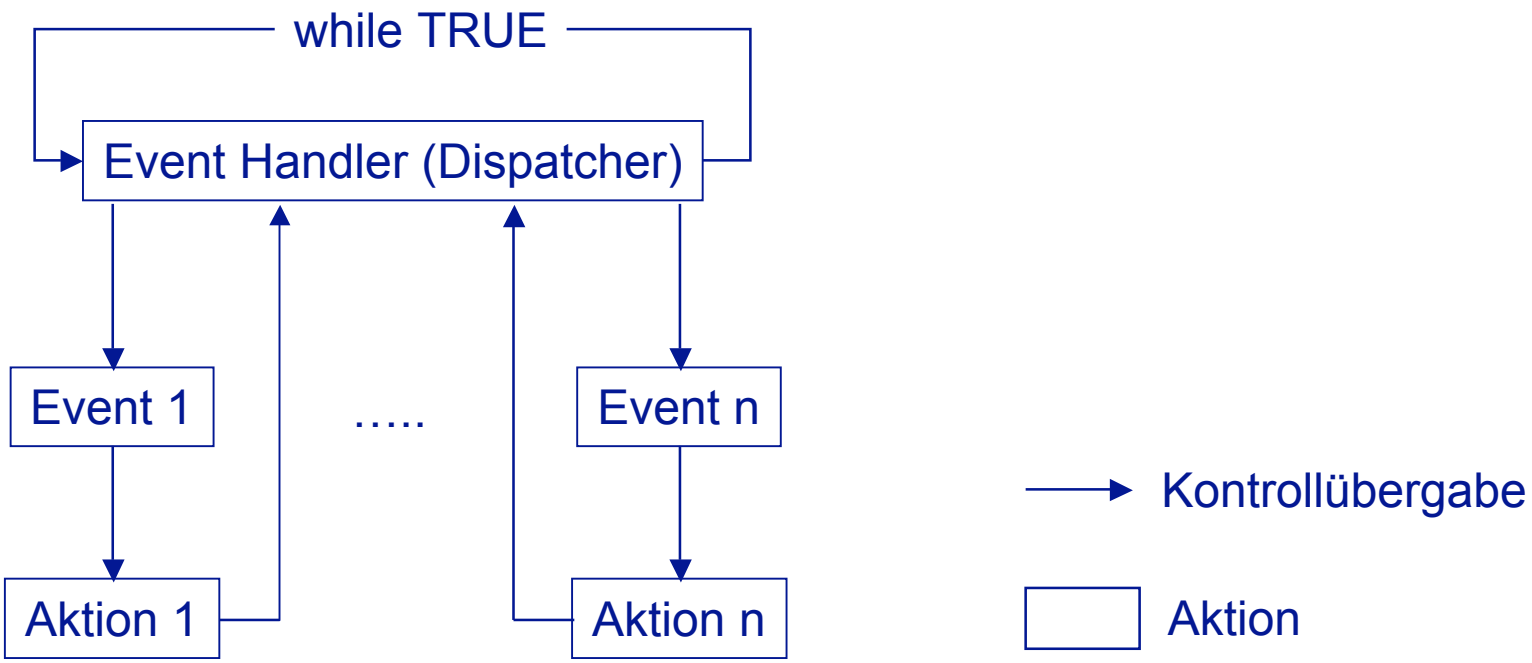
6.3 Ereignisgesteuerte Programmierung

Im Gegensatz zur klassischen Programmierung geht man bei der ereignisgesteuerten Programmierung davon aus, dass ein Programm immer aktiv bleibt, bis es explizit beendet wird. Die Beabeitung geschieht durch Präsentation von Benutzeroberflächen, auf denen jeweils eine Eingabe vorgenommen und abgeschickt wird. Nach Abschluss einer Plausibilitätsprüfung für die Eingaben wird die zugehörige Berechnung ausgeführt und dann mit dem nächste Eingabe-Bildschirm fortgefahren.

