WiSe 21/22

GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Prof. Dr. Winfried Kurth
Alex Tavkhelidze

Praktikum Computergrafik

Folien zu #1

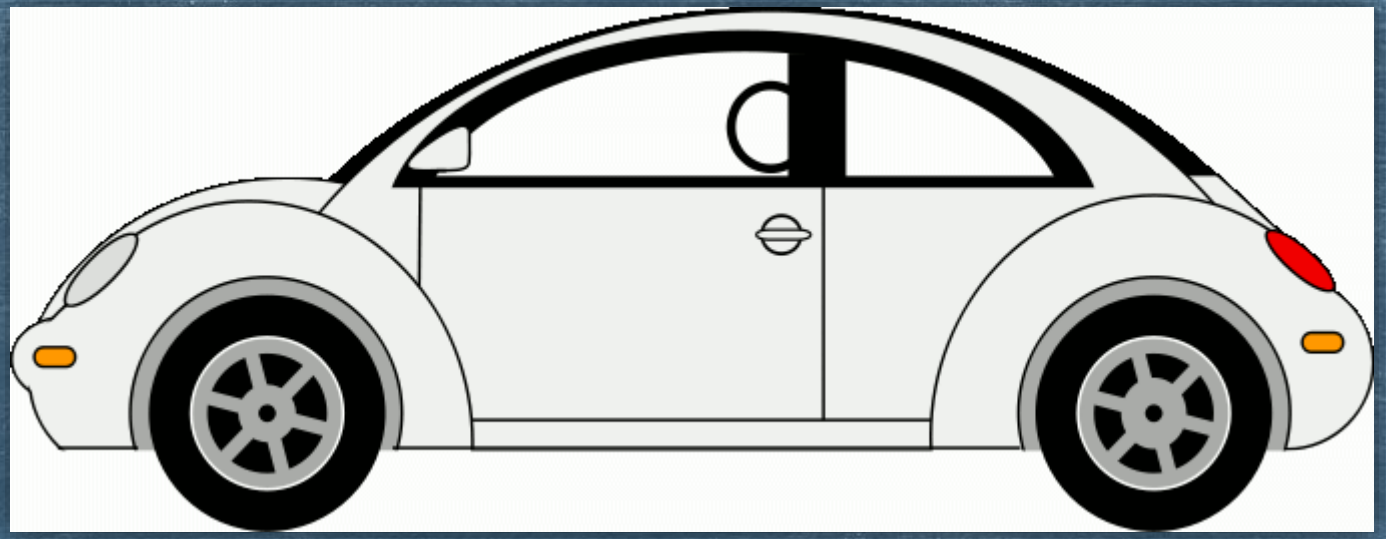
Einführung: grafische Primitive

Fenstereinstellungen

Exportieren in Eclipse

Weiteres: Objektbibliotheken

3D Objekte

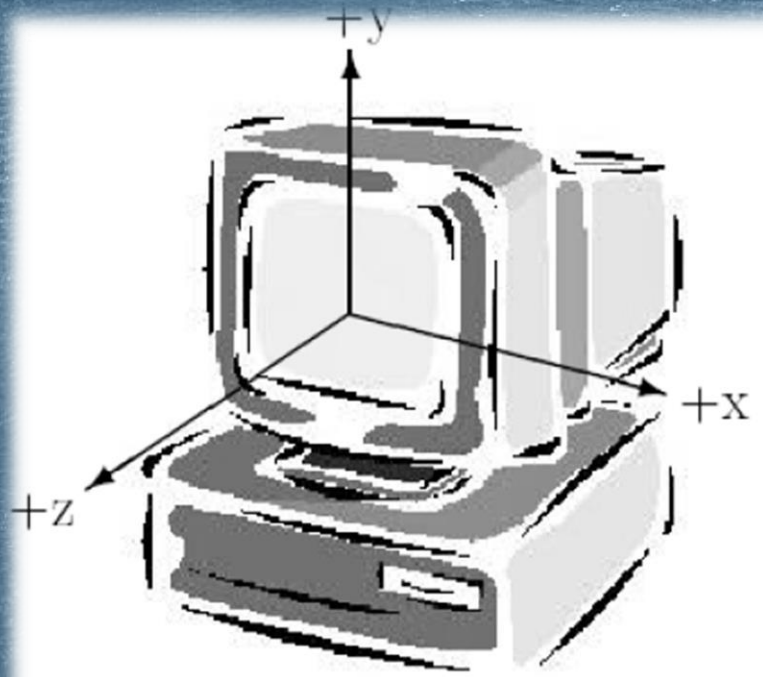


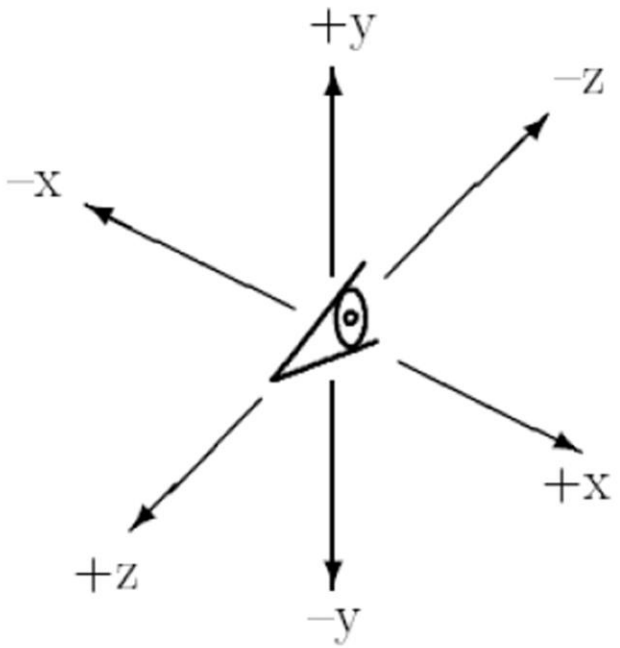
Grafische Primitive in OpenGL

- Mit einfachen Bausteinen erstellt man (hoch)komplexe Strukturen
- Die Bausteine nennt man Primitive in OpenGL

Originalquelle der Animation: https://upload.wikimedia.org/wikipedia/commons/b/b7/Beetle.gif

Koordinatensystem in OpenGL

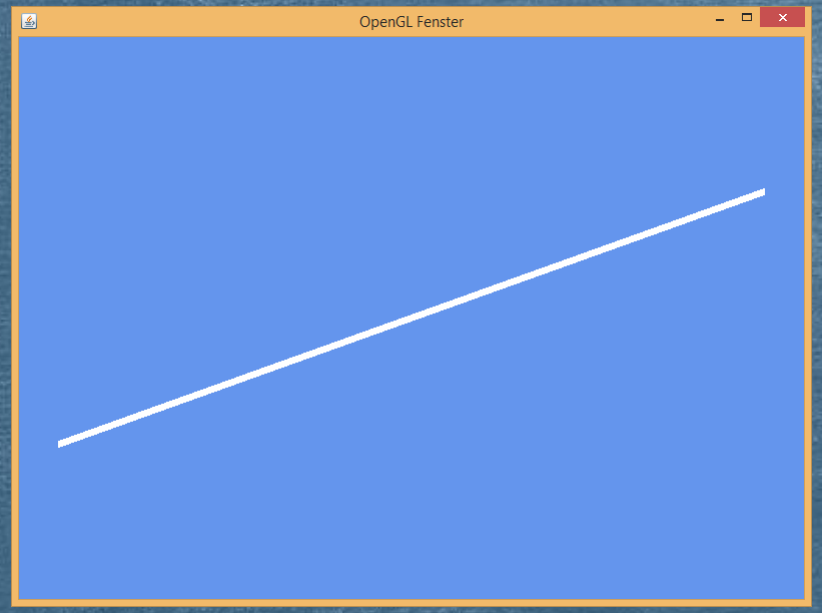




Bestimmung eines 3D-Knotens (Punktes) erfolgt durch den Befehl glVertex3f(float x, float y, float z)

