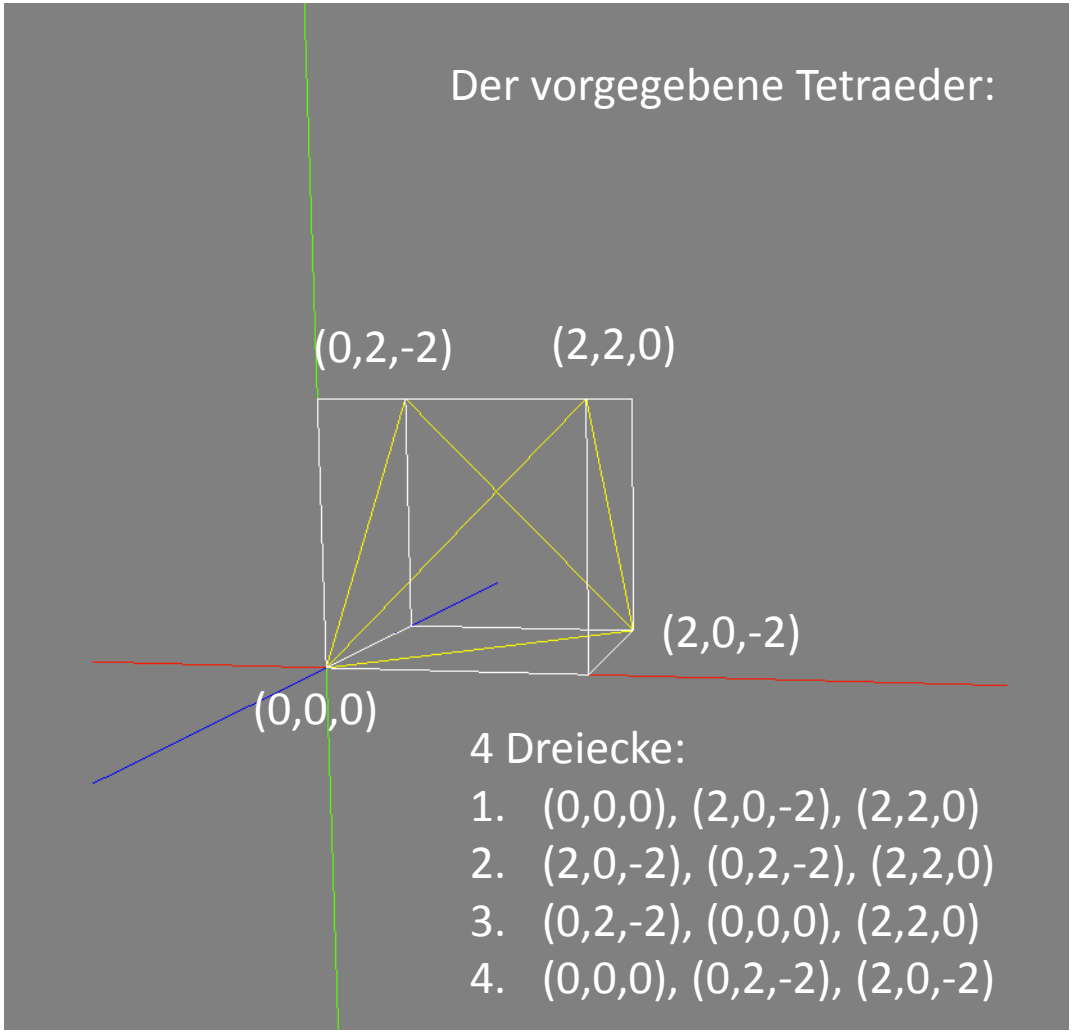
Computergrafik 1

Blatt 5

Aufgabe 1


Der vorgegebene Tetraeder:
(0,2,-2)
(2,2,0)
(2,0,-2)
(0,0,0)
4 Dreiecke:
1. (0,0,0), (2,0,-2), (2,2,0)
2. (2,0,-2), (0,2,-2), (2,2,0)
3. (0,2,-2), (0,0,0), (2,2,0)
4. (0,0,0), (0,2,-2), (2,0,-2)

Aufgabe 1 (2)

Die Ebenengleichung: $ax + by + cz + d = 0$,

wobei (a,b,c) der Normalenvektor der Ebene ist.

Wie berechnet man den Normalenvektor?

Aufgabe 1 (3)

Dreieck: P1 = (2,0,-2), P2 = (0,2,-2), P3 = (2,2,0)

