CARL VON OSSIETZKY universität OLDENBURG

Torus-Knoten

Johannes Diemke

Übung im Modul OpenGL mit Java
Wintersemester 2010/2011

Torus-Knoten

(p, q)-Torus-Knoten

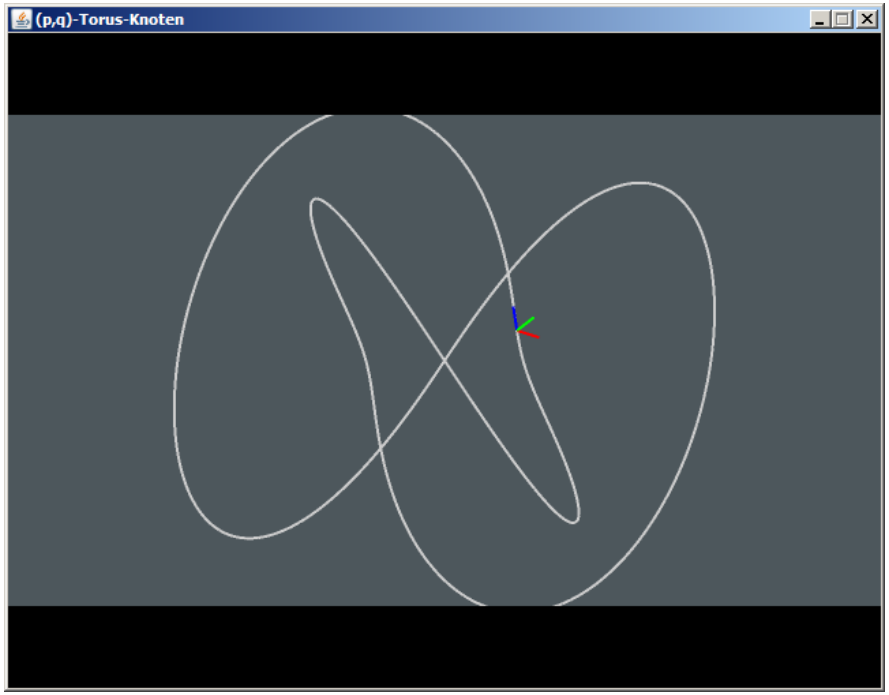
- Wird im $\mathbb{R}^3$ durch eine Raumkurve $r(\varphi)$ beschrieben
- p und q müssen relativ prim zueinander sein
  - Zwei natürliche Zahlen sind *teilerfremd* oder *relativ prim*, wenn es keine natürliche Zahl außer der Eins gibt, die beide Zahlen teilt
  - Zwei Zahlen a und b sind dann teilerfremd, wenn ihr $ggT(a, b) = 1$ ist

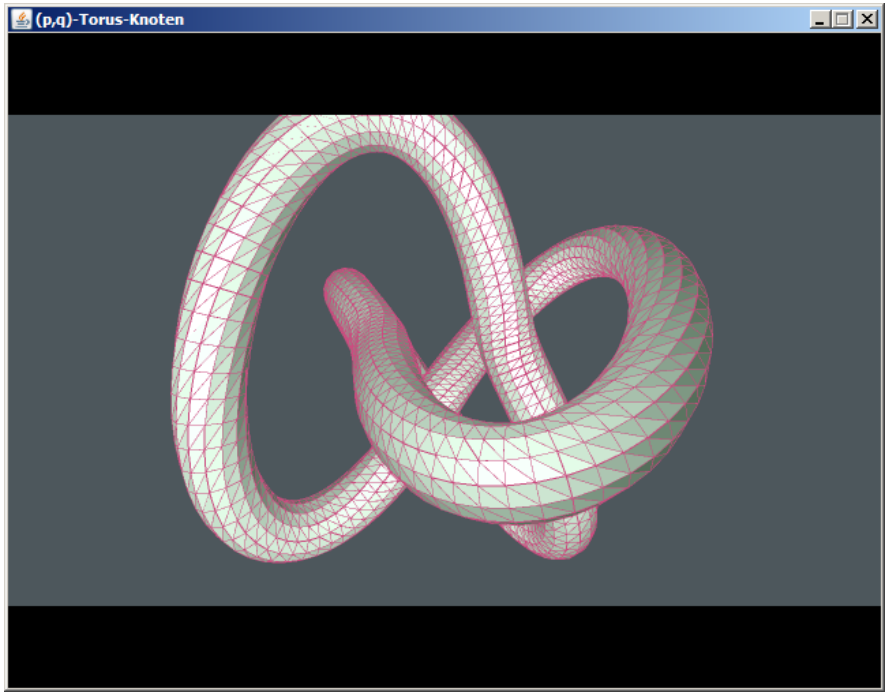
$$r(\varphi) = \begin{pmatrix} (\cos(q\varphi) + 2) \cdot \cos(p\varphi) \\ (\cos(q\varphi) + 2) \cdot \sin(p\varphi) \\ \sin(q\varphi) \end{pmatrix}, \quad 0 \leq \varphi \leq 2\pi$$