(Nischwitz *et al.*, 2011)

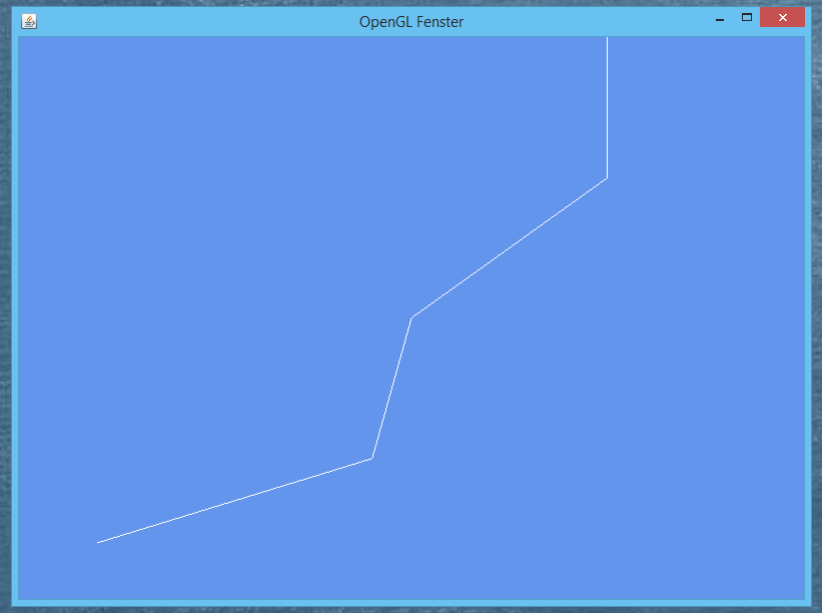
Liniensegmente




```
@Override
public void display(GLAutoDrawable drawable) {
    // TODO Auto-generated method stub

    GL2 gl = drawable.getGL().getGL2();
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);

        final float w = 7.0f;

        gl.glLineWidth(w);

        gl.glBegin(GL2.GL_LINES);

            gl.glVertex3f(-0.9f, -0.45f, 0.0f);
            gl.glVertex3f( 0.9f, 0.45f, 0.0f);

        gl.glEnd();

    gl.glFlush();

}
```

- Der Parameter GL2.GL_LINES wandelt zwei Knoten (Vertices) in ihre Verbindungstrecke um
- Die Linienbreite setzt man mit der Methode glLineWidth(float width)

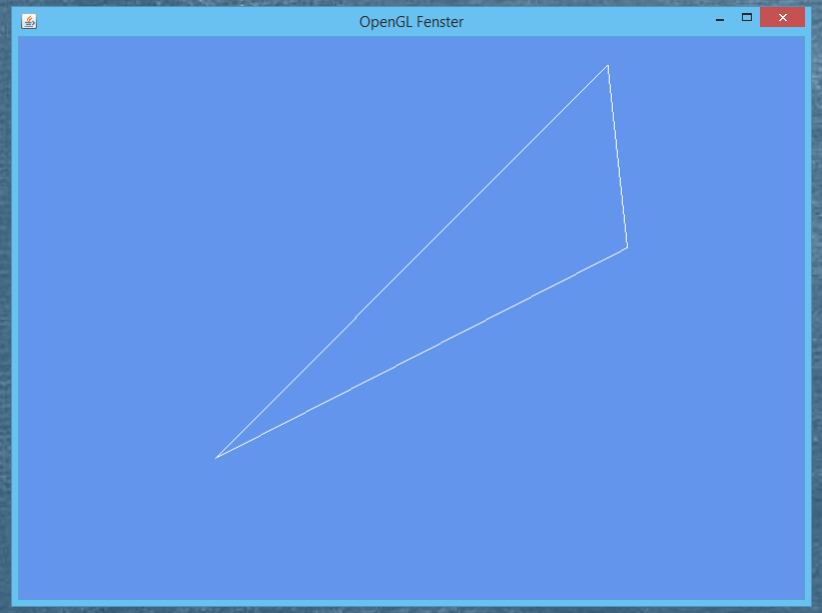
Linienzüge




```
@Override
public void display(GLAutoDrawable drawable) {
    // TODO Auto-generated method stub

    GL2 gl = drawable.getGL().getGL2();
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);

    gl.glBegin(GL2.GL_LINE_STRIP);

        gl.glVertex3f(-0.8f,-0.8f, 0.0f);
        gl.glVertex3f(-0.1f,-0.5f, 0.0f);
        gl.glVertex3f( 0.0f, 0.0f, 0.0f);
        gl.glVertex3f(0.5f, 0.5f, 0.0f);
        gl.glVertex3f(0.5f, 1.0f, 0.0f);

    gl.glEnd();

    gl.glFlush();

}
```

- Der Parameter GL2.GL_LINE_STRIP interpretiert die Liste der Vertices als Endpunkte eines Linienzuges

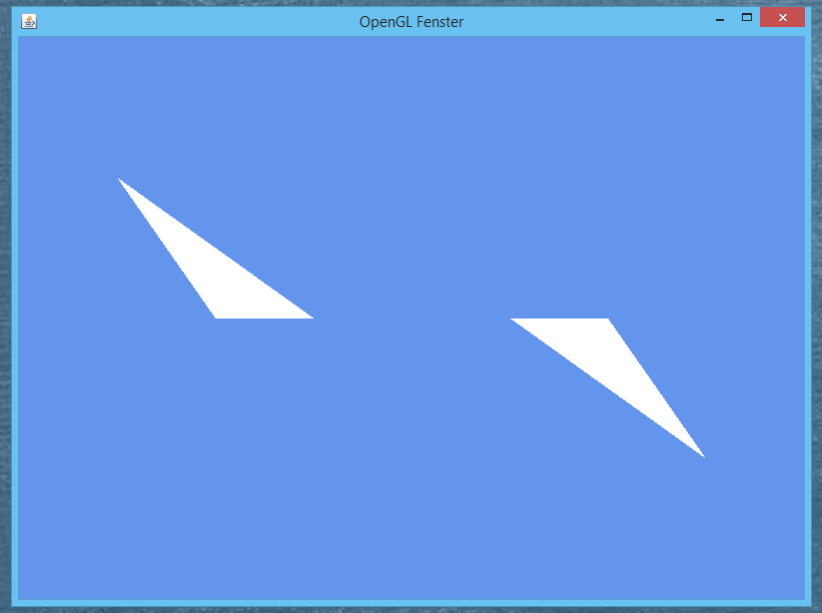
geschlossene Linienzüge



- Der Parameter GL2.GL_LINE_LOOP bildet aus der Liste der Vertices einen geschlossenen Linienzug

```
@Override
public void display(GLAutoDrawable drawable) {
    // TODO Auto-generated method stub

    GL2 gl = drawable.getGL().getGL2();
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);

    gl.glBegin(GL2.GL_LINE_LOOP);

        gl.glVertex3f(-0.5f,-0.5f, 0.0f);
        gl.glVertex3f( 0.55f, 0.25f, 0.0f);
        gl.glVertex3f( 0.5f, 0.9f, 0.0f);

    gl.glEnd();

    gl.glFlush();

}
```

