OpenGL Programme für Windows und BeOS schreiben

Inhalt

1.Vorwort

Da ich mich gerade etwas in OpenGL einarbeite habe ich versucht meine Programme sowohl unter Windows als auch unter BeOS zum Laufen zu Bringen. Mit Hilfe der GLUT-Bibliothek ist dies ohne Änderungen am Code möglich. Das Programm muss lediglich unter dem jeweiligen Betriebssystem neu compiliert werden.

Was muss ich vorher wissen?

Du solltest Dich schon einigermassen gut mit der Programmiersprache C auskennen.

2. GLUT installieren

2.1 GLUT-Bibliothek unter BeOS

Als erstes brauchst Du die Developer Tools, diese können auf BeZip.de heruntergeladen werden (in der Pro. Version schon enthalten). Jetzt brauchst Du auch noch die BeOS-GLUT-Bibliothek die ebenfalls auf BeZip zu finden ist. Nach dem Downlad der GLUT-Bibliothek muss die zip-Datei mit einem Doppelklick entpackt (das Verzeichnis, in das GLUT entpackt wird, ist beliebeig) werden. Jetzt muss ein Terminal Fenster geöffnet werden indem Du im Be Menü unter Applications (Programme) auf Terminal klickst. Im Terminal muss in das Verzeichnis gewechselt werden in dem sich die entpackte Version der GLUT-Bibliothek befindet. Ein Verzeichnis wechselt man im Terminal übrigends mit ***cd Verzeichnisname***, um ein Verzeichnis zurück zu kommen benutzt man den Befehl ***cd ..***

Jetzt muss nur noch setup.sh eingetippt werden und die nötigen Dateien werden selbständig an die richtigen Verzeichnisse kopiert und ein kleines Video abgespielt. Das wars, jetzt kannst Du mit GLUT und OpenGL unter BeOS arbeiten.

2.2 GLUT-Bibliothek unter Windows

Unter Windows ist die Installation der GLUT-Bibliothek leider nicht so einfach wie unter BeOS, da es unter Windows viele verschiedene C-Compiler gibt. Zuerst muss die passende Windows-GLUT-Bibliothek heruntergeladen und entpackt werden. In dem Verzeichnis in das die GLUT-Bibliothek entpackt wurde befinden sich nun einige Dateien. Diese Dateien müssen in folgende Verzeichnisse kopiert werden:

Dateiname	Kopierziel
gl.h glut.h glu.h	Verzeichnis Deines C-Compilers\include\gl
opengl32.lib glut32.lib glu32.lib	Verzeichnis Deines C-Compilers\lib
opengl32.dll glut32.dll glu32.dll	Windows 9x: c:\windows\system Windows NT/2000: winnt\system32

Ja nachdem welchen Compiler Du verwendest musst Du ein Projekt anlegen und neben Deiner *.c(pp) Datei die Dateien opengl32.lib, glut32.lib sowie glu32.lib hinzufügen.

3.1 Das erste Programm unter BeOS

Starte zunächst BeIDE (im Be-Menü unter Applications (Programme)) und erstelle ein neues Projekt, indem Du auf File -> New Project klickst. Jetzt erscheint ein Fenster in dem Du die Art des Projekts festlegen kannst. Hier wählst Du Emty Project aus und läst das Feld Create Folder markiert. Nachdem Du jetzt auf Create geklickt hast erschein ein Fenster indem Du Deinem Projekt einen Namen geben kannst. Ich habe hier mal Programm1.prj eingegeben aber jeder andere Name geht natürlich auch. Achte nur darauf das Du die Endung .prj anhängst. Wenn Du jetzt auf Save klickst erscheint Dein Projektfenster.