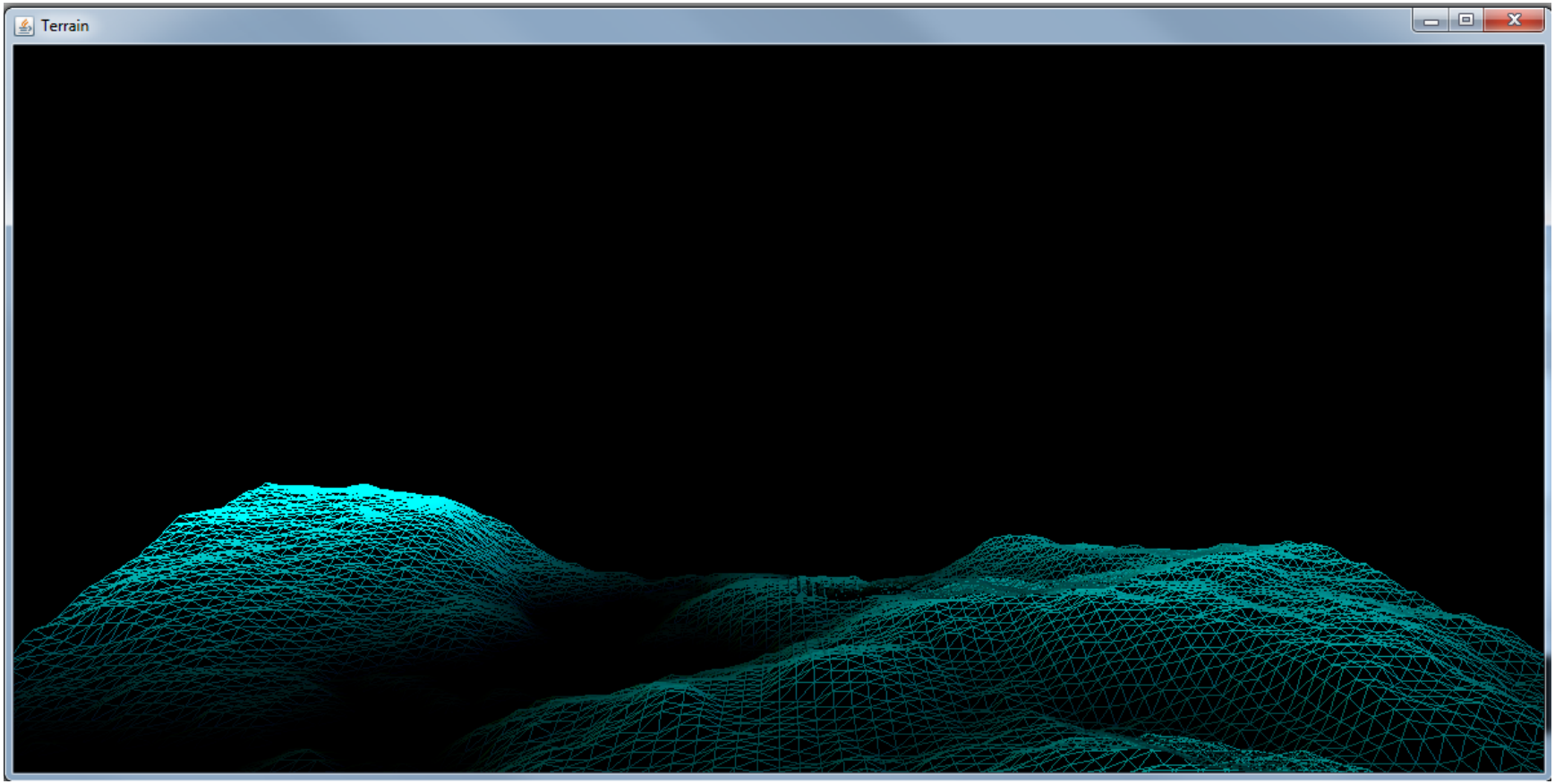
1. Schritt: Zwei Vektoren in der Ebene finden:
   1. $\vec{v} = \overrightarrow{P1P2}$ = P2 – P1 = (-2, 2, 0)
   2. $\vec{w} = \overrightarrow{P1P3}$ = P3 – P1 = (0, 2, 2)
2. Schritt: Kreuzprodukt berechnen
   - $\vec{v} \times \vec{w} = \begin{pmatrix} v2w3-v3w2 \\ v3w1-v1w3 \\ v1w2-v2w1 \end{pmatrix}$ = **(4, 4, -4)** -> Das ist der Normalenvektor
3. Ebenengleichung aufstellen: 4x + 4y – 4z + d = 0
4. Beliebigen Punkt (hier P1) der Ebene einsetzen um d zu berechnen:
   - 4*2 + 4*0 – (4*-2) + d = 0
   - 8 + 8 + d = 0
   - d = -16
5. x + y -z -4 = 0 -> Das ist die Ebenengleichung

Aufgabe 1 (4)

4 Dreiecke – 4 Ebenengleichungen:

1. (0,0,0), (2,0,-2), (2,2,0) -> x – y + z + 0 = 0
2. (2,0,-2), (0,2,-2), (2,2,0) -> x + y – z – 4 = 0
3. (0,2,-2), (0,0,0), (2,2,0) -> x – y – z + 0 = 0
4. (0,0,0), (0,2,-2), (2,0,-2) -> x + y + z + 0 = 0