Torus-Knoten

Hülle entlang einer Raumkurve

1. Berechnen des Frenet-Rahmen (Dreibein)
2. Aufspannen von Polygonen





Torus-Knoten

Hülle entlang einer Raumkurve (Forts.)

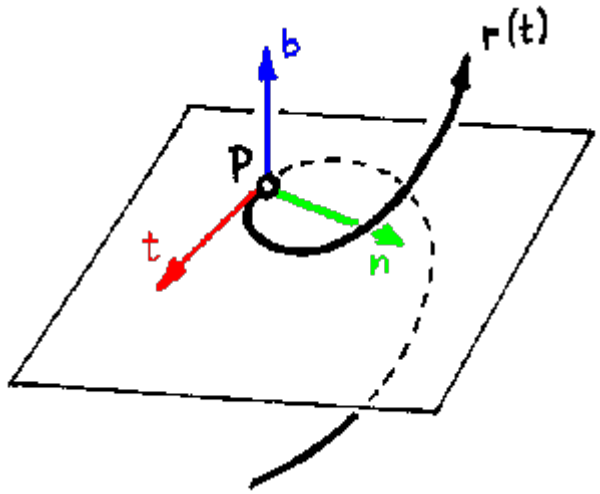
- Beispielhaft anhand folgender Raumkurve

$$r(\varphi) = \begin{pmatrix} 0.5 \cdot (2 + \sin(q\varphi)) \cdot \cos(p\varphi) \\ 0.5 \cdot (2 + \sin(q\varphi)) \cdot \cos(q\varphi) \\ 0.5 \cdot (2 + \sin(q\varphi)) \cdot \sin(p\varphi) \end{pmatrix}, \quad 0 \leq \varphi \leq 2\pi$$

Torus-Knoten

Frenet-Rahmen

- Bildet eine Orthonormalbasis
- Ist an jedem Punkt der Kurve gesondert definiert (begleitendes Dreibein)



- Idee: Entlang der Raumkurve Ringe in der von der Einheitsnormale und Einheitsbinormale aufgespannten Ebene zeichnen

Torus-Knoten

Frenet-Rahmen (Forts.)

Definition (Frenet-Rahmen)

Sei $I \subset \mathbb{R}^3$ ein Intervall und sei $r : I \to \mathbb{R}^3$ eine Raumkurve von der wir annehmen, dass sie differenzierbar und frei von Wendepunkten ist. Das durch die Raumkurve gleitende Dreibein, auch Frenet-Rahmen genannt, ist eine Orthonormalbasis bestehend aus den drei Vektoren $T(\varphi)$, $N(\varphi)$ und $B(\varphi)$ definiert durch:

$$T(\varphi) = \frac{r'(\varphi)}{\|r'(\varphi)\|}, \quad N(\varphi) = \frac{T'(\varphi)}{\|T'(\varphi)\|}, \quad B(\varphi) = T(\varphi) \times N(\varphi)$$

Torus-Knoten

Frenet-Rahmen (Forts.)

