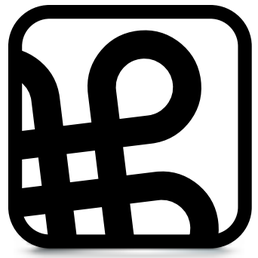


Macoun’11

Grundlagen der 3D-Grafik

Daniel Dönigus

Ablauf

- Einführung
- Primitive/Modelle
- 3D-Transformationen
- Projektionsmatrizen
- Beleuchtung
- Texturen

Einleitung

Um was gehts?

- Abbildung von 3D-Objekten auf einem 2D-Bildschirm
- Behandlung von wichtigen Themen
- Formeln nachschlagen

Frameworks

- DirectX
- OpenGL
  - OpenGL 1.x (Fixed Rendering Pipeline)
  - OpenGL 2.0 (Full Programmable)
  - GLKit

Primitive & Modelle

Grafische Primitive

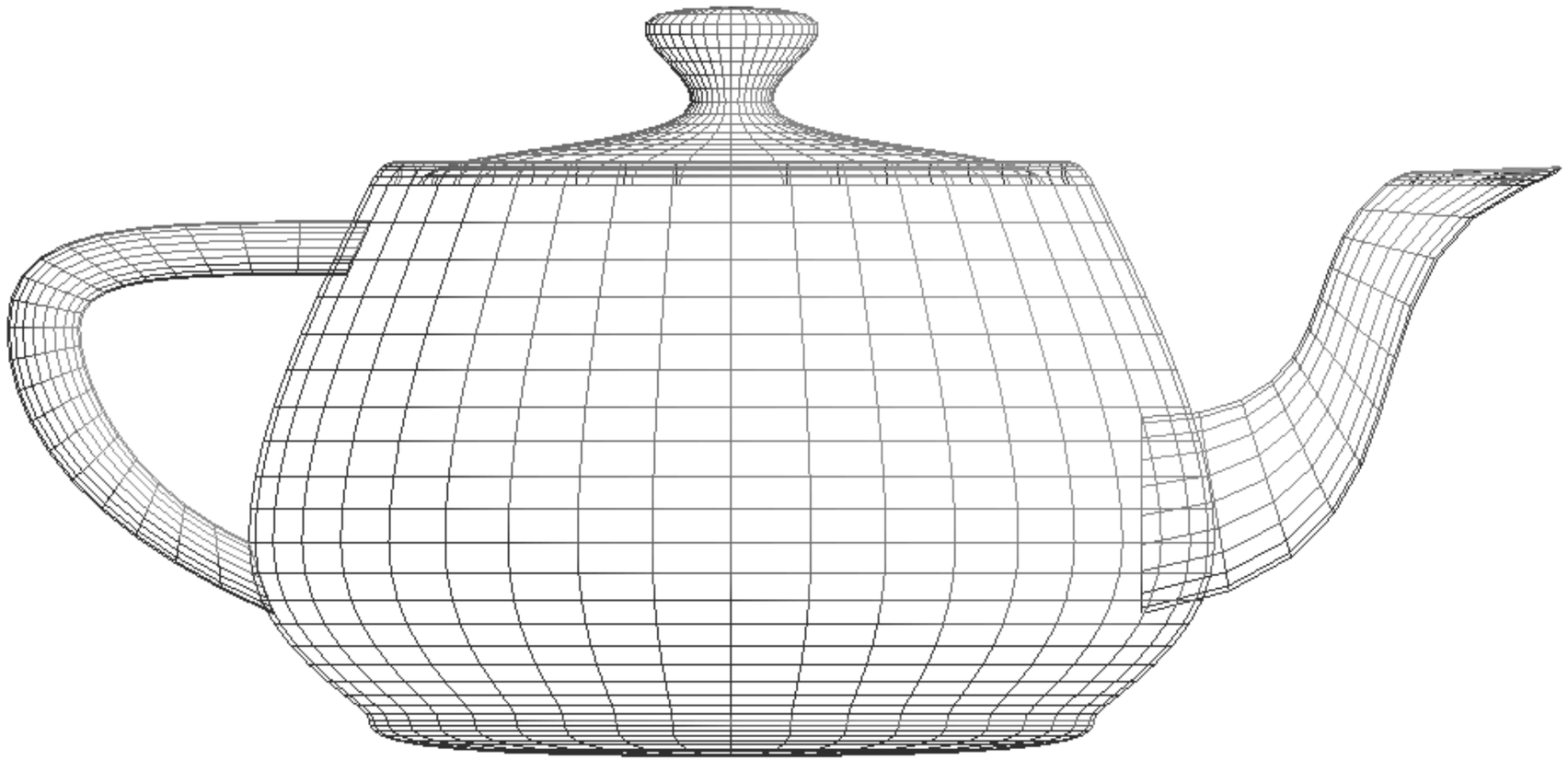
- Punkte (Vertex)
- Linien
- Dreiecke

```
GL_POINT
```

```
GL_LINE
```

```
GL_TRIANGLE
```

