

WiSe 18/19



GEORG-AUGUST-UNIVERSITÄT  
GÖTTINGEN

Prof. Dr. Winfried Kurth  
Alex Tavkhelidze

# Praktikum Computergrafik

*Folien für Extras*

Übung 2: 3D Objekte

Freiformgeometrie

Animation

# Bibliotheken für geometrische Objekte (1)

## Bibliothek

com.jogamp.opengl.**GL2**

## Geometrische Objekte

### Geometrische Primitive:

<i>Linien</i>	<i>Vielecke (Polygone)</i>	
<i>Punkte</i>	<i>Dreiecke</i>	<i>Vierecke</i>

**GL2.GL\_POINTS**  
glPointSize()

**GL2.GL\_TRIANGLES**

**GL2.GL\_QUADS**

### Weitere geometrischen Objekte:

<i>Scheiben</i>	<i>Tetraeder</i>	<i>Würfel</i>
-----------------	------------------	---------------

### Freiformgeometrie:

<i>Bézierkurven</i>	<i>Bézierflächen</i>
---------------------	----------------------

# Bibliotheken für geometrische Objekte (2)

## Bibliothek

## Geometrische Objekte

com.jogamp.opengl.glu.GLU

### Quadriken:

<i>Scheibe</i> gluDisk()	<i>Scheibenausschnitt</i> gluPartialDisk()
<i>Kugel</i> gluSphere()	<i>Zylinder</i> gluCylinder()

### Basische 3D Objekte:

com.jogamp.opengl.util.gl2.GLUT

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

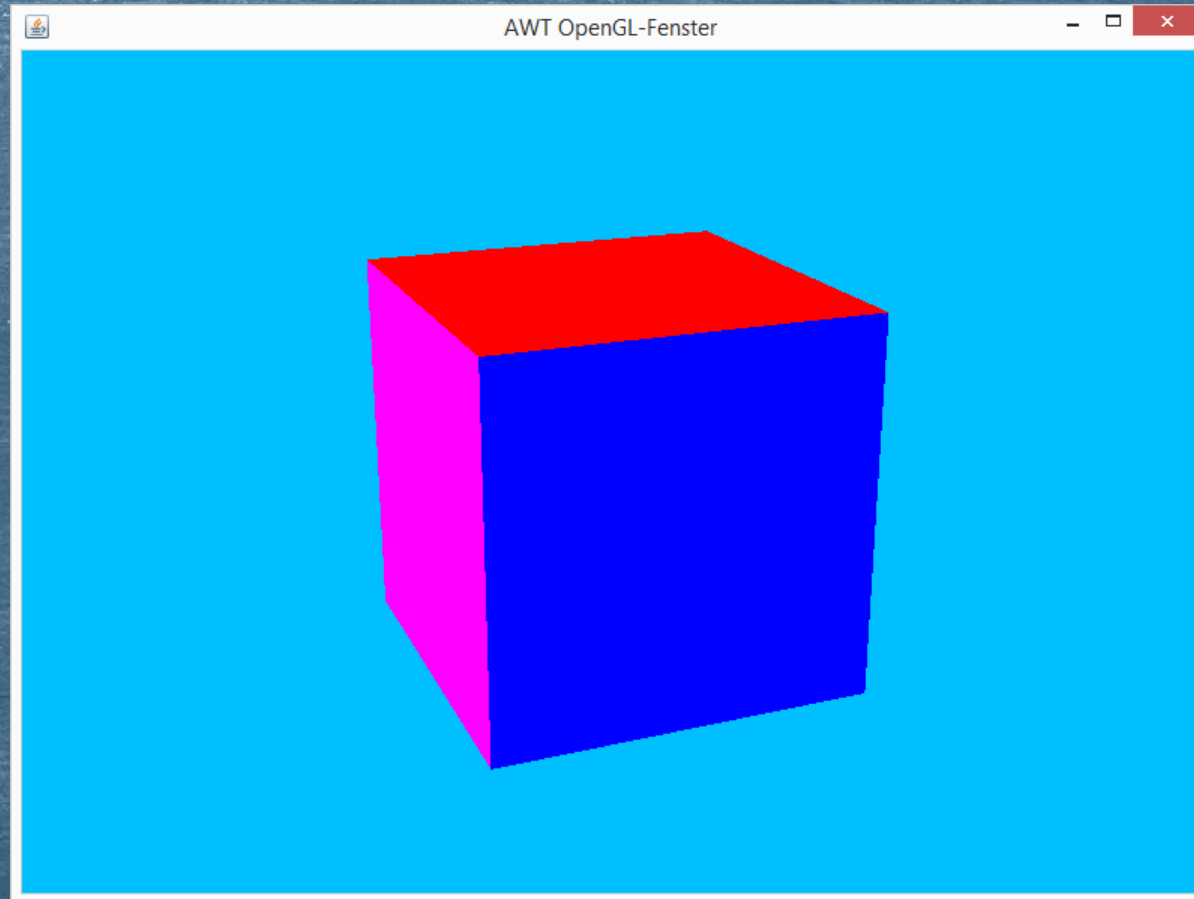
### Weitere 3D Objekte:

