Ereignisse und Fenstersysteme

Die ereignisgesteuerte Arbeitsweise ist meist an Fenstersysteme gebunden – mit wenigen Gegenbeispielen

Im folgenden:

Fenstersysteme mit Ereignisbehandlung

Ereignisse und Fenstersysteme

Bindung eines Fenstersystems an ein Betriebssystem (BS)

- prägt einheitliches „Look & Feel“ der damit laufenden Sw
- öffnet die BS-Nutzungsgemeinde für neue Sw-Produkte
- ermöglicht gemeinsame Publizität, insb. f. kleine Anbieter

Unabhängigkeit der Fenstersysteme von Betriebssystemen

- erleichtert Portierung auf mehrere BSe u. BS-Versionen
- lockert Abhängigkeit von dominierenden BS-Anbietern
- erlaubt spez. Optimierung (Laufzeit, Aussehen, Speicher)

GLUT

OpenGL ist unabhängig von Fenstersystemen; dies erfordert für Grafik-Sw die Adoption

- entweder eines ‚nativen' Fenstersystems oder
- einer Programmierschnittstelle (Application Programming Interface, API) zur Anbindung an Fenstersysteme.

Eine solche Programmierschnittstelle ist GLUT (*OpenGL Utility Toolkit* - Aussprache „wie *glutttony*").

Aktuelle Version: 3.7 (3.7.6)

Literatur:

- M.J.Kilgard: „OpenGL Programming for the X Window System", Addison-Wesley 1996
- E.Angel: „Interactive Computer Graphics: A Top-Down Approach Using OpenGL", Addison-Wesley 2006

GLUT

HTML-Online-Dokumentation zu GLUT:

- Links zu Quellen und Dokumentation:
  www.opengl.org/resources/libraries/glut

- Benutzungshandbuch:
  www.opengl.org/documentation/specs/glut/spec3/spec3.html
  www.opengl.org/resources/libraries/glut/spec3/spec3.html
  www.opengl.org/resources/libraries/glut/glut-3.spec.pdf – bzw.:
  http://homepages.thm.de/christ/>Computergrafik>aktuell
- Einführung in GLUT:
  http://mindfuck.de-brauwer.be/articles/glut/
- Quellen u. Dokumentation für MS-Windows zu GLUT:
  www.xmission.com/~nate/glut.html

GLUT

Funktionalität durch GLUT:

- Einrichtung und Handhabung mehrerer Grafik-Fenster
- Ereignisverarbeitung durch Callbacks *(event processing)*
- Unterstützung vielfältiger Eingabegeräte
- Routinen zur Einhaltung von Zeitvorgaben *(timer)*
- Nutzung “ereignisloser” *("idle”)* Zeitintervalle
- Erzeugung von Pop-Up-Menü-Kaskaden
- Unterprogramme zur Generierung geometrischer Körper
- Mehrere Bitmap-/ Vektor-Schriftarten *(raster/ stroke fonts)*
- Diverse Funktionen zur Fenster- und Overlayverwaltung

GLUT

- bietet ein (fast) komplettes API für d. native Fenstersystem
  - ➥ Sw-Entwicklung ohne Einarbeitung in Fenstersystem
- überläßt Fenster-Einrichtung dem nativen Fenstersystem
  - ➥ Erhaltung des plattformeigenen Look & Feel
- wird (wurde) regelmäßig aktualisiert
  - ➥ Befolgung von Trends, Beseitigung von Bugs
- ist kostenlos erhältlich inkl. Source-Code
  - ➥ Verwendbarkeit in Ausbildung
  - ➥ Anpassung an individuelle Sw-Entwicklung möglich

Design-Philosophie:

- Fenster-Einrichtung u. -Verwaltung mit wenigen Aufrufen (für Ein-Fenster-Applikationen genügt Callback +Start) :

  ➥ Schnelle Erlernbarkeit, Augenmerk auf Fenster-Inhalt

- Aufrufe mit möglichst knappen Parameterlisten, Zeiger nur für (Eingabe von) Zeichenketten, keine Zeiger-Rückgabe durch GLUT:

  ➥ Leichte Handhabung, Fehlerrobustheit

- Keine Verwendung v. Daten des nativen Fenstersystems (Fenster-Handler, Schriftarten):

  ➥ Unabhängigkeit vom nativen Fenstersystem

"Handler": Behandlungsroutine

Design-Philosophie: (Forts.)

- Übernahme der Ereignisverarbeitung, Überlassung von OpenGL-Displaylisten der Applikation:

  ➥ Einschränkung gleichzeitiger Verwendung weiterer Ereignis-verarbeitung, keine Einschränkung des OpenGL-Einsatzes

- Zustandshaftes *(stateful)* API:

  Zustand: Datensatz mit d. Beschreibung des Systems für die Bedürfnisse der Anwendung

  ➥ Einfache Applikationen durch Voreinstellungen *(default/ current window/ menu);* wiederholter Datentransfer (interessant z.B. bei unsicheren Verbindungen) wird vermieden

Echtzeit ! (*real time*)