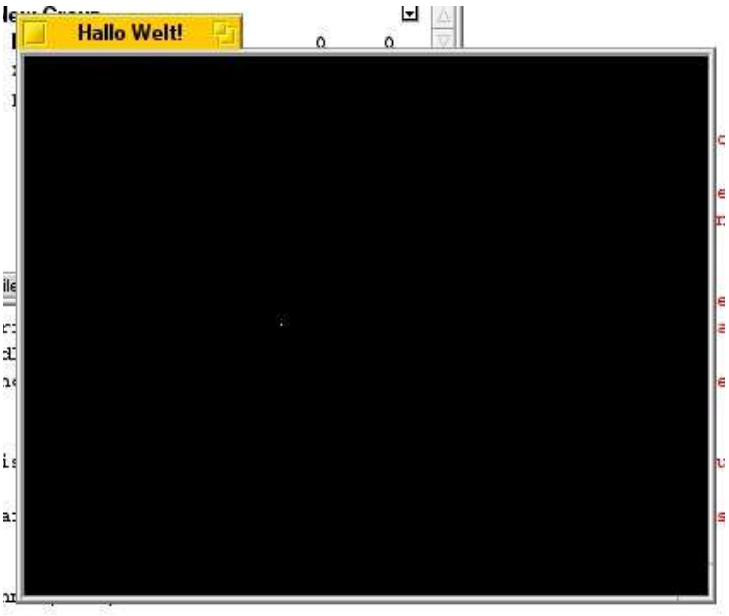
Jetzt müssen noch die nötigen Bibliotheken hinzugefügt werden indem Du auf AddFiles im Menü Project klickst. Als erstes muss die Datei libGl.so hinzugefügt werden. Diese ist im Verzeichnis /boot/beos/system/lib/libGl.so zu finden. Jetzt fehlt noch die Datei libgl.so, die im Verzeichnis /boot/home/config/lib/libglut.so zu finden ist.

Wechsle jetzt in den Texteditor der noch im Hintergrund laufen müste und gib folgenden Code ein:

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

void zeichnen(void);

int main(int argc, char **argv)
{
        glutInit(&argc, argv);                              //Initialsierung der GLUT-Bibliothek
        glutInitDisplayMode(GLUT_SINGLE  | GLUT_RGB);       //Initialisierung des Bildschirms(Art des
                                                            //Zwischenspeichers, Farben)
        glutInitWindowSize(400,300);                        //Fenstergrösse festlegen(Breite, Höhe)
        glutInitWindowPosition(100,100);                    //Position des Fensters
        glutCreateWindow("Hallo Welt!");                    //Fenstertitel

        glClearColor(0.0,0.0,0.0,0.0);                      //Hintergrundfarbe
        glMatrixMode(GL_PROJECTION);                        //Art des Koordinatensytems festlegen
        glLoadIdentity();
        glOrtho(0.0,400.0,0.0,300.0,-1.0,1.0);              //Koordinatenverteilung festlegen
                                                            //x-> 0.0 ... 400.0
                                                            //y- > 0.0 ... 300.0

        glutDisplayFunc(&zeichnen);     //Festelgen der Funktion die zum zeichnen auf den
                                        //Bildschirm verwendet wird

        glutMainLoop();                                     //eine Art Endlosschleife des Programms
}

void zeichnen(void)
{
        glClear(GL_COLOR_BUFFER_BIT);                       // Fenster löschen

        glBegin(GL_POINTS);                                 //Zeichenmodus setzen (einzelne Pixel)
                glColor3f(1.0,1.0,1.0);                     //Zeichenfarbe wählen
                                                            //(Rot-,Grün-,Blauanteil)
                glVertex2i(150,150);                        //Koordinaten an denen der Punkt gesetzt
                                                            //werden soll
        glEnd();

        glFlush();                                          //Zeichnen
}
```

Nachdem Du den Code eingetippt hast muss das ganze noch gespeichert werden indem Du auf *File -> Save* klickst. Der Name spielt wieder keine Rolle, er sollte nur mit *.cpp* enden. Du solltest das Feld *Add to Project* aktivieren damit die Datei Deinem Projekt hinzugefügt wird. Jetzt kannst Du das ganze Programm compilieren und linken indem Du auf *Make im Menü Project* klickst. Wenn keine Fehler aufgetreten sind kannst Du das Programm nun laufen lassen indem Du auf Run im Menü Project klickst.

Das ganze müste dann so aussehen:



3.2 Das erste Programm unter Windows

In Deiner Entwicklungsumbebung solltest Du ein neues Projekt (Terminal-Anwendung) erstellen und die Dateien opengl32.lib, glut32.lib und glu32.lib hinzufügen. Wer den C++ Builder verwendet braucht die Dateien nicht hinzuzufügen da sich der C++ Builder diese Dateien selber zusammensucht. Anschliessend muss wie unter BeOS eine C++ Datei mit dem obigen Code erstellt werden. Nachdem das Programmm dann compiliert und gelinkt worden ist sollte man zum gleichen Ergebnis kommen wie unter BeOS.

4. Beschreibung des ersten Programms

Initialisierung des Fensters

