



Material, Texturen und Transparenz

Christoph Bichlmeier

22 May 2007

chair for computer aided medical procedures
department of computer science | technische universität münchen



Material

- Materialieigenschaften 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

```
image=LoadBMP("logo.bmp"); //eigene Ladefunktion
glEnable(GL_TEXTURE_2D);
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);
```

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***/
CAMP::Bitmap bitmptex;
if (!bitmptex.load(filename[0], false)){
    std::cout << "mytexture konnte nicht geladen werden." << std::endl;
}else{
    glEnable(GL_TEXTURE_2D);
    glGenTextures (1, &texture[0]);
    glBindTexture (GL_TEXTURE_2D, texture[0]);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

    glTexImage2D(GL_TEXTURE_2D, 0, 3, bitmptex.getWidth(),
bitmptex.getHeight(), 0, GL_BGR_EXT, GL_UNSIGNED_BYTE, bitmptex.getData());

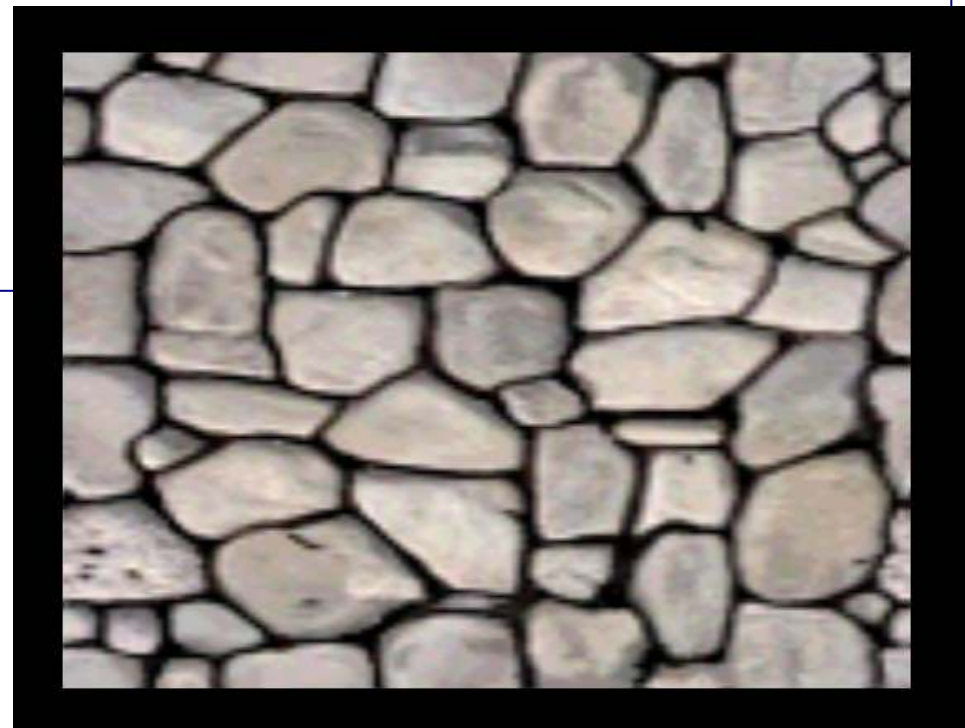
    gluBuild2DMipmaps(GL_TEXTURE_2D, 3, bitmptex.getWidth(),
bitmptex.getHeight(), GL_BGR_EXT, GL_UNSIGNED_BYTE, bitmptex.getData());