<b>Tetraeder</b> Drahtgitter	glutWireTetrahedron()	<b>Teekanne</b> Drahtgitter	glutWireTeapot()
<b>Tetraeder</b> Volumen	glutSolidTetrahedron()	<b>Teekanne</b> Volumen	glutSolidTeapot()

# 3D-Objekte: Würfel (1)

Volumen, ohne Farbverlauf

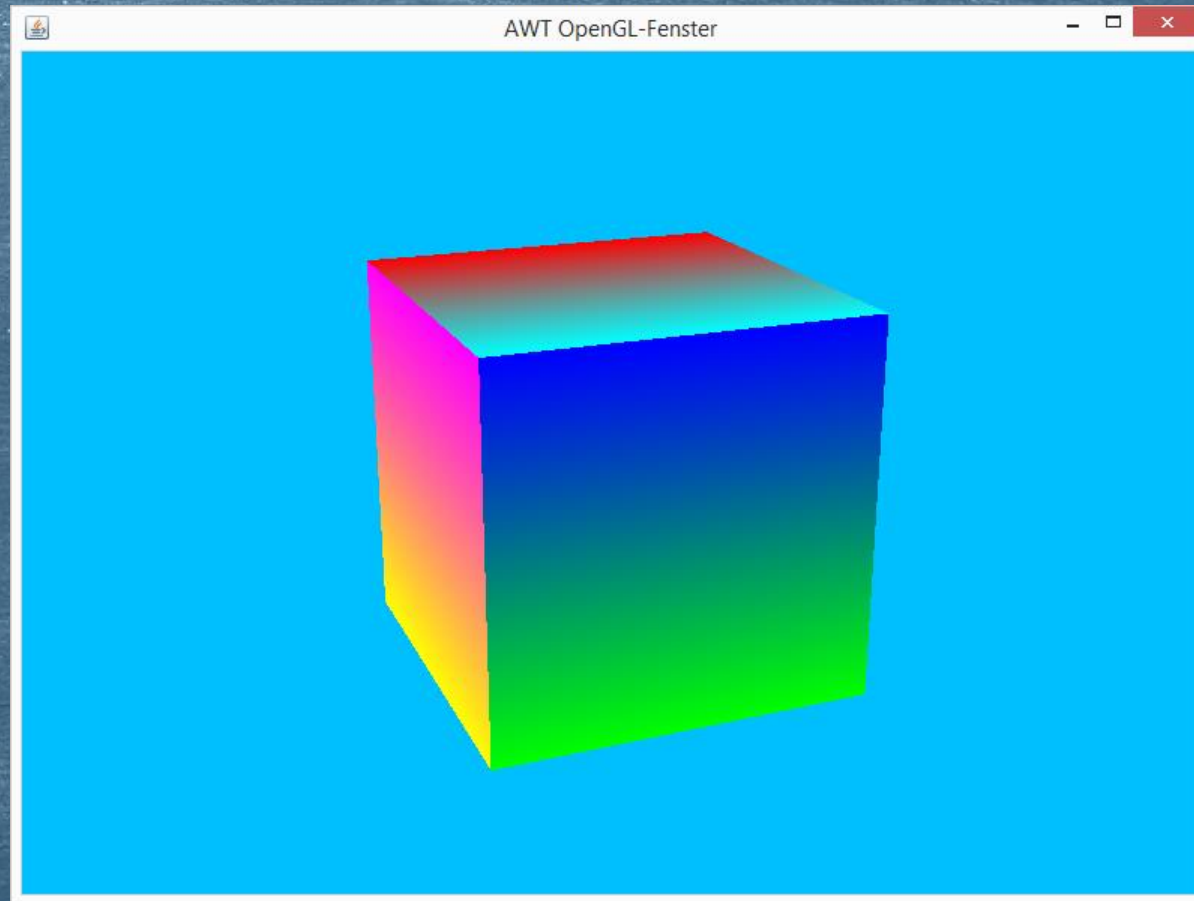
```
void drawCube(GL2 gl) {  
  
    gl.glBegin(GL2.GL_QUADS);  
  
        // Obere Fläche  
        gl.glColor3f( 1f,0f,0f ); //red color  
        gl.glVertex3f( 0.7f, 0.7f, -0.7f ); // Top Right  
        gl.glVertex3f( -0.7f, 0.7f, -0.7f ); // Top Left  