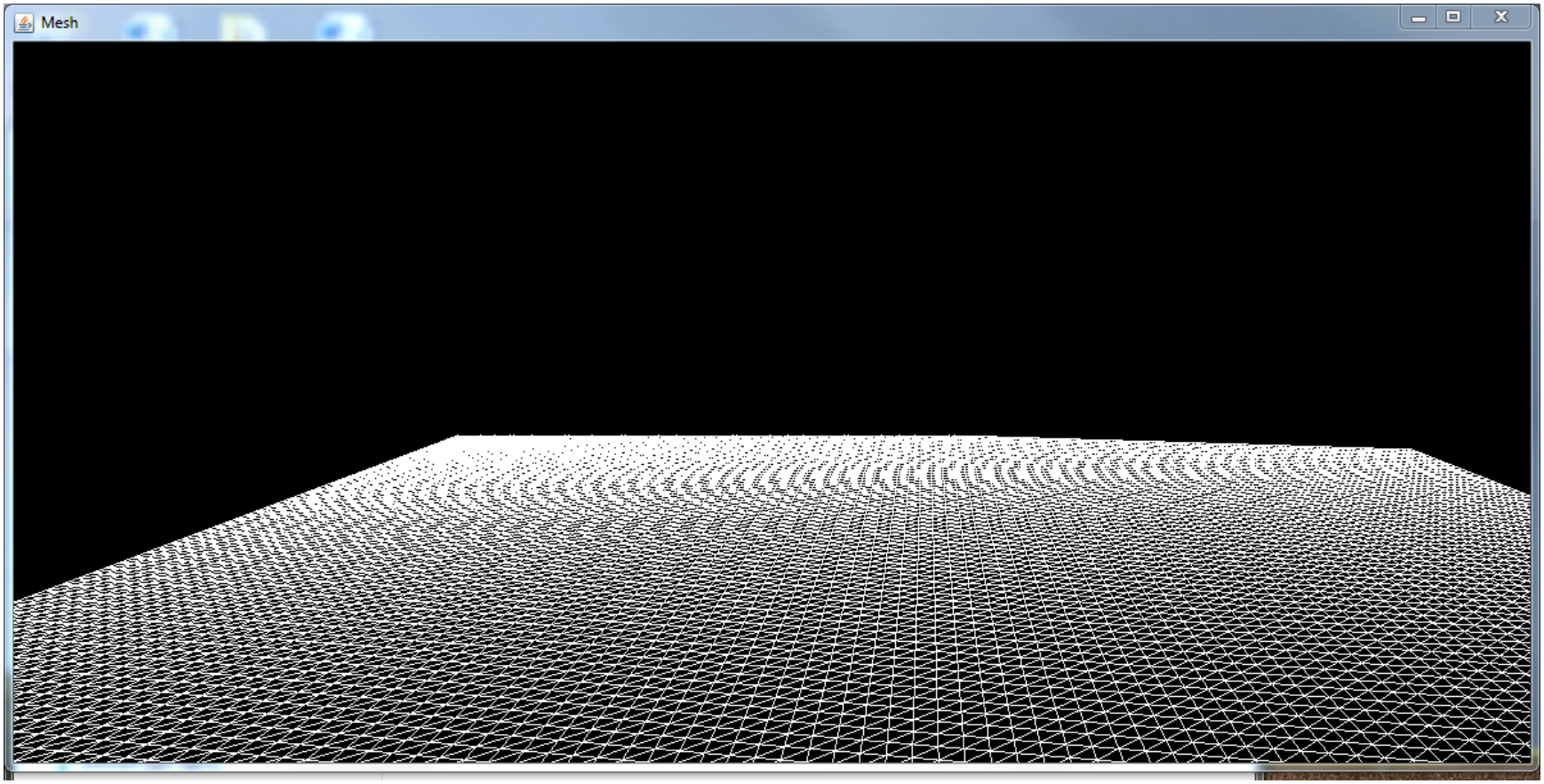
Terrain



Aufgabe 2 i

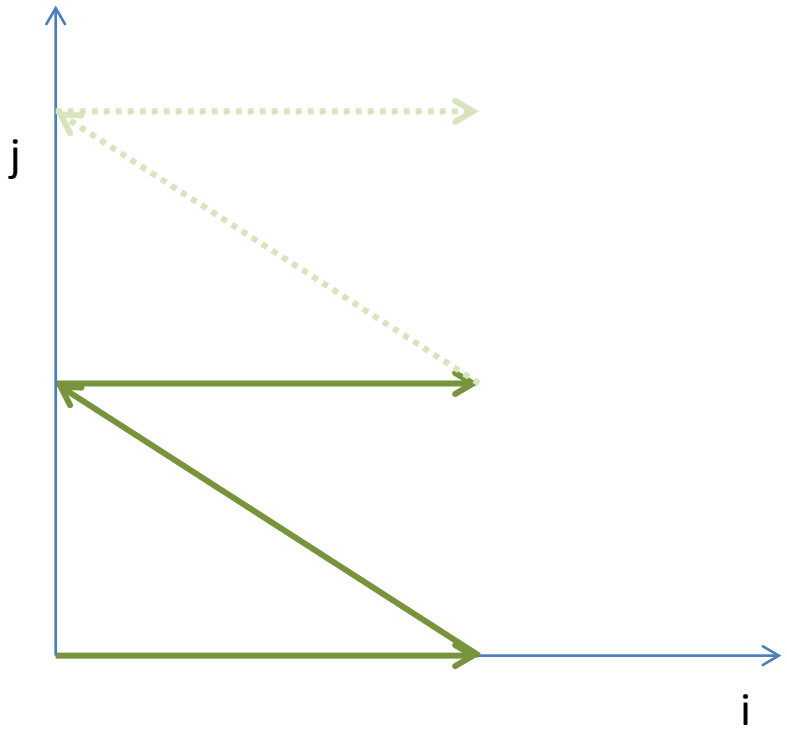
- GL_TRIANGLES
  - Zeichnet individuelle Dreiecke
  - Benötigt 3 Punkte
  - Bei n Punkten werden n/3 Dreiecke gezeichnet. Wenn n kein Vielfaches von 3 ist, werden übrige Punkte ignoriert.
- GL_TRIANGLE_STRIP
  - Sequenz von Dreiecken, Kanten werden geteilt
  - 1. Dreieck: Punkte 1-3
  - 2. Dreieck: Punkte 2-4 usw.
  - Bei n Punkten werden n-2 verbundene Dreiecke gezeichnet

