Farbmischung

- Farbmischung zwischen halbdurchlässigen Oberflächen und dem Hintergrund (Anwendung von Alpha-Werten)

  glEnable, **glDisable** mit GL_BLEND

  glBlendEquation mit GL_FUNC_ADD (Standardwert), GL_FUNC_SUBSTRACT, GL_FUNC_REVERSE_SUBSTRACT, GL_MIN, GL_MAX

  glBlendFunc(int *sFactor*, int *dFactor*), Tabelle mit Argumenten: `http://www.glprogramming.com/red/chapter06.html`

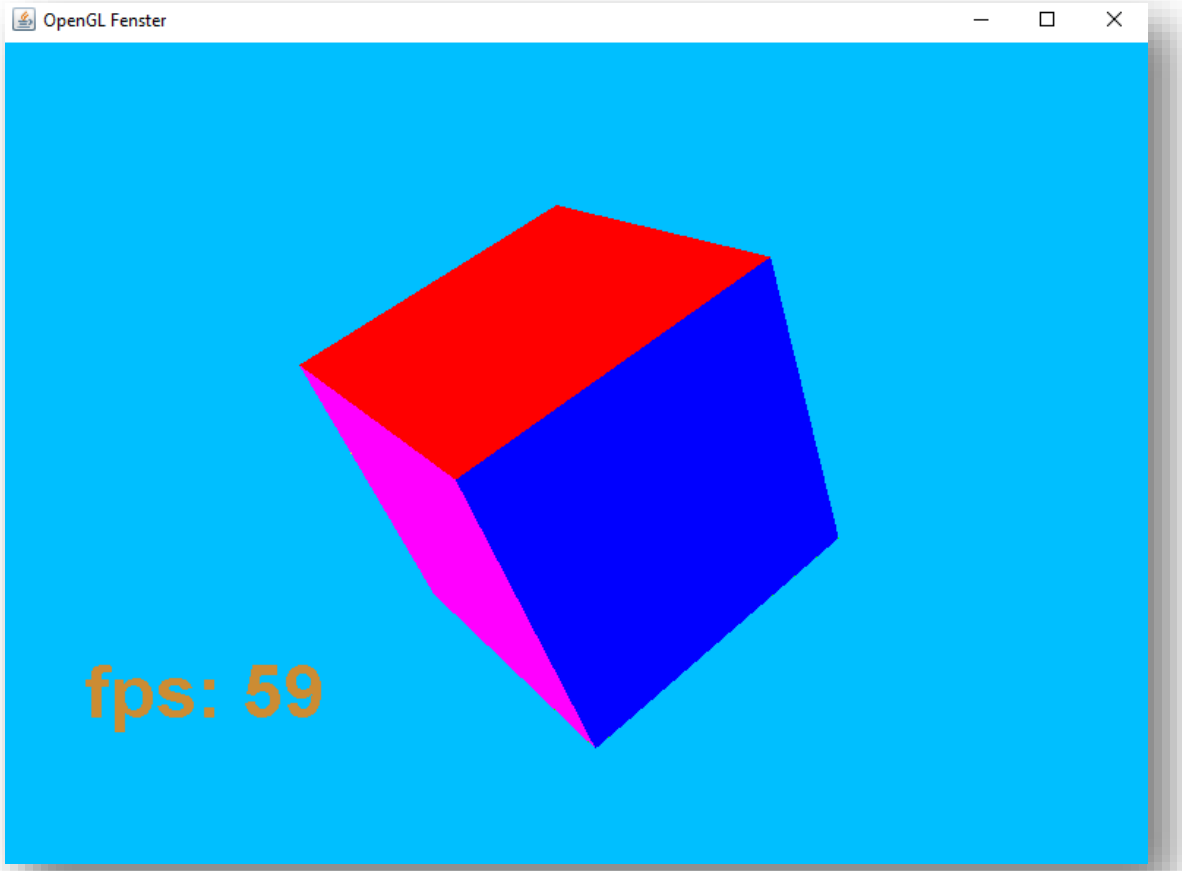
Alpha-Wert (Glas) = 0.7, (Teapot) = 1 (Milchglass, lässt 30% Licht durch)
gl.glBlendFunc(GL.GL_SRC_ALPHA, GL.GL_ONE_MINUS_SRC_ALPHA);