Ein Beispiel für die ereignisgesteuerte Programmierung ist das SAP-System R/3 mit der Sprache ABAP. Ein weiteres Beispiel sind die Sprache Tcl und der „Tk Toolkit“.

Ablaufschema der Ereignissteuerung

Das grundsätzliche Ablaufschema sieht aus wie folgt:



Beispiele für Events

- Tastaturklick
- Mausbewegung
- Mausklick
- …

Eventbasiert arbeitende Toolkits

- Java Swing
- SAP R/3 mit ABAP
- Athena Widgets des X Window Systems
- Interviews
- Fresco

  => Übersetzung in C - / C++ - Code

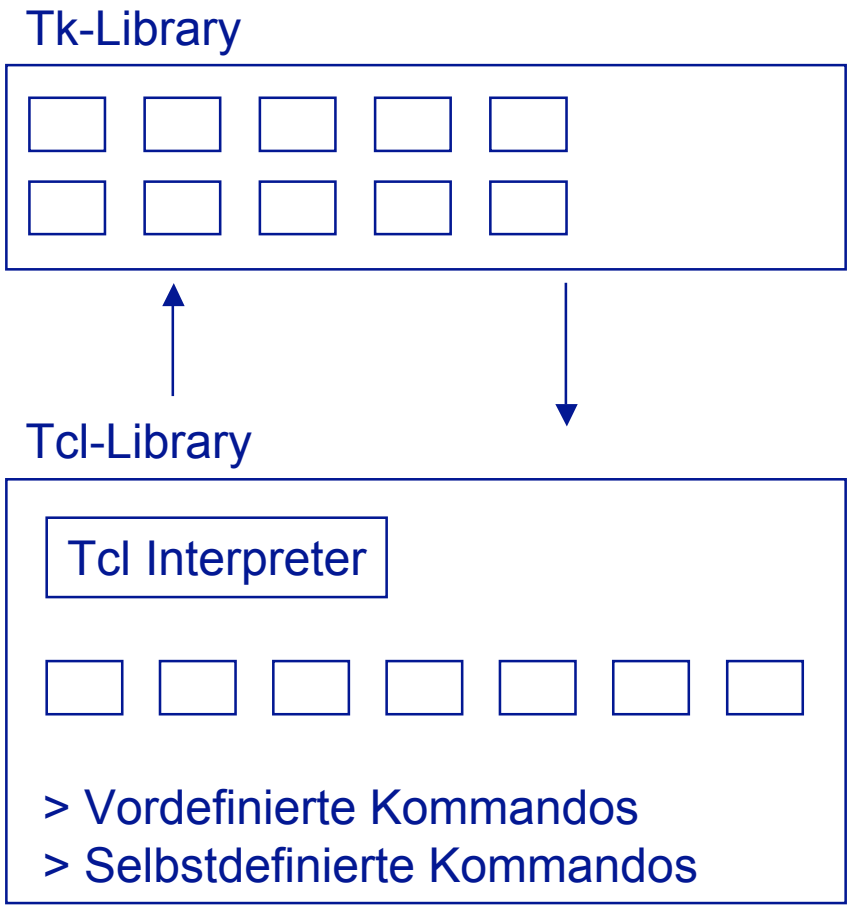
- Tcl/Tk

  => Interpretation

Der Tcl/Tk-Toolkit

- Seit 1987 von John K. Ousterhout entwickelt
- X11-basiert
- frei erhältlich
- Tcl: Script-Programmiersprache