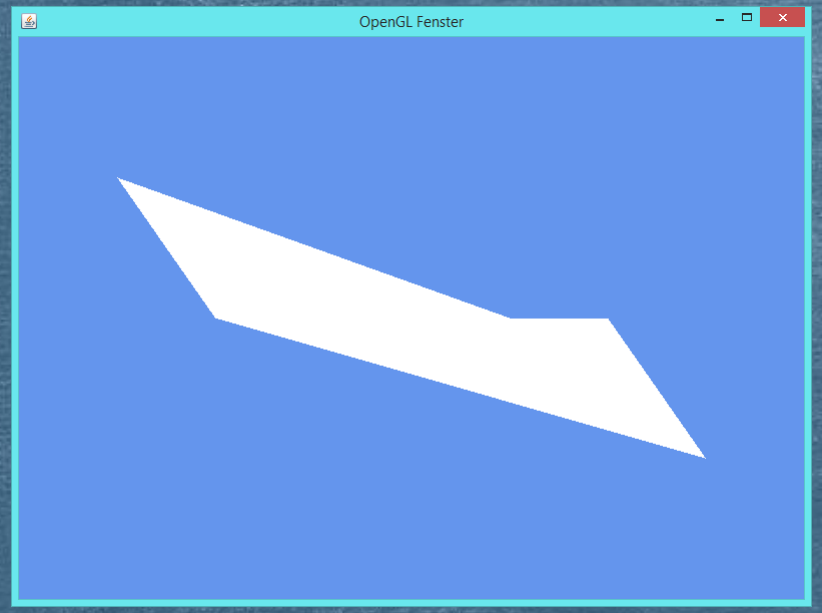
Dreiecke




```
@Override
public void display(GLAutoDrawable drawable) {
    // TODO Auto-generated method stub

    GL2 gl = drawable.getGL().getGL2();
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);

    gl.glBegin(GL2.GL_TRIANGLES);

        gl.glVertex3f(0.50f, 0.0f, 0.0f);
        gl.glVertex3f(0.75f, -0.50f, 0.0f);
        gl.glVertex3f(0.25f, 0.0f, 0.0f);

        gl.glVertex3f(-0.50f, 0.0f, 0.0f);
        gl.glVertex3f(-0.75f, 0.50f, 0.0f);
        gl.glVertex3f(-0.25f, 0.0f, 0.0f);

    gl.glEnd();

    gl.glFlush();

}
```

- Der Parameter GL2.GL_TRIANGLES fasst die Dreier von Punkten als Dreiecke zusammen

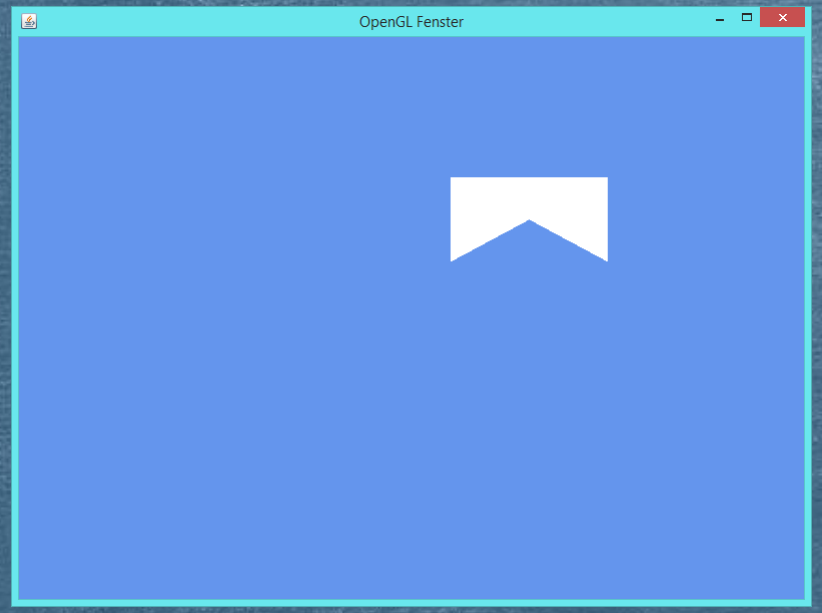
Dreiecksstreifen (1)




```
@Override
public void display(GLAutoDrawable drawable) {
    // TODO Auto-generated method stub

    GL2 gl = drawable.getGL().getGL2();
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);

    gl.glBegin(GL2.GL_TRIANGLE_STRIP);

        gl.glVertex3f( 0.50f,  0.0f,  0.0f);
        gl.glVertex3f( 0.75f,-0.5f,  0.0f);
        gl.glVertex3f( 0.25f,  0.0f,  0.0f);

        gl.glVertex3f(-0.50f,  0.0f,  0.0f);
        gl.glVertex3f(-0.75f,  0.5f,  0.0f);
        gl.glVertex3f(-0.25f,  0.0f,  0.0f);

    gl.glEnd();;

    gl.glFlush();

}
```

- Der Parameter GL2.GL_TRIANGLE_STRIP interpretiert die Liste der Vertices als Gruppe verbundener Dreiecke

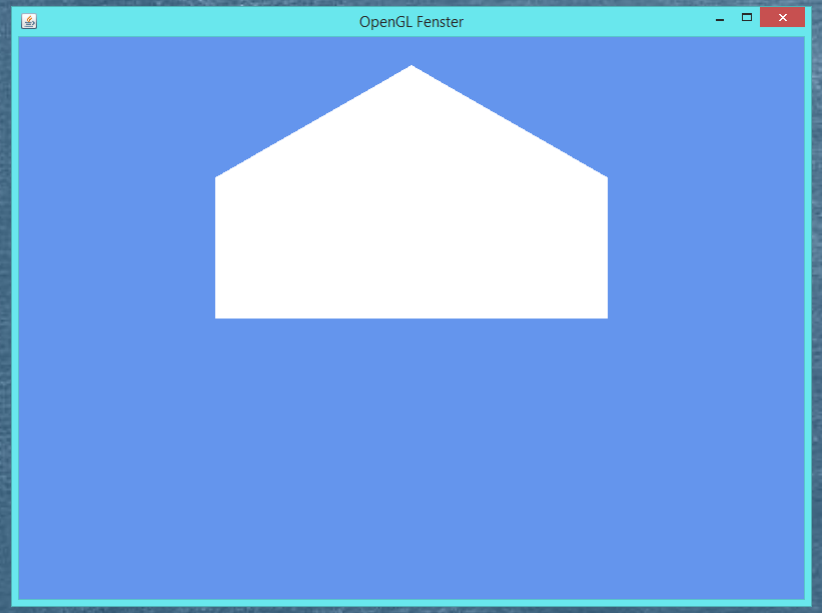
Dreiecksstreifen (2)




```
@Override
public void display(GLAutoDrawable drawable) {
    // TODO Auto-generated method stub

    GL2 gl = drawable.getGL().getGL2();
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);

    gl.glBegin(GL2.GL_TRIANGLE_STRIP);

        gl.glVertex3f( 0.1f, 0.2f, 0.0f);
        gl.glVertex3f( 0.1f, 0.5f, 0.0f);
        gl.glVertex3f( 0.5f, 0.5f, 0.0f);
        gl.glVertex3f( 0.5f, 0.2f, 0.0f);

    gl.glEnd();;

    gl.glFlush();

}
```

- Welches Bauverfahren des Parameters GL2.GL_TRIANGLE_STRIP ergibt sich aus dem Vergleich beider Dreicksstreifen?

