3D-SIMULATION VON BEWEGUNGEN ENTLANG GEODÄTISCHER LINIEN AUF DER ERDKUGEL MIT JAVA

DOZENT: UWE HAHNE

28.06.2023

MATHEMATIK UND SIMULATION, MIB 2, SOSE 23

NIKLAS RIEDINGER, CLARA GEY

AUFGABENSTELLUNG

- Gegeben sind zwei Koordinaten P und Q auf der Erdkugel
- Gesucht wird die kürzeste erdoberflächennahe (Flug-)Verbindung zwischen P und Q
- Diese Verbindung ist in einer Java-3D-Anwendung darzustellen und zu animieren

GPS-Koordinaten

↓

3D-Koordinaten

↓

Bildschirmkoordinaten

GEOGRAPHISCHE KUGELKOORDINATEN ZU 3D KOORDINATEN

$$(\theta, \varphi) \mapsto \begin{pmatrix} r \cdot \cos(\theta) \cdot \cos(\varphi) \\ r \cdot \cos(\theta) \cdot \sin(\varphi) \\ r \cdot \sin(\theta) \end{pmatrix}$$

GPS-Koordinaten

3D-Koordinaten

θ := Breitengrad
φ := Längengrad
r := Kugelradius

3D PUNKTE ZU BILDSCHIRMPUNKTEN

Punkte im Raum werden in homogener Koordinatenform

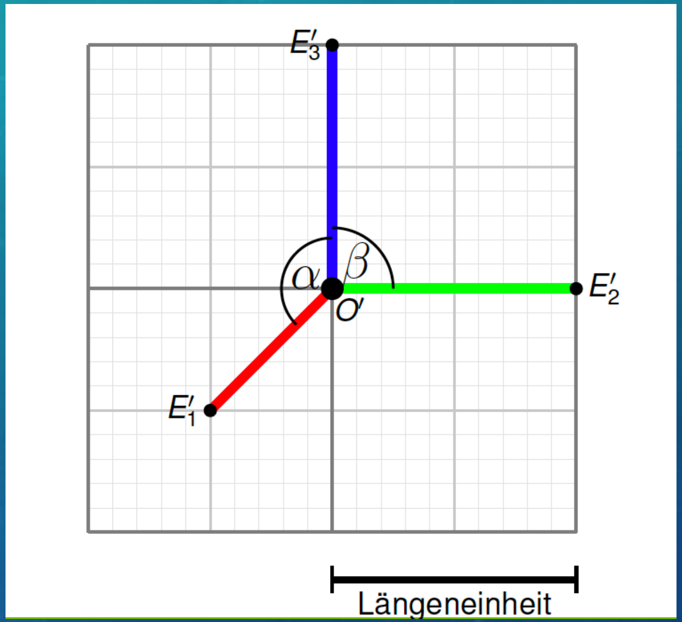
$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix}$$

von rechts an die Projektionsmatrix

$$M(s_1, \alpha) = \begin{bmatrix} -s_1 \cdot \sin(\alpha) & 1 & 0 & \frac{w}{2} \\ -s_1 \cdot \cos(\alpha) & 0 & -1 & \frac{h}{2} \end{bmatrix}$$

multipliziert und ergeben einen Punkt auf dem Bildschirm, wobei der Ursprung in der Mitte abgebildet wird.

s_1:= Skalierungsfaktor von $\overrightarrow{O'E'_1}$
w := Bildschirmbreite
h := Bildschirmhöhe



Beispiel mit den Werten $s_1 = \frac{\sqrt{2}}{2}$ und $\alpha = 135°$

3D-Koordinaten

Bildschirmkoordinaten

GEODÄTISCHE LINIE

- Bestimmung des Winkels δ zwischen den Ortsvektoren der beiden Punkte P und Q:

$$\cos(\delta) = \frac{\vec{p} \cdot \vec{q}}{\|\vec{p}\| \cdot \|\vec{q}\|}$$

- Aus diesem Winkel parametrisieren wir eine Kurve auf dem Äquator:

$$g_0(t) = \begin{pmatrix} r \cdot \cos(t) \\ r \cdot \sin(t) \\ 0 \end{pmatrix}, t \in [0, \delta]$$