Alpha-Wert (Alles) = 1.0
gl.glBlendFunc(GL.GL_ONE, GL.GL_ONE);

Alpha-Wert (Alles) = 1.0
gl.glBlendEquation(GL.GL_FUNC_REVERSE_SUBTRACT);
gl.glBlendFunc(GL.GL_ONE, GL.GL_ONE);

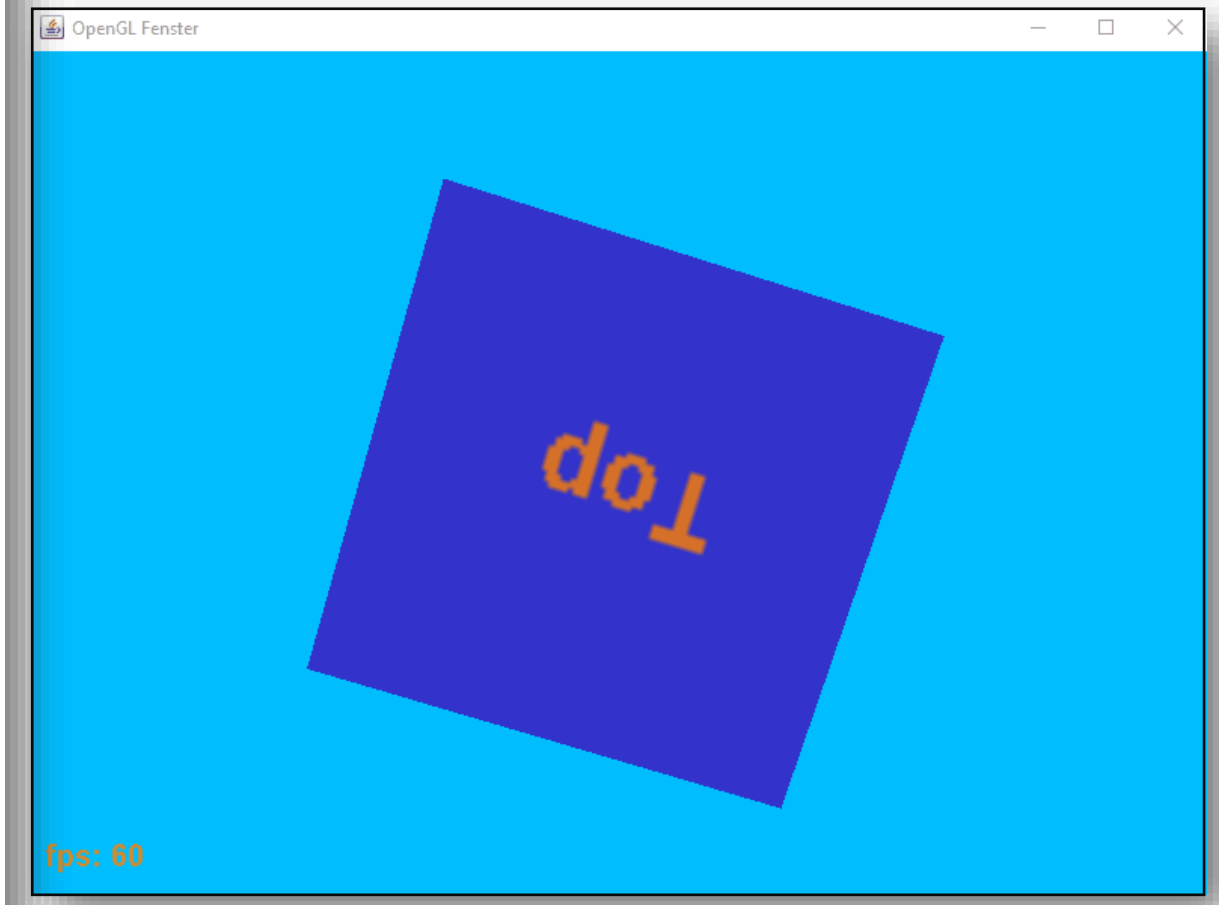
Einfaches Textrendering in OpenGL

```
...
import com.jogamp.opengl.util.FPSAnimator;
import com.jogamp.opengl.util.awt.TextRenderer;
...

public class Main implements GLEventListener {

        ...
        int fps;

        TextRenderer renderer;

        ...
    public static void main(String[] args) {

        ...
        final FPSAnimator animator = new FPSAnimator( glcanvas, 60,true );

        animator.start();

        ...
    }


    @Override
    public void init(GLAutoDrawable drawable) {

        ...
        renderer = new TextRenderer(new Font("SansSerif", Font.BOLD, 50));

        ...
    }


    @Override
    public void display(GLAutoDrawable drawable) {

            ...
            renderer.beginRendering(drawable.getSurfaceWidth(), drawable.getSurfaceHeight());
                renderer.setColor(1.0f, 0.5f, 0.0f, 0.8f);
                // 30 und 100 stehen für X- und Y-Koordinaten des Schaufensters (Viewport)
                renderer.draw("  fps: " + fps, 30, 100);
            renderer.endRendering();

            ...
    }

    ...
```