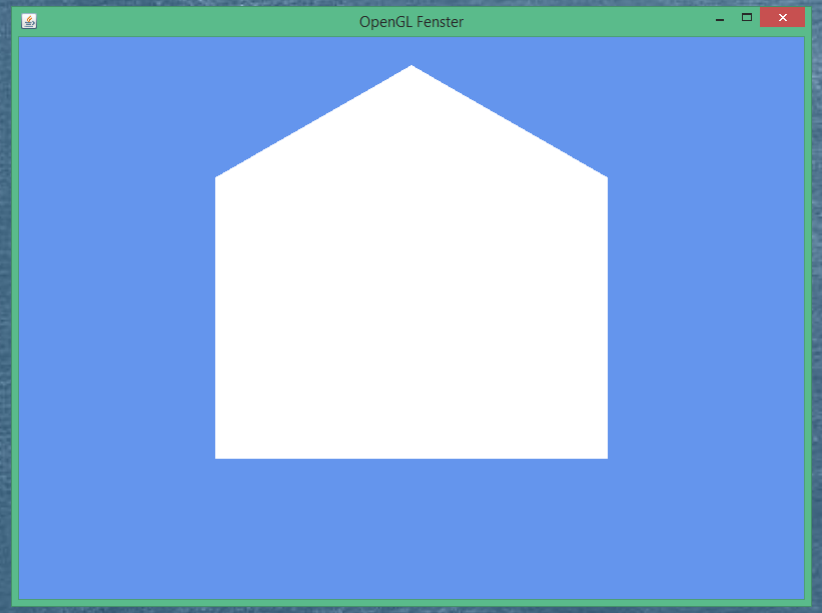
Polygone



- Der Parameter GL2.GL_POLYGON nutzt die Liste der Knoten als Spitze eines Polygons

```
@Override
public void display(GLAutoDrawable drawable) {
    // TODO Auto-generated method stub

    GL2 gl = drawable.getGL().getGL2();
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);

    gl.glBegin(GL2.GL_POLYGON);

        gl.glVertex3f(-0.5f, 0.0f, 0.0f);
        gl.glVertex3f( 0.5f, 0.0f, 0.0f);
        gl.glVertex3f( 0.5f, 0.5f, 0.0f);
        gl.glVertex3f( 0.0f, 0.90f, 0.0f);
        gl.glVertex3f(-0.5f, 0.5f, 0.0f);

    gl.glEnd();

    gl.glFlush();

}
```

einfachste Zusammensetzung von Bausteinen



- Erfolgt durch die nacheinanderfolgende Verwendung der gl.glBegin … gl.glEnd Code-Ausschnitte

```
@Override
public void display(GLAutoDrawable drawable) {
    // TODO Auto-generated method stub

    GL2 gl = drawable.getGL().getGL2();
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);

    gl.glBegin(GL2.GL_POLYGON);

        gl.glVertex3f(-0.5f, 0.0f, 0.0f);
        gl.glVertex3f( 0.5f, 0.0f, 0.0f);
        gl.glVertex3f( 0.5f, 0.5f, 0.0f);
        gl.glVertex3f( 0.0f, 0.9f, 0.0f);
        gl.glVertex3f(-0.5f, 0.5f, 0.0f);

    gl.glEnd();

    gl.glBegin(GL2.GL_POLYGON);

        gl.glVertex3f(-0.5f,-0.5f, 0.0f);
        gl.glVertex3f(-0.5f, 0.0f, 0.0f);
        gl.glVertex3f( 0.5f, 0.0f, 0.0f);
        gl.glVertex3f( 0.5f,-0.5f, 0.0f);

    gl.glEnd();

    gl.glFlush();

}
```

Weitere Grafik-Primitive

Befehl	Beschreibung
GL_POINTS	Zeichnet jeweils einen Punkt
GL_QUADS	Zeichnet mehrere 4-Ecke (aus jeweils 4 Vertices)
GL_QUAD_STRIP	Zeichnet einen Streifen von 4-Ecken
GL_TRIANGLE_FAN	Zeichnet eine Serie von 3-Ecken, die sich einen Mittelpunkt teilen: zB v0,v1,v2, dann v0,v2,v3, dann v0,v3,v4, usw.

Fenstereinstellungen
Farbe

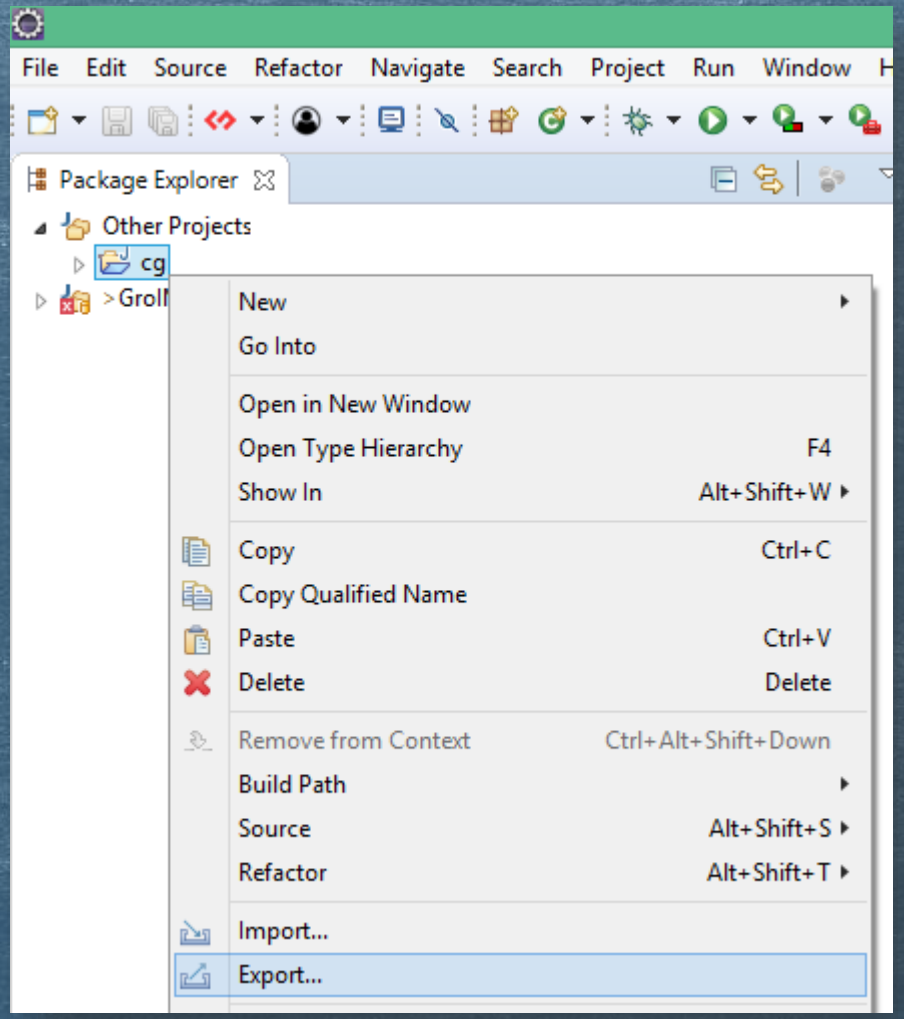
- Die Farbtabelle mit der von 0 bis 1 verlaufenden RGB-Werten ist beispielsweise unter diesem Link abrufbar:

www.rgbtool.com

```
@Override
public void init(GLAutoDrawable drawable) {
    // TODO Auto-generated method stub

    GL2 gl = drawable.getGL().getGL2();

2   /*Mit diesem Befehl setzt sich eine andere Farbe mit Gleitkommazahlen
     * als Parameter,die dem RGBA-Format entsprechen */
    gl.glClearColor(0.863f, 0.078f, 0.235f, 1.0f);

}

@Override
public void display(GLAutoDrawable drawable) {
    // TODO Auto-generated method stub

    GL2 gl = drawable.getGL().getGL2();

1   // Löschung des Farbpuffers (Schwarz setzt sich defaultmäßig)
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT);

    // call your draw code here

    gl.glFlush();

}
```