Modell



Modell

```
glBegin(GL_TRIANGLES);
  glVertex3f(-0.5, 0.5, 6.0);
  glVertex3f( 0.0, 0.0, 7.0);
  glVertex3f( 0.5, 0.5, 6.0);

  glVertex3f( 0.5,-0.5, 6.0);
  glVertex3f( 0.0, 0.0, 7.0);
  glVertex3f(-0.5,-0.5, 6.0);
glEnd();
```

Lineare Algebra

Operanden

Vektor

$$\vec{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$$

Matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$$

Matrix-Vektor-Multiplikation

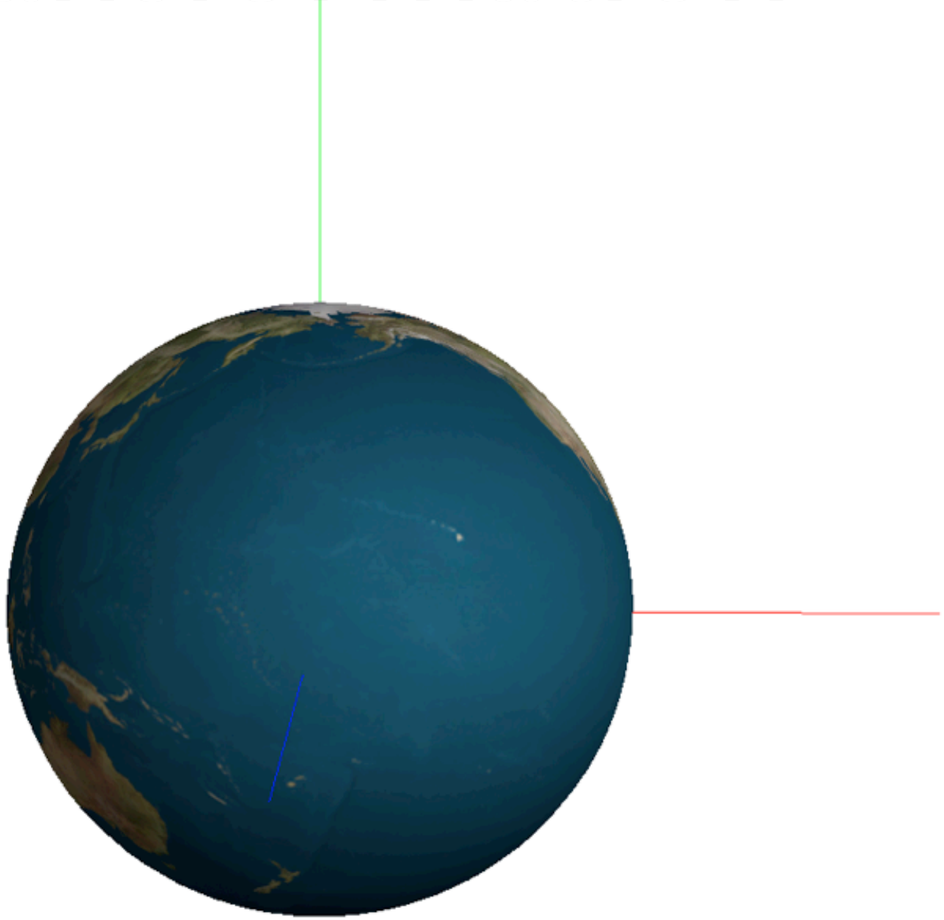
$$\mathbf{A} \cdot \vec{v} = \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} a_{11}v_x + a_{21}v_y + a_{31}v_z \\ a_{12}v_x + a_{22}v_y + a_{32}v_z \\ a_{13}v_x + a_{23}v_y + a_{33}v_z \end{pmatrix}$$

