

Tutorial 1: Mit glut ein Fenster oeffnen und dann mit OpenGL ein Rechteck zeichnen

Wer sich mit 3D-Grafik beschäftigen möchte, sprich sich mit der Programmierung auseinandersetzen will, steht vor einer Wahl:

- Entweder D3D oder OpenGL

In diesem Tutorial möchte ich mich mit OpenGL beschäftigen. Deswegen vorweg einige Informationen zu OpenGL:

- 1.) OpenGL liefert vorab keine 3D-Engine, man muss sich schon selbst die Mühe machen. Die OpenGL-API stellt einen die Funktionalität zur Verfügung, die Mühe des Codens liegt dann in der Hand des Programmierers.
- 2.) OpenGL ist im Gegensatz zu D3d (die 3D-API der Microsoft-Jungs) nicht objektorientiert, das kann man als Nachteil oder aber als Vorteil sehen, je nach Geschmack. Natürlich kann man die Funktionalität selbst in die entsprechenden Objekte kapseln, allerdings dazu in den weiteren Tuts mehr.
- 3.) OpenGL gibt es anders als D3D für fast alle OS (z.B. Windows, Linux, Solaris usw...).

Nun denn, genug der Hintergründe, erster Schritt sollte es sein, alle benötigten Header-Dateien und Librarys von der Seite www.opengl.org herunterladen. Des weiteren brauchen wir die nötigen GLUT-Libs sowie deren Header. Diese findet man unter:

<http://www.reality.sgi.com/opengl/glut3/glut3.html>

Um plattformunabhängig Fenster mit zugehörigen Funktionen erzeugen und verwalten zu können, muss die GLUT-Lib genutzt werden.

Für Windows:

```
#include <windows.h>
#include <gl/glut.h>
```

Und für Linux:

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

Durch diese Header werden alle relevanten Prototypen der OpenGL- und GLUT-Libs definiert. Ausserdem werden dort alle relevanten Makro's definiert, die für den Gebrauch der OpenGL-API notwendig sind.

Die Funktion, in der das Rechteck in das Fenster gezeichnet wird, heisst `RenderScene(...)`. Dort wird zunächst mit der Funktion `glClear(...)` der Hintergrund des Fensters gelöscht. `GL_COLOR_BUFFER_BIT` gibt dabei die Art an, wie der Hintergrund des Fensters gelöscht werden soll: der Color und der Pixel-Buffer des Fensters soll gelöscht werden. Ein Buffer stellt dabei einen Speicherbereich dar, der von OpenGL verwaltet wird. Die Funktion `glClear(...)` löscht nun diesen Buffer, der den Inhalt des Fensters repräsentiert. Danach wird mit der Funktion `glColor3f(r, g, b)` eine Farbe zum Zeichnen definiert. Die entsprechenden Anteile in Rot, Grün und Blau ergeben zusammen die Zeichenfarbe (kein Anteil 0.0f, voller Anteil 1.0f). In diesem Fall ergibt das die Farbe rot.

Danach wird in dieser besagten Zeichenfarbe ein rot gefülltes Rechteck mit der

Funktion glRectF(float x1, float y1, float x2, float y2)

```
// Funktion zum Malen der Szene
void RenderScene(void) {
    // Loesche das Fenster mit der aktuell gesetzten
    // Loeschfarbe
    glClear(GL_COLOR_BUFFER_BIT);

    // Setze die aktuelle Zeichenfarbe zu rot
    // ala
    glColor3f(1.0f, 0.0f, 0.0f);

    // Zeichne ein Rechteck
    glRectf(100.0f, 150.0f, 150.0f, 100.0f);

    // Flush Zeichenoperationen
    glFlush();
}
```

Die Anweisung glFlush() gibt an, dass keine weitere Anweisungen zum Zeichnen mehr folgen.

In der Funktion SetupRC

```
// Setze Renderstate
void SetupRC(void)
{
    glClearColor(0.0f, 0.0f, 1.0f, 1.0f);
}
```

wird die Hintergrundfarbe festgelegt (in diesem Fall handelt es sich dabei um blau).