```
1:      glutInit(&argc, argv);                          //Initialsierung der GLUT-Bibliothek
2:      glutInitDisplayMode(GLUT_SINGLE  | GLUT_RGB);   //Initialisierung des Bildschirms(Art des
                                                        //Zwischenspeichers, Farben)
3:      glutInitWindowSize(400,300);                    //Fenstergrösse festlegen(Breite, Höhe)
4:      glutInitWindowPosition(100,100);                //Position des Fensters
5:      glutCreateWindow("Hallo Welt!");                //Fenstertitel
```

In der ersten Zeile wird die GLUT-Bibliothek initialisiert. Das Programm wird mit der Grafischen Oberfläche des Betriebssystems abgestimmt. Wenn dies nicht gelinkt wird eine Fehlermeldung ins Terminal ausgegeben.

In der zweiten Zeile wird die Art des Zwischenspeichers auf GLUT_SINGLE gesetzt. Soll sich in dem Programm irgendwas im Fenster ändern sollte statt GLUT_SINGEL GLUT_DOUBLE verwendet werden. Wird GLUT_DOUBLE verwendet muss am Ende der Zeichenfunktion der Zwischenspeicher getauscht werden, dies geschieht mit der Funktion glutSwapBuffers(); . In dem Beispiel wird mit GLUT_RGB noch die Art der Farben bestimmt. Alle diese Initialisierungswerte werden durch ein Oderzeichen (|) verknüpft.

Die dritte Zeile dient dazu die Grösse des Fensters fest zu legen. In unserem Beispiel währe das Fenster 400 Pixel breit und 300 Pixel hoch.

Die 4. Zeile der Initialisierung dient dazu das Fenster an eine bestimmte Position auf dem Bildschirm zu setzten. Als Referenzpunkt dient dazu die linke obere Ecke des Fensters. Unter BeOS ist es die linke obere Ecke unterhalb der gelben Titelzeile des Fensters.

Mit der 5. Zeile wird das Fenster schliesslich erzeugt. Dabei muss ein Titel für das Fenster übergeben werden.

Weitere Einstellungen

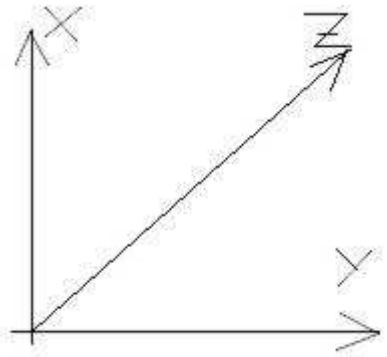
```
1:      glClearColor(0.0,0.0,0.0,0.0);                  //Hintergrundfarbe
2:      glMatrixMode(GL_PROJECTION);                    //Art des Koordinatensytems festlegen
3:      glLoadIdentity();
4:      glOrtho(0.0,400.0,0.0,300.0,-1.0,1.0);          //Koordinatenverteilung festlegen
                                                        //x-> 0.0 ... 400.0
                                                        //y- > 0.0 ... 300.0
```

In der ersten Zeile wird die Hintergrundfarbe im Fenster bestimmt. Dazu wird ein RGB-Wert übergeben, sowie ein Alpha-Wert. Der Alpha-Wert kann beispielsweise dazu verwendet werden um eine Transparente Farbe zu kenzeichnen. Fürs erste kannst Du ihn auf 0.0 lassen. Alle diese Angaben müssen zwischen 0.0 und 1.0 liegen.

Ein paar RGB-Farbwerte

Rot	Grün	Blau	Ergebnis
0.0	0.0	0.0	Schwarz
1.0	1.0	1.0	Weis
1.0	0.0	0.0	Rot
1.0	1.0	0.0	Gelb
0.0	1.0	0.0	Grün
0.0	1.0	1.0	Türkis
0.0	0.0	1.0	Blau
0.5	0.5	0.5	Grau

Die Zeilen 2-4 dienen dazu ein Koordinatensystem zu erzeugen. In der Zeile 4 weden dazu die maximalwerte übergeben. Die ersten beiden Übergabewerte bestimmen die Einheiten der X-Achse. In unserem Beispiel würde diese von 0.0 bis 400.0 laufen. Die nächsten beiden Werte bestimmen die Einheiten der Y-Achse die in unserem Beispiel von 0.0 bis 300. laufen. Die letzten beiden Werte bestimmen die Einheiten der z-Achse, die "in den Bildschirm hinein läuft". Das folgende Bild zeigt das Koordinatensystem beim Zeichnen:



Zeichnende Funktion festlegen