Aufgabe 2 ii



Aufgabe 2 ii (2)

```
double[][] terrain = new double[100][100];
…
gl.glPolygonMode(GL2.GL_FRONT_AND_BACK, GL2.GL_LINE );//Wireframe-Modus
…
for (int i = 0; i<terrain.length-1;i++){
gl.glBegin(GL2.GL_TRIANGLE_STRIP);

        for (int j = 0; j<terrain[0].length-1;j++){

        gl.glVertex3d(i,0, j);

        gl.glVertex3d(i+1,0, j);

        gl.glVertex3d(i,0, j+1);

        gl.glVertex3d(i+1,0, j+1);

        }
gl.glEnd();
}
```



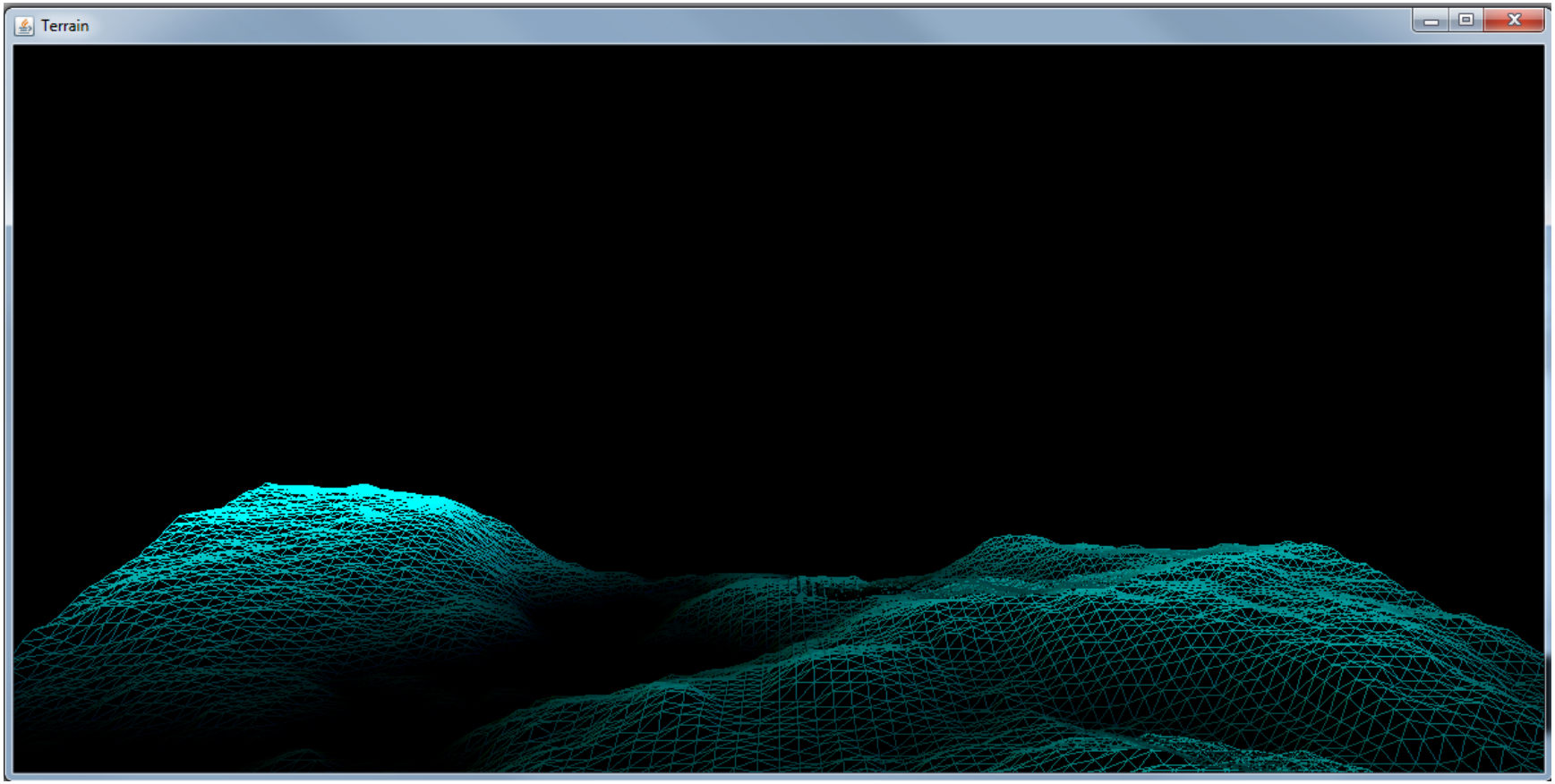
Aufgabe 2 iii

```
private void generateHeightField(){

int[] pixels =
ImageLoader.loadImage("http://www.medien.ifi.lmu.de/lehre/ss12/cg1/uebung/heightmap.png");

for(int i = 0; i<terrain.length;i++){
    for(int j = 0; j<terrain[0].length;j++){
    terrain[i][j] = (double)pixels[i+j*100]/5.0 - 5; // Individuelle Berechnung der Höhe aus den Grauwerten
    }
}

}
```