- Berechnen der beiden Ableitungen $r'(\varphi)$ und $r''(\varphi)$

$$r'(\varphi) = \begin{pmatrix} 0.5\cos(q\varphi)q\cos(p\varphi) - 0.5(2+\sin(q\varphi))\sin(p\varphi)p \\ 0.5\cos(q\varphi)^2 q - 0.5(2+\sin(q\varphi))\sin(q\varphi)q \\ 0.5\cos(q\varphi)q\sin(p\varphi) + 0.5(2+\sin(q\varphi))\cos(p\varphi)p \end{pmatrix}$$

$$r''(\varphi) = \begin{pmatrix} -0.5\sin(q\varphi)q^2\cos(p\varphi) - \cos(q\varphi)q\sin(p\varphi)p - 0.5(2+\sin(q\varphi))\cos(p\varphi)p^2 \\ -1.5\cos(q\varphi)q^2\sin(q\varphi) - 0.5(2+\sin(q\varphi))\cos(qt)q^2 \\ -0.5\sin(q\varphi)q^2\sin(p\varphi) + \cos(q\varphi)q\cos(p\varphi)p - 0.5(2+\sin(q\varphi))\sin(p\varphi)p^2 \end{pmatrix}$$

Torus-Knoten

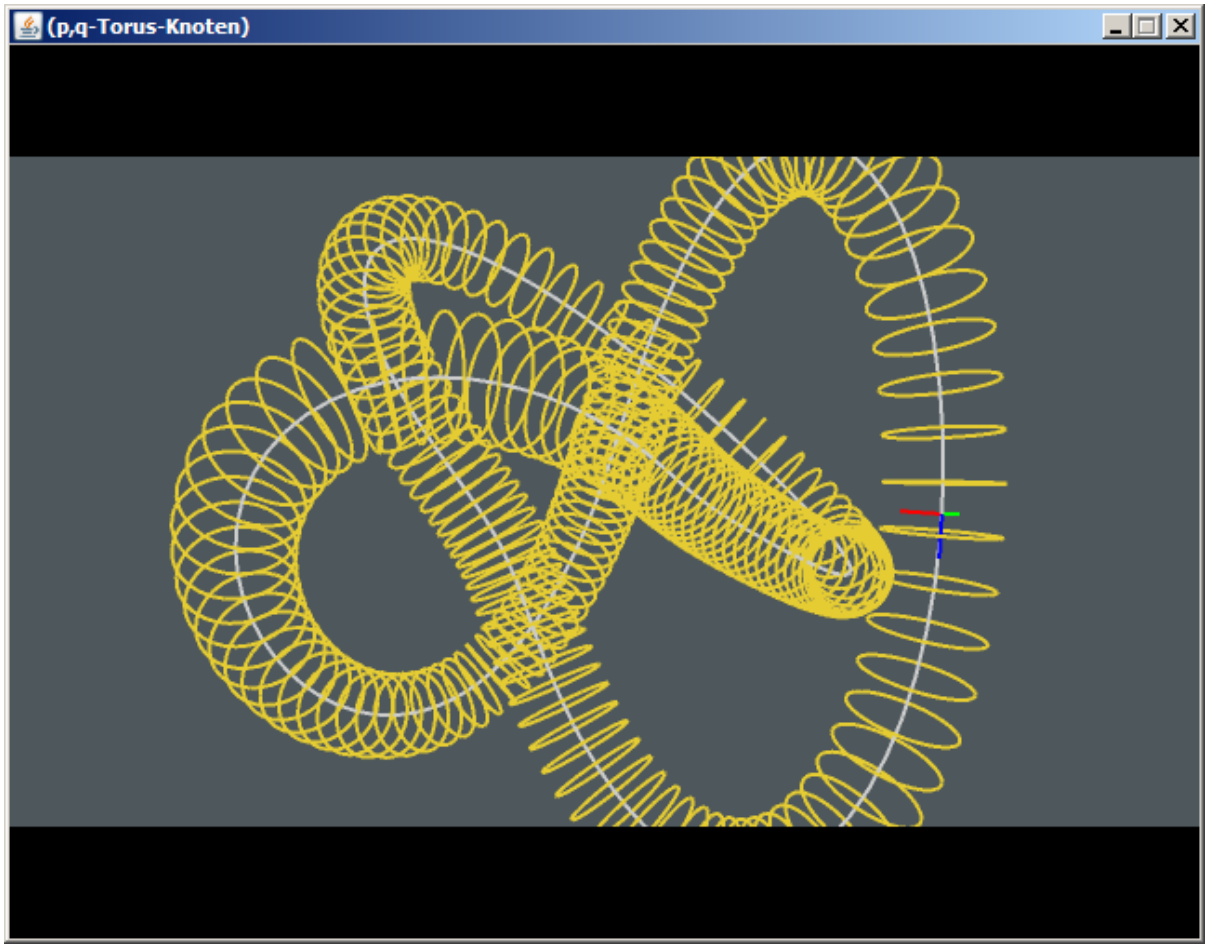
Aufspannen von Ringen

- Aufspannen von Ringen in der von der Einheitsnormale und Einheitsbinormale aufgespannten Ebene

```
gl.glBegin(GL2.GL_LINE_LOOP);
    for(int j=0; j < segments; j++) {
        x = radius * Math.cos(2 * PI / segments * j));
        y = radius * Math.sin(2 * PI / segments * j));

        // Affine Transformation
        point = normal(t)*x + binormal(t)*y + curve(t);
        gl.glVertex3f(point.x, point.y, point.z);
    }
gl.glEnd();
```

Torus-Knoten

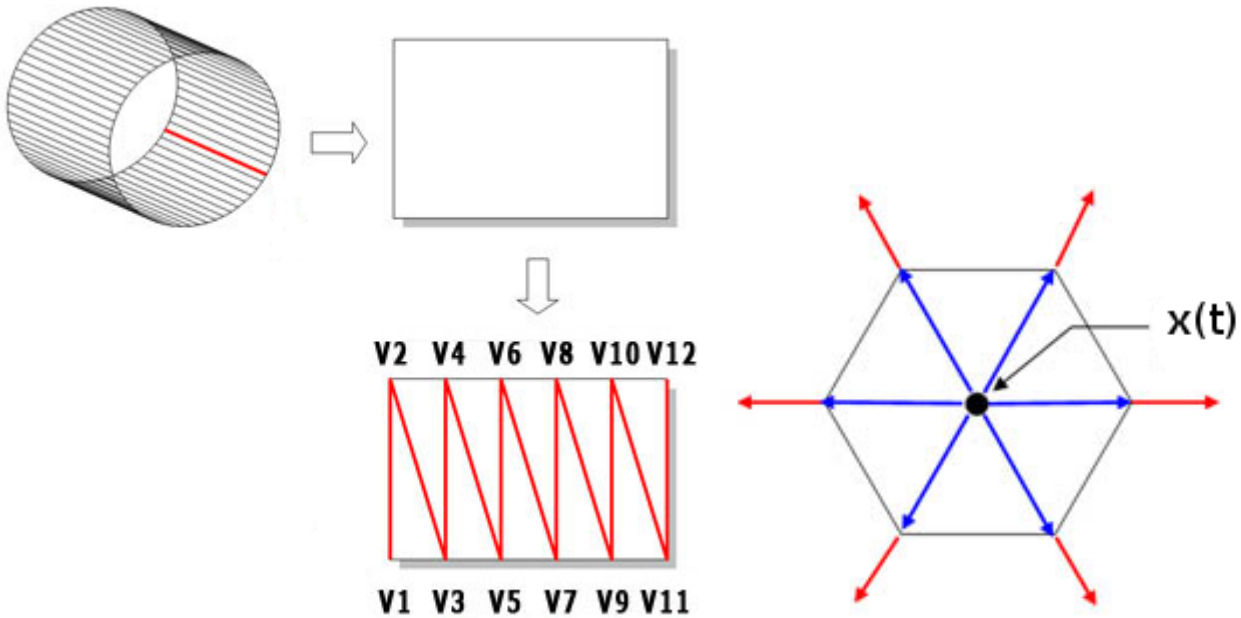
Aufspannen von Ringen (Forts.)