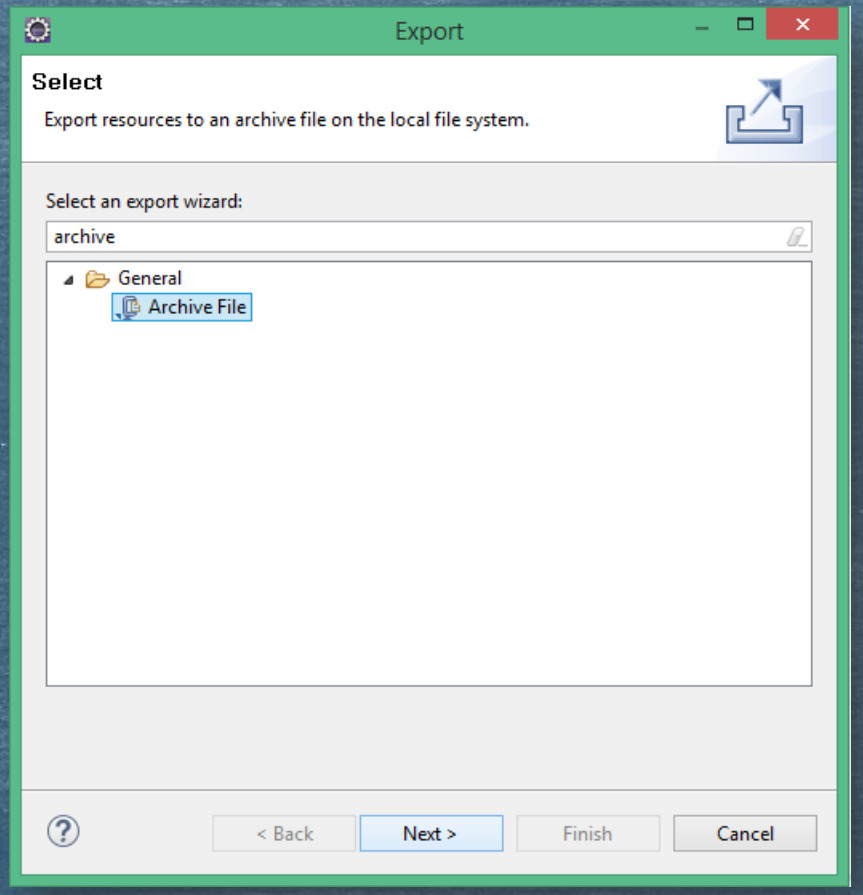
Exportieren von Java-Projekten in Eclipse

Schritt 1 von 4



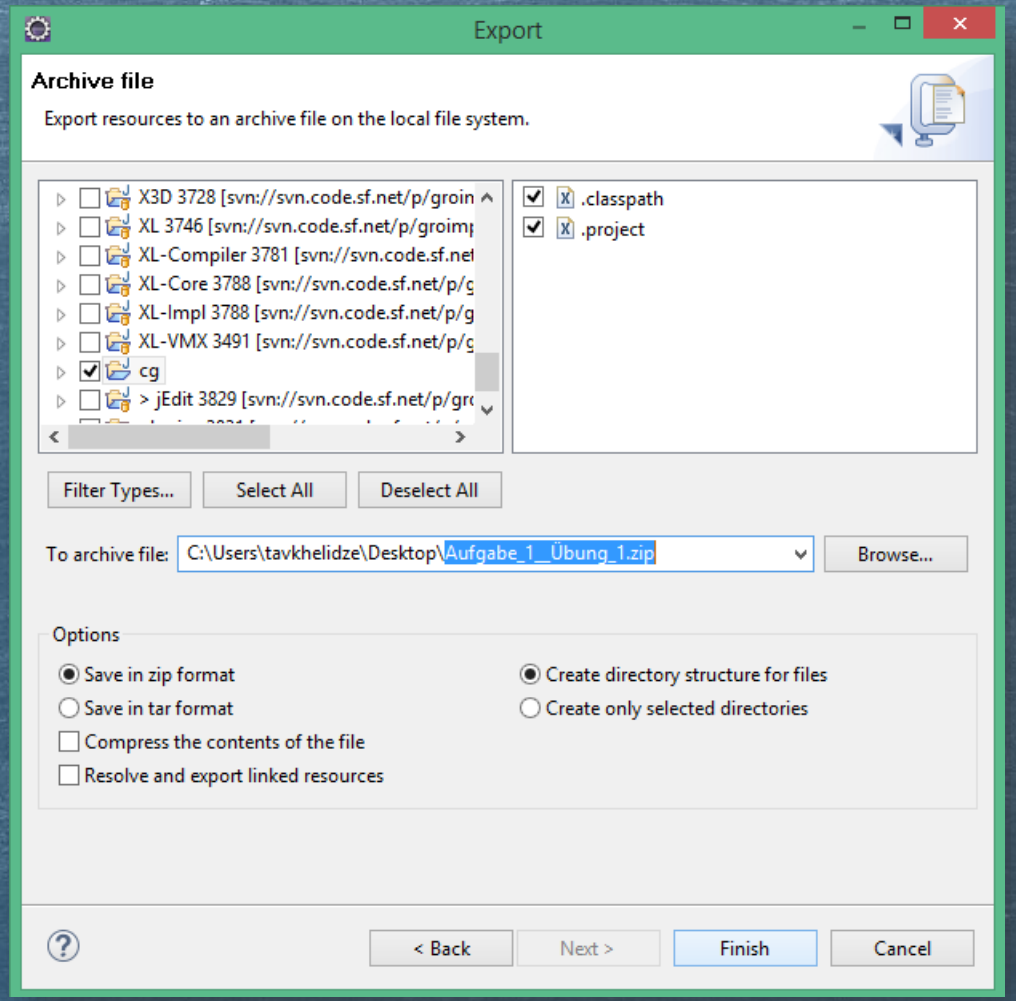
Exportieren von Java-Projekten in Eclipse

Schritt 2 von 4



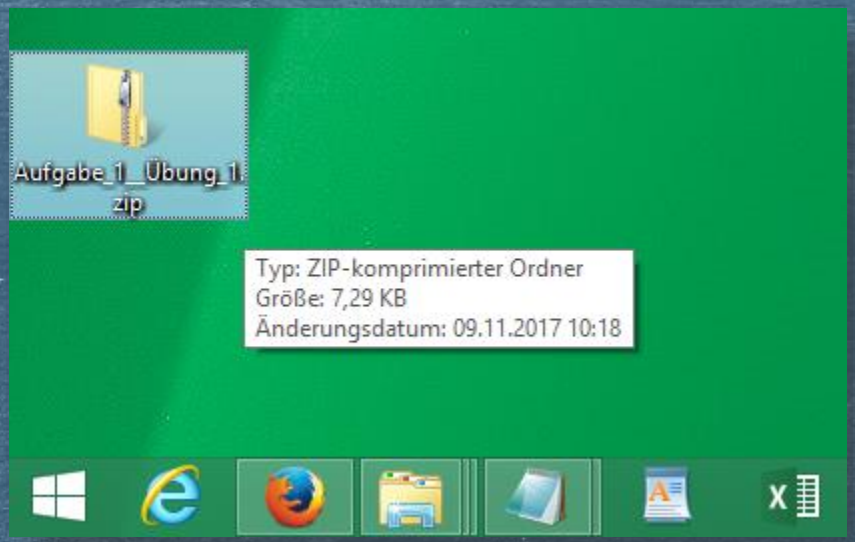
Exportieren von Java-Projekten in Eclipse

Schritt 3 von 4



Exportieren von Java-Projekten in Eclipse

Schritt 4 von 4



Bibliotheken für geometrische Objekte (1)

Bibliothek

com.jogamp.opengl.GL2

Geometrische Objekte

Geometrische Primitive:

Linien	*Vielecke (Polygone)*	
Punkte	*Dreiecke*	*Vierecke*
GL2.GL_POINTS glPointSize()	GL2.GL_TRIANGLES	GL2.GL_QUADS

Weitere geometrischen Objekte:

Scheiben	*Tetraeder*	*Würfel*

Freiformgeometrie:

Bézierkurven	*Bézierflächen*

Bibliotheken für geometrische Objekte (2)

Bibliothek

com.jogamp.opengl.glu.GLU

com.jogamp.opengl.util.gl2.GLUT

Geometrische Objekte

Quadriken:

Scheibe gluDisk()	*Scheibenausschnitt* gluPartialDisk()
Kugel gluSphere()	*Zylinder* gluCylinder()

Basische 3D Objekte:

Würfel Drahtgitter	glutWireCube()	Kugel Drahtgitter	glutWireSphere()
Würfel Volumen	glutSolidCube()	Kugel Volumen	glutSolidSphere()
Kegel Drahtgitter	glutWireCone()	Torus Drahtgitter	glutWireTorus()
Kegel Volumen	glutSolidCone()	Torus Volumen	glutSolidTorus()