- Mehrere, differenziert aufgerufene Callbacks

  ➥ Vermeidung unnötigen Codes je nach Art der Applikation

- Logische Organisation in 10 ‚Sub-APIs‘: ⇨

GLUT - Unterschnittstellen

1. Initialisierung:

- Initialisierung des Fenstersystems,
- Überprüfug der **`main()`**-Parameter,
- Setzen der Voreinstellungen für Fenster-Position, Größe, Darstellungsmodus (Single/ Double Buffer)
- 4 Routinen:


```
void glutInit(int *argc, char **argv);
                                    /*Parameter aus main()*/
void glutInitWindowSize(int width, int height);
                                        /*def.: (300,300)*/
void glutInitWindowPosition(int x, int y);
  /*def.: (-1,-1): dem nativen Fenstersystem ueberlassen*/
void glutInitDisplayMode(unsigned int mode);
                            /*def.: (GLUT_RGB | GLUT_SINGLE)*/
```

GLUT - Unterschnittstellen

2. Start der Ereignisverarbeitung:

- Letzte Anweisung (meist in **`main()`**) vor Überlassung der Programmsteuerung an GLUT und den dort registrierten Callbacks der Anwendung – keine Rückkehr!
- 1 Routine:

```
void glutMainLoop(void);
```

3. Fenster-Verwaltung:

- Fenster-Generierung und -Steuerung
- 18 Routinen:

```
int glutCreateWindow(char *name);/*intVar<=Fenst.einrichten*/
int glutCreateSubWindow(int win,int x,int y,int wid,int hig);
void glutSetWindow(int win);/*Setzen:win=>aktuelles Fenster*/
int glutGetWindow(void);  /*Abfragen:int<=aktuelles Fenster*/
```

GLUT - Unterschnittstellen

3. Fenster-Verwaltung (Forts.):

```
                          /*meist: "execution upon return to GLUT"*/
void glutDestroyWindow(int win);
void glutPostRedisplay(void);   /*Kennzeichnung: aktualisieren!*/
void glutSwapBuffers(void);
void glutPositionWindow(int x, int y);
void glutReshapeWindow(int width, int height);
void glutFullScreen(void);          /*may vary by window system*/
void glutPopWindow(void);         /*Aendert Fenster-Reihenfolge*/
void glutPushWindow(void);        /*     "     "          "     */
void glutShowWindow(void);             /*macht Fenster sichtbar*/
void glutHideWindow(void);           /*macht Fenster unsichtbar*/
void glutIconifyWindow(void);
void glutSetWindowTitle(char *name);
void glutSetIconTitle(char *name);
void glutSetCursor(int cursor);/*23 cursor images GLUT_CURSOR..*/
```

(Test7.exe)

GLUT - Unterschnittstellen

4. Overlay-Verwaltung:

- Einrichtung u. Nutzung von Overlay-Hw (falls vorhanden)
- 6 Routinen:

```
void glutEstablishOverlay(void);
void glutUseLayer(GLenum layer);
void glutRemoveOverlay(void);
void glutPostOverlayRedisplay(void);
void glutShowOverlay(void);
void glutHideOverlay(void);
```

GLUT - Unterschnittstellen

5. Menü-Verwaltung:

- Menü-Generierung und -Steuerung
- 11 Routinen:

```
int  glutCreateMenu(void (*func)(int value));/*Callback-Reg.*/
void glutSetMenu(int menu);
int  glutGetMenu(void);
void glutDestroyMenu(int menu);
void glutAddMenuEntry(char *name, int value);
   /*Eintrag und an glutCreateMenu-Callback uebergebener Wert*/
void glutAddSubMenu(char *name, int menu);
void glutChangeToMenuEntry(int entry, char *name, int value);
void glutChangeToSubMenu(int entry, char *name, int menu);
void glutRemoveMenuItem(int entry);
void glutAttachMenu(int button);          /*Maustaste~Menue*/
void glutDetachMenu(int button);
```

(Glutmech.exe)

6. Callback-Registrierung:

- Anmeldung von Applikationsfunktionen (Callbacks).
- Callback-Aufruf durch die Verarbeitungsschleife der GLUT- Ereignisse *(GLUT event processing loop)*.
- Drei Callback-Typen:
  - Fenster-Callbacks – zur Änderung von Größe, Form, Sichtbarkeit von Fenstern bzw. zur Anzeige des fertiggestellten Fensterinhalts
  - Menü-Callbacks – zur Menü-Darstellung
  - Globale Callbacks – für die Einbeziehung von Zeitabläufen und Menü-Nutzung.
- insg. 20 Routinen:

6. Callback-Registrierung (Forts.):

- **Fenster-Callbacks:**

```
void glutDisplayFunc(void(*func)(void)); /*CB f.aktuelles Fenster*/

void glutOverlayDisplayFunc(void (*func)(void));

void glutReshapeFunc(void (*func)(int width, int height));

void glutKeyboardFunc(void(*func)(unsigned char key,int x,int y));

void glutMouseFunc(void(*func)(int button,int state,int x, int y));

void glutMotionFunc(void(*func)(int x,int y));   /*button pressed*/

void glutPassiveMotionFunc(void(*func)(int x,int y));  /*no press*/

void glutVisibilityFunc(void (*func)(int state));
                /* state == GLUT_NOT_VISIBLE oder GLUT_VISIBLE  */

void glutEntryFunc(void (*func)(int state));   /*Maus im Fenster?*/
                      /* state == GLUT_LEFT oder GLUT_ENTERED */
```

6. Callback-Registrierung (Forts.):

- Fenster-Callbacks (Forts.):

```
void glutSpecialFunc(void (*func)(int key, int x, int y));
     /* key == Funktions-, Cursor- (Pfeil-) und Spezialtasten */

void glutSpaceballMotionFunc(void (*func)(int x, int y, int z));

void glutSpaceballRotateFunc(void (*func)(int x, int y, int z));

void glutSpaceballButtonFunc(void (*func)(int button, int state));

void glutButtonBoxFunc(void (*func)(int button, int state));

void glutDialsFunc(void (*func)(int dial, int value));

void glutTabletMotionFunc(void (*func)(int x, int y));

void glutTabletButtonFunc(void (*func)(int button, int state,
   int x, int y));
```