        gl.glVertex3f( -0.7f, 0.7f, 0.7f ); // 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

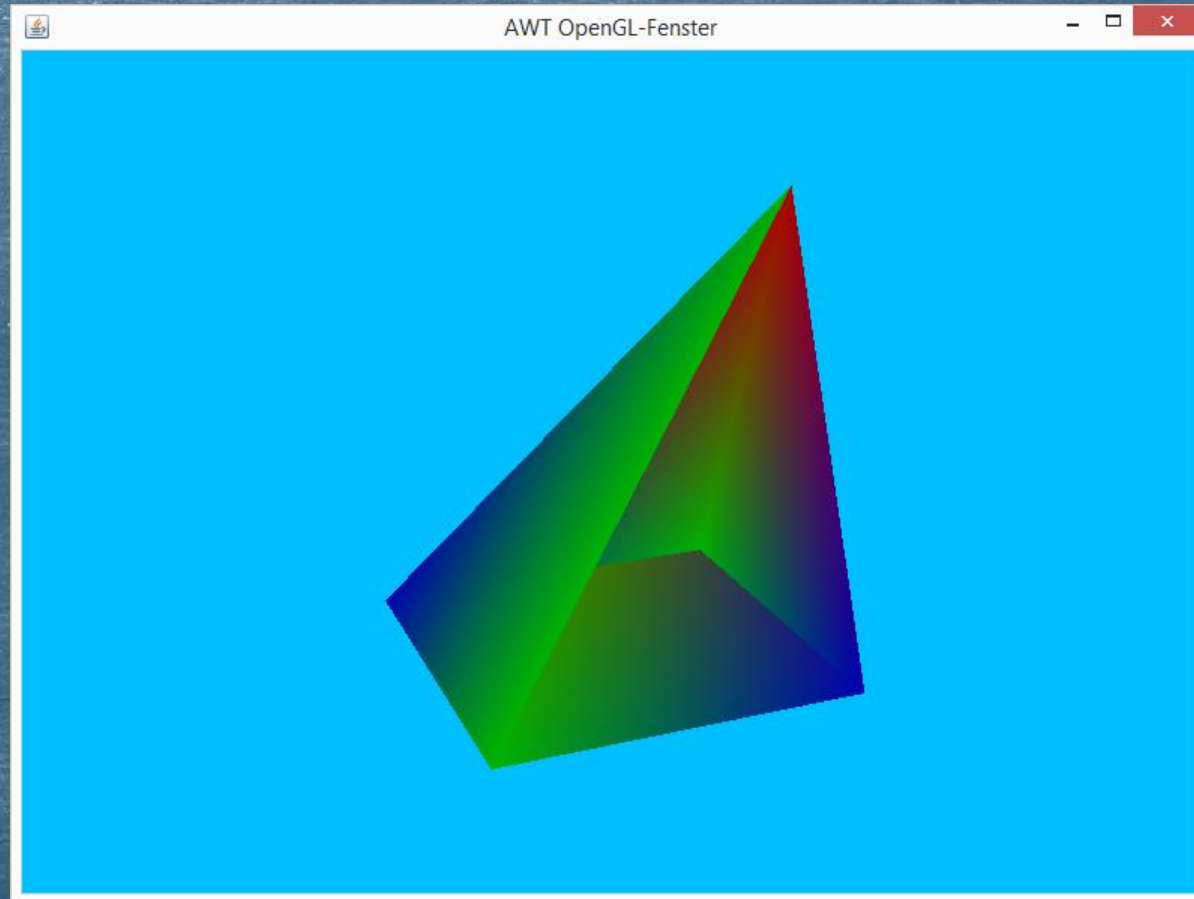
```
void drawCube(GL2 gl) {  
  
    gl.glBegin(GL2.GL_QUADS);  
  
    // Obere Fläche  
    gl.glColor3f( 1f,0f,0f ); //red color  
    gl.glVertex3f( 0.7f, 0.7f, -0.7f ); // Top Right  
    gl.glVertex3f( -0.7f, 0.7f, -0.7f); // Top Left  
    gl.glColor3f( 0f,1f, 1f ); //sky blue (blue+green)  
    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