Arbeitsweise des Tcl/Tk- Toolkit

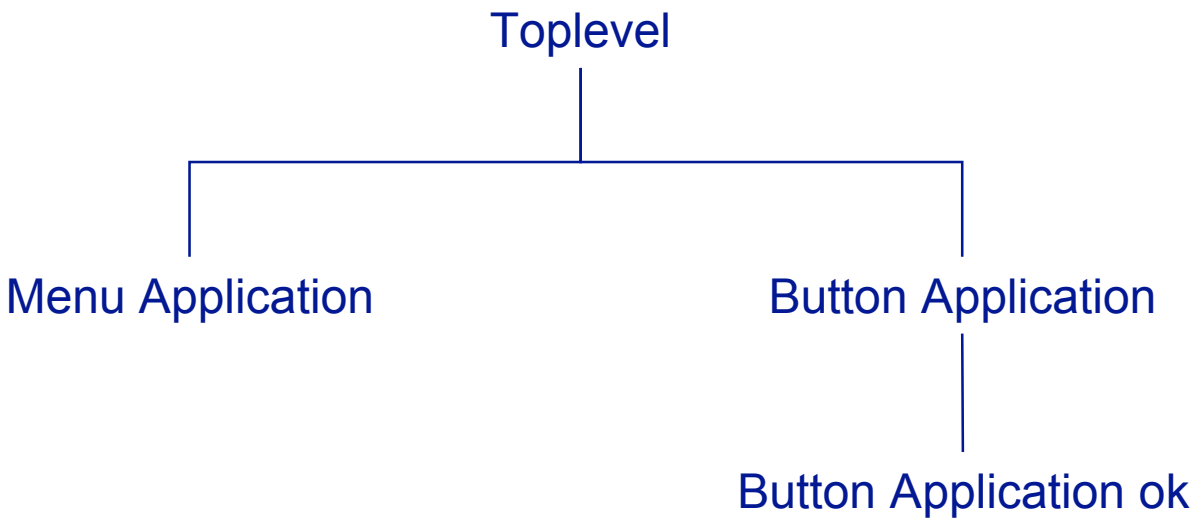


Tk

- Elegante, schnelle Erstellung von graphischen Benutzeroberflächen
- Verwendung von Widgets (allgemeiner Begriff für ein graphisches Objekt)
  - Buttons
  - Sliders
  - Menu-Boxen
  - etc.
- interaktive Testmöglichkeiten durch eine spezielle Unix-Shell (wish)
- Erweiterbarkeit um selbst definierte Prozeduren (wahlweise in Tcl oder C)

Fensterverwaltung in Tk

Hierarchische Fensterverwaltung



Widgets in Tk

- Widgets: Sammelbegriff für graphische Objekte
- 15 Widget-Klassen

Beispiele:

- Buttons
- Menus (pop-up, pull-down)
- Frames
- Labels (Darstellung von Text)
- Messages (Darstellung von Text)
- Listboxes (Darstellung von Text)
- Scrollbars
- Entry-Widgets

Der Geometrie-Manager

- Darstellung der Widgets auf dem Bildschirm
- Gestaltung der Widgets (Größe, Position etc.)

  Unterteilt in:

- **Placer**: Gestaltungshilfe zur Anordnung der Fenster an festen Koordinaten in vorgegebener Größe
- **Packer**: Mächtiger als Placer
  - Anordnung von Gruppen untergeordneter Fenster eines Levels
  - automatische Anpassung von Form und Größe
  - Verwaltung der Fensterhierarchie in einer “packing list“
  - Realisation der Fenster mittels Durchlaufen der “packing list“
  - Verwaltung der geometrischen Aspekte eines Fensters (Breite, Höhe, Position im Vorgängerfenster)

Der Window-Manager

- Jede Anwendung in Tk kommuniziert mit einem Window-Manager durch den Befehl ”wm”
  - fvwm
  - twm
  - olwm
  - mwm
  - etc.
- interaktive Veränderung der Fenstergröße durch den Benutzer oder
- Festlegung unveränderbarer Größen

Beispiele für Window-Manager-Aufrufe

Beispiel 1: wm minsize 50 10

wm maxsize 100 30

Veränderungen durch den Benutzer möglich

Beispiel 2: wm -geometry 100 x 30

feste Fenstergröße

Alle Toplevel-Fenster von Tk werden dem Window-Manager unterstellt.