```
glutDisplayFunc(&zeichnen);     //Festelgen der Funktion die zum zeichnen auf
den Bildschirm verwendet wird
```

Beim Aufruf der Funktion wird ein Zeiger auf eine Funktion übrgeben die die Anweisungen zum Zeichen auf den Bildschirm enthält. In unserem Beispiel währe dies die Funktion zeichnen() :

```
void zeichnen(void)
{
1:      glClear(GL_COLOR_BUFFER_BIT);                   // Fenster löschen

2:      glBegin(GL_POINTS);                             //Zeichenmodus setzen (einzelne Pixel)
3:              glColor3f(1.0,1.0,1.0);                 //Zeichenfarbe wählen
                                                        //(Rot-,Grün-,Blauanteil)
4:              glVertex2i(150,150);                    //Koordinaten an denen der Punkt gesetzt
                                                        //werden soll
5:      glEnd();

6:      glFlush();                                      //Zeichnen
}
```

Mit der ersten Zeile wird zunächst der gesamte Fensterinhalt gelöscht, d.h. auf die zuvor eingestellte Hintergrundfarbe gesetzt. Mit dem Befehl in Zeile 2 wird der Zeichenmodus gesetzt und gleichzeitig angezeigt das die Folgendenen Anweisungen zum Zeichnen einzelner Pixel auf dem Bildschirm

verwendet werden.

In Zeile 3 wird die Farbe des Pixels als RGB-Wert eingestellt. Die 4. Zeile bestimmt die Position des Pixels im Fenster.

Mit dem Aufruf der Funktion glEnd() in Zeile 5 das setzen der Pixel abgeschlossen und mit der Anweisung in Zeile 6 ausgeführt.

glutMainLoop();

Das Programm wird schliesslich mit der Funktion glutMainLoop abgeschlossen, die das Fenster erst schliesst wenn der Benutzer das Programm beendet. Vereinfacht könnte man sagen das es sich hierbei um eine Endlosschleife der main()-Funktion handelt.

5. Geometrische Objekte

Natürlich können mit OpenGL nicht nur einzelne Pixel gesetzt werden. Die Art des Objekts wird mit dem Übergabewert von glBegin() bestimmt. Die folgende Tabelle zeigt welche Übergabewerte möglich sind. Die nötigen Punkte werden wie im obigen Beispiel mit glVertex2i(x,y) angegeben.

Übergabewert	Bedeutung	Wieviele Punkte müssen mindestens angegeben werden
GL_POINTS	einzelne Pixel	1
GL_LINES	Linie	2
GL_LINE_STRIP	mehrere Linien wobei das Ende der Vorherigen der Anfang der Nächsten ist	3
GL_LINE_LOOP	wie GL_LINE_STRIP, wobei das Ende der Letzten Linie mit dem Anfang der Ersten verbunden wird	3
GL_POLYGON	wie GL_LINE_LOOP, jedoch ausgefüllt mit der Zeichenfarbe	3
GL_QUADS	ausgefülltes Viereck	4
GL_QUADS_STRIP	zusammengesetzte Vierecke	6
GL_TRIANGLE	Dreieck	3
GL_TRIANGLE_STRIP	zusamengesetzte Dreiecke	4
GL_TRIANGLE_FAN	ausgehend von einem Mittelpunkt werden Dreiecke erzeugt, indem immer zwei Punkte miteinander und mit dem Mittelpunkt verbunden werden	4

5.1 Beispiele

Die nächsten Beispiele kannst Du ganz einfach umsetzen indem Du den Bereich