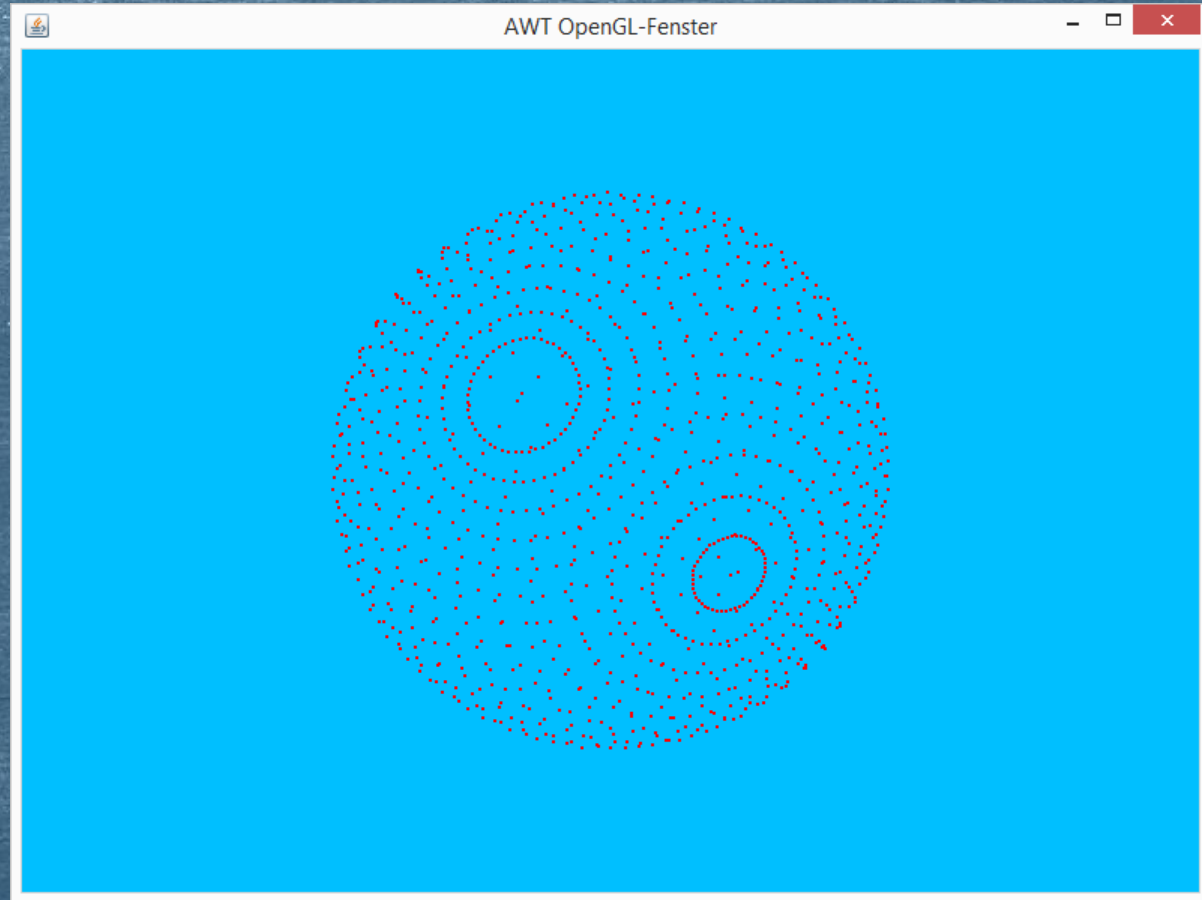
```
void drawTetrahedron(GL2 gl) {  
  
    gl.glBegin(GL2.GL_TRIANGLES);  
  
    // Vordere Fläche  
    gl.glColor3f( 0.7f, 0.0f, 0.0f ); // Red  
    gl.glVertex3f( 0.7f, 1.0f, 0.0f ); // Top  
    gl.glColor3f( 0.0f, 0.7f, 0.0f ); // Green  
    gl.glVertex3f( -0.7f, -0.7f, 0.7f ); // Left  
    gl.glColor3f( 0.0f, 0.0f, 0.7f ); // Blue  
    gl.glVertex3f( 0.7f, -0.7f, 0.7f ); // Right  
  
    // Rechte Fläche  
    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  
  
    // Linke Fläche  
    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  
  
    // Untere Fläche  
    gl.glColor3f( 0.0f, 0.7f, 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  
  
    gl.glEnd();  
}
```