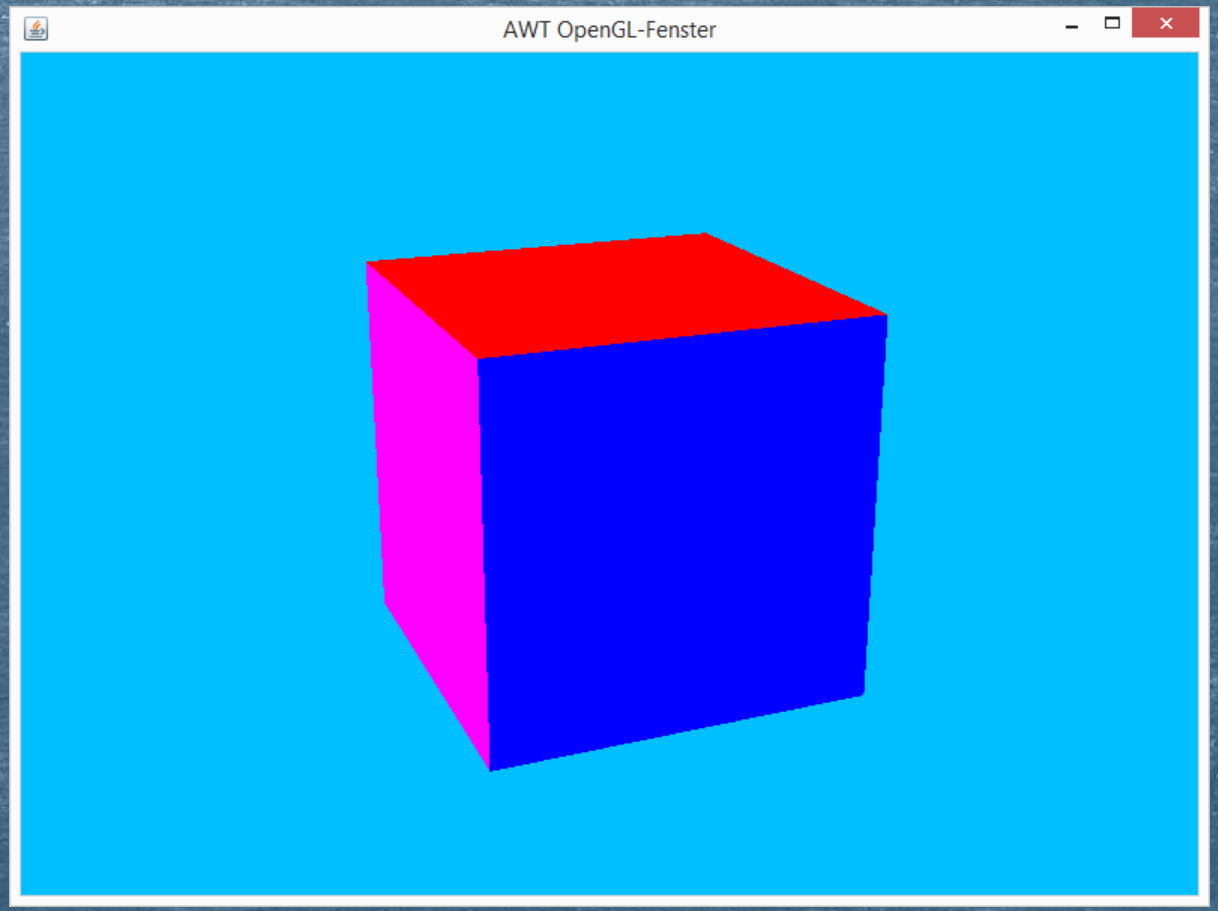
Weitere 3D Objekte:

Tetraeder Drahtgitter	glutWireTetrahedron()	**Teekanne** Drahtgitter	glutWireTeapot()
Tetraeder Volumen	glutSolidTetrahedron()	**Teekanne** Volumen	glutSolidTeapot()

3D-Objekte: Würfel (1)

Volumen, ohne Farbverlauf

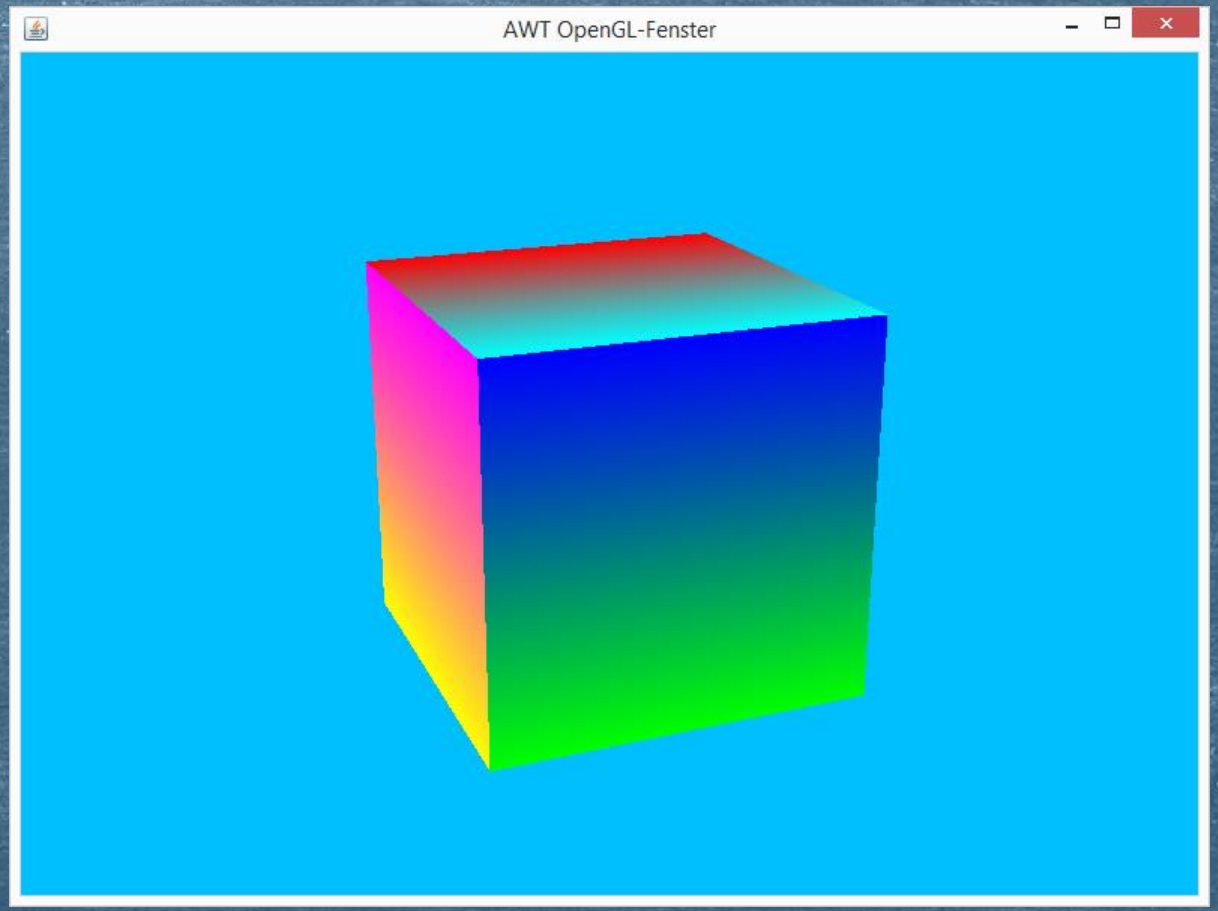
```
void drawCube(GL2 gl) {

    gl.glBegin(GL2.GL_QUADS);

      // Obere Fläche
    gl.glColor3f( 1f,0f,0f );   //red color
    gl.glVertex3f( 0.7f, 0.7f, -0.7f ); // Top Right
    gl.glVertex3f( -0.7f, 0.7f, -0.7f); // Top Left
    gl.glVertex3f( -0.7f, 0.7f, 0.7f ); // Bottom Left
    gl.glVertex3f( 0.7f, 0.7f, 0.7f ); // Bottom Right

      // Untere Fläche
    gl.glColor3f( 0f,1f,0f ); //green color
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Top Right
    gl.glVertex3f( -0.7f, -0.7f, 0.7f ); // Top Left
    gl.glVertex3f( -0.7f, -0.7f, -0.7f ); // Bottom Left
    gl.glVertex3f( 0.7f, -0.7f, -0.7f ); // Bottom Right

      // Vordere Fläche
    gl.glColor3f( 0f,0f,1f ); //blue color
    gl.glVertex3f( 0.7f, 0.7f, 0.7f ); // Top Right
    gl.glVertex3f( -0.7f, 0.7f, 0.7f ); // Top Left
    gl.glVertex3f( -0.7f, -0.7f, 0.7f ); // Bottom Left
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Bottom Right

      // Hintere Fläche
    gl.glColor3f( 1f,1f,0f ); //yellow (red+green)
    gl.glVertex3f( 0.7f, -0.7f, -0.7f ); // Bottom Left
    gl.glVertex3f( -0.7f, -0.7f, -0.7f ); // Bottom Right
    gl.glVertex3f( -0.7f, 0.7f, -0.7f ); // Top Right
    gl.glVertex3f( 0.7f, 0.7f, -0.7f ); // Top Left

      // Linke Fläche
    gl.glColor3f( 1f,0f,1f ); //purple (red+green)
    gl.glVertex3f( -0.7f, 0.7f, 0.7f ); // Top Right
    gl.glVertex3f( -0.7f, 0.7f, -0.7f ); // Top Left
    gl.glVertex3f( -0.7f, -0.7f, -0.7f ); // Bottom Left
    gl.glVertex3f( -0.7f, -0.7f, 0.7f ); // Bottom Right

      // Rechte Fläche
    gl.glColor3f( 0f,1f, 1f ); //sky blue (blue+green)
    gl.glVertex3f( 0.7f, 0.7f, -0.7f ); // Top Right
    gl.glVertex3f( 0.7f, 0.7f, 0.7f ); // Top Left
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Bottom Left
    gl.glVertex3f( 0.7f, -0.7f, -0.7f ); // Bottom Right

    gl.glEnd();
}
```