Vektor-Addition

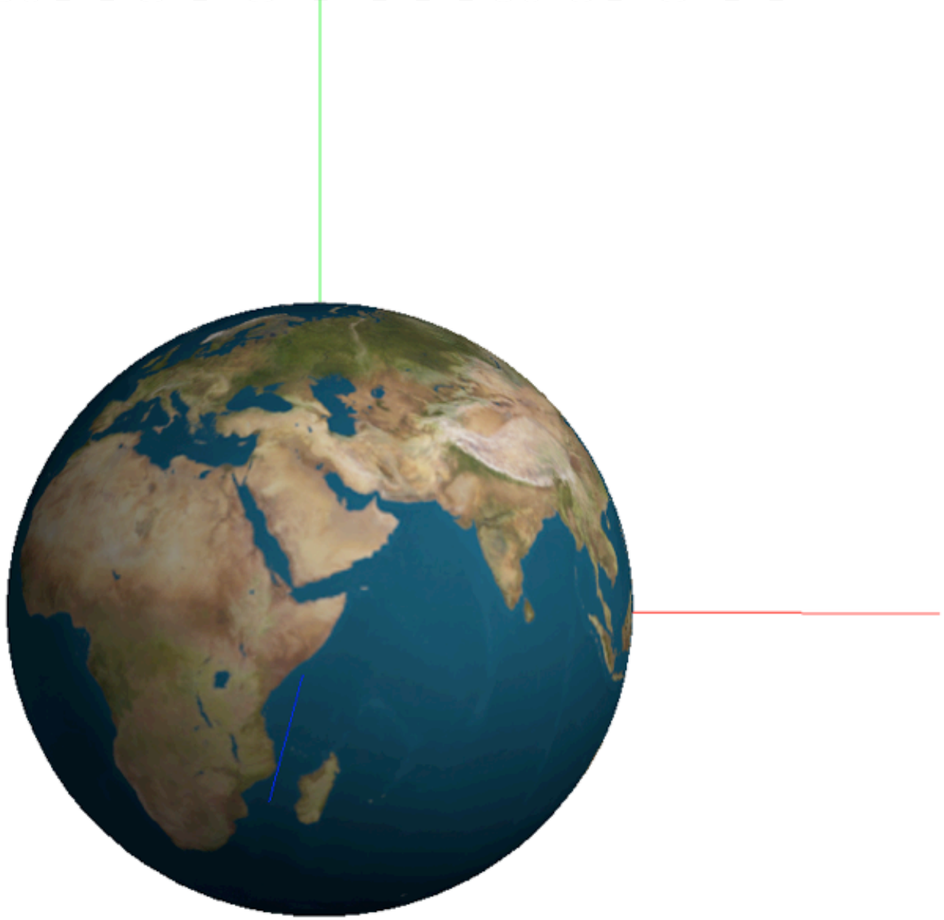
$$\vec{v} + \vec{w} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix} = \begin{pmatrix} v_x + w_x \\ v_y + w_y \\ v_z + w_z \end{pmatrix}$$

Transformationen

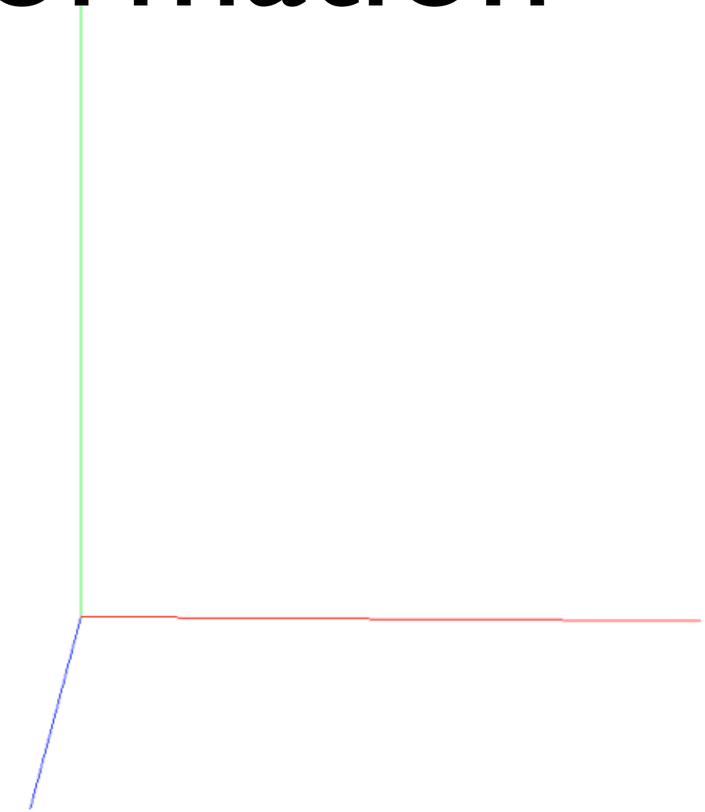
Transformation



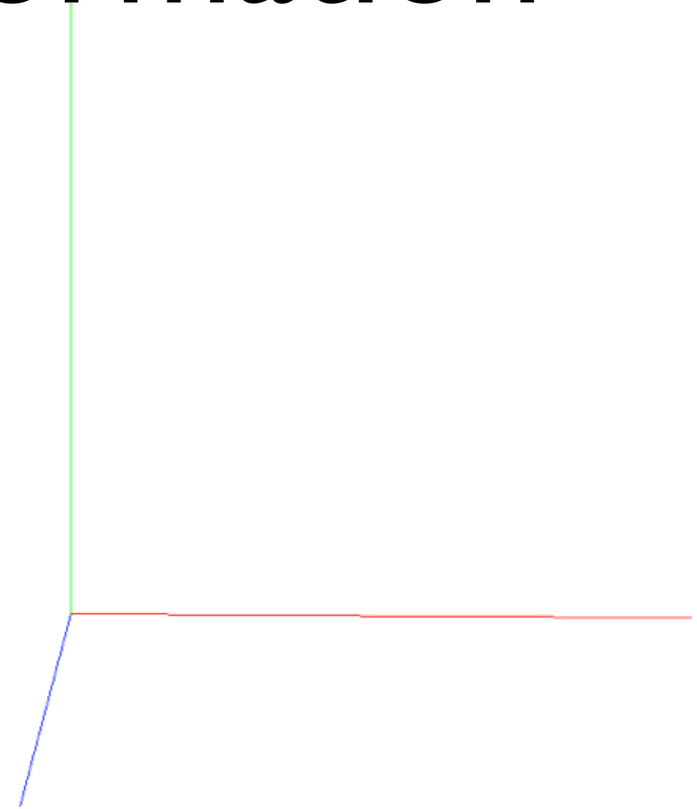
Transformation



Transformation



Transformation



Skalierung

$$\begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glScalef(x, y, z);
```