# 3D-Objekte: Kugel (1)

## GLU, Punktwolke

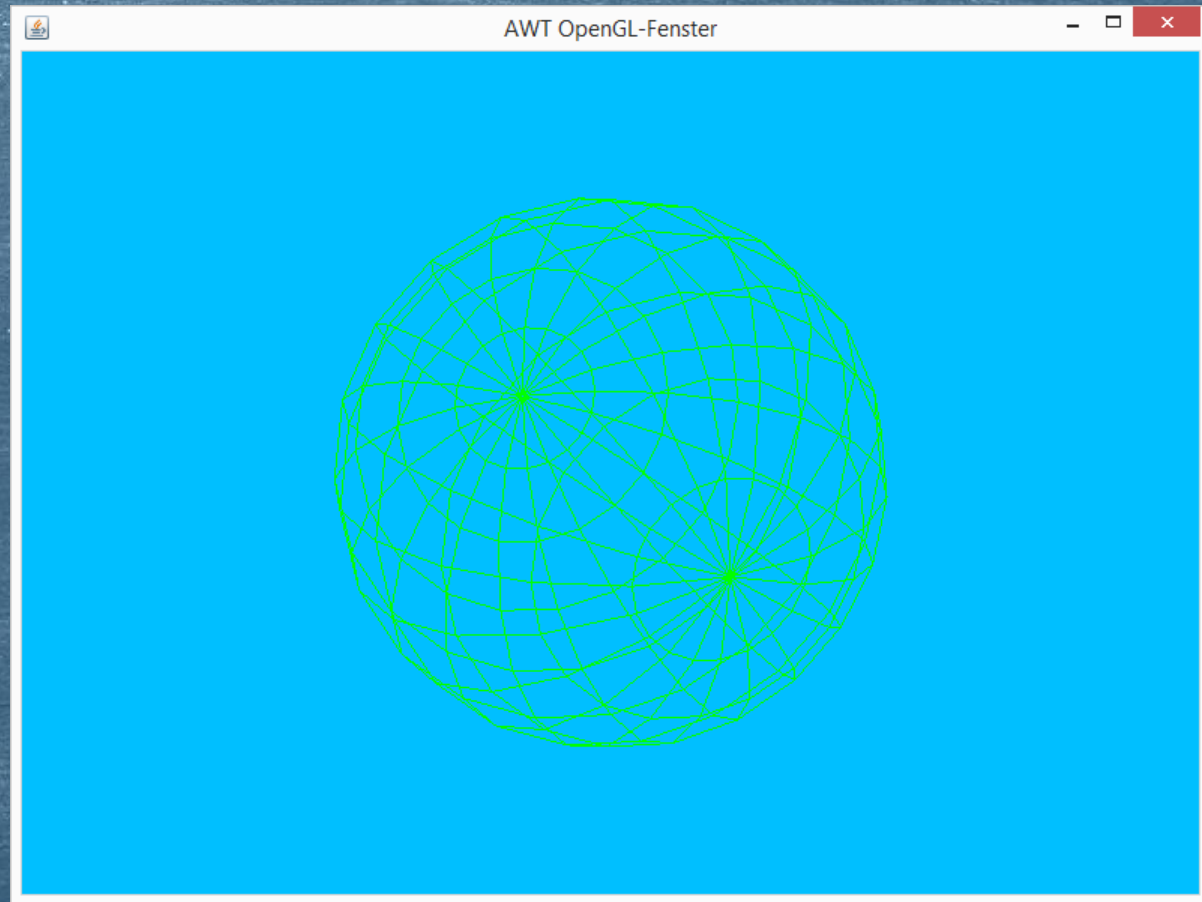
```
void drawGluPCSphere(GLU glu, GL2 gl) {  
  
    GLUquadric quad;  
    quad = glu.gluNewQuadric();  
  
    gl.glPointSize(2f);  
    glu.gluQuadricDrawStyle(quad, GLU.GLU_POINT);  
  
    gl.glColor3f(1,0,0); // rot  
  
    glu.gluSphere(quad,1,50,25);  
  
}
```



