

Modul: Programmierung B-PRG Grundlagen der Programmierung 1


V13 Entwicklung von User Interfaces (1)

Prof. Dr. Detlef Krömker
Professur für Graphische Datenverarbeitung
Institut für Informatik
Fachbereich Informatik und Mathematik (12)

Wo stehen wir ... der rote Faden!

Woche	Montag 12 c.t.-14 Uhr – Hörsaal VI	Freitag 9.30-11 Uhr – Hörsaal V
16.10.	V 00 Begrüßung und Einführung (heute, diese Vorlesung)	V 01 Computer – Algorithmus – Programm
23.10.	V02 Programmieren – Erste Schritte	V 03 Kontrollstrukturen 1 (Verzweigungen + Schleifen)
30.10.	V 04 Elementare Datentypen + Operatoren 1 (int, bool, none)	V 05 Elementare Datentypen + Operatoren 2 (String)
01.11.	EPR 3: Erste größere Programmierarbeit im 2er-Team	
06.11.	V 06 Kontrollstrukturen 2 (Funktionen)	V 07 Elementare Datentypen + Operatoren 3 (Float)
13.11.	V 08 Allererste Schritte im Software-Engineering (Module)	V 09 Aggregierte Datentypen in Python (Builtins)
15.11.	EPR 4: Zweite größere Programmierarbeit im 2er-Team (Fahrstuhl)	
20.11.	V 10 Software-Tests (aus PS2)	V 11 Iteration und Rekursion
22.11.	PS2 Erste Aufgabe: ... Testen	

Die inhaltliche Grobstruktur von PRG 1 und EPR Vorlesungen Teil 2



Woche	Montag 12 c.t.-14 Uhr – Hörsaal VI	Freitag 9.30-11 Uhr – Hörsaal V
27.11.	V 12 OO-Programmierung – Erste Schritte	V 13 Uis systematisch entwickeln
29.11.	EPR 5: Dritte größere Programmierarbeit im 2er-Team (GUI)	
04.12.	V 14 GUIs programmieren 1	V 15 GUIs programmieren 2 + Iterativ vs. Rekursiv vs. Inkrementell
11. 12	V 16 OO-Entwurf und –Design 1	V 17 OO-Entwurf und –Design 2
13.12.	EPR 6: Vierte größere Programmierarbeit im 2er-Team (Partner*innewechsel!)	
18.12.	V 18 Beispiele zu Objekten und (selbstprogrammierten) Datentypen	schon Weihnachspause ;-)
20.12.	PS2 Zweite Aufgabe:	

Sachen, die Sie wissen sollten, an die Sie denken sollen: Spannende Zeiten!

Schon gemerkt: **Wir feiern diese Woche Bergfest!**

Wichtige bevorstehende Ereignisse:

Nächste und übernächste Woche: Evaluation der Tutorien.
(Feedback ist ganz wichtig)

Ab EPR 05: Partnerwechsel beim Pair Programming – spätestens in EPR 06 muss der Wechsel stattfinden!

Bedeutung und Kosten von "guten" User Interfaces (Benutzungsschnittstellen)

- Marktvorteile für Unternehmen
 - Beispiele: Google, Amazon, Microsoft
 - Benutzungsschnittstelle oft entscheidend für den Kauf eines Systems
 - Benutzungsschnittstelle hat zentralen Einfluss auf die Zufriedenheit der Nutzer
 - Eine gute Benutzungsschnittstelle kann Arbeitsprozesse verkürzen und damit zu Einsparungen beitragen
 - Mittlerer Anteil des Sourcecodes für die Benutzungsschnittstelle am Gesamtsystem: **48%**
- ➔ Die **Hälfte des Aufwands** zur Entwicklung eines Softwaresystems fällt typischerweise **für die Benutzungsschnittstelle** an!