```
GLKMatrixStackScale(stack, x, y, z);
GLKMatrix4 multiplied = GLKMatrix4Scale(matrix, x, y, z);
```

Rotationen (X-Achse)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glRotatef(alpha, 1, 0, 0);
```

```
GLKMatrixStackRotateX(stack, alpha);
GLKMatrix4 multiplied = GLKMatrix4RotateX(matrix, alpha);
```

Rotationen (Y-Achse)

$$\begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glRotatef(beta, 0, 1, 0);
```

```
GLKMatrixStackRotateY(stack, beta);
GLKMatrix4 multiplied = GLKMatrix4RotateY(matrix, beta);
```

Rotationen (Z-Achse)

$$\begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glRotatef(gamma, 0, 0, 1);
```

```
GLKMatrixStackRotateZ(stack, gamma);
GLKMatrix4 multiplied = GLKMatrix4RotateZ(matrix, gamma);
```

Homogene Koordinaten

Vektor

$$\vec{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \\ 1 \end{pmatrix}$$

Matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{21} & a_{31} & t_x \\ a_{12} & a_{22} & a_{32} & t_y \\ a_{13} & a_{23} & a_{33} & t_z \\ p_1 & p_2 & p_3 & 1 \end{pmatrix}$$

Translation (Verschiebung)

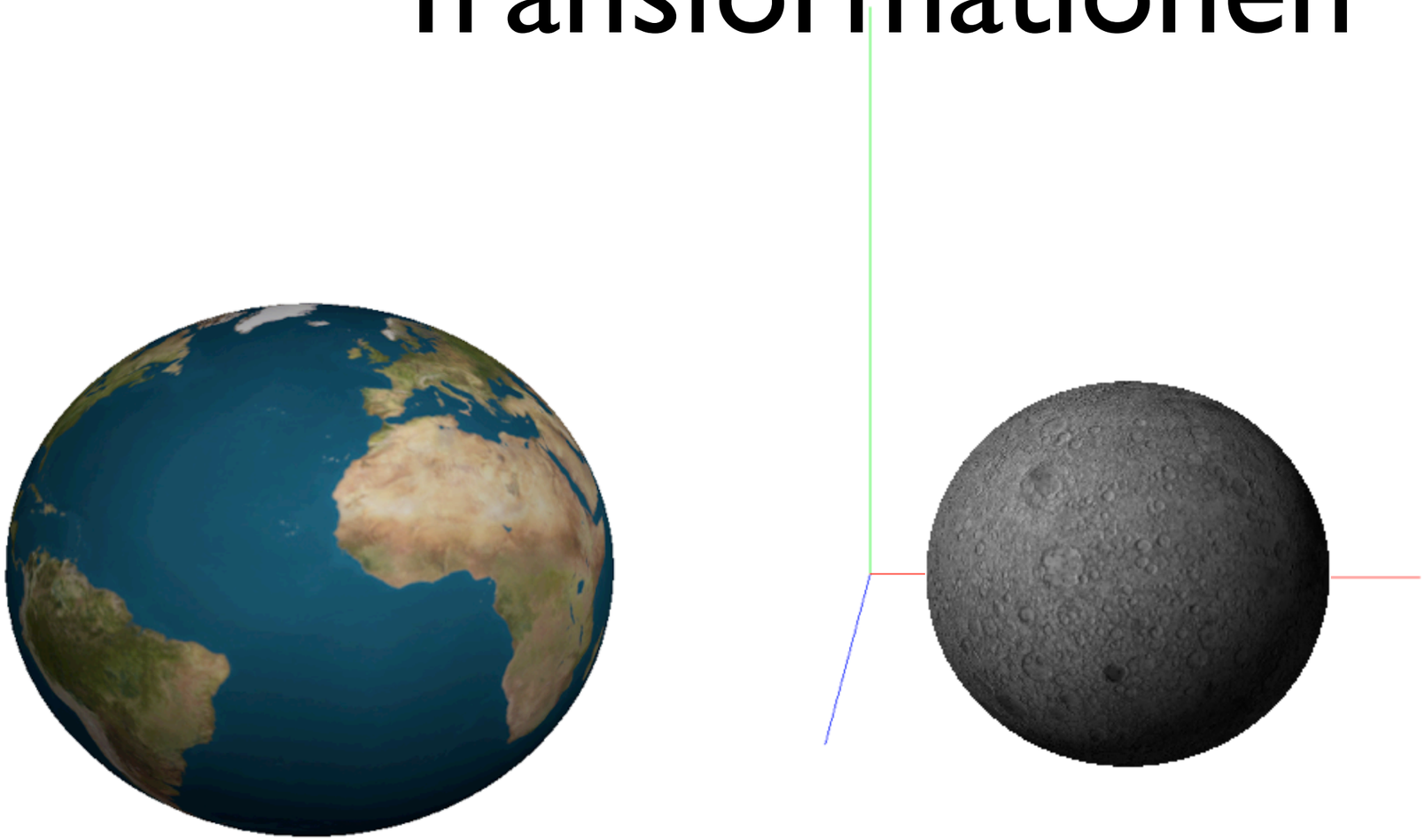
$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
glTranslatef(x, y, z);
```