    glDisable(GL_TEXTURE_2D);
}
```



Beispielcode

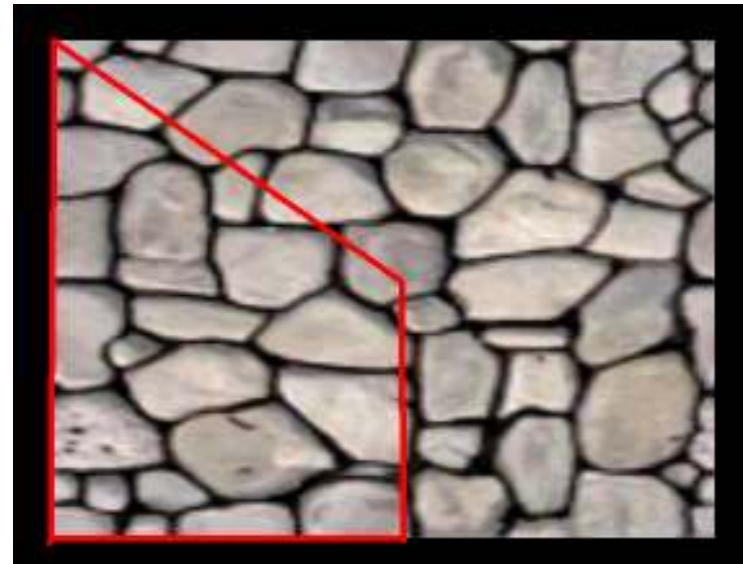
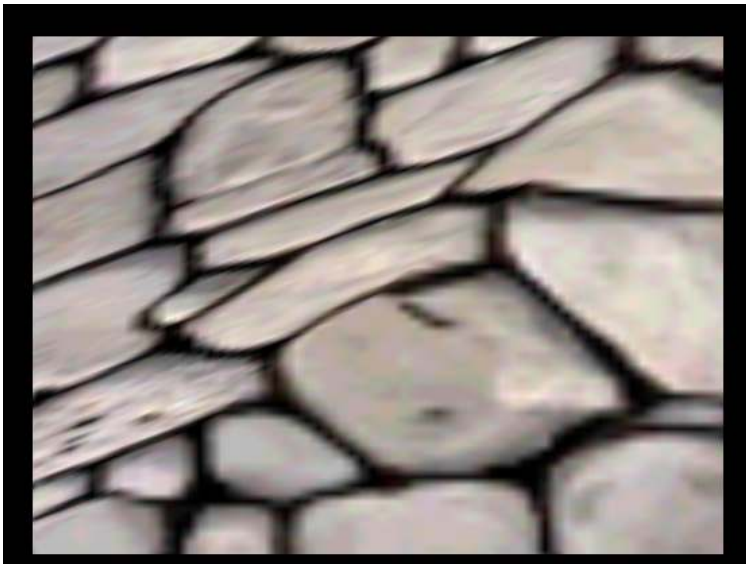
```
/**draw textured object***/
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texture[0]);
glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f);
    glVertex3f(-0.8f, -0.8f, 0);
    glTexCoord2f(1.0f, 0.0f);
    glVertex3f( 0.8f, -0.8f, 0);
    glTexCoord2f(1.0f, 1.0f);
    glVertex3f( 0.8f,  0.8f, 0);
    glTexCoord2f(0.0f, 1.0f);
    glVertex3f(-0.8f,  1.0f, 0);
glEnd();
glDisable(GL_TEXTURE_2D);
```





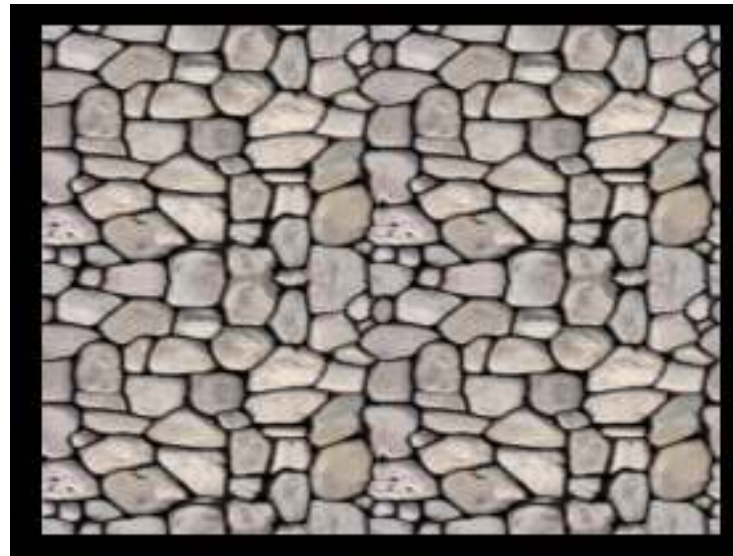
```

glBegin(GL_QUADS);
glTexCoord2f(0,0); glVertex3f( -0.8f, -0.8f, 0.0f);           // unten links
glTexCoord2f(0,0.5); glVertex3f( 0.8f, -0.8f, 0.0f);         // Unten Rechts
glTexCoord2f(0.5,0.5); glVertex3f( 0.8f, 0.8f, 0.0f);         // oben Rechts
glTexCoord2f(1,0); glVertex3f( -0.8f, 0.8f, 0.0f);           // oben links
glEnd();
    