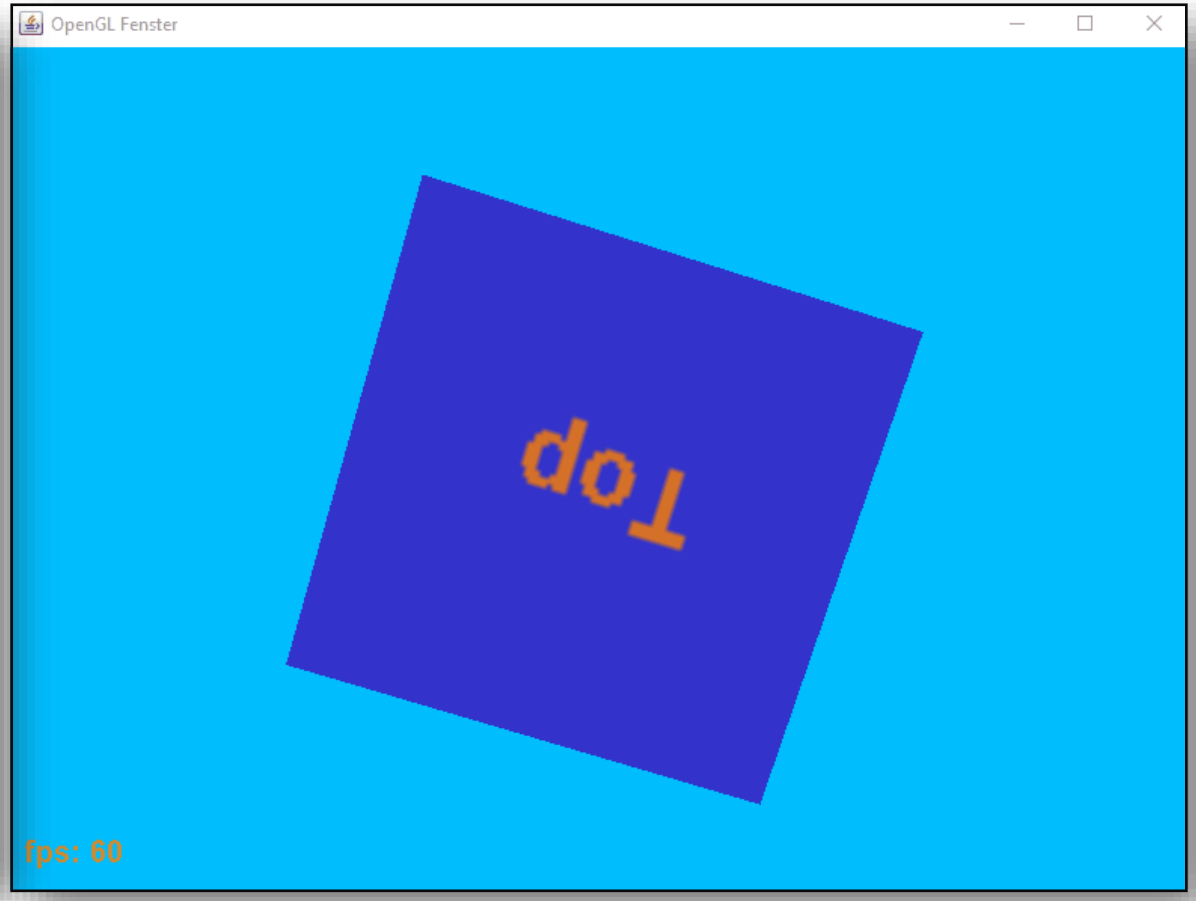
Zeichnen auf Objektflächen (1)

```
...
import java.awt.Font;
import java.awt.geom.Rectangle2D;
import com.jogamp.opengl.util.awt.TextRenderer;
...

public class Main implements GLEventListener {
...
        private TextRenderer renderer;
        private float textScaleFactor;

    public static void main(String[] args) {
...
    }

    @Override
    public void init(GLAutoDrawable drawable) {

...
        renderer = new TextRenderer(new Font("SansSerif", Font.BOLD, 20));
...
        Rectangle2D bounds = renderer.getBounds("Bottom");
        float w = (float) bounds.getWidth();
        textScaleFactor = 0.01f;
    }

    @Override
    public void display(GLAutoDrawable drawable) {
...
        drawFace(gl, 1.0f, 0.2f, 0.2f, 0.8f, "Top"); ...
        drawFace(gl, 1.0f, 0.8f, 0.2f, 0.2f, "Front"); ...
        drawFace(gl, 1.0f, 0.2f, 0.8f, 0.2f, "Right"); ...
        drawFace(gl, 1.0f, 0.8f, 0.8f, 0.2f, "Back"); ...
        drawFace(gl, 1.0f, 0.2f, 0.8f, 0.8f, "Left"); ...
        drawFace(gl, 1.0f, 0.8f, 0.2f, 0.8f, "Bottom");
...
            renderer.beginRendering(drawable.getSurfaceWidth(), drawable.getSurfaceHeight());
                renderer.setColor(1.0f, 0.5f, 0.0f, 0.8f);
                renderer.draw("  fps: " + fps, 0, 20);
            renderer.endRendering();

    }
```



