl
  - iPod befüllen: tk_gnupod.tcl

- Anmerkung: es werden teilweise ein paar Zusatzpakete benötigt

Zum Abschluss ein paar Hinweise...

Programmierstil (allgemein)

- "Es gibt 1000 Wege nach Rom...", jeder muss seinen eigenen „Tcl-Stil“ finden
- geschachtelte Klammern ({}, []) nur soweit, wie man selbst den Code noch versteht
- möglichst Tcl/Tk nur im Standardsprachumfang verwenden: unkomplizierte Weitergabe der Programme
- Fehlerbehandlung nicht vergessen!
- wenn möglich plattformunabhängigen Code schreiben bzw. entsprechende Spezifika unterscheiden

Programmierstil (Tk)

- wenn möglich nur Geometriemanager pack und grid verwenden: automatische Anordnung der Widgets
- zuerst Oberfläche vollständig konzipieren und als „Programmhülse“ realisieren, dann erst Funktionalität
- wenn möglich grafische Elemente nicht immer neu zeichnen, sondern nur die Eigenschaften modifizieren

- ...und ganz wichtig: Immer ein Tcl/Tl-Manual griffbereit haben!

Einsatzgebiete für Tcl/Tk

- eigentlich überall da, wo es nicht auf extreme Performance ankommt (→ Interpretersprache)
- man kommt mit wenigen Codezeilen schnell zum Ziel, da die einzelnen Syntax-Elemente sehr flexibel und mächtig sind
- es gibt ein paar Mechanismen, bei denen man sich in anderen Programmiersprachen sehr schwer tun würde
- GUI-Entwicklung
- auch für sehr komplexe Anwendungen tauglich
- plattformunabhängige Anwendungen

Danke...!

<u>Achtung Werbung:</u>
heute 16:00 Uhr; V3; Vortrag „uBasic – eine plattformunabhängige BASIC-Interpreterbibliothek“