Inhalt

- Die historische Entwicklung der User Interfaces
- Eigenschaften von Command Line Interfaces: CLIs
- Entwicklung von Command Line Interfaces
- Eigenschaften von GUIs
- Entwicklung von Graphischen User Interfaces

Übersicht zur HCI (Human Computer Interaction) Entwicklung -- Paradigmenwechsel

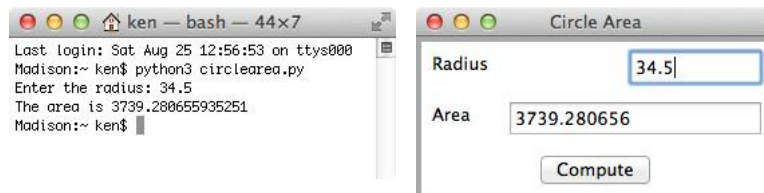
60er / 70er / 80er	80er/90er	ab 2000
Operator bedient Maschine Batch Command Line Interface Interface	User benutzt Tool Graphical User Interface	Manager delegiert an Assistenten Attentive User Interface
TUI -- CLI		
GUI -- WIMP		
ZUI (?) ... TUI (?)		

7

Programmieren 1 / EPR

Prof. Dr. Detlef Krönker

Vergleich CLIs (TextUIs) und GraphicalUIs Charakteristika



1. Lese die Eingabe `input()`
2. Verarbeitung
3. Ausgabe `print()`,
ggf. mit `.format()`
4. Gehe zu Schritt 1.

EVA-Prinzip

1. Layout und pop-up des Fensters.
2. Warte auf ein 'User Event'.
3. Bearbeite das 'User Event'.
4. Gehe zu Schritt 2.

Machen wir mit dem Modul `tkinter`
Event-based Programming

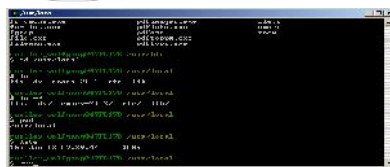
8

Programmieren 1 / EPR

Prof. Dr. Detlef Krönker

TUI - TextUIs speziell: CLI: Command Language Interface

- Befehle werden als **Text** mit der **Tastatur** eingegeben
- Systemantwort als **Text** auf dem **Bildschirm**
- Historisch die erste (echte, ursprüngliche) Form der Interaktion Mensch-Maschine (vorher überhaupt keine eigentliche Interaktion, z.B. Lochkarten, Drucker)
- Beispiele
 - UNIX Shell, MS-DOS Shell
 - Datenbankabfragen, z.B. SQL
 - Programmiersprachen, Skriptsprachen
- In Python: `print()` und `input()`



9

Programmieren 1 / EPR

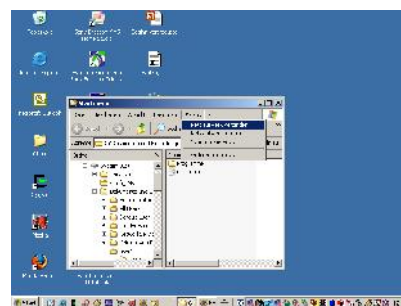
Prof. Dr. Detlef Krönker

GUIs – Das WIMP-Paradigma

Immer noch **grundlegendes Paradigma** zur Gestaltung der Mensch-Maschine-Interaktion (MMI)

- **WIMP (Windows, Icons, Menues, Pointer)**
- **Werkzeugmetapher (Tool)**
- **Direct Manipulation**

zumindest bei Desktop-Rechnern und Laptops



10

Programmieren 1 / EPR

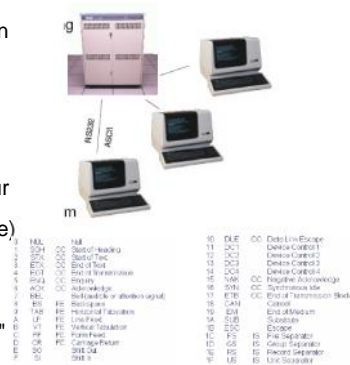
Prof. Dr. Detlef Krönker

Attentive User Interface ... Future?

- ▶ Erste größere Änderungen ab 2000 (2005) mit der weiten Verbreitung der Touch-Interfaces (Smartphones, Tablets) hin zu ZUIs (Zoomable User Interface, skalierbare Benutzungsoberfläche).
- ▶ Andere Technologien: BCI (Brain Computer Interface), Natürliche Benutzungsschnittstellen (Natural User Interface, NUI), Sprachbasierte Benutzungsschnittstellen (Voice User Interfaces, VUI), Perceptual User Interface PUI, Gegenständliche Benutzerschnittstellen (Tangible User Interface; auch TUI)
- ▶ ... aber alle Ansätze sind in der Forschungsphase, ... keine stabile Situation ...