6. Callback-Registrierung (Forts.):

- **Menü-Callbacks:**

```
void glutMenuStatusFunc(void(*func)(int status,int x,int y));
   [auch: void glutMenuStateFunc(void (*func)(int status));]
   /* status == GLUT_MENU_IN_USE oder GLUT_MENU_NOT_IN_USE */
```

- **Globale Callbacks:**

```
void glutIdleFunc(void (*func)(void));  /*wenn kein Ereignis*/

void glutTimerFunc(unsigned int msecs,(*func)(int val),val);
            /*msecs Millisekunden spaeter: Aufruf func(val)*/
```

7. Verwaltung von Farbtabellen mit Farbindex

(*Color Index Colormap Management)* – 3 Routinen:

```
void glutSetColor(int cell, GLfloat red, GLfloat green,
                            GLfloat blue);      /*LUT-Eintrag*/
GLfloat glutGetColor(int cell, int component);
void glutCopyColormap(int win);
```

8. Zustands-Abfrage *(State Retrieval)* – 5 Routinen:

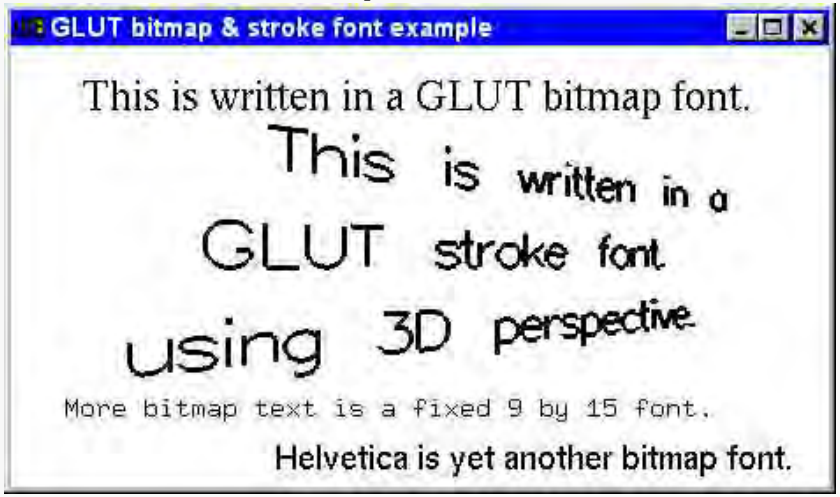
```
int glutGet(GLenum state);
    /* state: 1 v.35 Zustandskennungen, z.B.GLUT_WINDOW_X */
int glutLayerGet(GLenum info);          /* Overlay-Nutzung */
int glutDeviceGet(GLenum info);
                /* info: 1 von 10 GLUT_HAS_... (Maus etc.) */
int glutGetModifiers(void);
                  /* GLUT_ACTIVE_SHIFT, ..._CTRL, ..._ALT */
int glutExtensionSupported(char *extension);/*OpenGL-Extens.*/
```

9. Wiedergabe von Bitmap- und Vektor-Schriftarten

(Font Rendering) – 9 Routinen:

```
void glutBitmapCharacter(void *font, int character);
```

font aus: **GLUT_BITMAP_8_BY_13**
GLUT_BITMAP_9_BY_15
GLUT_BITMAP_TIMES_ROMAN_10
GLUT_BITMAP_TIMES_ROMAN_24
GLUT_BITMAP_HELVETICA_10
GLUT_BITMAP_HELVETICA_12
GLUT_BITMAP_HELVETICA_18




```
int glutBitmapWidth(GLUTbitmapFont font, int character)
                    /*gibt Breite der Bitmap-Schrift in Pixel*/
void glutStrokeCharacter(void *font, int character);
```

font aus: **GLUT_STROKE_ROMAN**
GLUT_STROKE_MONO_ROMAN

```
int glutStrokeWidth(GLUTstrokeFont font, int character);
```

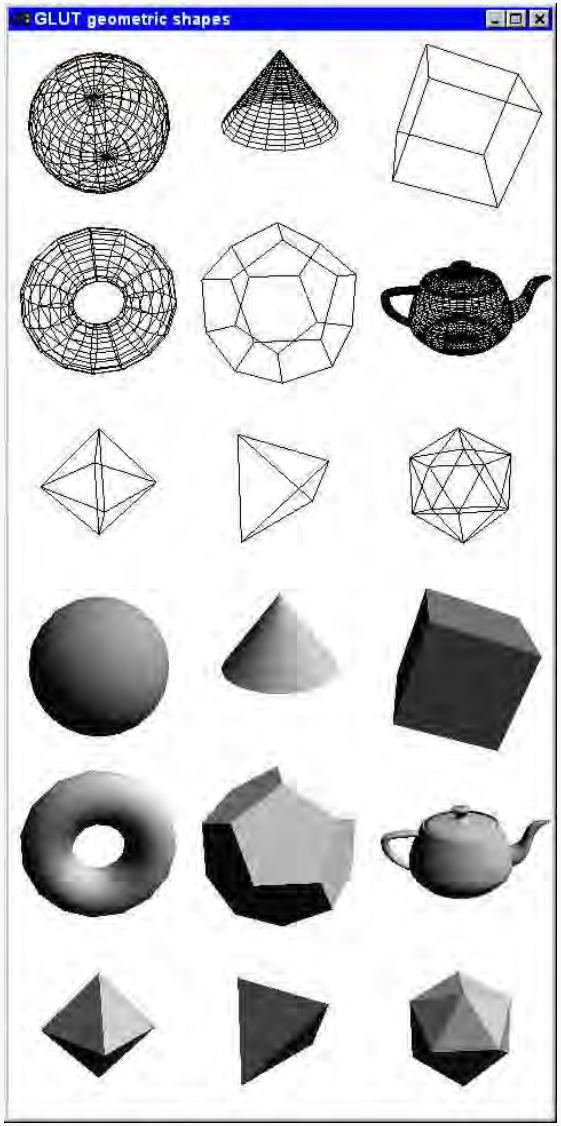
(s.u.)