3D-Objekte: Würfel (2)

Volumen, mit Farbverlauf

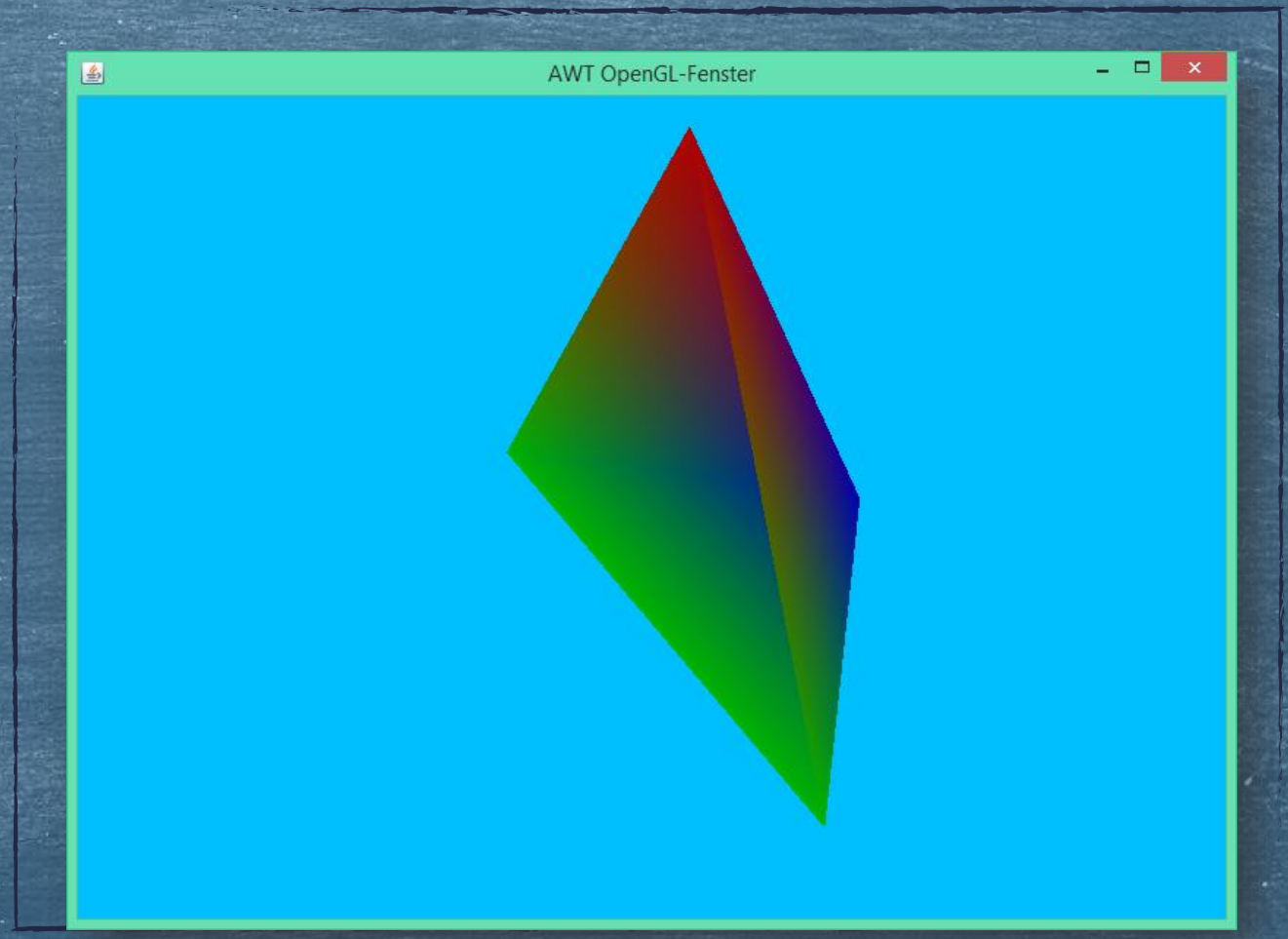
```
void drawCube(GL2 gl) {

    gl.glBegin(GL2.GL_QUADS);

      // Obere Fläche
    gl.glColor3f( 1f,0f,0f );   //red color
    gl.glVertex3f( 0.7f, 0.7f, -0.7f ); // Top Right
    gl.glVertex3f( -0.7f, 0.7f, -0.7f); // Top Left
    gl.glColor3f( 0f,1f, 1f ); //sky blue (blue+green)
    gl.glVertex3f( -0.7f, 0.7f, 0.7f ); // Bottom Left
    gl.glVertex3f( 0.7f, 0.7f, 0.7f ); // Bottom Right

      // Untere Fläche
    gl.glColor3f( 0f,1f,0f ); //green color
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Top Right
    gl.glVertex3f( -0.7f, -0.7f, 0.7f ); // Top Left
    gl.glVertex3f( -0.7f, -0.7f, -0.7f ); // Bottom Left
    gl.glVertex3f( 0.7f, -0.7f, -0.7f ); // Bottom Right

      // Vordere Fläche
    gl.glColor3f( 0f,0f,1f ); //blue color
    gl.glVertex3f( 0.7f, 0.7f, 0.7f ); // Top Right
    gl.glVertex3f( -0.7f, 0.7f, 0.7f ); // Top Left
    gl.glColor3f( 0f,1f,0f ); //green color
    gl.glVertex3f( -0.7f, -0.7f, 0.7f ); // Bottom Left
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Bottom Right

      // Hintere Fläche
    gl.glColor3f( 1f,1f,0f ); //yellow (red+green)
    gl.glVertex3f( 0.7f, -0.7f, -0.7f ); // Bottom Left
    gl.glVertex3f( -0.7f, -0.7f, -0.7f ); // Bottom Right
    gl.glVertex3f( -0.7f, 0.7f, -0.7f ); // Top Right
    gl.glVertex3f( 0.7f, 0.7f, -0.7f ); // Top Left

      // Linke Fläche
    gl.glColor3f( 1f,0f,1f ); //purple (red+green)
    gl.glVertex3f( -0.7f, 0.7f, 0.7f ); // Top Right
    gl.glVertex3f( -0.7f, 0.7f, -0.7f ); // Top Left
    gl.glColor3f( 1f,1f,0f ); //yellow (red+green)
    gl.glVertex3f( -0.7f, -0.7f, -0.7f ); // Bottom Left
    gl.glVertex3f( -0.7f, -0.7f, 0.7f ); // Bottom Right

      // Rechte Fläche
    gl.glColor3f( 0f,1f, 1f ); //sky blue (blue+green)
    gl.glVertex3f( 0.7f, 0.7f, -0.7f ); // Top Right
    gl.glVertex3f( 0.7f, 0.7f, 0.7f ); // Top Left
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Bottom Left
    gl.glVertex3f( 0.7f, -0.7f, -0.7f ); // Bottom Right

    gl.glEnd();
}
```