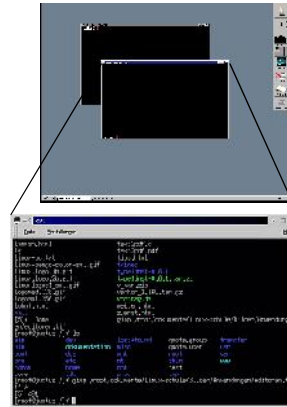
Eigenschaften von Command Line Interfaces CLI: Hauptform beim Timesharing Betrieb (... 1985)

- ▶ **ASCII und ANSI Steuerzeichen**
ASCII (American Standard Code for Information Interchange)
- ▶ Kennzeichen: „beliebig“ lange Listen von (Eingaben und Systemantworten -- EVA)
- ▶ **Wenig Formatierung möglich:**
 - ▶ die ersten 32 Zeichen als Steuerzeichen für Kommunikation mit Terminal reserviert
 - ▶ ANSI (American National Standard Institute) Escape Sequenzen:
werden durch ESC (1B) eingeleitet, z.B. ESC[1A = scroll up
 - ▶ zusätzliche Steuerung, Farbe, etc.
 - ▶ weitgehend identisch mit dem „legendären“ DEC VT100“ Steuerzeichen



Eingabeaufforderung (command *prompt*) ab ca. 1985

- ab 1985 Einführung der fensterorientierten Oberflächen
- ermöglichen es, **mehrere virtuelle Terminals** zu öffnen.
- werden als **Eingabeaufforderung**, **command prompt** oder **shell** bezeichnet
- jede Eingabeaufforderung ist eigenständiges virtuelles Terminal
- kann gut zur Netzwerk-Verbindung zwischen Rechnern verwendet werden
 - heute meist über TCP/IP
 - Telnet: Terminal Emulator über TCP/IP

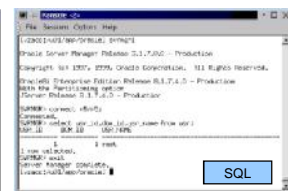
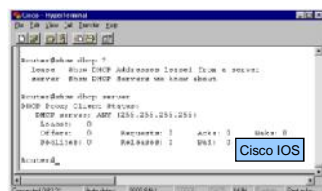
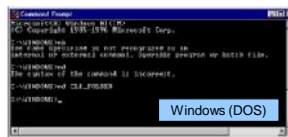


13

Programmieren 1 / EPR

Prof. Dr. Detlef Krönker

Typische Nutzungen



14

Programmieren 1 / EPR

Prof. Dr. Detlef Krönker

Vorteil der CLI Schnittstellen - **Mächtigkeit**

- Kommandosprachen haben (potentiell) einen großen Umfang von Befehlen
- der Wortschatz kann auf die Anforderung zugeschnitten werden, Beispiele dir, rm,...
- meist einfache Syntax
 - verbraucht wenig Ressourcen z.B. Befehl "-" Optionen Argument
 - ggf. einfach zu memorieren (trotzdem noch schwer genug)
- Erweiterbar: häufig benötigte Befehlssequenzen können abgespeichert und wieder abgerufen werden; Makros z.B. BAT-Dateien

Weitere Vorteile des CLI

- **Flexibilität**
 - Benutzer kann „jederzeit“ einen beliebigen Befehl eingeben
 - Keine Reihenfolge der Befehle vorgegeben
- **Schnell und effizient – "effizienteste" Benutzungsschnittstelle**
Voraussetzung dafür:
 - geübter Benutzer mit guten Schreibmaschinenkenntnissen und
 - CLI ist gut entworfen
- **Schonender Umgang mit Bildschirmplatz**
 - (meist) nur eine Zeile für Eingabe (unbedingt) benötigt
 - Rest für Ausgabe
- **Geringer Bandbreitenbedarf, "nur Text"**

Nachteile der CLI-Schnittstellen (1)