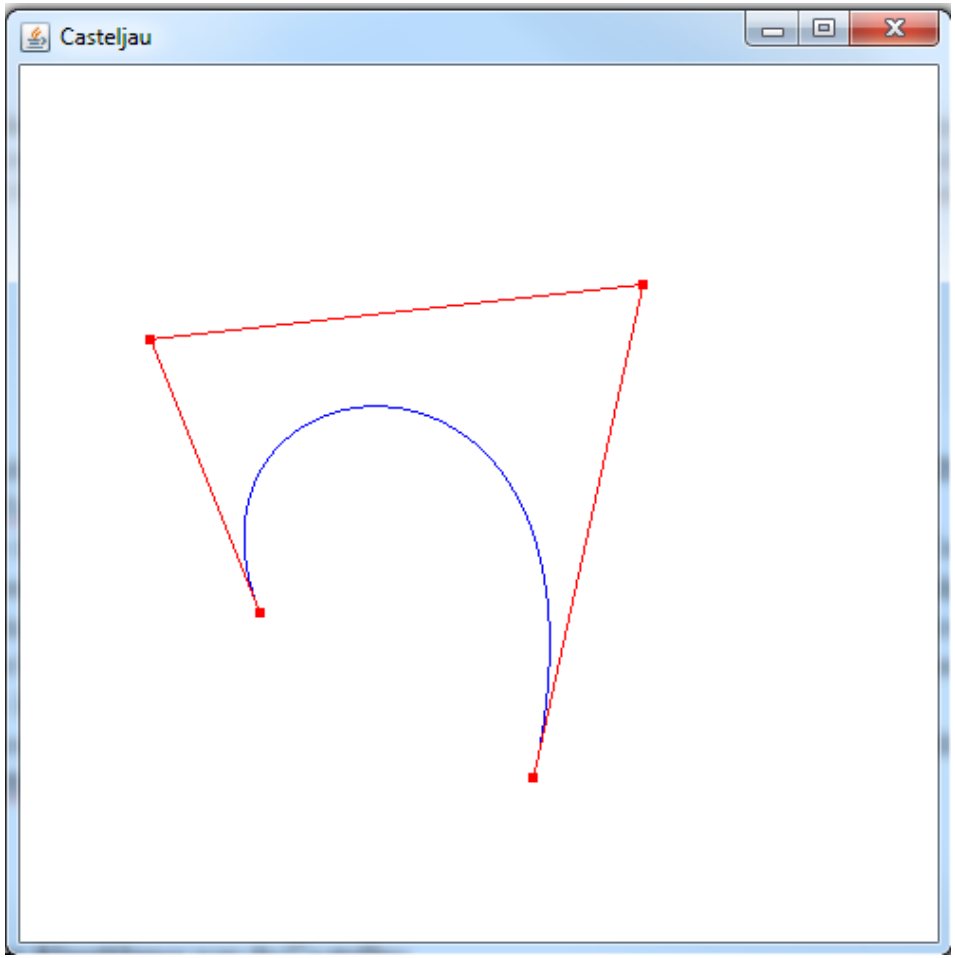
Aufgabe 2 iv

```
for (int i = 0; i<terrain.length-1;i++){
gl.glBegin(GL2.GL_TRIANGLE_STRIP);

        for (int j = 0; j<terrain[0].length-1;j++){

        gl.glColor3d(0, terrain[i][j]/10.0, terrain[i][j]/10.0); // Farbe
        gl.glVertex3d(i,terrain[i][j], j); // y-Koordinate bekommt entsprechenden Wert

        gl.glColor3d(0, terrain[i+1][j]/10.0, terrain[i][j]/10.0);
        gl.glVertex3d(i+1,terrain[i+1][j], j);

        gl.glColor3d(0, terrain[i][j+1]/10.0, terrain[i][j]/10.0);
        gl.glVertex3d(i,terrain[i][j+1], j+1);

        gl.glColor3d(0, terrain[i+1][j+1]/10.0, terrain[i][j]/10.0);
        gl.glVertex3d(i+1,terrain[i+1][j+1], j+1);
        }
gl.glEnd();
}
```