Zeichnen auf Objektflächen (2) – drawFace() //benutzerdefiniert

```
...

private void drawFace(GL2 gl, float faceSize, float r, float g, float b, String text) {

    float halfFaceSize = faceSize / 2;

    gl.glColor3f(r, g, b);
    gl.glBegin(GL2.GL_QUADS);
    gl.glVertex3f(-halfFaceSize, -halfFaceSize, halfFaceSize);
    gl.glVertex3f( halfFaceSize, -halfFaceSize, halfFaceSize);
    gl.glVertex3f( halfFaceSize,  halfFaceSize, halfFaceSize);
    gl.glVertex3f(-halfFaceSize,  halfFaceSize, halfFaceSize);
    gl.glEnd();

    renderer.begin3DRendering();
        gl.glDisable(GL2.GL_DEPTH_TEST);

        Rectangle2D bounds = renderer.getBounds(text);
        float w = (float) bounds.getWidth();
        float h = (float) bounds.getHeight();
        renderer.draw3D(text,
                  w / -2.0f * textScaleFactor,
                  h / -2.0f * textScaleFactor,
                  halfFaceSize,
                  textScaleFactor);
    renderer.end3DRendering();
}
```

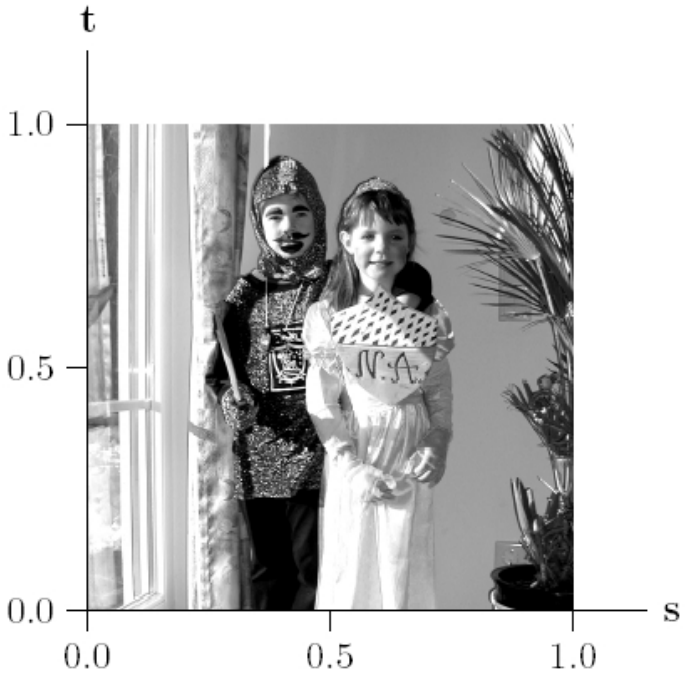


Texture-Mapping

- Eingesetzt, um die Oberfläche von Objekten mit einer Struktur zu überziehen $\rightarrow$ realist. Bilder
- Texturkoordinaten spezifizieren wie die Textur auf die Geometrie abgebildet wird

  glTexCoord*

```
gl.glBegin(GL.GL_QUADS);
   gl.glTexCoord2f(0.0f, 0.0f);
   gl.glVertex2f(-5.0f, -5.0f);
   gl.glTexCoord2f(1.0f, 0.0f);
   gl.glVertex2f(5.0f, -5.0f);
   gl.glTexCoord2f(1.0f, 1.0f);
   gl.glVertex2f(5.0f, 5.0f);
   gl.glTexCoord2f(0.0f, 1.0f);
   gl.glVertex2f(-5.0f, 5.0f);
gl.glEnd();
```



(Nischwitz *et al.*, 2011)

Erzeugen eines Textur-Objektes - Beispiel

```
...
Texture textureTree;

public void init(GLAutoDrawable drawable) {
    ...
    textureTree = TextureIO.newTexture(new File("images/poplar.png"); }
}
public void display(GLAutoDrawable drawable) {
    ...
    drawTree(gl);
}
void drawTree(GL gl) {
    textureTree.enable();
    textureTree.bind();
    ...
    gl.glBegin(GL.GL_QUADS);
        gl.glTexCoord2f(0.0f, 0.0f);
        gl.glVertex2f(-0.45f, 0.0f);
        gl.glTexCoord2f(1.0f, 0.0f);
        gl.glVertex2f(0.45f, 0.0f);
        gl.glTexCoord2f(1.0f, 1.0f);
        gl.glVertex2f(0.45f, 1.3f);
        gl.glTexCoord2f(0.0f, 1.0f);
        gl.glVertex2f(-0.45f, 1.3f);
    gl.glEnd();

    textureTree.disable();
}
```