# 3D-Objekte: Kugel (2)

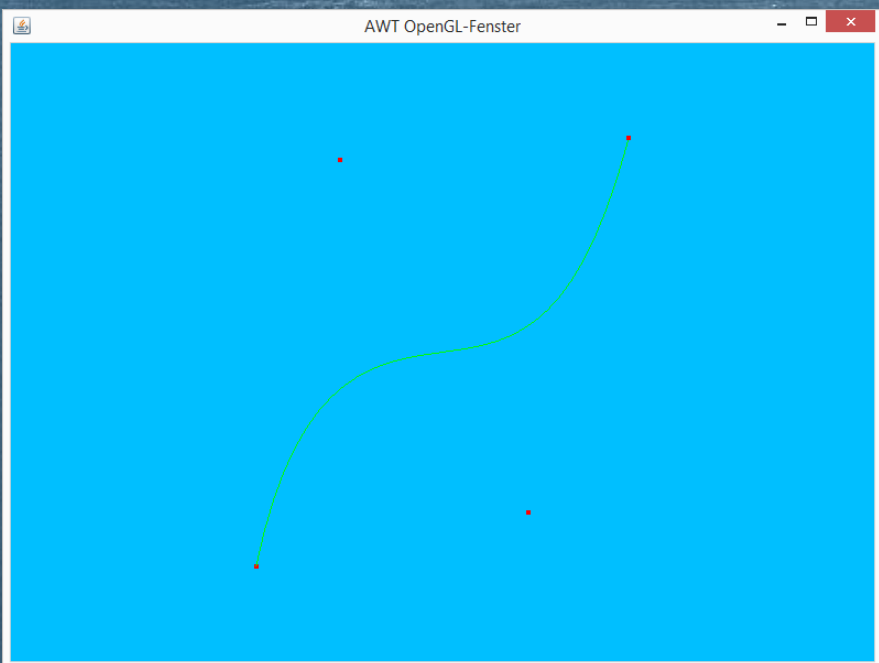
## GLUT, Drahtgitter

```
void drawGlutWiSphere(GLUT glut, GL2 gl) {  
    gl.glColor3f(0f, 1f, 0f); // Green  
    glut.glutWireSphere(1, 20, 10);  
}
```



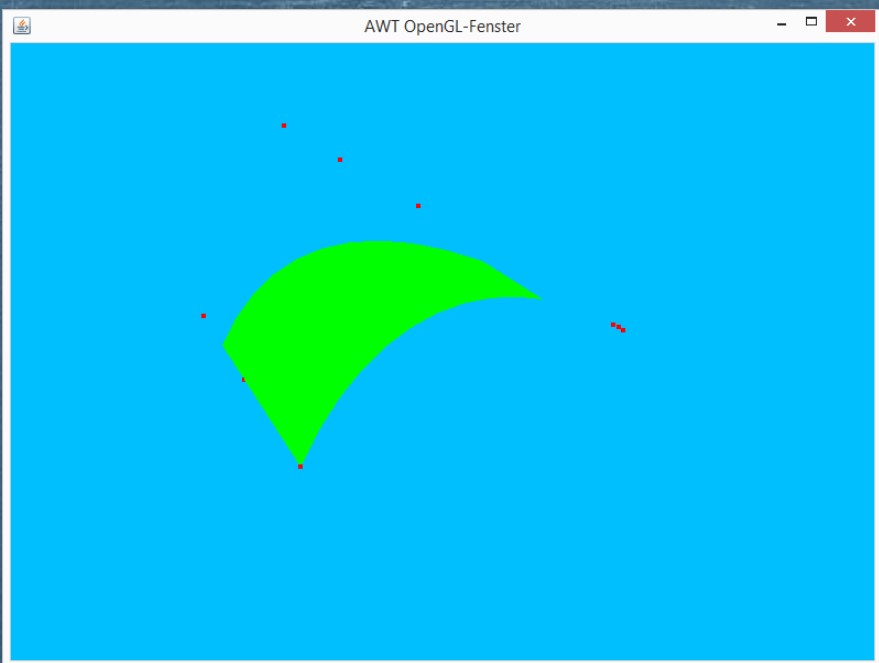


# Freiformgeometrie: Bézierkurve



```
void drawBezCur (GL2 gl) {  
  
    int numPoints = 4;  
    float ctrlPoints[] = {  
        -1.0f, -1.0f, 0.0f, // erster Kontrollpunkt (Endpunkt)  
        -0.5f, 1.0f, 0.0f, // zweiter Kontrollpunkt  
        0.5f, -1.0f, 0.0f, // dritter Kontrollpunkt  
        1.0f, 1.0f, 0f}; // vierter Kontrollpunkt (Endpunkt)  
  
    // Deklaration einer Variablen vom Typ Gleitkommazahlenpuffer und Angabe von passender Zielgröße dieses Pufferelements  
    FloatBuffer ctrlPointsBf = Buffers.newDirectFloatBuffer(ctrlPoints.length);  
  
    // Einspeisen von Daten aus dem Datenfeld ctrlPoints in den deklarierten Puffer  
    for (int i = 0; i < ctrlPoints.length; i++) { ctrlPointsBf.put(ctrlPoints[i]); }  
    ctrlPointsBf.rewind();  
  
    // legt die Kontrollpunkte der Kurve fest  
    gl.glMap1f(GL2.GL_MAP1_VERTEX_3, 0.0f, 1f, 3, numPoints, ctrlPointsBf);  
  
    // Aktiviert die Erstellung von x-,y-,z-Verteckoordinaten wenn glMapGrid1f und glEvalMesh1 aufgerufen sind  
    gl.glEnable(GL2.GL_MAP1_VERTEX_3);  
  
    // Zeichnen von Kontrollpunkten (man darf dies weglassen und nur die Kurve rendern lassen)  
    gl.glPointSize(4f);  
    gl.glBegin(GL2.GL_POINTS);  
    for (int i = 0; i < numPoints; i++) {  
        gl.glColor3f(1f,0,0);  
        gl.glVertex3f(ctrlPoints[3*i], ctrlPoints[3*i+1], ctrlPoints[3*i+2]);  
    }  
    gl.glEnd();  
  
    // erzeugt ein eiddimensionales Punkte-Gitter  
    gl.glMapGrid1f(30, 0.0f, 1f);  
    // berechnet und rendert die Kurve (wertet die Kurve in Gitterpunkten aus)  
    // Anstelle des Parameters GL_LINE (Linienzug) steht auch GL_POINT (Punktewolke) zur Verfügung  
    gl.glColor3f(0f,1f,0);  
    gl.glEvalMesh1(GL2.GL_LINE, 0, 30);  
}
```

