Hardwarenahe Software-Entwicklung

2 Wiederholung: Bibliotheken

Wie muß man grundsätzlich vorgehen, wenn man . . .

- anderen etwas Fertiges anbieten möchte?
- nutzen möchte, was andere bereits fertig geschrieben haben?

2 Wiederholung: Bibliotheken

2.1 Der Präprozesesor

#include: Text einbinden

- **#include** <stdio.h>: Standard-Verzeichnisse – Standard-Header
- **#include** "answer.h": auch aktuelles Verzeichnis – eigene Header

#define VIER 4: Text ersetzen lassen – Konstante definieren

- Kein Semikolon!
- Berechnungen in Klammern setzen:
  #define VIER (2 + 2)
- Konvention: Großbuchstaben

2 Wiederholung: Bibliotheken

2.1 Der Präprozesesor

#include: Text einbinden

- **#include** <stdio.h>: Standard-Verzeichnisse – Standard-Header
- **#include** "answer.h": auch aktuelles Verzeichnis – eigene Header

#define VIER 4: Text ersetzen lassen – Konstante definieren

- Kein Semikolon!
- Berechnungen in Klammern setzen:
  #define VIER (2 + 2)
- Konvention: Großbuchstaben

2 Wiederholung: Bibliotheken

2.1 Der Präprozesesor

. . . hat eigentlich nichts mit Bibliotheken zu tun.

#include: Text einbinden

- **#include** <stdio.h>: Standard-Verzeichnisse – Standard-Header
- **#include** "answer.h": auch aktuelles Verzeichnis – eigene Header

#define VIER 4: Text ersetzen lassen – Konstante definieren

- Kein Semikolon!
- Berechnungen in Klammern setzen:
  #define VIER (2 + 2)
- Konvention: Großbuchstaben

2 Wiederholung: Bibliotheken

2.2 Bibliotheken einbinden

Inhalt der Header-Datei: externe Deklarationen

```
extern int answer (void);
extern int printf (__const char *__restrict __format, ...);
```

Funktion wird „anderswo“ definiert

- separater C-Quelltext: mit an `gcc` übergeben
- vorcompilierte Bibliothek: `-lfoo`
  = Datei libfoo.a in Standard-Verzeichnis

2.3 Bibliothek verwenden (Beispiel: OpenGL)

- Include-Dateien:

  ```
  #include <GL/gl.h>
  #include <GL/glu.h>
  #include <GL/glut.h>
  ```

- Compiler-Aufruf:

  ```
  gcc -Wall -O cube.c -lGL -lGLU -lglut -o cube
  ```

- Funktionen aufrufen: glutInit (&argc, argv);
- Konstante: GLUT_RGBA
- Datentypen: GLfloat
- Array übergeben:

  ```
  GLfloat light0_position[] = {1.0, 2.0, −2.0, 1.0};
  glLightfv (GL_LIGHT0, GL_POSITION, light0_position);
  ```

- Funktion übergeben (*Callback*):

  ```
  void draw_cube (void)
  { ... }

  glutDisplayFunc (draw_cube);
  ```

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include "opengl−magic.h"

float t = 0.0;

void draw (void)
{ ... }

void timer_handler (int value)
{ ... }

int main (int argc, char **argv)
{
   init_opengl (&argc, argv, "Orbit");
   glutDisplayFunc (draw);
   glutTimerFunc (50, timer_handler, 0);
   glutMainLoop ();
   return 0;
}
```

```
void draw (void)
{
   glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glPushMatrix ();
   glRotatef (23.44, 1.0, 0.0, −1.0);
   set_material_color (0.7, 0.7, 1.0);
   glutSolidSphere (0.25512, 63, 20);
   glRotatef (30.0 ∗ t, 0.0, −1.0, 0.0);
   glTranslatef (0.9, 0.0, 0.0);
   set_material_color (1.0, 0.7, 0.0);
   glutSolidSphere (0.06952, 31, 10);
   glPopMatrix ();
   glFlush ();
   glutSwapBuffers ();
}

int main (int argc, char ∗∗argv)
{
   ...
   glutDisplayFunc (draw);
   ...
}
```

```
float t = 0.0;

void draw (void)
{ ... }

void timer_handler (int value)
{
   t += 0.05;
   glutPostRedisplay ();
   glutTimerFunc (50, timer_handler, 0);
}

int main (int argc, char **argv)
{
   ...
   glutDisplayFunc (draw);
   glutTimerFunc (50, timer_handler, 0);
   ...
}
```

2.3 Bibliothek verwenden (Beispiel: OpenGL)

Aufgabe:

Für welche elementaren geometrischen Körper
stellt die GLUT-Bibliothek Zeichenroutinen zur Verfügung?

Lösung: In die .h-Dateien schauen!

```
glutSolidCube (GLdouble size);
glutWireCube (GLdouble size);
glutSolidSphere (GLdouble radius, GLint slices, GLint stacks); ... Wire ...
glutSolidCone (GLdouble base, GLdouble height, GLint slices, GLint stacks);
glutSolidTorus (GLdouble innerRadius, GLdouble outerRadius,
                GLint sides, GLint rings);
glutSolidDodecahedron (void);
glutSolidOctahedron (void);
glutSolidTetrahedron (void);
glutSolidIcosahedron (void);
glutSolidTeapot (GLdouble size);
glutSolidRhombicDodecahedron (void);
glutSolidSierpinskiSponge (int num_levels, GLdouble offset[3], GLdouble scale);
glutSolidCylinder (GLdouble radius, GLdouble height, GLint slices, GLint stacks);
```

Hardwarenahe Software-Entwicklung

Hardwarenahe Software-Entwicklung

Hardwarenahe Software-Entwicklung

Hardwarenahe Software-Entwicklung

Hardwarenahe Software-Entwicklung

Hardwarenahe Software-Entwicklung

Programmierung des AVR ATmega

PORTA, PORTB, PORTC, PORTD: Ausgabe-Ports
Was man in diese Variablen schreibt, wird bitweise durch die LEDs dargestellt.

DDRA, DDRB, DDRC, DDRD: *Data Direction Registers*
Durch eine 1 macht man den entsprechenden Port zum Output-Port;
ansonsten ist er ein Input-Port.

Beispiele:

- DDRA = 0xff;
  konfiguriert PORTA komplett als Output-Port.
- DDRB = 0xf0;
  konfiguriert die oberen vier Bits von PORTB als Output-Port,
  die unteren vier als Input-Port.

_delay_ms (137): 137 Millisekunden warten

Praktikumsaufgabe

Programmieren Sie eine Animation in einer LED-Spalte:
Ein roter Lichtpunkt wandert auf und ab. (Zylonen / Knight Rider)