3D-Objekte: Tetraeder

Durchsichtig, mit Farbverlauf

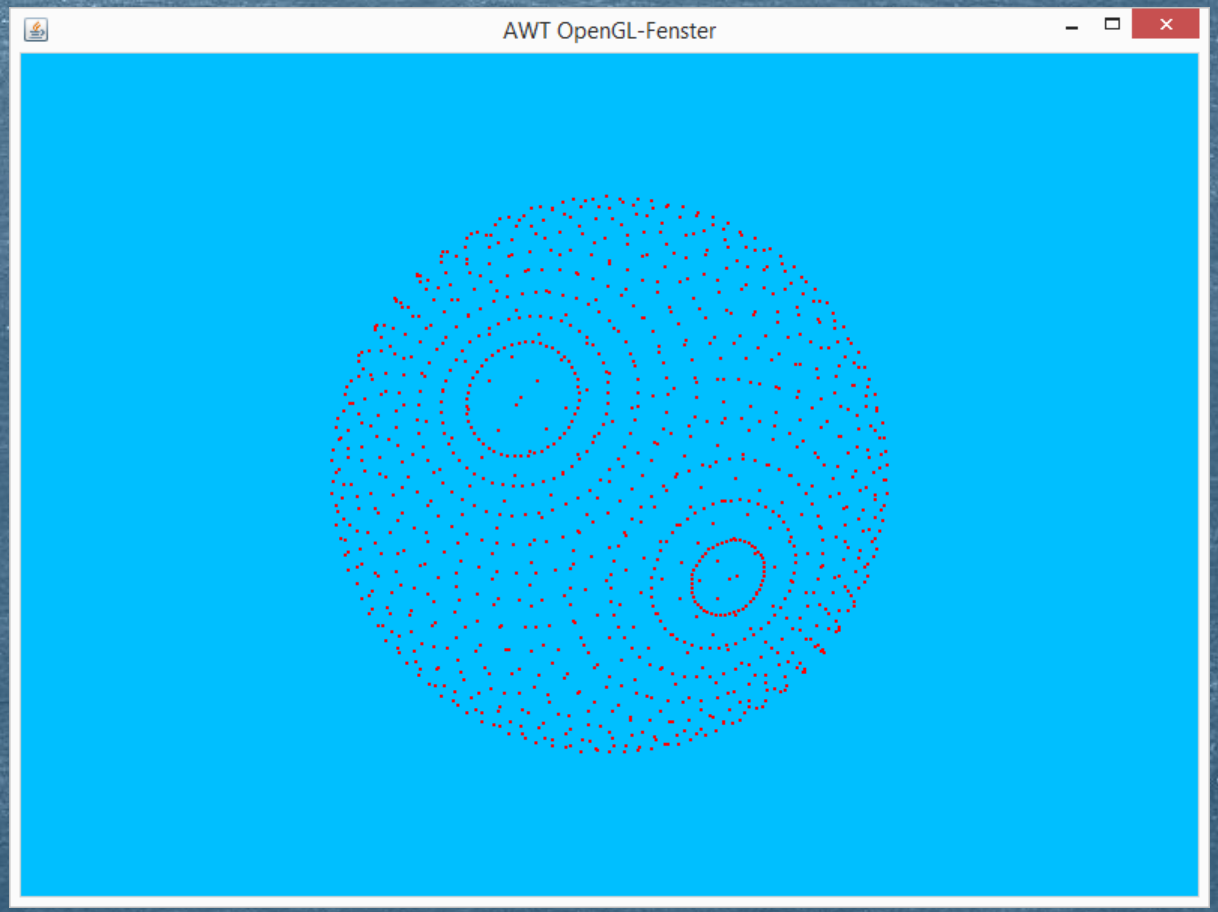
```
    void drawTetrahedron(GL2 gl) {

    gl.glBegin(GL2.GL_TRIANGLES);

     // erste Seitenfläche
    gl.glColor3f( 0.7f, 0.0f, 0.0f ); // Red
    gl.glVertex3f( 0.7f, 1.0f, 0.0f ); // Top
    gl.glColor3f( 0.0f, 0.7f, 0.0f ); // Green
    gl.glVertex3f( -0.7f, -0.7f, 0.7f ); // Left
    gl.glColor3f( 0.0f, 0.0f, 0.7f ); // Blue
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Right

    // zweite Seitenfläche
    gl.glColor3f( 0.7f, 0.0f, 0.0f ); // Red
    gl.glVertex3f( 0.7f, 1.0f, 0.0f ); // Top
    gl.glColor3f( 0.0f, 0.0f, 0.7f ); // Blue
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Left
    gl.glColor3f( 0.0f, 0.7f, 0.0f ); // Green
    gl.glVertex3f( 0.7f, -0.7f, -0.7f ); // Right

    // dritte Seitenfläche
    gl.glColor3f( 0.7f, 0.0f, 0.0f ); // Red
    gl.glVertex3f( 0.7f, 1.0f, 0.0f ); // Top
    gl.glColor3f( 0.0f, 0.7f, 0.0f ); // Green
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Left
    gl.glColor3f( 0.0f, 0.0f, 0.7f ); // Blue
    gl.glVertex3f( 0.7f, -0.7f, -0.7f ); // Right

    // GrundFläche
    gl.glColor3f( 0.0f, 0.7f, 0.0f ); // Red
    gl.glVertex3f( -0.7f, -0.7f, 0.7f ); // Top
    gl.glColor3f( 0.0f, 0.0f, 0.7f ); // Blue
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Left
    gl.glColor3f( 0.0f, 0.7f, 0.0f ); // Green
    gl.glVertex3f( 0.7f, -0.7f, -0.7f ); // Right

    gl.glEnd();
}
```



3D-Objekte: Kugel (1)

GLU, Punktewolke

```
    void drawGluPCSphere(GLU glu, GL2 gl) {

    GLUquadric quad;
    quad = glu.gluNewQuadric();

    gl.glPointSize(2f);
    glu.gluQuadricDrawStyle(quad, GLU.GLU_POINT);

    gl.glColor3f(1,0,0); // rot

    glu.gluSphere(quad,1,50,25);

}
```



3D-Objekte: Kugel (2)

GLUT, Drahtgitter

```
    void drawGlutWiSphere(GLUT glut, GL2 gl) {

    gl.glColor3f(0f, 1f, 0f); // Green

    glut.glutWireSphere(1, 20, 10);
}
```