- ▶ **Schwierige Erlernbarkeit**
Befehle müssen auswendig gelernt werden
 - ▶ ebenfalls Parameter und Optionen der Befehle
 - ▶ Effizienz durch kurze (aber meist kryptische) Befehle
- ▶ **Schnelles Vergessen garantiert**
 - ▶ Bei Nichtgebrauch gehen Befehlssyntax schnell wieder vergessen
→ **nicht geeignet für gelegentliche Benutzung**
- ▶ **Schreibmaschinen-Fertigkeit**
nur wenn diese hoch, dann effizienteste Benutzungsschnittstelle
- ▶ **Fehleranfällig:** Tippfehler sind schnell geschehen

Nachteile der CLI-Schnittstellen (2)

- ▶ Kommandos müssen z.T. in bestimmten **Reihenfolgen** eingegeben werden.
- ▶ **Schwierig, Fehler zu korrigieren** oder andere Lösungsstrategien umzusetzen.
- ▶ Ggf. müssen **längere Eingabesequenzen wiederholt werden**, um eine Kleinigkeit zu ändern.
- ▶ **Sind auf Text-Ein-/Ausgabe beschränkt.**
- ▶ In der "realen Welt" werden **CLIs von GUIs komplett dominiert ...**

Wie entwirft man eine Kommandosprachen-Struktur?

Beschreibung der geforderten Funktionalität durch Analyse des Anwendungsgebietes

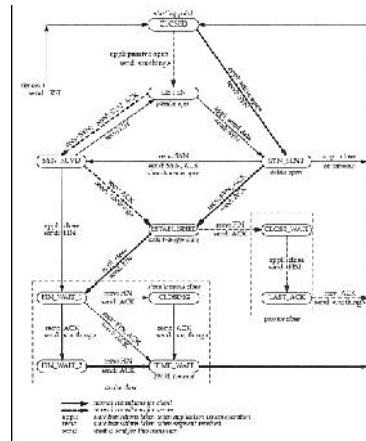
1. Auflisten aller User Tasks

- Bei großen Anzahlen eine Spalte mit der erwarteten Häufigkeit der Nutzung anlegen: Häufigste Aufgaben sollten am einfachsten zu merken und auszuführen sein.

2. Auflistung aller Interface-Objekte

Unterstützende Technik

- Transitionsdiagramme (rechts)
(Zustand (Knoten) – Übergang (Pfeil))



Weitere wichtige Schritte

- Abstrahieren der Liste in Interface Actions und Objekte
- Entwicklung und Beschreibung der Low-level Interface Syntax
- Spezielle Evaluierung destruktiver Aktionen (z.B. Löschen eines Objektes)
Bereitstellung von Undo-Mechanismen (teilweise sehr aufwendig)
- Identifizieren von Fehlersituationen und Formulierung adäquater Fehlermeldungen
- Einführung zusätzlicher Shortcuts für Experten, z.B. über Makros oder Mechanismen zur Anpassung von Systemparametern

Strukturierung von Kommandos

(Wenn die Nutzerbasis deutschsprachig ist, dürfen Kommandos natürlich in Deutsch sein!)

1. Einfache Kommandos (Befehle ohne Argumente)

- Beispiele: look, go, move, ... end

2. Befehle mit Argumenten und Optionen – drei Möglichkeiten

- Befehl mit Argumenten
 - `copy source dest`
- Befehle mit Keyword Labels
 - `copy From=source To=dest`
- Befehle mit Optionen
 - `PRINT/3,HQ FILEA`
 - `PRINT FILEA -3, HQ`

3. Hierarchische Kommandostrukturen (führt hier zu weit → HCI)

21

Programmieren 1 / EPR

Prof. Dr. Detlef Krönker

Empfehlungen

- **Kongruente (Begriffs-)Paare und hierarchische Formen sind vorteilhaft.** (Carroll '82)

(Kongruenz: Benutzung **bedeutungsvoller Begriffspaare**, z.B. Open/Close – links/rechts – oben unten)
- **Konsistente Ordnung von Argumenten ist vorteilhaft.** (Barnard '81)
- **Spezifische Begriffe (sonst selten genutzte) sind vorteilhaft gegenüber allgemeinen, sie sollten zusätzlich gut unterscheidbar sein.** (Barnard '81 und Black und Moran '82)
- **Kurze Kommandos** (z.B. durch Abkürzungen) sind gut, haben aber Nachteile bzgl. Retention. Abkürzung sollten Kommandozeileingaben um mehr als 2 Zeichen verkürzen