Aufgabe 2 v



Aufgabe 3



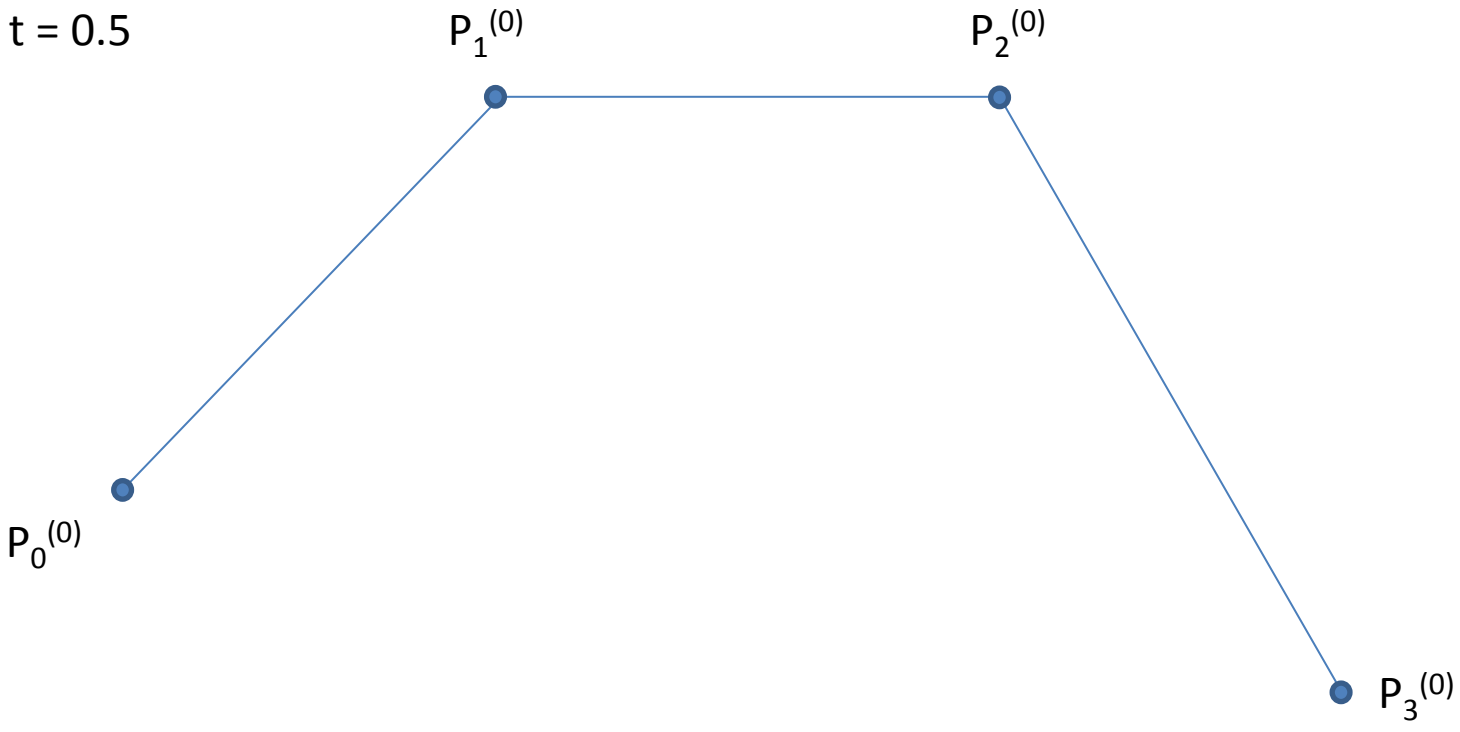
De-Casteljau-Algorithmus

- Pseudocode (Quelle: Wikipedia)

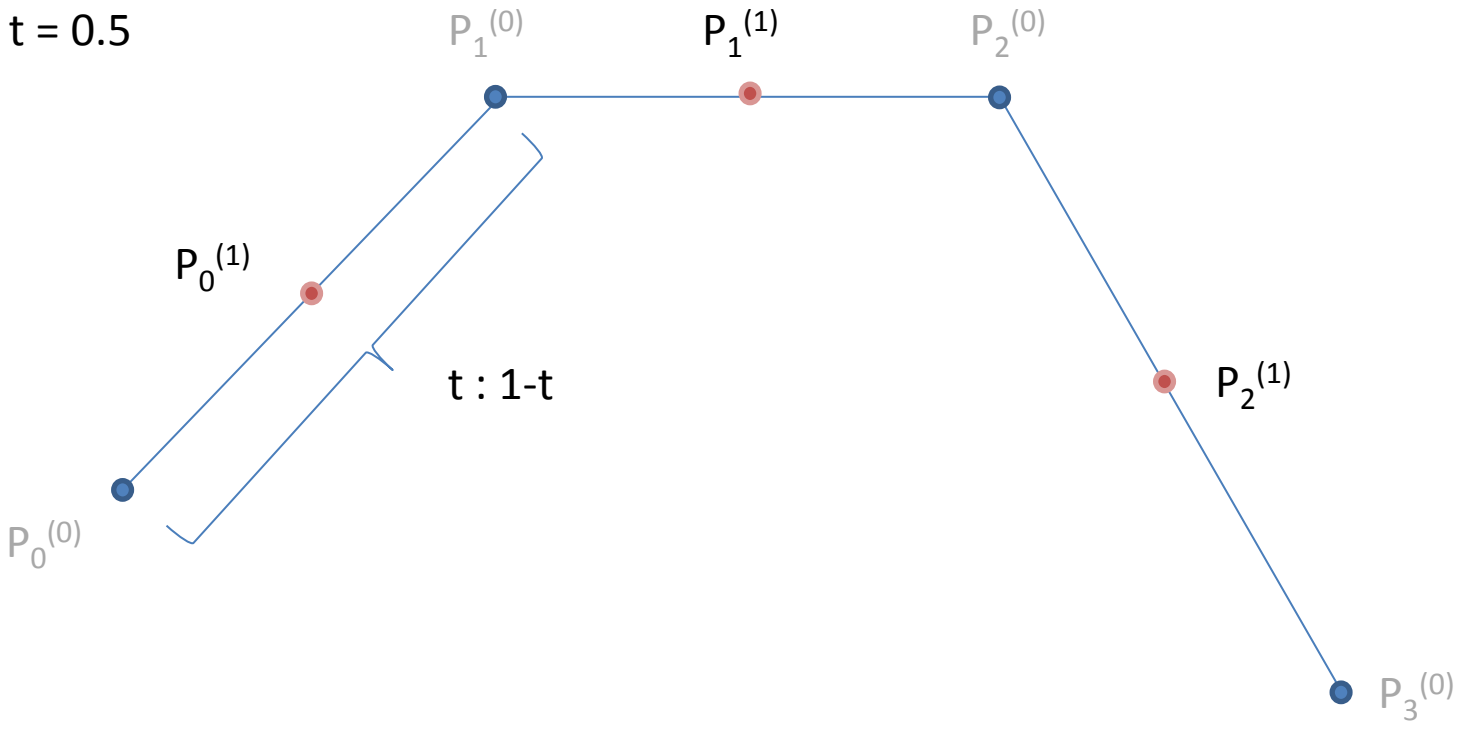
```
BEGIN
    FOR i:=0..n
        P_i^(0) := P_i

    FOR j:=1..n
        FOR i:=0..(n-j)
            // Unterteilung mit Faktor t
            P_i^(j) = (1 - t_0) · P_i^(j-1) + t_0 · P_(i+1)^(j-1)

    RETURN P_0^(n)
END
```