GEODÄTISCHE LINIE

- Diese Kurve wird einer Drehung D unterworfen, so dass
  - der Punkt $g_0(0)$ auf den Punkt P
  - und der Punkt $g_0(\delta)$ auf den Punkt Q

  abgebildet wird

- Wir erhalten die Parametrisierung unserer geodätischen Kurve:

$$g(t) = D(g_0(t)), t \in [0, \delta]$$

GEODÄTISCHE LINIE

- Wir benötigen eine Drehmatrix $[D]$, welche die Einheitsvektoren folgendermaßen abbildet:
  - $\hat{x}$ auf den Einheitsvektor $\hat{p} \coloneqq \frac{\vec{p}}{\|\vec{p}\|}$
  - $\hat{z}$ auf den Einheitsvektor $\hat{n} \coloneqq \frac{\vec{p}\times\vec{q}}{\|\vec{p}\times\vec{q}\|}$
  - $\hat{y}$ auf den Einheitsvektor $\hat{u} \coloneqq \frac{\vec{n}\times\vec{p}}{\|\vec{n}\times p\|}$
- Somit ergibt sich die Matrix $[D] = [(\hat{p}) \quad (\hat{u}) \quad (\hat{n})]$

GEODÄTISCHE LINIE

$$g(t) = D(g_0(t)) = [(\hat{p}) \quad (\hat{u}) \quad (\hat{n})] \cdot \begin{pmatrix} r \cdot cos(t) \\ r \cdot sin(t) \\ 0 \end{pmatrix}$$

$$= r \cdot \cos(t) \cdot \hat{p} + r \cdot \sin(t) \cdot \hat{u}$$

KLASSEN FÜR VECTOR, MATRIX & COORDINATE

```java
public class Vector {
  private double x, y, z;
}
```

```java
public class Matrix {
  private double[][] data;
}
```

```java
public class Coordinate {
  private double longitude;
  private double latitude;
  private double r;
}
```

MATHEMATISCHE FORMELN IN JAVA

Kartesische Koordinaten aus Kugelkoordinaten:

$$(\theta, \varphi) \mapsto \begin{pmatrix} r \cdot \cos(\theta) \cdot \cos(\varphi) \\ r \cdot \cos(\theta) \cdot \sin(\varphi) \\ r \cdot \sin(\theta) \end{pmatrix}$$

```java
public Vector toCartesian() {
  return new Vector(
    this.r * Math.cos(Math.toRadians(this.latitude)) * Math.cos(Math.toRadians(this.longitude)),
    this.r * Math.cos(Math.toRadians(this.latitude)) * Math.sin(Math.toRadians(this.longitude)),
    this.r * Math.sin(Math.toRadians(this.latitude))
  );
}
```

MATHEMATISCHE FORMELN IN JAVA

3D-Punkte zu Bildschirmpunkten:

$$\begin{bmatrix} -s_1 \cdot \sin(\alpha) & 1 & 0 & \frac{w}{2} \\ -s_1 \cdot \cos(\alpha) & 0 & -1 & \frac{h}{2} \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix}$$

```java
public Vector doScreenProjection(Vector v) throws Exception {
        Matrix vm = new Matrix(new double[][]{
                {v.x()},
                {v.y()},
                {v.z()},
                {1.0}
        });

        Matrix result = this.multiply(vm);

        return new Vector(
                result.data[0][0],
                result.data[1][0],
                0
        );
    }
```

MATHEMATISCHE FORMELN IN JAVA

$$\cos(\delta) = \frac{\vec{p} \cdot \vec{q}}{\|\vec{p}\| \cdot \|\vec{q}\|}$$

$$\hat{p} \coloneqq \frac{\vec{p}}{\|\vec{p}\|}$$

$$\hat{n} \coloneqq \frac{\vec{p} \times \vec{q}}{\|\vec{p} \times \vec{q}\|}$$

$$\hat{u} \coloneqq \frac{\vec{n} \times \vec{p}}{\|\vec{n} \times p\|}$$

$$g(t) = r \cdot \cos(t) \cdot \hat{p} + r \cdot \sin(t) \cdot \hat{u}$$

```java
Vector p = coordStart.toCartesian();
Vector q = coordEnd.toCartesian();

double delta = Math.acos(
      p.dot(q) / (p.length() * q.length())
);

Vector p_u = p.copy().normalize();
Vector n_u = p.cross(q).normalize();
Vector u_u = n_u.cross(p_u).normalize();

// ...

Vector v = p_u.multiply(r * Math.cos(t)).add(u_u.multiply(r * Math.sin(t)));
```