22

Programmieren 1 / EPR

Prof. Dr. Detlef Krönker

Details: Vergleiche Kongruenz und Hierarchien (nach Carroll 1982): Beispiel: Steuerung eines Roboterarms

CONGRUENT		NONCONGRUENT	
Hierarchical	Non-hierarchical	Hierarchical	Non-hierarchical
MOVE ROBOT FORWARD	ADVANCE	MOVE ROBOT FORWARD	GO
MOVE ROBOT BACKWARD	RETREAT	CHANGE ROBOT BACKWARD	BACK
MOVE ROBOT RIGHT	RIGHT	CHANGE ROBOT RIGHT	TURN
MOVE ROBOT LEFT	LEFT	MOVE ROBOT LEFT	LEFT
MOVE ROBOT UP	STRAIGHTEN	CHANGE ROBOT UP	UP
MOVE ROBOT DOWN	BEND	MOVE ROBOT DOWN	BEND
MOVE ARM FORWARD	PUSH	CHANGE ARM FORWARD	POKE
MOVE ARM BACKWARD	PULL	MOVE ARM BACKWARD	PULL
MOVE ARM RIGHT	SWING OUT	CHANGE ARM RIGHT	PIVOT
MOVE ARM LEFT	SWING IN	MOVE ARM LEFT	SWEEP
MOVE ARM UP	RAISE	MOVE ARM UP	REACH
MOVE ARM DOWN	LOWER	CHANGE ARM DOWN	DOWN
CHANGE ARM OPEN	RELEASE	CHANGE ARM OPEN	UNHOOK
CHANGE ARM CLOSE	TAKE	MOVE ARM CLOSE	GRAB
CHANGE ARM RIGHT	SCREW	MOVE ARM RIGHT	SCREW
CHANGE ARM LEFT	UNSCREW	CHANGE ARM LEFT	TWIST

23

Programmieren 1 / EPR

Prof. Dr. Detlef Krönker

Details: Vergleiche Kongruenz und Hierarchien (nach Carroll 1982)

CONGRUENT		NONCONGRUENT	
Hierarchical	Non-hierarchical	Hierarchical	Non-hierarchical
MOVE ROBOT FORWARD	ADVANCE	MOVE ROBOT FORWARD	GO
MOVE ROBOT BACKWARD	RETREAT	CHANGE ROBOT BACKWARD	BACK
MOVE ROBOT RIGHT	RIGHT	CHANGE ROBOT RIGHT	TURN
MOVE ROBOT LEFT	LEFT	MOVE ROBOT LEFT	LEFT
MOVE ROBOT UP	STRAIGHTEN	CHANGE ROBOT UP	UP
MOVE ROBOT DOWN	BEND	MOVE ROBOT DOWN	BEND
MOVE ARM FORWARD	PUSH	CHANGE ARM FORWARD	POKE
MOVE ARM BACKWARD	PULL	MOVE ARM BACKWARD	PULL
MOVE ARM RIGHT	SWING OUT	CHANGE ARM RIGHT	PIVOT
MOVE ARM LEFT	SWING IN	MOVE ARM LEFT	SWEEP
MOVE ARM UP	RAISE	MOVE ARM UP	REACH
MOVE ARM DOWN	LOWER	CHANGE ARM DOWN	DOWN
CHANGE ARM OPEN	RELEASE	CHANGE ARM OPEN	UNHOOK
CHANGE ARM CLOSE	TAKE	MOVE ARM CLOSE	GRAB
CHANGE ARM RIGHT	SCREW	MOVE ARM RIGHT	SCREW
CHANGE ARM LEFT	UNSCREW	CHANGE ARM LEFT	TWIST
Subjective Ratings (1 = Best, 5 = Worst)			
	1.88	1.63	1.81
Test Scores	14.88	14.63	7.25
Errors	0.50	2.13	4.25
Omissions	2.00	2.50	4.15

?