# Freiformgeometrie: Bézierfläche

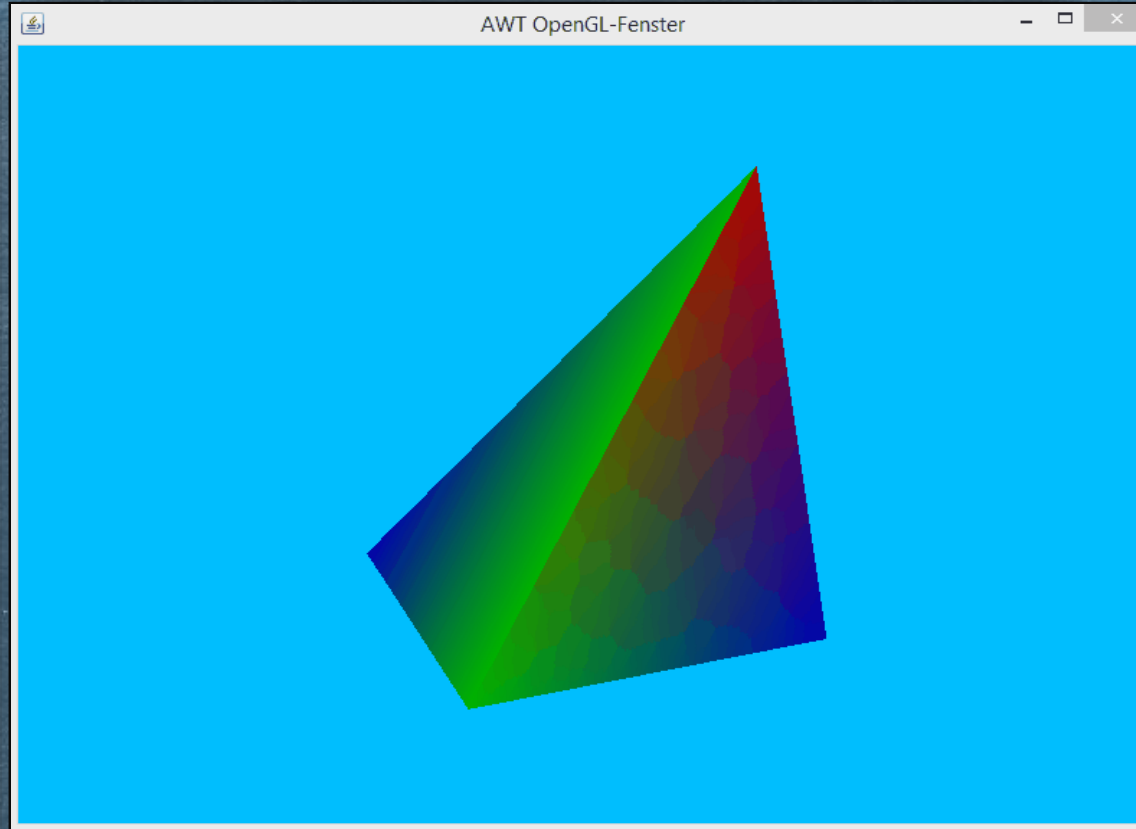


```
void drawBezSur (GL2 gl) {  
  
    int numPlines = 3; // Anzahl der Linienzüge im Gitternetz der Fläche (siehe Details unter http://www-lehre.inf.uos.de/~cg/2008/skript/node132.html)  
    float ctrlPoints[] = {  
        // Erster Linienzug  
        -1.0f, 0.0f, 1.0f, // erster Kontrollpunkt  
        -0.5f, 1.0f, 1.0f, // zweiter Kontrollpunkt  
        1.0f, 0.0f, 0.05f, // dritter Kontrollpunkt  
        // Zweiter Linienzug  
        -1.0f, 0.0f, 0.0f, // erster Kontrollpunkt  
        -0.5f, 1.0f, 0.0f, // zweiter Kontrollpunkt  
        1.0f, 0.0f, 0.0f, // dritter Kontrollpunkt  
        // Dritter Linienzug  
        -1.0f, 0.0f, -1.0f, // erster Kontrollpunkt  
        -0.5f, 1.0f, -1.0f, // zweiter Kontrollpunkt  
        1.0f, 0.0f, -0.05f}; // dritter Kontrollpunkt  
  
    // Deklaration einer Variablen vom Typ Gleitkommazahlenpuffer und Angabe von passender Zielgröße dieses Pufferelements  
    FloatBuffer ctrlPointsBf = Buffers.newDirectFloatBuffer(ctrlPoints.length);  
  
    // Einspeisen von Daten aus dem Datenfeld ctrlPoints in den deklarierten Puffer  
    for (int i = 0; i < ctrlPoints.length; i++) { ctrlPointsBf.put(ctrlPoints[i]); }  
    ctrlPointsBf.rewind();  
  
    // legt die Kontrollpunkte der Fläche fest - mehr Details unter https://wiki.delphigl.com/index.php/glMap2  
    gl.glMap2f(GL2.GL_MAP2_VERTEX_3, 0.0f, 4f, 3, 3, 0.0f, 4f, 9, 3, ctrlPointsBf);  
  
    // Aktiviert die Erstellung von x-, y-, z-Verteikordinaten wenn glMapGrid2f und glEvalMesh2 aufgerufen sind  
    gl.glEnable(GL2.GL_MAP2_VERTEX_3);  
  
    // Zeichnen von Kontrollpunkten (man darf dies weglassen und nur die Fläche rendern lassen)  
    gl.glPointSize(4f);  
    gl.glBegin(GL2.GL_POINTS);  
    for (int i = 0; i < numPlines*3; i++) {  
        gl.glColor3f(1f,0,0);  
        gl.glVertex3f(ctrlPoints[3*i], ctrlPoints[3*i+1], ctrlPoints[3*i+2]);  
    }  
    gl.glEnd();  
  
    // erzeugt ein zweidimensionales Punkte-Gitter - mehr Details unter https://wiki.delphigl.com/index.php/glMapGrid  
    gl.glMapGrid2f(10, 0.0f, 3f, 10, 0f, 3f);  
    // berechnet und rendert die Fläche (wertet die Fläche in Gitterpunkten aus) - mehr Details unter https://wiki.delphigl.com/index.php/glEvalMesh  
    // Anstelle des Parameters GL_FILL (gefüllte Fläche) stehen auch GL_LINE (Drahtgitter) und GL_POINT (Punktewolke) zur Verfügung  
    gl.glColor3f(0f,1f,0);  
    gl.glEvalMesh2(GL2.GL_FILL, 0, 10, 0, 10);  
}
```