Torus-Knoten

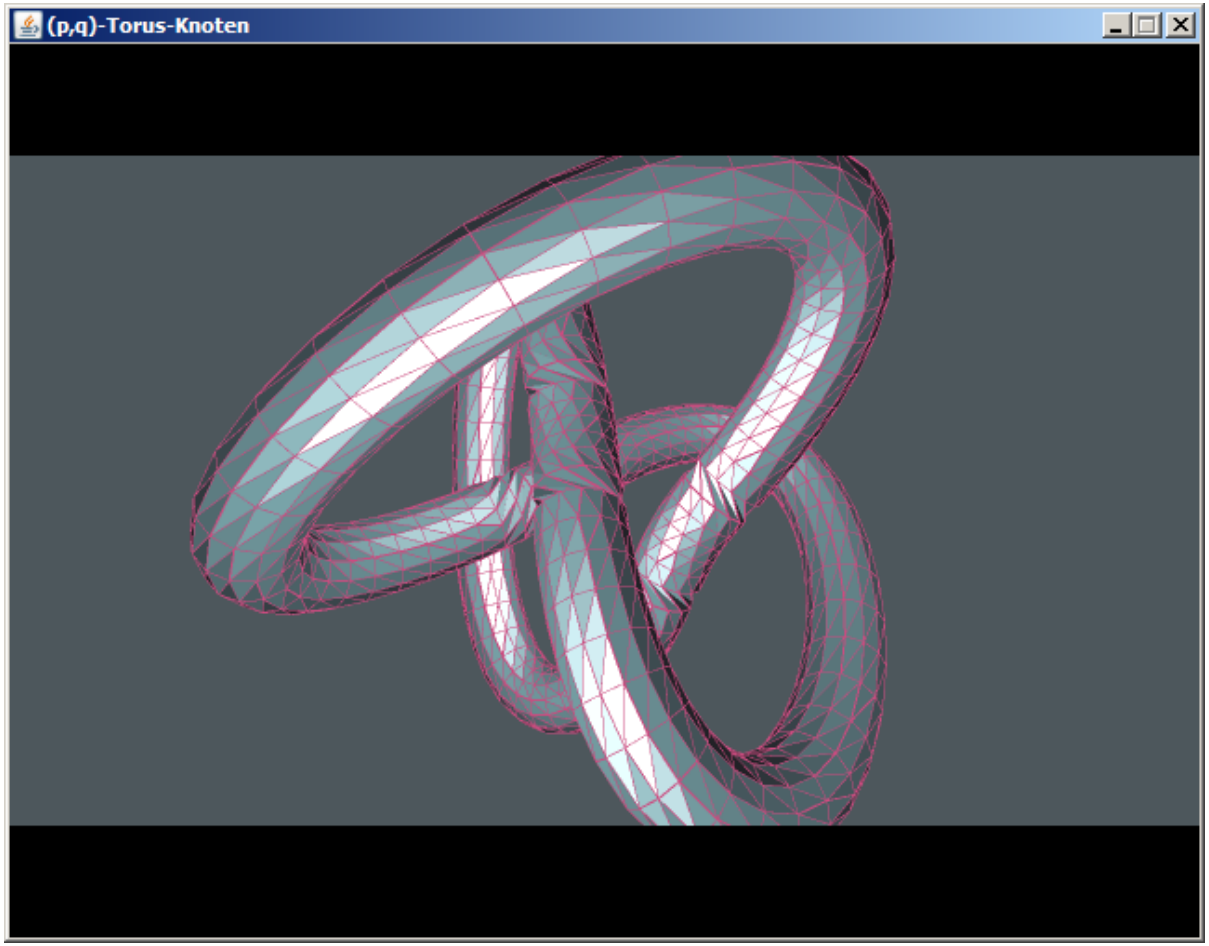
Aufspannen von Polygonen

- Aufspannen von Polygonen zwischen den Ringen und Berechnung der Vertex-Normalen



Torus-Knoten

Zwischenstand



Torus-Knoten

Zwischenstand (Forts.)