http://de.wikipedia.org/wiki/De-Casteljau-Algorithmus

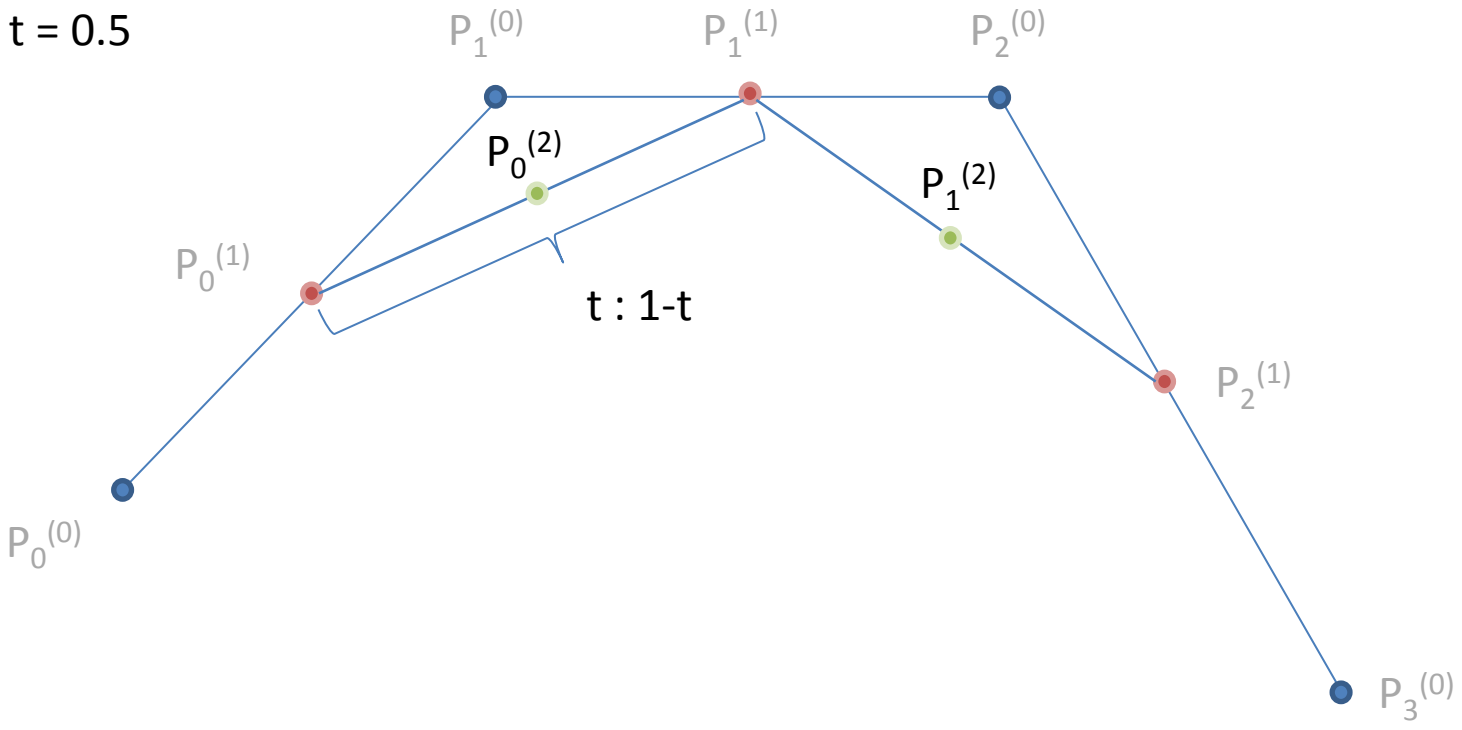
Schritt 1



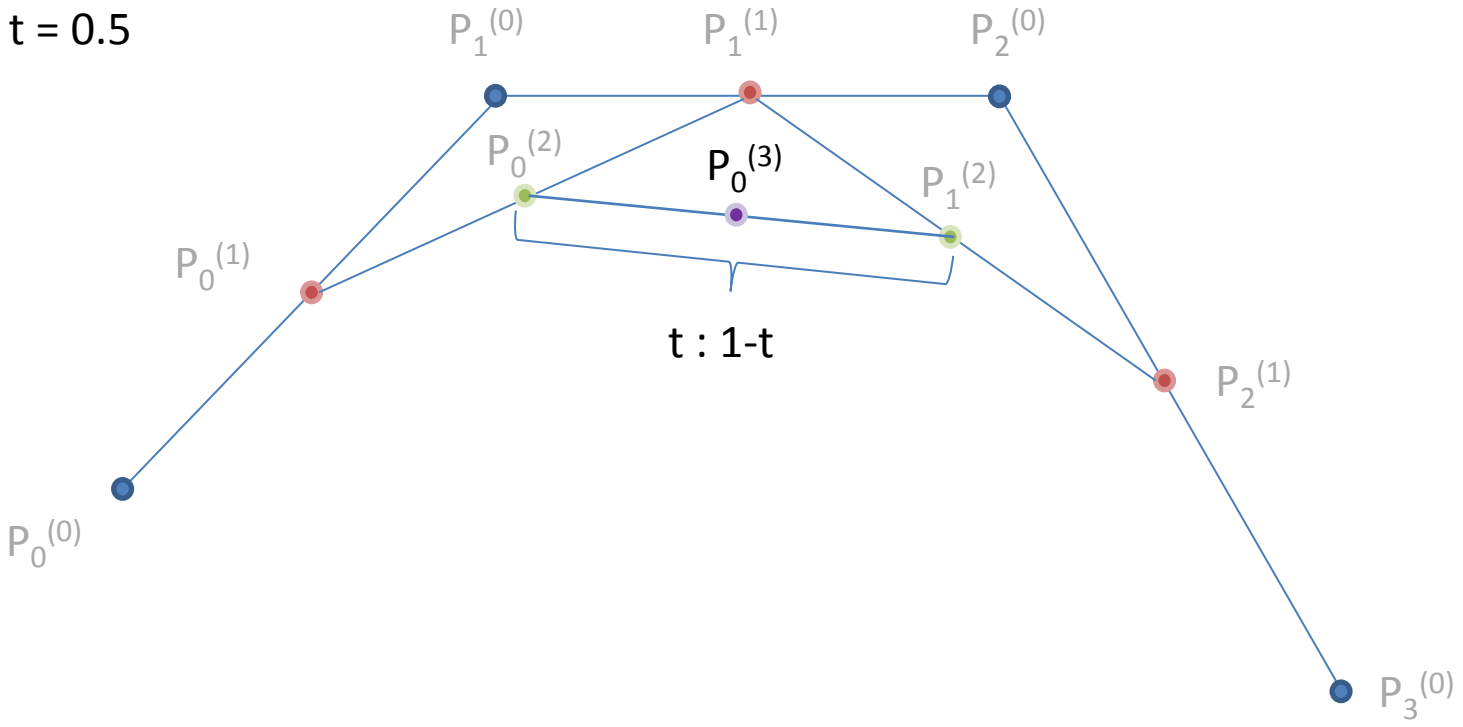
Schritt 2



Schritt 3



Schritt 4



$P_0^{(3)}$ wird zurückgegeben. Führt man diesen Algorithmus für viele t im Intervall von 0 bis 1 aus, erhält man eine Annäherung an die entsprechende Bézierkurve.

De-Casteljau-Algorithmus in Java (1)

```
public class Casteljau {
private ArrayList<Point2D.Double> ctrlPoints = new ArrayList<Point2D.Double>(); //Liste für die Kontrollpunkte
public Casteljau(){
        // vier Kontrollpunkte festlegen und der Liste hinzufügen
        ctrlPoints.add(new Point2D.Double(3,-2));
        ctrlPoints.add(new Point2D.Double(1,3));
        ctrlPoints.add(new Point2D.Double(10,4));
        ctrlPoints.add(new Point2D.Double(8,-5));
        //Berechnung der Kurvenannäherung
        evalCasteljau();
}

public void evalCasteljau(){
        for(double i = t;i<=1.0;i+=0.02){ //t von 0 bis 1 hochzählen, Granularität frei wählbar
                tmp = getCasteljauPoint(i); //Hier wird der Algorithmus durchgeführt
                curvePoints.add(tmp); //Das Ergebnis wird einer Liste von Kurvenpunkten angehängt
                }
}
…
```

De-Casteljau-Algorithmus in Java (2)

```