GLUT - Unterschnittstellen

10. Wiedergabe Geometrischer Figuren

(Geometric Shape Rendering)–18 Routinen:

`glutSolidSphere,`	`glutWireSphere`
`glutSolidCube,`	`glutWireCube`
`glutSolidCone,`	`glutWireCone`
`glutSolidTorus,`	`glutWireTorus`
`glutSolidDodecahedron,`	`glutWireDodecahedron`
`glutSolidOctahedron,`	`glutWireOctahedron`
`glutSolidTetrahedron,`	`glutWireTetrahedron`
`glutSolidIcosahedron,`	`glutWireIcosahedron`
`glutSolidTeapot,`	`glutWireTeapot`



GLUT

Anmerkungen zu GLUT:

- GLUT-Koordinaten (Bildschirm, Fenster) in Pixel; Ursprung oben links (wie d. meisten Fenstersysteme ⇔ OpenGL: math. Koord.).
- Fenster-, Menü- u. Menüpunkt-Kennungen beginnen mit 1.
- GLUT-Header enthält OpenGL- u. GLU-Header:
  `#include <GL/glut.h>`
- **`glutInit`** soll zur Hauptinitialisierung genau einmal, möglichst am Programm-Anfang aufgerufen werden. Nur Aufrufe mit **`glutInit`**–Präfix dürfen davor stehen (z.B. zum Setzen von Voreinstellungen).
- GLUT übernimmt u.a.:
  - die Festlegung des Zeitpunktes für Fenster-Aktionen (Darstellung, Aktualisierung etc. immer erst nach Rückgabe der Kontrolle an die Ereignisverarbeitung von GLUT)
  - die Behandlung mehrerer verwandter Aufrufe (z.B. nach mehrmaligem **`glutPostRedisplay`** oder sich gegenseitig aufhebenden Fenster-Aktivierungen).

GLUT

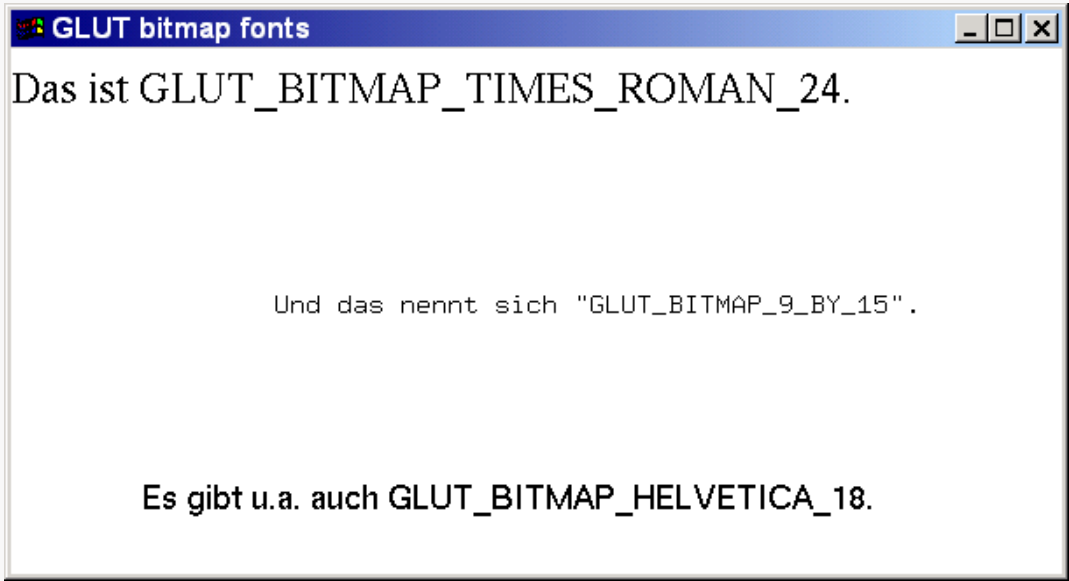
Neben GLUT-Unterschnittstellen: OpenGL-Aufrufe, z.B.:

- Cursor setzen an Position (x, y): **-1.≤ x,y ≤ +1.:**
  `void glRasterPos2f(GLfloat x, GLfloat y);`
  Untere linke Fenster-Ecke bei (**-1.;-1.**)
- Vor Fortsetzung vorausgegangene Aufrufe ausführen:
  `void glFlush(void);` `/*(vgl. fflush)*/`
- Fenster löschen:
  `void glClear(GLbitfield mask);`
  Für **mask** hier immer: **GL_COLOR_BUFFER_BIT**
- Zeichenfarbe wählen (GLUT-Def.: **(1.,1.,1.)**):
  `void glColor3f(GLfloat red, GLfloat green,`
  `GLfloat blue);`
- Lösch-/Hintergrundfarbe wählen (GLUT-Def.: **(0.,0.,0.,1.)**):
  `void glClearColor(GLclampf red, GLclampf green,`
  `GLclampf blue, GLclampf alpha);`
  Farbkomponenten werden intern auf den Bereich [0,1] begrenzt („clamped“).