```
GLKMatrixStackTranslate(stack, x, y, z);
GLKMatrix4 multiplied = GLKMatrix4Translate(matrix, x, y, z);
```

Matrixstack

Transformationen



Wichtige Stacks

- Model-View-Stack

```
glMatrixMode(GL_MODELVIEW);
```

- Projektionsstack

```
glMatrixMode(GL_PROJECTION);
```

Zugriffe auf den Stack

```
glPush();
glLoadMatrixf(&matrix);
glPop();
```

```
GLKMatrixStackPush();
GLKMatrixStackLoadMatrix4(matrix);
GLKMatrixStackPop();
```

Projektionen

Orthographische Projektion

$$\begin{pmatrix} \frac{2}{(right-left)} & 0 & 0 & t_x \\ 0 & \frac{2}{top-bottom} & 0 & t_y \\ 0 & 0 & -\frac{2}{(farVal-nearVal)} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}, \text{ mit}$$

$$t_x = -\frac{right+left}{right-left}, t_y = -\frac{farVal+nearVal}{farVal-nearVal} \text{ und } t_z = -\frac{(farVal+nearVal)}{(farVal-nearVal)}$$

```
glOrtho(left, right, bottom, top, nearVal, farVal);
```

```
GLKMatrix4MakeOrtho(left, right, bottom, top, nearVal, farVal);
```

Perspektivische Projektion

$$\begin{pmatrix} \frac{f}{aspect} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{(zFar+zNear)}{(zNear-zFar)} & \frac{(2 \cdot zFar \cdot zNear)}{(zNear-zFar)} \\ 0 & 0 & -1 & 0 \end{pmatrix}, \text{ mit}$$

$$f = cot\left(\frac{fovy}{2}\right)$$