24

Programmieren 1 / EPR

Prof. Dr. Detlef Krönker

Benamungen und Abkürzungen

- Beispiel: UNIX (**leider kein gutes Beispiel!**)
 - **mkdir** (make directory)
 - **ls** (list directory)
 - **cd** (change directory)
 - **rm** (remove file)
 - **pwd** (print working directory)
- Problem mit diesen Abkürzungen
 - Keine feste Regel, die festlegt, wie man von einem Befehl auf die Abkürzung kommt
 - Fehlen von Standards
 - darum ...

Strategien (Regeln) für Abkürzungen

- **Einfaches Abschneiden**
 - Verwendung des 1. + 2. (+ 3. ...) Zeichens des Kommandos
 - Problem: Kommandos müssen unterscheidbar sein
- **Vowel Drop** – Weglassen von Vokalen
- **Erster und letzter Buchstabe**
 - Verwendung des ersten und letzten Buchstaben des Kommandos, da diese besonders deutlich wahrgenommen werden.
- **Erster Buchstabe eines jeden Worts einer Phrase**
 - Beispiel: **lf** für List Full
 - Typischerweise verwendbar mit hierarchischem Konzept
- **Standard-Abkürzungen verwenden**
- **Phonics**
 - Beispiel: **XQT** für execute
 - Problem: eher im amerikanischen Sprachraum üblich

Richtlinien für Abkürzungen

(nach Ehrenreich und Porcu, 1982)

- Verwende **maximal 2 verschiedene Regeln zur Abkürzung**
 - Einfache Hauptregel (siehe letzte Folie)
 - Abweichende Zweitregel im Fall von Konflikten (bei Anwendung der Hauptregel)
 - Anwendungen der Zweitregel sollte durch Marker (z.B. *) dargestellt werden
 - Zweitregel sollte so wenig wie möglich verwendet werden
- **Anwender müssen die Regeln kennen**
- **Einheitliche Länge der Kommandos ist zu bevorzugen**
- Abkürzungen **sollten immer angeboten** werden. Ausnahme: zu viele ähnliche Aktionen

Aber: Vom Computer generierte Messages sollten **keine Abkürzungen** verwenden, wenn es nicht ein Platzproblem bei der Darstellung gibt

Zusammenfassung Command Line Interfaces

- Ganz schön viel Entwurfsarbeit, bevor man das erste input() – print() Paar schreiben kann.
- Diese Arbeit müssen Sie auch dokumentieren →
 - Source Code
 - Benutzerhandbuch oder Benutzungshandbuch

Eigenschaften von GUIs

- WIMP
- Widgets
- Entwicklung mit entsprechenden Bibliotheken
- Direct Manipulation

Zentrale Eigenschaften der WIMP-Interfaces (1)

- **Windows** (Fenster):
Ein abgegrenzter Bereich auf dem Bildschirm. Überlappend oder nebeneinander. In der Computergrafik meist (etwas genauer) **Viewport** genannt. Programme nutzen ein Fenster um ihre Ausgaben als Text, Grafik oder Steuerelemente: Buttons, Slider, etc. darzustellen.
- **Icons** (Ikonen, Symbole, kompakte Bilder, Graphisches Zeichen, Grapheme): Zeichen repräsentieren entweder eine **Funktion** oder ein **Objekt**, z.B. ein Schere um etwas auszuschneiden oder ein Mülleimer um etwas zu löschen.

Sie sollen **schneller 'aufnehmbar'/erfassbar als Text sein**. Beispiele sind u.a. Warnsymbole, Werkzeugicons, Navigationspfeile.

Zentrale Eigenschaften der WIMP-Interfaces (2)

► **Menue** (Menü):

Meist ausklappbare **Auswahlliste** von (aktuell verfügbaren) Kommandos, oft in Form von Text, zwischen denen der Nutzer auswählen kann. Häufig Kontextabhängig, d.h. zeigt nur die Kommandos, die aktuell möglich sind. Pull-Down-Menüs klappen vom Fensterrand nur nach unten (down) auf. Das Pull-Up-Menüs (am unteren Fensterrand) und Pop-Up-Menüs (an beliebigen Stellen).