- Die Normale und Binormale machen sprunghaft 180-Grad-Drehungen in den Wendepunkten der Kurve
- Tangente und Normale sind in den Wendepunkten parallel

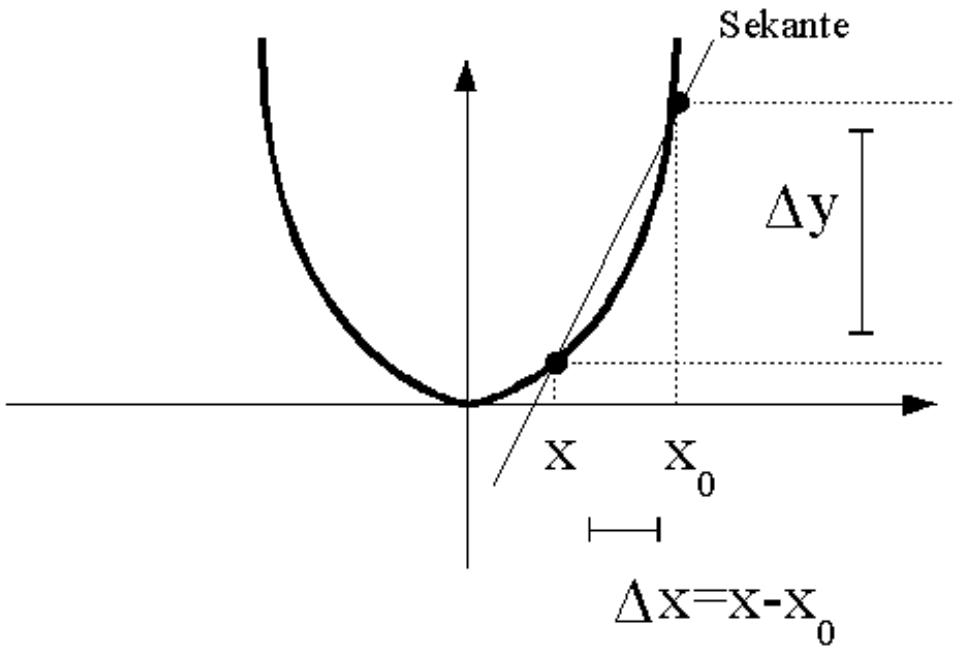
Lösung

- Dreibein nicht über zweite Ableitung konstruieren
- Approximation über beliebigen Vektor, der nie parallel ist

Torus-Knoten

Approximation

- Vektor muss entsprechend der Raumkurve gewählt werden, damit er nie parallel zur Tangente ist
- Weiterhin kann im selben Zug auch gleich die Tangente selbst approximiert werden $\rightarrow$ *forward difference*



Torus-Knoten

Approximation (Forts.)

- symbolische vs. numerische Ableitung (Forward Difference):

```
point1 := curve(t);
point2 := curve(t + delta);

unitTangent  := normalize(point2 - point1);
unitNormal   := crossProduct(unitTangent, vec3(0, 1, 0));
unitBinormal := crossProduct(unitTangent, unitNormal);
```

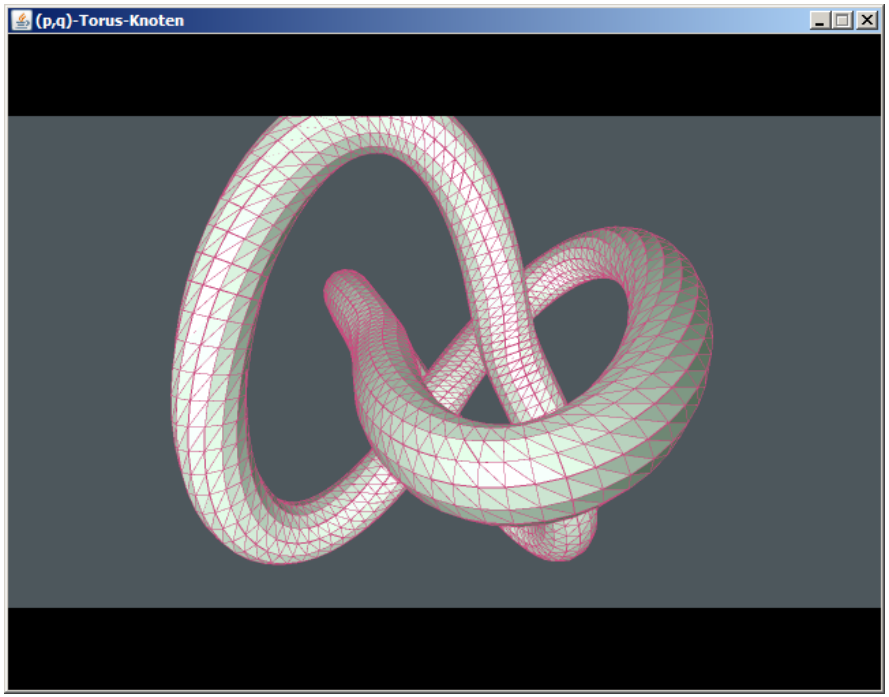
- oder alternativ (Central Difference):