```
gluPerspective(fovy, aspect, zNear, zFar);
```

```
GLKMatrix4MakePerspective(fovy, aspect, zNear, zFar);
```

Beleuchtung

Einschränkungen (Blinn-Phong)

- Kein Schattenwurf
- Keine Lichtreflexionen
- Keine Metalleffekte
- ...

Normalen

$$\vec{s} \times \vec{t} = \begin{pmatrix} s_2 \cdot t_3 - s_3 \cdot t_2 \\ s_3 \cdot t_1 - s_1 \cdot t_3 \\ s_1 \cdot t_2 - s_2 \cdot t_1 \end{pmatrix}$$

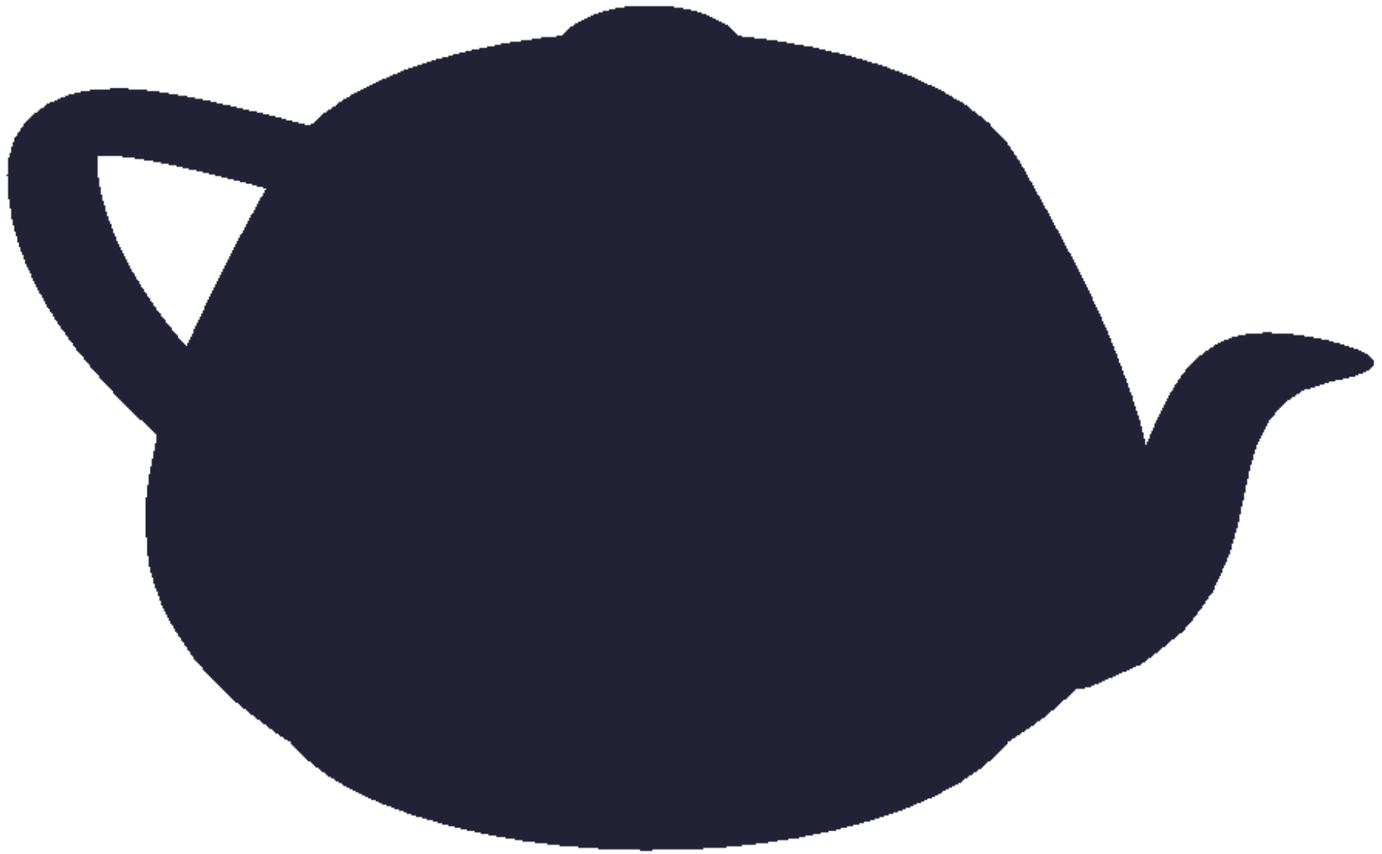