Wenn der Benutzer die Fenstergrösse variieren will, sollte trotzdem der Fensterinhalt eintspendend angepasst werden. Um dieses gewährleisten zu können, stellen wir die Funktion ChangeSize(...) zur Verfügung, In dieser wird dafür gesorgt, dass der Viewport auf die aktuellen Fensterweiten gelegt wird. Die Funktion glViewport(x1, y1, x2, y2) erledigt dieses. Der Viewport-Bereich liegt also genau im Fenster. Um die Seitenverhältnisse des gefüllten Rechteckes beibehalten zu können, skaliert das Programm in der Funktion ChangeSize das Rechteck so, dass die Seitenverhältnisse beibehalten werden.

```
void ChangeSize(GLsizei w, GLsizei h)
{
    // Es darf keine Division durch Null durchgeführt werden
    if (h==0)
        h = 1;

    // Setze den Viewport auf die aktuelle Dimension des Fensters
    glViewport(0, 0, w, h);

    // Wähle die Projektionsmatrix aus
```

```

glMatrixMode(GL_PROJECTION);

// Hilfsmatrix auf Null setzen (wird zur Berechnung von Translationen
// und Rotationen gebraucht
glLoadIdentity();

// Wenn Breite kleiner der Höhe, dann skaliere Breite
// ansonsten die Höhe
if (w<=h)
    glOrtho(0.0f, 250.f, 0.0f, 250.f * h/w, 1.0, -1.0);
else
    glOrtho(0.0f, 250.f *w/h, 0.0f, 250.f, 1.0, -1.0);

// Alle Transformationen werden auf das Tragwerk angewandt
// (Transformations-Matrix)
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}

```

Zunächst wird einer Division by zero vorgebeugt. Dann wird der Viewport auf die aktuelle Fensterposition gesetzt. Je nach Abmasse des Fensters wird nun der Fensterinhalt auf die Breite oder die Höhe skaliert. Danach wird mit `glMatrixMode(GL_MODELVIEW)` die Modelview-Matrix ausgewählt und auf null gesetzt.

Abschliessend muss GLUT noch wissen, welche Programmfunktionen wo ausgeführt werden soll. Zunächst wird mit `glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB)` GLUT gesagt, dass nur ein Fenster genutzt wird, in dem die Farben über Farbanteile (Rot, Grün und Blau) festgelegt werden. Dann öffnet die Funktion `glutCreateWindow(char *Name)` ein GLUT-Fenster, welches in diesem Fall den Namen "First Steps" erhält. Über die Funktion `glutDisplayFunc(void (func*) (void))` wird der Pointer auf die Callback-Funktion übergeben, die den Zeichen-Prozess übernimmt. Die Funktion `glutReshapeFunc(void (*func))` übergibt den Pointer auf die Callback-Funktion, die das Neuzeichnen des Fensters übernimmt, wenn dessen Grösse verändert wurde. `RenderScene` wird über der `glutDisplayFunc(void (*func) (void))` als Render-Funktion festgelegt. Die Funktion `glutReshapeFunc(...)` übergibt die Funktion `ChangeSize` zur Änderung der Fenstergrösse. Nun muss nur noch der `RenderContext` gesetzt werden und die Hauptschleife gestartet werden (`glutMainLoop(void)`). Compiliert man den Source, so wird von GLUT ein Fenster geöffnet, in dem ein Rechteck gezeichnet wird.

```

void main(void) {
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("First Steps");
    glutDisplayFunc(RenderScene);
    glutReshapeFunc(ChangeSize);
}

```

```
SetupRC();  
    glutMainLoop();  
}
```

Im nächsten Tutorial werde ich dann ein 3D-Objekt erzeugen, dass translatiert und rotiert wird. Bis denne Kimmi

Fragen, Korrekturen?

hp : <http://www.sir-kimmi.de>

email: sir-kimmi@gmx.de