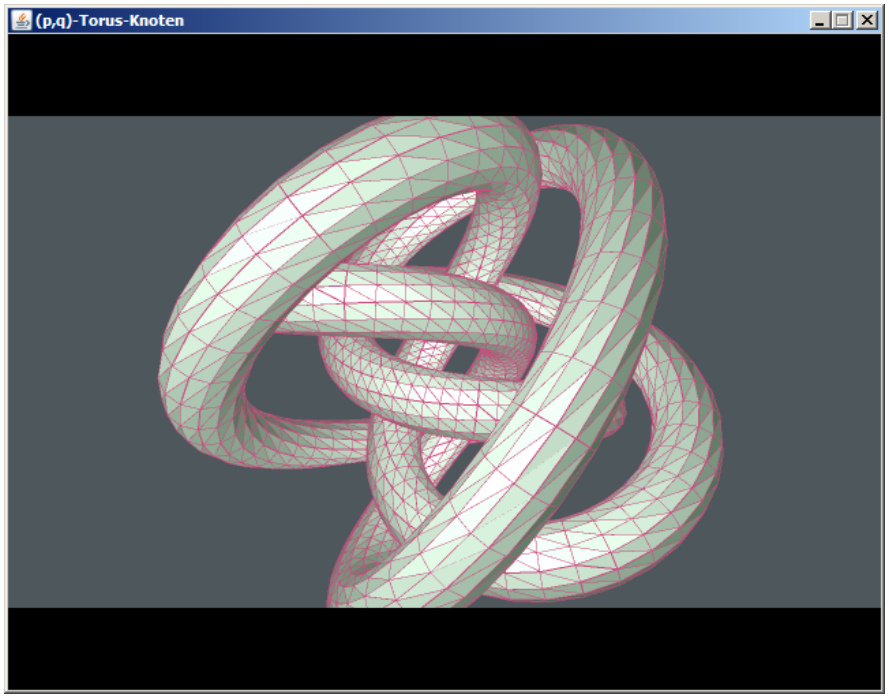
```
point1 := curve(t - delta/2);
point2 := curve(t + delta/2);

unitTangent  := normalize(point2 - point1);
unitNormal   := crossProduct(unitTangent, vec3(0, 1, 0));
unitBinormal := crossProduct(unitTangent, unitNormal);
```

Torus-Knoten

Ergebnis

- $(2, 3)$- und $(5, 3)$-Torus-Knoten mit Flat-Shading





Torus-Knoten

Ergebnis (Forts.)

- $(2, 3)$-Torus-Knoten mit Phong-Shading und Schatten

Literatur

- Dave Shreiner
  OpenGL Programming Guide
  http://www.opengl-redbook.com/
- Richard S. Wright, Benjamin Lipchak und Nicholas Haemel
  OpenGL SuperBibel
  http://www.starstonesoftware.com/OpenGL/
- Randi J. Rost
  OpenGL Shading Language
  http://www.3dshaders.com/
- Tomas Akenine-Möller, Eric Haines und Naty Hoffman
  Real-Time Rendering
  http://www.realtimerendering.com/

Literatur

- Edward Angel
  Interactive Computer Graphics
  http://www.cs.unm.edu/~angel/
- Gerald Farin und Dianne Hansford
  Practical Linear Algebra
  http://www.farinhansford.com/books/pla/
- Fletcher Dunn und Ian Parberry
  3D Math Primer for Graphics and Game Development
  www.gamemath.com/