```
glNormal3f(1, 0, 0);
```

Ambientes Licht

$$I_{\text{ambient}} = I_{\text{a}} k_{\text{a}}$$

- Lichtkonstante (Ambient)
- Materialkonstante (Ambient)

Ambientes Licht

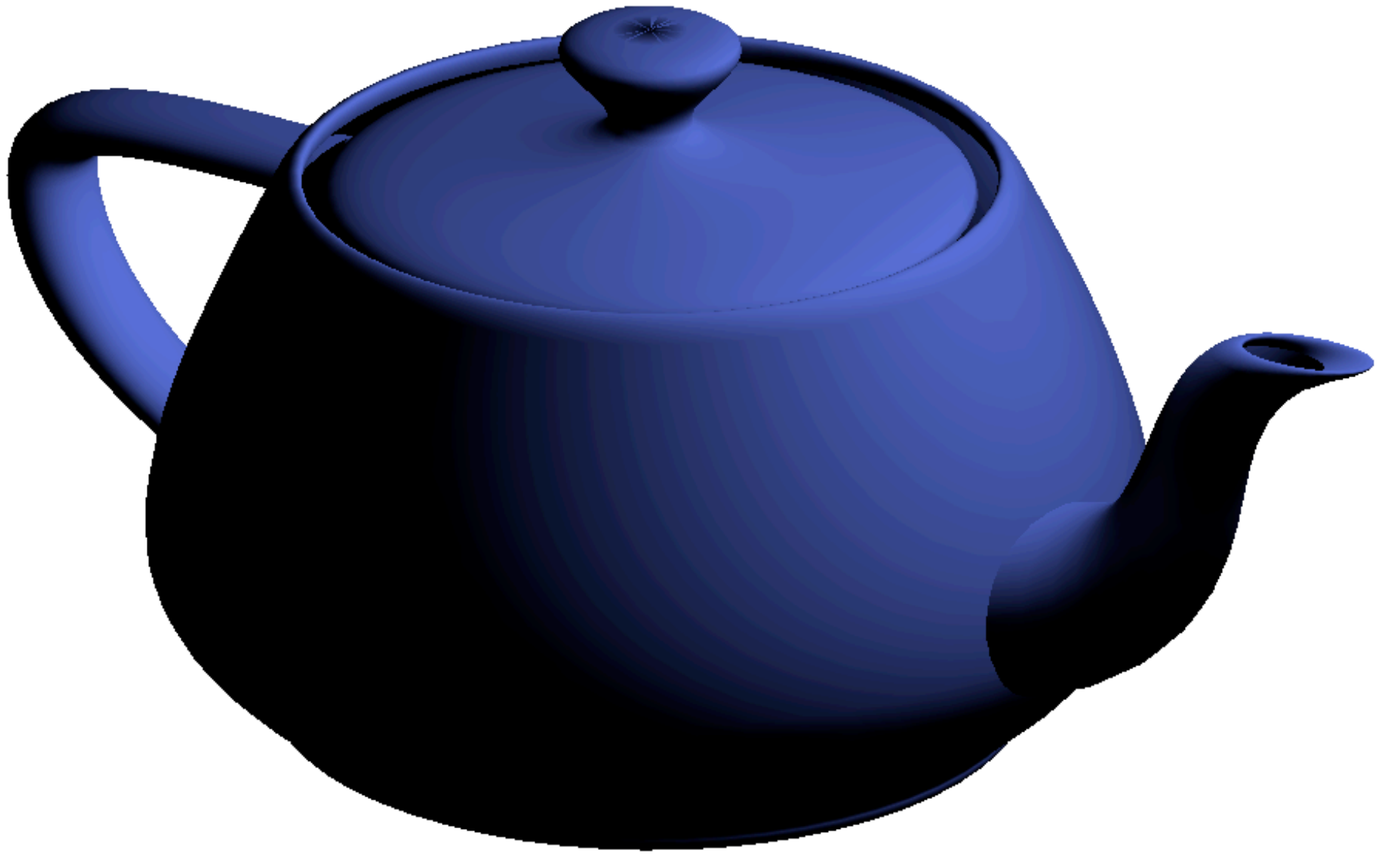


Diffuses Licht

$$I_{\text{diffuse}} = I_{\text{in}} k_{\text{d}} \cos\varphi = I_{\text{in}} k_{\text{d}} \left(\vec{l} \cdot \vec{n} \right)$$

- Lichtkonstante (Diffus)
- Materialkonstante (Diffus)