GLUT

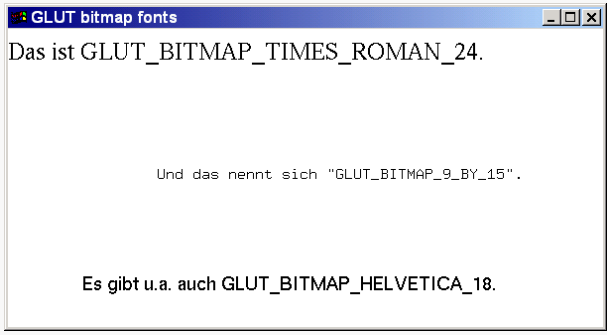
Beispiel: Text-Ausgabe mit GLUT:

```
int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(600, 300);
  glutCreateWindow("GLUT bitmap fonts");
  glClearColor(1.0, 1.0, 1.0, 1.0);
  glColor3f(0., 0., 0.);
  glutDisplayFunc(display);
  glutMainLoop();
  return 0;
}
```



Beispiel: Text-Ausgabe mit GLUT (Forts.)

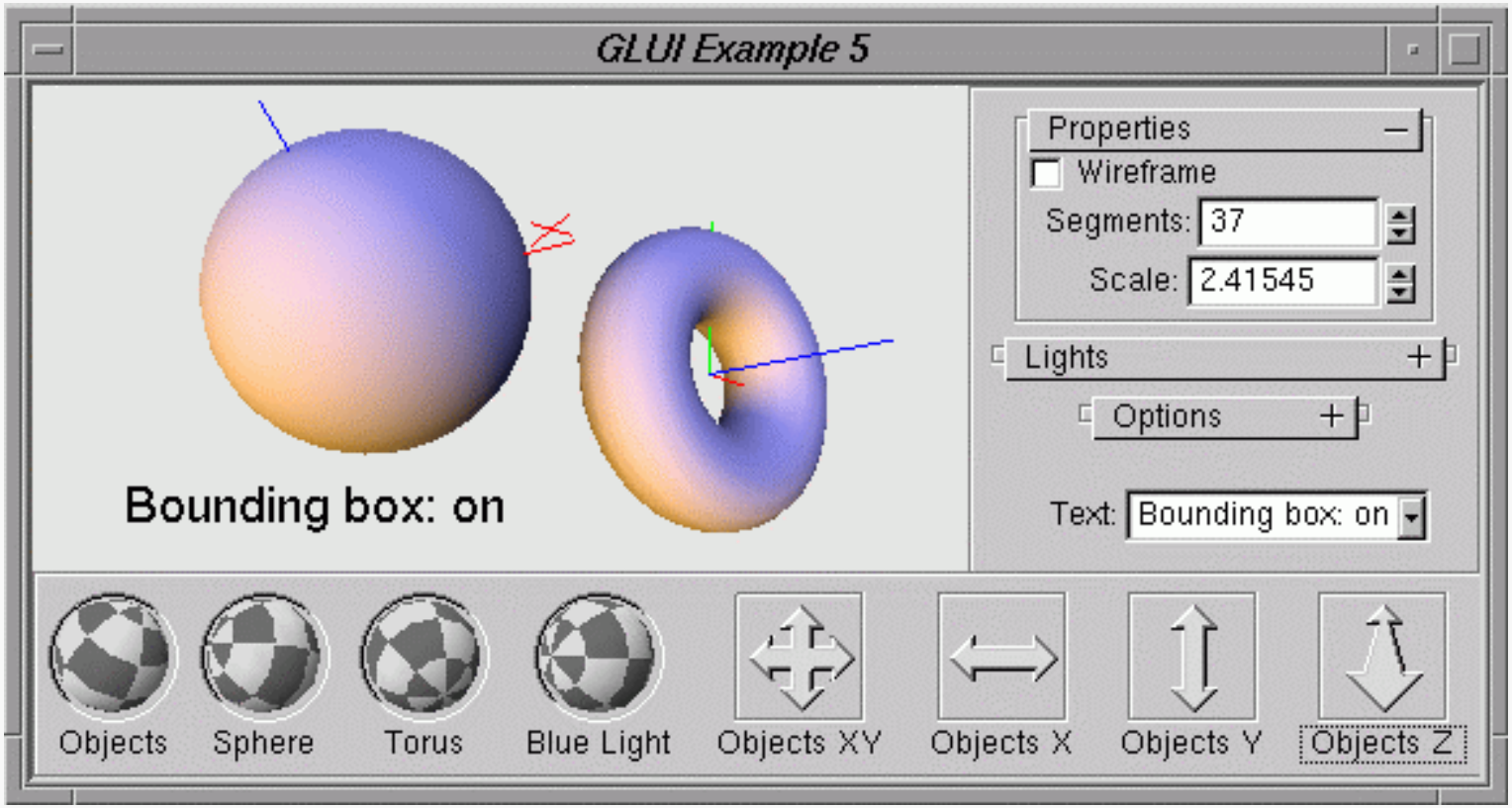
```
#include <GL/glut.h>
void bmpOut(GLfloat x, GLfloat y, void *font, char *string)
{ int len, i;
  glRasterPos2f(x, y);
  len = (int) strlen(string);
  for (i = 0; i < len; i++)
    glutBitmapCharacter(font, string[i]);
}
void display(void)
{ glClear(GL_COLOR_BUFFER_BIT);
  bmpOut(-1.f, 0.8f, GLUT_BITMAP_TIMES_ROMAN_24,
         "Das ist GLUT_BITMAP_TIMES_ROMAN_24.");
  bmpOut(-0.5f, 0.0f, GLUT_BITMAP_9_BY_15,
     "Und das nennt sich \"GLUT_BITMAP_9_BY_15\".");
  bmpOut(-0.75f, -0.75f, GLUT_BITMAP_HELVETICA_18,
     "Es gibt u.a. auch GLUT_BITMAP_HELVETICA_18.");
  glFlush();
}
```