# Animation

```
void animTetrahedron(GL2 gl) {  
  
    gl.glBegin(GL2.GL_TRIANGLES);  
  
    // Vordere Fläche  
    gl.glColor3f( 0.7f, 0.0f, 0.0f ); // Red  
    gl.glVertex3f( 0.7f, 1.0f+x, 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  
  
    // Rechte Fläche  
    gl.glColor3f( 0.7f, 0.0f, 0.0f ); // Red  
    gl.glVertex3f( 0.7f, 1.0f+x, 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  
  
    // Linke Fläche  
    gl.glColor3f( 0.7f, 0.0f, 0.0f ); // Red  
    gl.glVertex3f( 0.7f, 1.0f+x, 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  
  
    // Untere Fläche  
    gl.glColor3f( 0.0f, 0.7f, 0.0f ); // Red  
    gl.glVertex3f( 0.7f, 1.0f+x, 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  
  
    gl.glEnd();  
}
```

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
  
    // The animator  
    final FPSAnimator animator = new FPSAnimator(glcanvas, 60, true);  
    animator.start();  
}
```



```
@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);  
  
    // ----- zeichnet den definierten (siehe unten) Tetraeder (wird schrittweise animiert) -----  
    animTetrahedron(gl);  
    if (x<0.5f) {x +=0.005;} else {x =0f;};  
}
```