Zentrale Eigenschaften der WIMP-Interfaces (2)

► **Pointer** (Zeigegerät, z.B. Mouse (Maus)).

Ein Symbol (**Cursor**), das die Position der Maus auf dem Bildschirm repräsentiert und sich bewegt, wenn sich auch die Maus bewegt und das man durch (Mouse-)Click bedienen kann, z.B. Fensterflächen verschieben, Icons markieren, Menüpunkt auswählen, Buttons drücken, Schieberegler verschieben.

Je nach Anwendungskontext repräsentiert der Mauszeiger mit einem geeigneten Symbol (Pfeil, Einfügemarke, Greifhand, Zeigefingerhand, Warteschleife usw.) die vom Anwender momentan gewählte Zeige- und Bearbeitungsposition.

Geschichte des WIMP-Interfaces

- Zur Geschichte des WIMP-Interfaces sehen Sie z.B. <http://mark13.org/utf8/wimp>. (Nett gemacht!)
- Ein **erster Meilenstein** der WIMP-Geschichte ist das am Stanford Research Institute unter der Leitung von Douglas Engelbart entwickelte *NLS* (oN-Line System), welches **1968 öffentlich präsentiert** wurde. Einige NLS-Entwickler wechselten Anfang der 1970er Jahre ans Xerox PARC, um das WIMP-gestützte Smalltalk (Programmiersprache) zu realisieren, welches wiederum für Apples Mac OS Pate stand.
- Die Computersysteme veränderten sich vor allem in den 1980er Jahren dramatisch: PCs, Workstations, etc.

Entwicklung von GUIs

Hierzu benutzt man sogenannte "**GUI-Toolkits**" oder **GUI-Bibliotheken** oder **widget toolkit**,

Man benötigt zwei Anbindungen:

- Zum Betriebssystem (Unix, Windows, Mac-OS) und zur
- Programmiersprache

Es gibt hunderte solcher Bibliotheken oder Frameworks. Siehe https://de.wikipedia.org/wiki/Liste_von_GUI-Bibliotheken. Manche erlauben ein sogenanntes Visual Programming.

Wir benutzen

- **Tk** als Python Bibliothek **tkinter**.

Widget (1)

Bedeutungen im modernen Englisch:

1. *Kleines mechanisches Gerät oder Geräteteil, dessen Namen unbekannt ist.*
2. *Ein massenproduziertes Objekt einer Fabrik.*
3. **Softwareentwicklung: Komponente einer grafischen Benutzungsoberfläche.**
4. *Vorrichtung in Bierdosen, wodurch das Bier beim Ausschenken schäumt.*



Widget (2)

Herkunft:

- Wird seit den 1930er Jahren möglicherweise als Abwandlung des Wortes *gadget* (= *Vorrichtung, Zubehör-(gerät)*) verwendet.
<https://de.wiktionary.org/wiki/Widget>
- In der Informatik wahrscheinlich ursprünglich aus dem Projekt Athena (MIT), und bezeichneten **ein mit einem Fenster assoziiertes Objekt**, daraus resultiert das Silbenkurzwort aus **Wi(ndow)** und **(Ga)dget**

Widget

Ein **Steuerelement** (*graphical control element* oder auch **widget**) ist ein Interaktionselement in einer grafischen Benutzeroberfläche (GUI), beispielsweise eine Schaltfläche oder eine Bildlaufleiste.

Jedes widget hat verschiedene Eigenschaften (Attribute), wie Größe, Position auf dem Bildschirm, usw.

Gebräuchliche widgets (1)

(Liste aus <https://de.wikipedia.org/wiki/Steuerelement> - teilweise überarbeitet)

- ▶ **Formular:** Objekt mit diversen meist Text-Eingabefeldern und Auswahlelementen
- ▶ **Symbol, Icon:** Ein grafisches Symbol, meist als Repräsentant einer Datei oder eines Fensters oder des dazugehörigen Programms.
- ▶ **Menü:** Meist nur einmal in einem Programm oder Fenster vorkommende Auswahlliste mit Drop-Down-Listen für die grundlegenden Funktionen eines Programms. Kann auch außerhalb des Programmfensters platziert sein, z. B. am oberen Bildschirmrand bei MacOS und einigen Linux-Oberflächen
- ▶ **Symbolleiste, Werkzeugleiste:** Ein Bereich mit mehreren, meist bildhaften Schaltflächen (Icons) für Programmfunktionen. Kann ein Menü ergänzen oder auch ganz ersetzen.
- ▶ **Bezeichnungsfeld, Label:** Für einfachen Text.