GLUTbmpFont.exe

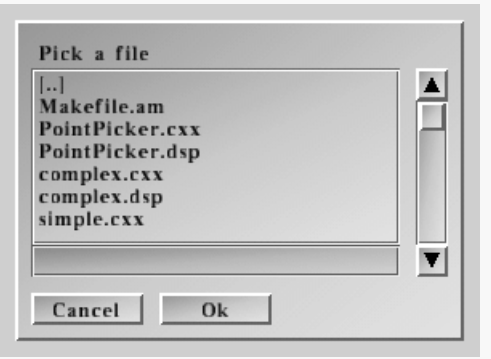
GLUT - basierte Benutzungsschnittstellen

- **GLUI 2.1**: **GL**UT-based C++ **U**ser **I**nterface
  (Paul Rademacher, Univ. of North Carolina)
  www.cs.unc.edu/~rademach/glui/



- **PLIB 1.4.2**: A **P**ortable Games **Lib**rary / SDK (Steve J. Baker)

  http://plib.sourceforge.net/



  PUI: A **P**icoscopic **U**ser **I**nterface (part of PLIB)
  http://plib.sourceforge.net/pui/index.html


Mooeee!


- **MUI**: Micro User Interface (Tom Davis, Silicon Graphics Inc.)

www.opengl.org//resources/code/samples/mjktips/mui/mui.html

www.opengl.org//resources/code/samples/glut_examples/mui/mui.html

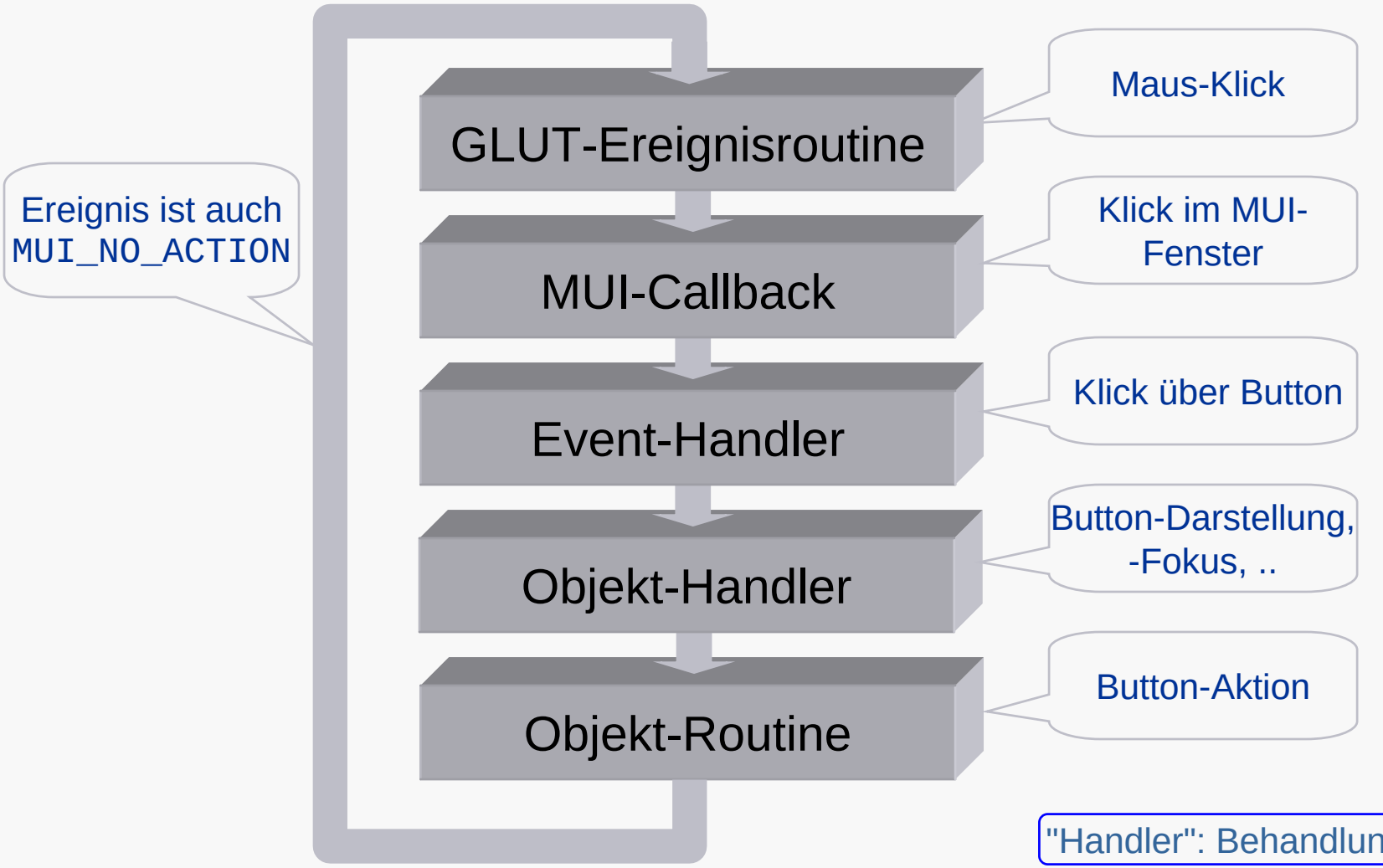
MUI

MUI...