Beispiele für die Oberflächenentwicklung in Tcl/Tk (1)

Programmierung von buttons:

```
button    .top       -text    ”Top button”
pack      .top
button    .bottom    -text    ”Bottom button”
pack      .bottom
```

Anmerkung:

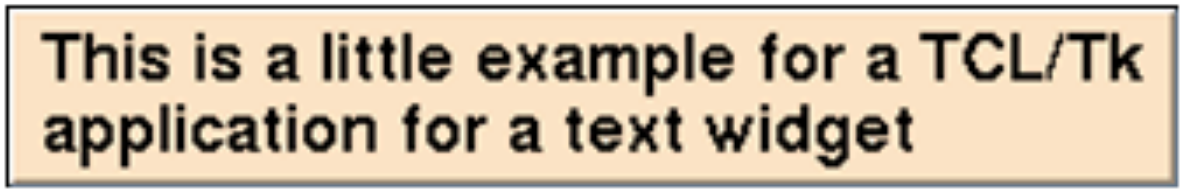
Die Anweisung ”pack” zeigt ein zuvor eingegebenes Widget auf dem Bildschirm an.

Beispiele für die Oberflächenentwicklung in Tcl/Tk (2)

Programmierung von Textfeldern

message .msg -width 8c -justify left -relief raised

-text ”This is a little example of a TCL/Tk application for a text widget”



Erstellung einer einfachen Oberfläche (1)

Programmierung eines Scrollbars

```
# ! /usr/local/bin/wish -f
frame       .mBar   -relief flat     -bd 2
pack        .mBar   -side top        -fill x
scrollbar   .s      -relief flat     -command ".t yview"
pack        .s      -side right      -fill y
```

Anmerkung:

Ein Slider dient dazu, eine größere Liste von Texteinträgen in einem Feld übersichtlich darzustellen. Mit Hilfe der Pfeilbuttons kann der Text horizontal und/oder vertikal gescrollt werden.

Erstellung einer einfachen Oberfläche (2)

Vorbereitung des Gesamtfensters:

```
text      .t     -relief raised     -bd 2      -y scrollcommand ”.s set” \
-setgrid true
pack      .t     -side left         -fill both -expand 1
wm title . ”PI I Application”
wm Iconname . ”PI I”
wm minsize . 10 10
```

Anmerkung:

Die ”wm”-Aufrufe rufen den unter X laufenden Window-Manager auf. Sie geben den Fensternamen und die Größe des Fensters an.

Einrichten eines einfachen Menüs (1)

```
menubutton   .mBar.file     -text “Menu“    -underline 0   -menu .mBar.file.m
menu         .mBar.file.m
.mBar.file.m  add  cascade  -label ”Example” -underline 0
                            -menu .mBar.file.m.apps
.mBar.file.m  add command -label ”Quit”    -underline 0   -command exit
menu         .mBar.file.m.apps              -command exampleApplication
                                            -text “PRG“
pack         .mBar.file     -side left
proc exampleApplication { }{ }
```

Einrichten eines einfachen Menüs (2)

Anmerkung

Durch die ”command”-Anweisung werden den Buttons bzw. den Menüeinträgen Aktionen zugewiesen. Diese können vom System vorgegeben sein, wie die ”exit”-Anweisung oder vom Anwender selbst definiert werden. Dann ist ein proc-Block einzugeben.

Erweiterung der Oberfläche um einen Button

```
button    .oK       -text OK -command {
                    set label OK
                    destroy   .oK
}
pack      .oK       -side right
```

Anmerkung:

Durch die Option "-side" wird die Position eines Widgets spezifiziert. Möglich sind ”left”, ”right”, ”top” und ”bottom”. Die Positionierung wird dann vom Tk - Interpreter automatisch durchgeführt.

Zusammenfassung

- Bei der ereignisgesteuerten Programmierung verwendet der Programmierer einen Satz von vorgegebenen Routinen zur Gestaltung der Benutzeroberfläche.
- Die nach dem Abschicken eines Bildschirms erforderlichen internen Operationen werden wie üblich in einer höheren Programmiersprache eingefügt.
- Moderne Toolkits bieten eine sehr mächtige Unterstützung für die ereignisgesteuerte Programmierung.