```
glBegin(GL_POINTS);                          //Zeichenmodus setzen (einzelne Pixel)
        glColor3f(1.0,1.0,1.0);              //Zeichenfarbe wählen
                                             //(Rot-,Grün-,Blauanteil)
        glVertex2i(150,150);                 //Koordinaten an denen der Punkt gesetzt
                                             //werden soll
glEnd();
```

durch folgendes ersetzt, oder den neuen Code einfach darunter schreibst:

Linie	`glBegin(GL_LINES);`
	`  glColor3f(1.0,1.0,1.0);`
	`  glVertex2i(100,150);`
	`  glVertex2i(200,150);`
	`glEnd();`

zusammengesetze Linien	`glBegin(GL_LINE_STRIP);`
	`  glColor3f(1.0,1.0,1.0);`
	`  glVertex2i(100,150);`
	`  glVertex2i(200,150);`
	`  glVertex2i(150,200);`
	`glEnd();`

zusammengesetzte Linien
Anfang und Ende verbunden
	`glBegin(GL_LINE_LOOP);`
	`  glColor3f(1.0,1.0,1.0);`
	`  glVertex2i(100,150);`
	`  glVertex2i(200,150);`
	`  glVertex2i(150,200);`
	`glEnd();`

Polygon	`glBegin(GL_POLYGON);`
	`  glColor3f(1.0,1.0,1.0);`
	`  glVertex2i(100,150);`
	`  glVertex2i(200,150);`
	`  glVertex2i(150,200);`
	`glEnd();`

Viereck	`glBegin(GL_QUADS);`
	`  glColor3f(1.0,1.0,1.0);`
	`  glVertex2i(100,150);`
	`  glVertex2i(200,150);`
	`  glVertex2i(150,200);`
	`  glVertex2i(50,200);`
	`glEnd();`

Zusammengesetze Vierecke	`glBegin(GL_QUAD_STRIP);`
	`  glColor3f(1.0,1.0,1.0);`
	`  glVertex2i(50,150);`
	`  glVertex2i(50,200);`
	`  glVertex2i(100,150);`
	`  glVertex2i(100,200);`
	`  glColor3f(1.0,1.0,0.0);`
	`  glVertex2i(150,150);`
	`  glVertex2i(150,200);`
	`glEnd();`

Dreieck	`glBegin(GL_TRIANGLE);`
	`glColor3f(1.0,1.0,1.0);`
	`glVertex2i(50,150);`
	`glVertex2i(50,200);`
	`glVertex2i(100,150);`
	
	`glEnd();`

Zusammengesetze Dreiecke	`glBegin(GL_TRIANGLE_STRIP);`
	`glColor3f(1.0,1.0,1.0);`
	`glVertex2i(50,150);`
	`glVertex2i(50,200);`
	`glVertex2i(100,150);`
	`glColor3f(1.0,1.0,0.0);`
	`glVertex2i(100,200);`
	`glEnd();`

Dreiecke aus Mittelpunkt	`glBegin(GL_TRIANGLE_FAN);`
	`glColor3f(1.0,1.0,1.0);`
	`glVertex2i(50,150);  //Nullpunkt`
	`glVertex2i(100,180);`
	`glVertex2i(150,150);`
	`glColor3f(1.0,1.0,0.0);`
	`glVertex2i(100,100);`
	`glColor3f(1.0,0.0,0.0);`
	`glVertex2i(75,100);`
	`glEnd();`

6. Tastaturereignisse mit GLUT auswerten

Um eine Tastatureingabe mit GLUT zu ermöglichen braucht man zunä:nachst eine GLUT-Funktion namens glutKeyboardFunc(), dieser wird in der main()-Funktion ein Zeiger auf eine Funktion übergeben, die das Tastaturereignis auswertet. Diese Funktion muss selbst erstellt werden, die Ãœbergabewerte der Funktion sind dabei fest vorgegeben, der Name kann jedoch selbst gewählt werden. Der Funktionskopf sieht damit so aus:

void tastatur(unsigned char key, int x, int y);
key entspricht der Taste die gedrückt wurde und die beiden Integerwerte x und y entsprechen den Koordinaten des Mauszeigers.

Wie das ganze im Programm aussieht siehst Du im Beispiel das nach der Mausverarbeitung folgt

7. Mausereignisse mit GLUT auswerten

Die Verarbeitung von Mausereignissen ähnelt der verarbeitung der Tastaturereignisse. Auch hier wird zunächst eine GLUT-Funktion benötigt, der ein Zeiger auf eine Funktion übergeben wird die das Mauserreignis verarbeitet. Die benötigte GLTU Funktion heisst glutMouseFunc(), ihr muss ein Zeiger auf eine Funktion übergeben werden die folgende Übergabewerte besitzt:

void maus(int button, int state, int x, int y);

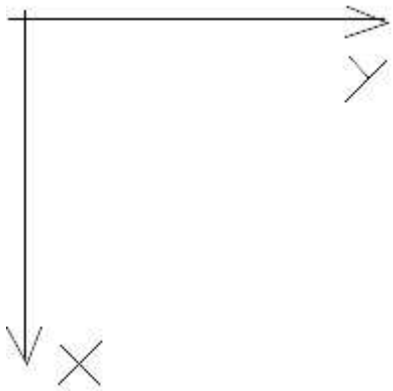
button gibt an welche Taste gedrückt wurde

GLUT_LEFT_BUTTON	Linke Maustaste
GLUT_MIDDLE_BUTTON	Mittlere Maustaste
GLUT_RIGHT_BUTTON	Rechte Maustaste

state gibt an ob die Taste gedrückt wurde oder losgelassen wurde

GLUT_DOWN	Taste gedrückt
GLUT_UP	Taste losgelassen

x und y geben die Position des Mauszeigers an. Diese werden wie in fogenden Bild erzeugt:



8. Ein Beispiel mit Maus und Tastatur

Was sich im vergleich zu Beispiel 1 verändert hat habe ich rot markiert!