```





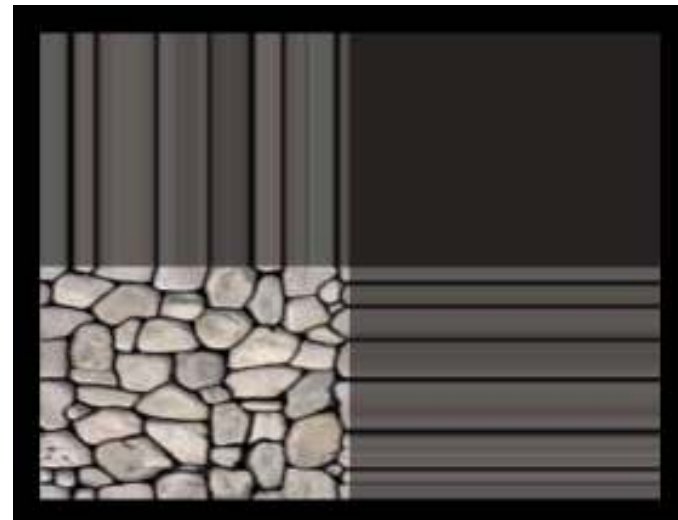
```
glEnable(GL_TEXTURE_2D);  
glBindTexture (GL_TEXTURE_2D, texture[0]);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);  
glBegin(GL_QUADS);  
glTexCoord2f(0,0); glVertex3f( -0.8f, -0.8f, 0.0f); // unten links  
glTexCoord2f(0,2); glVertex3f( 0.8f, -0.8f, 0.0f); // Unten Rechts  
glTexCoord2f(2,2); glVertex3f( 0.8f, 0.8f, 0.0f); // oben Rechts  
glTexCoord2f(2,0); glVertex3f( -0.8f, 0.8f, 0.0f); // oben links  
glEnd();  
glDisable(GL_TEXTURE_2D);
```





glTexParameter

```
glEnable(GL_TEXTURE_2D);  
glBindTexture(GL_TEXTURE_2D, texture[0]);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);  
glBegin(GL_QUADS);  
glTexCoord2f(0,0); glVertex3f(-0.8f, -0.8f, 0.0f); // unten links  
glTexCoord2f(0,2); glVertex3f(0.8f, -0.8f, 0.0f); // Unten Rechts  
glTexCoord2f(2,2); glVertex3f(0.8f, 0.8f, 0.0f); // oben Rechts  
glTexCoord2f(2,0); glVertex3f(-0.8f, 0.8f, 0.0f); // oben links  
glEnd();  
glDisable(GL_TEXTURE_2D);
```





Überblenden [Blending]

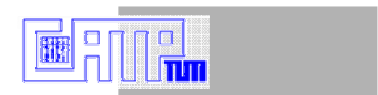
- OpenGL kann mit α Werten für die Deckfähigkeit der Farben umgehen: $\alpha=0$ ist komplett durchsichtig, $\alpha=1$ ist undurchsichtig und Werte dazwischen sind halbtransparent
- Mit `glEnable(GL_BLEND)` kann man das Blending einschalten
- Mit `glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)` bestimmt 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, 0.5f)`
- auch Texturen kann man überblenden, wenn man die Textur mit `GL_RGBA` generiert hat



Überblenden [blending]

Blending aktivieren und BlendFunktion setzen

```
glEnable(GL_BLEND);  
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);  
glColor4f(1,1,0.5,0.5); //halb transparent  
/* Tipp: sortiere Objekte von hinten nach vorne */
```

Blending Effekte deaktivieren

```
glDisable(GL_BLEND);
```



Visible Korean Human



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