- ..übernimmt von GLUT Ereignisse und Callbacks
- ..gehört zu den frühen GLUT-Benutzungsschnittstellen (seit GLUT 3.5, 1997)
- ..hat einen überschaubaren Umfang
- ..besitzt als einzige Dokumentation -neben dem veröffentlichten Quellcode- die persönlichen Notizen von Steve J. Baker (PLIB)

MUI

Wirkungsweise von MUI:



"Handler": Behandlungsroutine

MUI

Struktur eines typischen MUI-Programms:

```
#include <stdio.h>
#include <GL/glut.h>
#include <mui/mui.h>

int Win;
/*...*/

int main(int argc, char **argv)
{ glutInitWindowSize(250, 400);  /*für alle Fenster*/
  glutInit(&argc, argv);
  Win = glutCreateWindow("MUI@SysProg");/*Einrichtg*/
  makeUI();              /*mui-Widgets, Callbacks etc.*/
  muiInit();                  /*vgl. glutDisplayFunc()*/
  glutMainLoop();  /*Start d. Ereignis-Verarbeitung*/
  return (0);
}
```

„Widget“: window gadget („Fenster-Dingsbums“)

MUI

Die MUI-Initialisierung:

```
muiInit();
```

muß erfolgen vor der Übergabe an GLUT durch

```
glutMainLoop();
```

Zuvor muß (mindestens) eine Benutzungsschnittstellen-Liste (UI List) erstellt worden sein; Verwaltungsroutinen:

```
void muiNewUIList       (int listid);

void muiAddToUIList     (int listid, muiObject *obj);

void muiSetActiveUIList (int listid);

int  muiGetActiveUIList ();
```

(Es gibt keine Löschroutinen)

Struktur für MUI-Objekte (aus: **`Mui.h`**):

```
typedef struct muiobj                    /*Struktur-Name*/
{ enum  muiObjType type; /*Objekttypen,z.B.MUI_BUTTON*/
  int   xmin, xmax, ymin, ymax;      /* bounding box */
  short active;   /*1=pressed, type in (textbox) etc.*/
  short enable;      /*1=accessible; with solid text*/
  short select;     /*1=pressed (located at the time)*/
  short locate;   /*1=located; usually:cursor over it*/
  short visible;   /*1=drawn;not visible=>not enabled*/
  int   id;                        /*arbitary user data*/
  int   uilist;
  void *object;     /*Allg.(!) Zeiger: Untermenü o.a.*/
  enum muiReturnValue (*handler)(struct muiobj *obj,
                   int event, int value, int x, int y);
  void(*callback) (struct muiobj*,enum muiReturnValue);
} muiObject;                                /*Typ-Name*/
    /*muiReturnValue: Ereignisse,z.B.MUI_BUTTON_PRESS*/
```

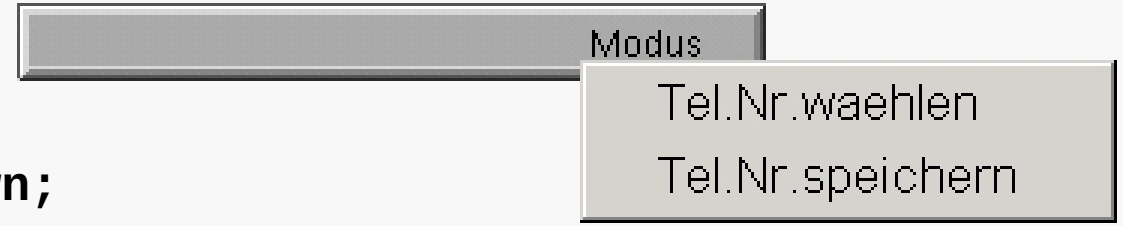
MUI-Widgets

Pull-Down-Menüs durch die MUI-Aufrufe:

```
muiObject *muiNewPulldown ();
void muiAddPulldownEntry (muiObject *obj, char *title,
                          int glut_menu,  int is_help);
```

in Kombination mit den entsprechenden GLUT-Routinen

Beispiel:




```
muiObject *pulldown;
int Menue= glutCreateMenu(menuCallback);
glutAddMenuEntry("Tel.Nr.waehlen", NOMSG);
glutAddMenuEntry("Tel.Nr.speichern", STORE);
muiNewUIList(1);          /*MUI display list number 1*/
pulldown = muiNewPulldown();
muiAddPulldownEntry(pulldown, "Modus", Menue, 1);
```

MUI-Widgets

Drei Sorten von **Buttons** (Schaltflächen):

- Button (Text darauf)
- Radio Button (Text seitlich).
- Tiny Radio Buttons (Text seitlich)


```
muiObject *muiNewButton          (int xmin, int xmax,
                                  int ymin, int ymax);
muiObject *muiNewRadioButton     (int xmin, int ymin);
muiObject *muiNewTinyRadioButton (int xmin, int ymin);
```