public Point2D.Double getCasteljauPoint(double t){

tempPoints.clear(); // temporäres Array leeren
int n = ctrlPoints.size()-1;

for(int i = 0; i<=n; i++){
        Point2D.Double p = new Point2D.Double(ctrlPoints.get(i).x,ctrlPoints.get(i).y);
        tempPoints.add(i, p); // Die vorgegebenen Kontrollpunkte temporär speichern
}

for (int k = 1; k<=n;k++){
        for(int i = 0; i<=n-k; i++){
                tempPoints.get(i).x = (1-t)*tempPoints.get(i).x + t*tempPoints.get(i+1).x;
                tempPoints.get(i).y = (1-t)*tempPoints.get(i).y + t*tempPoints.get(i+1).y;
        }
}
return tempPoints.get(0);
}
```

De-Casteljau-Algorithmus in Java (3)

```
class SceneView implements GLEventListener {

    Casteljau casteljau = new Casteljau();
    ArrayList<Point2D.Double> ctrlPoints = casteljau.getCtrlPoints();
    ArrayList<Point2D.Double> curvePoints = casteljau.getCurvePoints();
    …
public void display(GLAutoDrawable drawable) {
    GL2 gl = drawable.getGL().getGL2();
    gl.glClear(GL2.GL_COLOR_BUFFER_BIT); // clear background
    //Die Kurvenpunkte als LINE_STRIP zeichnen
    gl.glColor3d(0, 0, 1);//blue
    gl.glBegin(GL2.GL_LINE_STRIP);
    for(int i = 0; i<curvePoints.size();i++){
    gl.glVertex3d(curvePoints.get(i).x, curvePoints.get(i).y, 0);
    …
    }
    gl.glEnd();}
    …
```