- Für optimale Ergebnisse,
  Breite und Höhe als Zweierpotenz

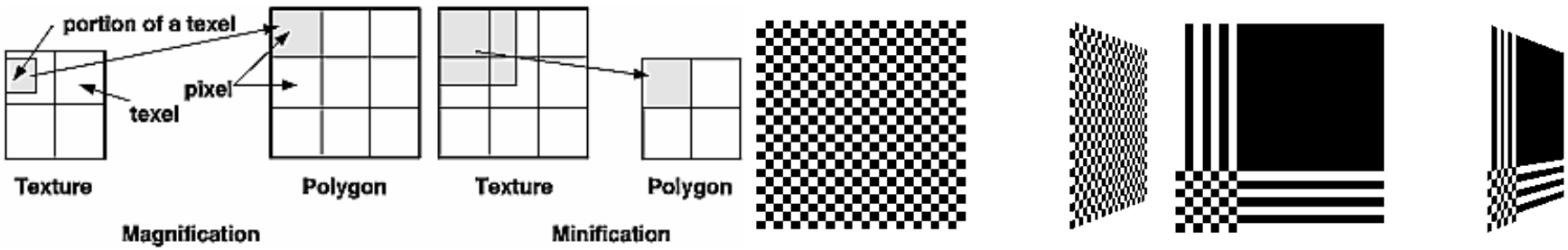
Textur-Filter

- Vergrößerung, Verkleinerung

```
gl.glTexParameteri(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MAG_FILTER, GL.GL_LINEAR);
gl.glTexParameteri(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MIN_FILTER, GL.GL_LINEAR);
```

- Texture Wraps

```
gl.glTexParameteri(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_WRAP_S, GL.GL_REPEAT);
gl.glTexParameteri(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_WRAP_T, GL.GL_REPEAT);

gl.glTexParameteri(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_WRAP_S, GL.GL_CLAMP);
gl.glTexParameteri(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_WRAP_T, GL.GL_CLAMP);
```



www.glprogramming.com/red/chapter09.html

Nebel

- Atmospherische Effekte $\rightarrow$ mehr Realismus

glFog*

name	param	Bedeutung
GL_FOG_MODE	GL_EXP	exponentiell abnehmende Sichtbarkeit, d.h. konstante Nebeldichte (11.2).
	GL_EXP2	exponentiell quadratisch abnehmende Sichtbarkeit, d.h. zunehmende Nebeldichte (11.4).
	GL_LINEAR	linear abnehmende Sichtbarkeit, d.h. hyperbolisch zunehmende Nebeldichte (11.5).
GL_FOG_COLOR	$\mathbf{g_n} = (R_n, G_n, B_n, A_n)^T$	Zeiger auf eine Nebelfarbe (nur in der Vektor-Form des Befehls möglich)
GL_FOG_DENSITY	a	Nebeldichte bei exp- und exp2-Funktion
GL_FOG_START	z_{start}	Entfernung, bei der ein linear ansteigender Nebel beginnt
GL_FOG_END	z_{end}	Entfernung, ab der ausschließlich die Farbe des linear ansteigenden Nebels zu sehen ist
GL_FOG_COORDINATE_SOURCE	GL_FRAGMENT_DEPTH	Entfernung (z-Wert) des Fragments vom Augenpunkt, wie sie im z-Buffer steht.
	GL_FOG_COORDINATE	z-Wert des Fragments, linear interpoliert aus den mit `glFogCoordfv()` an den Vertices festgelegten z-Werten.

exp

exp2

linear

(Nischwitz *et al.*, 2011)

Transparenz, Nebel - Beispiel

```
// do not draw the transparent parts of the texture
gl.glEnable(GL.GL_BLEND);
gl.glBlendFunc(GL.GL_SRC_ALPHA, GL.GL_ONE_MINUS_SRC_ALPHA);

// perform alpha-test, draw only if alpha > 0
gl.glEnable(GL.GL_ALPHA_TEST);
gl.glAlphaFunc(GL.GL_GREATER, 0);

// draw tree
...

gl.glDisable(GL.GL_ALPHA_TEST);
gl.glDisable(GL.GL_BLEND);
```

```
float[] fogColor = {1.0f, 1.0f, 1.0f, 1.0f};

gl.glEnable(GL.GL_FOG);

gl.glFogi(GL.GL_FOG_MODE, GL.GL_EXP);
gl.glFogf(GL.GL_FOG_DENSITY, 0.05f);
gl.glFogfv(GL.GL_FOG_COLOR, fogColor, 0);
```