



Open GL III – Material, Texturen und Transparenz

Christoph Bichlmeier

22 November 2006

chair for computer aided medical procedures

department of computer science | technische universität münchen



Material

- Mit dem Befehl kann man Materialeigenschaften von Objekten ändern

```
GLfloat material_diffuse = { 1, 0, 0, 1 };
GLfloat material_specular = { 1, 1, 1, 1 };
GLfloat material_shininess = { 100 };
glMaterialfv(GL_FRONT, GL_DIFFUSE, material_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, material_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, material_shininess);
```

- GL_DIFFUSE steht für den *matten* Farbanteil
- GL_SPECULAR steht für die Farbe an den glänzend reflektierenden Stellen
- GL_SHININESS setzt die Wert, wie stark das Material glänzt
- GL_AMBIENT setzt die Grundfarbe, die immer zu sehen ist
- Einige Werte für Materialien: <http://devernay.free.fr/cours/opengl/materials.html>
- Oder

```
glColorMaterial(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE )
glEnable(GL_COLOR_MATERIAL)
```

Beleuchtungsmodell verwendet hier Werte aus glColor(); **Schneller!**



Texturen

- Mächtiges Instrument um das Aussehen von Oberflächen zu beeinflussen → Texturemapping
- Pixelbild auf ein Oberfläche gelegt
- Vorgehensweise
Initialisierung:

- Bild in den Speicher laden
- Textur generieren

Benutzung:

- Texturemapping einschalten
`glEnable(GL_TEXTURE_2D)`
- Textur auswählen mit
`glBindTexture(GL_TEXTURE_2D, GLuint* texture)`
wird die Textur `texture` benutzt
- Evtl. Texturkoordinaten berechnen

```
image=LoadBMP("logo.bmp"); //eigene Ladefunktion
glGenTextures(1, &texture[0]);
glBindTexture(GL_TEXTURE_2D, texture[0]);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, sizeX,
sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE, image);
glEnable(GL_TEXTURE_2D);
```

Größe der Textur muss 2^n sein
(n ganzzahlig und ≤ 10)

Kann auch
`GL_LUMINANCE` (Graustufen),
`GL_RGBA` (mit Transparenz) oder
`GL_BGR_EXT` (rot und blau vertauscht)
etc. sein



Beispielcode

```
/**init texture***/
image=LoadBMP("logo.bmp"); //eigene Ladefunktion
glEnable(GL_TEXTURE_2D);
glGenTextures(1, &texture[0]);
glBindTexture(GL_TEXTURE_2D, texture[0]);
// Enable Linear Filtering
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, sizeX, sizeY, 0, GL_RGB,
GL_UNSIGNED_BYTE, image);

/**draw textured object***/
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texture[0]);
glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f);
    glVertex3f(-1.0f, -1.0f, 1.0f);
    glTexCoord2f(1.0f, 0.0f);
    glVertex3f( 1.0f, -1.0f, 1.0f);
    glTexCoord2f(1.0f, 1.0f);
    glVertex3f( 1.0f,  1.0f, 1.0f);
    glTexCoord2f(0.0f, 1.0f);
    glVertex3f(-1.0f,  1.0f, 1.0f);
glEnd(GL_QUADS);
```



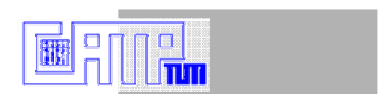
Überblenden [Blending]

- OpenGL kann mit α Werten für die Deckfähigkeit der Farben umgehen: $\alpha=0$ ist komplett durchsichtig, $\alpha=1$ ist undurchsichtig und Werte dazwischen sind halbtransparent
- Mit `glEnable(GL_BLEND)` kann man das Blending einschalten
- Mit `glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)` bestimmt man das Mischungsverhältnis

$$(R_s A_s + R_d (1 - A_s), G_s A_s + G_d (1 - A_s), B_s A_s + B_d (1 - A_s), A_s A_s + A_d (1 - A_s))$$

- Damit es geht, müssen die Farben in RGBA angegeben werden

- halbdurchsichtiges weiß:
`glColor4f(1.0f, 1.0f, 1.0f, 0.6f)`
- auch Texturen kann man überblenden, wenn man die Textur mit `GL_RGBA` generiert hat





Überblenden [blending]

- **Erst alle undurchsichtigen Objekte zeichnen mit**
`glDepthFunc(GL_LEQUAL);`
`glEnable(GL_DEPTH_TEST);`
- ***Blending* aktivieren und BlendFunktion setzen**
- `glEnable(GL_BLEND);`
- `glBlendFunc(GL_SRC_ALPHA, GL_ONE);`
`glColor4f(1,1,0.5,0.5); //halb transparent`
`/* zeichne Objekte von vorne nach hinten ... */`
- **Blending Effekte deaktivieren**
`glDisable(GL_BLEND);`



Hilfe, mein Bildaufbau ist zu langsam...

- Ohne Bildqualitätsänderungen
 - Möglichst nicht `glBegin(GL_POLYGON)` benutzen, man kann das Selbe mit `glBegin(GL_TRIANGLE_STRIP)` erreichen
 - Aufruflisten [displaylists] für große Objekte verwenden
 - Mipmaps anstatt normaler Texturen verwenden
 - Backface culling (unnötige Rückseitenberechnung vermeiden)
 - `glColorMaterial` statt `glMaterial` verwenden
- Mit Bildqualitätsänderung
 - Weniger (positionierte) Lichter verwenden
 - Shading auf `GL_FLAT` ändern
 - Weniger Polygone/Dreiecke anzeigen
 - Geringere Auflösung wählen