```
#include <GL/glut.h>
#include <stdlib.h>


void zeichnen(void);
void maus(int button, int state, int x, int y);     //Mausereignis verarbeiten
void tastatur(unsigned char key, int x, int y);//Tastaturereignis verarbeiten


int main(int argc, char **argv)
  {

        glutInit(&argc, argv);                          //Initialsierung der GLUT-Bibliothek
        glutInitDisplayMode(GLUT_SINGLE  | GLUT_RGB);   //Initialisierung des Bildschirms(Art

        //des Zwischenspeichers, Farben)
        glutInitWindowSize(400,300);                    //Fenstergrösse festlegen(Breite, Höhe)
        glutInitWindowPosition(100,100);                //Position des Fensters
        glutCreateWindow("Hallo Welt!");                //Fenstertitel

        glutMouseFunc(&maus);         //Reaktion auf Maus in Funktion maus(...)
        glutKeyboardFunc(&tastatur);  //Reaktion auf Tastatur in Funktion tastatur(...)


         glClearColor(0.0,0.0,0.0,0.0);                  //Hintergrundfarbe


         glMatrixMode(GL_PROJECTION);                    //Art des Koordinatensytems festlegen
         glLoadIdentity();
         glOrtho(0.0,400.0,0.0,300.0,-1.0,1.0);          //Koordinatenverteilung festlegen
                                                                  //x-> 0.0 ... 400.0
                                                 //y- > 0.0 ... 300.0
         glutDisplayFunc(&zeichnen);         //Festelgen der Funktion die zum zeichnen auf den
                                             // Bildschirm verwendet wird
         glutMainLoop();                                 //eine Art Endlosschleife des Programms


  }

void zeichnen(void)
  {
        glClear(GL_COLOR_BUFFER_BIT);                   // Fenster löschen
        glBegin(GL_POINTS);
        glColor3f(1.0,1.0,1.0);
        glVertex2i(150,150);
        glEnd();
        glFlush();                                      //Zeichnen
  }

void maus(int button, int state, int x, int y)//Mausereignis verarbeiten
  {
        switch(button)
        {
                case GLUT_LEFT_BUTTON:
                        if(state==GLUT_DOWN)//Taste gedrückt
                                glutSetWindowTitle("Maustaste gedrückt!"); //Titel des Fenster ändern
                        else
                                glutSetWindowTitle("Hallo Welt!"); //Titel des Fenster ändern
                        break;
                case GLUT_MIDDLE_BUTTON:
                        exit(0);
                        break;
                case GLUT_RIGHT_BUTTON:
                        glutSetWindowTitle("Rechte Maustaste gedrückt!");
                        break;
        }
  }
```

```
void tastatur(unsigned char key, int x, int y)//Tastaturereignis verarbeiten
  {
        switch(key)
        {
                case 27://ESC-Taste
                        exit(0);//Programm beenden
                        break;
                case 'e':
                case 'E':
                        exit(0);
                        break;
        }
  }
```

9. Timer

Timer dienen dazu, in einem Programm ein, bestimmtes Ereignis zu einer bestimmten Zeit zu starten. Beispielsweise eine Bewegung. Das erzeugen eines Timers funktioniert änlich wie die Reaktion auf Tastatur oder Maus. Jedoch muss eine Zeit (in Millisekunden) angegeben werden, nach der der Timer ein Ereignis startet und es muss eine Nummer für dem Timer angegeben werden. Dadurch ist es möglich mit mehreren Timern gleichzeitig zu arbeiten.

Und so sieht die GLUT-Timer-Funktion aus:
glutTimerFunc(int zeit , Zeiger auf Funktion, int nr);

Die Funktion auf die der Zeiger gerichtet ist braucht fogenden Prototyp:
void tname(int value);

Um einen Timer zu erzeugen der nach einer Sekunde ausgelöst wird, muss das ganze so aussehen:

```
. . .
glutTimerFunc(1000 , tname, 1);
. . .

void tname(int value)
{
. . ..
//mach was . . ..
}
```

Der Timer lösst nur einmal aus. Soll er ein Ereignis regelmässig auslössen so muss er immer wieder neu gestartet werden.