Gebräuchliche widgets (2)

- **Textfeld:** Ein- oder mehrzeiliges Feld zur Ein- oder Ausgabe von Zahlen oder Text.
- **Schaltfläche, Button:** Eine Fläche zum Klicken um damit eine Aktion auszulösen.
- **Auswahlkästchen, Checkbox:** Kann üblicherweise gesetzt oder nicht gesetzt sein, manchmal auch ausgegraut werden.
- **Optionsfeld, Radiobutton:** Zur Auswahl einer Möglichkeit aus einer Gruppe von zwei oder mehr.
- **Listenfeld, Listbox:** Zur Auswahl von Texten und anderen Objekten. Mehrzeilig oft mit vertikaler Bildlaufleiste.
- **Kombinationsfeld, Combobox:** Kombination aus Textfeld und Listenfeld, wobei das Listenfeld unterhalb des Textfeldes angezeigt wird oder ausgeklappt werden kann.

Gebräuchliche widgets (3)

- **Dropdown-Listenfeld, Dropdown-Liste:** Kombination aus schreibgeschütztem Textfeld und ausklappbarer Auswahlliste, so dass das Textfeld nur einen der Einträge der Liste annehmen kann.
- **Baum, Baumansicht:** Eine mit aufklappbaren Knoten hierarchisch gegliederte Auswahlliste. Bekanntestes Anwendungsbeispiel ist die Anzeige der Ordnerstruktur im Windows-Explorer.
- **Flexgrid, tabellarische Datenansicht:** Zur tabellarischen Anzeige von Daten. Die Daten werden in Zellen bearbeitet und können sortiert bzw. gruppiert werden.
- **Bildlaufleiste, Scrollbar:** Wird benutzt, um den Fensterinhalt vertikal oder horizontal zu verschieben oder Werte stufenlos einzustellen
- **Schieberegler, Slider:** Wird zur grafischen Auswahl von Werten benutzt. Optisch ähnlich der Bildlaufleiste.

Gebräuchliche widgets (4)

- ▶ **Fortschrittsbalken:** Zeigt den Fortschritt einer Operation an. Seit Windows XP gibt es auch eine Version mit durchlaufendem Balken, die verwendet werden kann, wenn der aktuelle Fortschritt (noch) nicht bekannt ist.
- ▶ **Statusleiste:** Ein Bereich am unteren Rand eines Fensters, in dem der Programmstatus oder ein Hilfetext angezeigt werden kann.
- ▶ **Registerkarte, Tab:** Für mehrseitige Dialogfenster oder mehrere Dokumente pro Fenster. Deren überstehende Bereiche zur Auswahl der einzelnen Karten werden auch Tabs genannt.
- ▶ **Gruppenfeld:** Ein Rahmen mit Überschrift zur Gruppierung inhaltlich zusammengehörender Steuerelemente.

Gebräuchliche widgets (5)

- ▶ **Multifunktionsleiste, Ribbon:** Eine Leiste, die die Vereinigung der Elemente Menü und Symbolleiste, sowie teilweise auch der Titelleiste darstellt.
- ▶ **Kalendersteuerelement:** Ein Kalenderwerkzeug zur Auswahl von Datumsangaben, gelegentlich ergänzt um eine Uhrzeit zur Auswahl eines Zeitpunkts.
- ▶ Ja, das sind sehr viele ... aber TK hat gar nicht alle Elemente und ein Prinzip gilt, gerade im HCI:

"Weniger ist mehr"

Fragen und (hoffentlich) Antworten

Zwischen-Zusammenfassung

Ooops - wir haben sehr viel kennengelernt ... jetzt muss nur noch das Programmieren folgen.

Das machen wir am kommenden Montag in

V14 Entwicklung von GUIs

Zusammen mit einigen Design-Überlegungen.

Und ... Danke für Ihre Aufmerksamkeit