Beschriftung:

```
void muiLoadButton (muiObject *button, char *label);
```

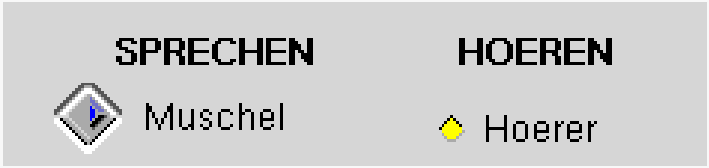
Verschaltung von je 2 Buttons, Abschaltung:

```
void muiLinkButtons(muiObject *obj1, muiObject *obj2);
void muiClearRadio ( muiObject *button );
```

MUI-Widgets

Text Label (Text-Bezeichnungsfeld)

```
muiObject *muiNewLabel (int xmin,int ymin,char *label);
muiObject *muiNewBoldLabel (int xmin,int ymin,char *label);
```



Änderung des Textes mit:

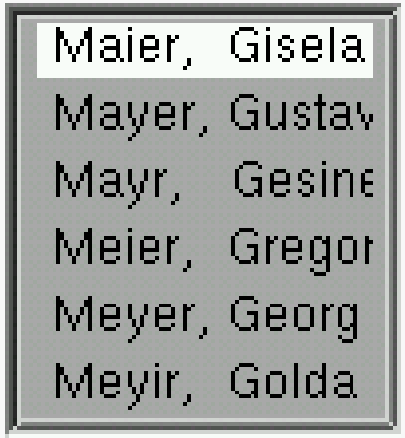
```
void muiChangeLabel (muiObject *obj,char *label);
```

Text Box (Textfeld)

```
muiObject *muiNewTextbox(int xmin, int xmax, int ymin);
char      *muiGetTBString   ( muiObject *obj);
void       muiClearTBString ( muiObject *obj);
void       muiSetTBString   ( muiObject *obj, char *s);
```

TextList (Textliste)

```
muiObject *muiNewTextList (int xmin, int ymin,
                           int xmax, int listheight);
void muiSetTLTop          (muiObject *obj, float p);
int  muiGetTLSelectedItem (muiObject *obj);
void muiSetTLStrings      (muiObject *obj, char **s);
```

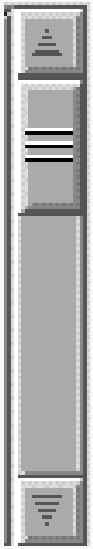


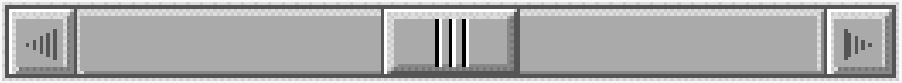
Vert. & horiz. Slider (Bildlaufleiste)

```
muiObject *muiNewVSlider (int xmin, int ymin, int ymax,
                          int scenter, int shalf);
muiObject *muiNewHSlider (int xmin, int ymin, int xmax,
                          int scenter, int shalf);

float    muiGetVSVal       (muiObject *obj);
float    muiGetHSVal       (muiObject *obj);

void     muiSetVSValue     (muiObject *obj, float val);
void     muiSetHSValue     (muiObject *obj, float val);
void     muiSetVSArrowDelta(muiObject *obj, int newd);
void     muiSetHSArrowDelta(muiObject *obj, int newd);
```





Allgemeine Funktionen für alle MUI-Objekte:

- **Zustand setzen und abfragen:**

```
void muiSetVisible (muiObject *obj, int state);
void muiSetActive  (muiObject *obj, int state);
void muiSetEnable  (muiObject *obj, int state);

int  muiGetVisible (muiObject *obj);
int  muiGetActive  (muiObject *obj);
int  muiGetEnable  (muiObject *obj);
```

- **Daten setzen und abfragen:**

```
void muiSetID (muiObject *obj, int id);
int  muiGetID (muiObject *obj);
```

Allgemeine Funktionen für alle MUI-Objekte: (Forts.)

- **Bounding box** (umhüllendes Rechteck) **abfragen:**

```
void muiGetObjectSize(muiObject *obj, int *xmin,
                      int *ymin,int *xmax,int *ymax);
```

- **Callback-Funktion für Objekt angeben:**

```
void muiSetCallback (muiObject *obj,
      void (*cb)(muiObject *, enum muiReturnValue));
```

- **Callback-Funktion für Klick ohne Ziel:**

```
void muiSetNonMUIcallback (void(*cb)(int x, int y));
```

Übung:

Sie wollen Ihre eigene Benutzungs-Schnittstelle für ein Telefon realisieren (**`Widgets.exe`**) und haben schon einiges geleistet (**`Uebung.exe`**). Stellen Sie Ihr Programm fertig mit Hilfe von GLUT und MUI - insbesondere:

- Fügen Sie den fehlenden Radio-Button für Stummschaltung zur Benutzungs-Oberfläche hinzu, und koppeln Sie zusammengehörende Radio-Buttons aneinander, damit sie nur alternativ wählbar sind.
- Koppeln sie die Textliste mit den Namen an die Textbox mit den Telefonnummern.
- Implementieren Sie die Meldungen, die in einem gleich großen, andockenden GLUT-Fenster der Telefon-UI auch bei Verschiebung folgen und nach 3 sec (GLUT-Timer) verschwinden.
- Eliminieren Sie zuletzt das Konsolen-Fenster.