- Cos des Winkels zwischen Normale und einfallendem Licht

Diffuses Licht

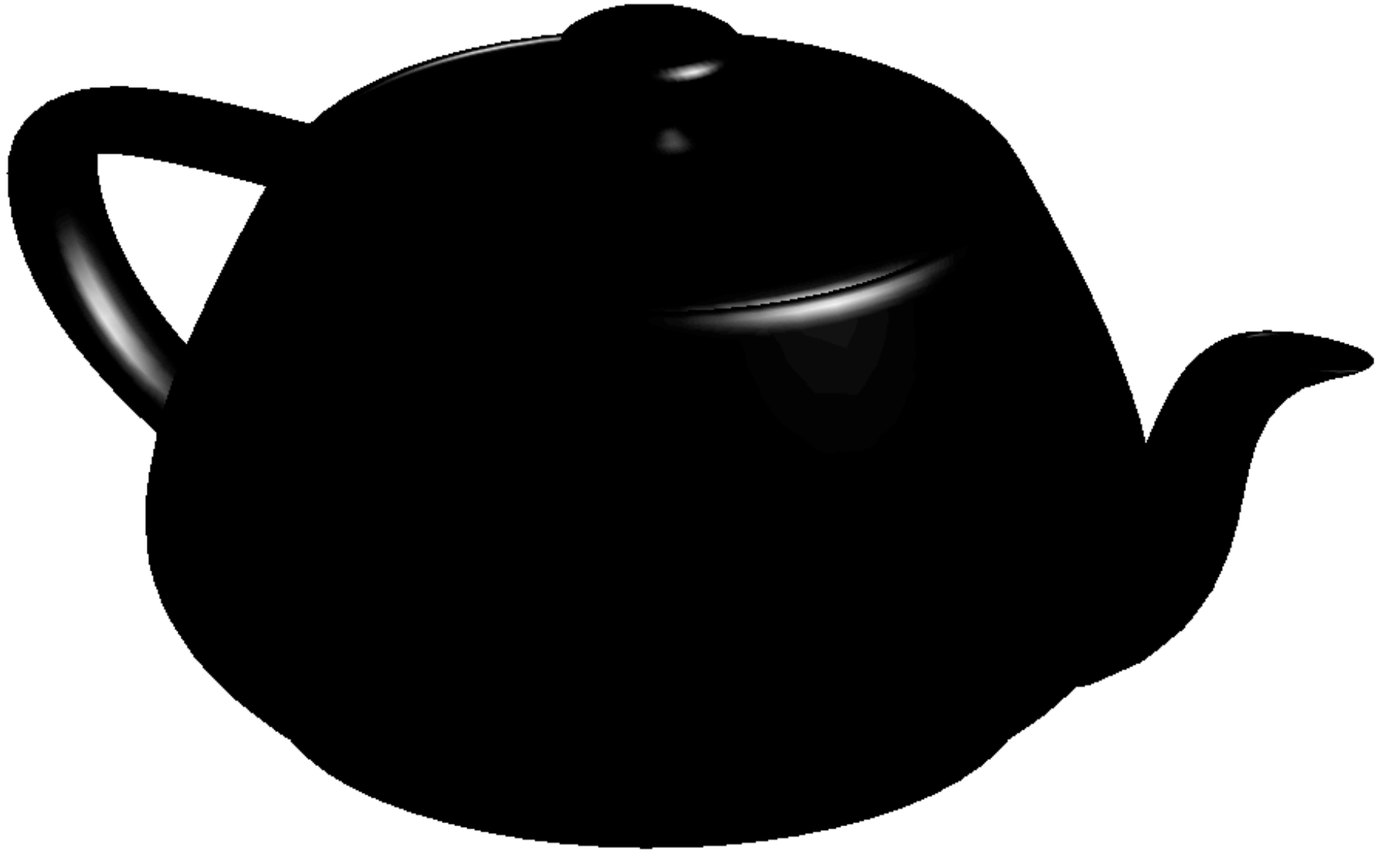


Spekulares Licht

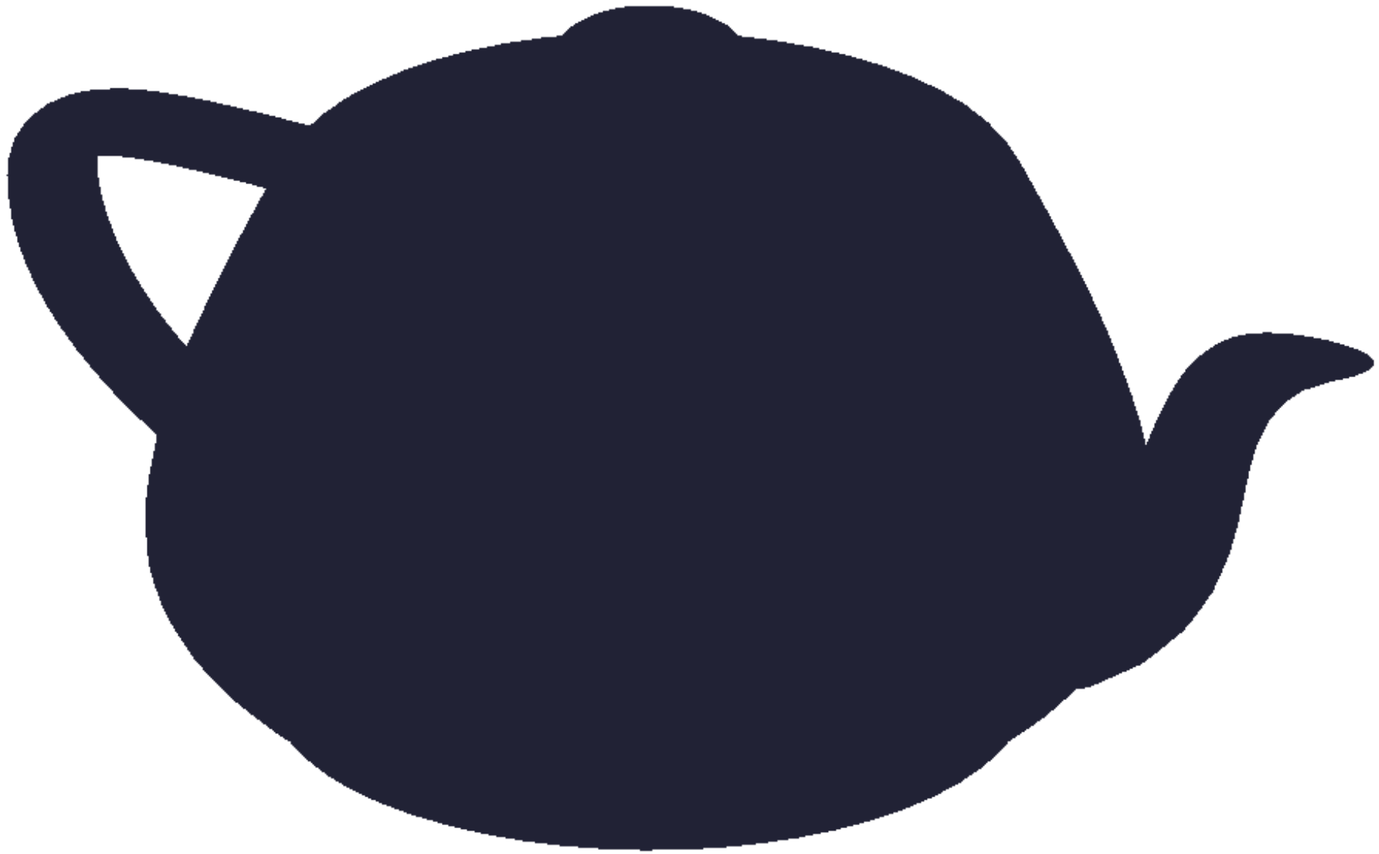
$$I_{\text{specular}} = I_{\text{in}} k_{\text{s}} \cos^{n} \theta = I_{\text{in}} k_{\text{s}} \left(\vec{r} \cdot \vec{v} \right)^{n}$$

- Lichtkonstante (Spekular)
- Materialkonstante (Spekular)
- Cos des Winkels zwischen Reflexionsrichtung und Betrachter
- Spiegelexponent

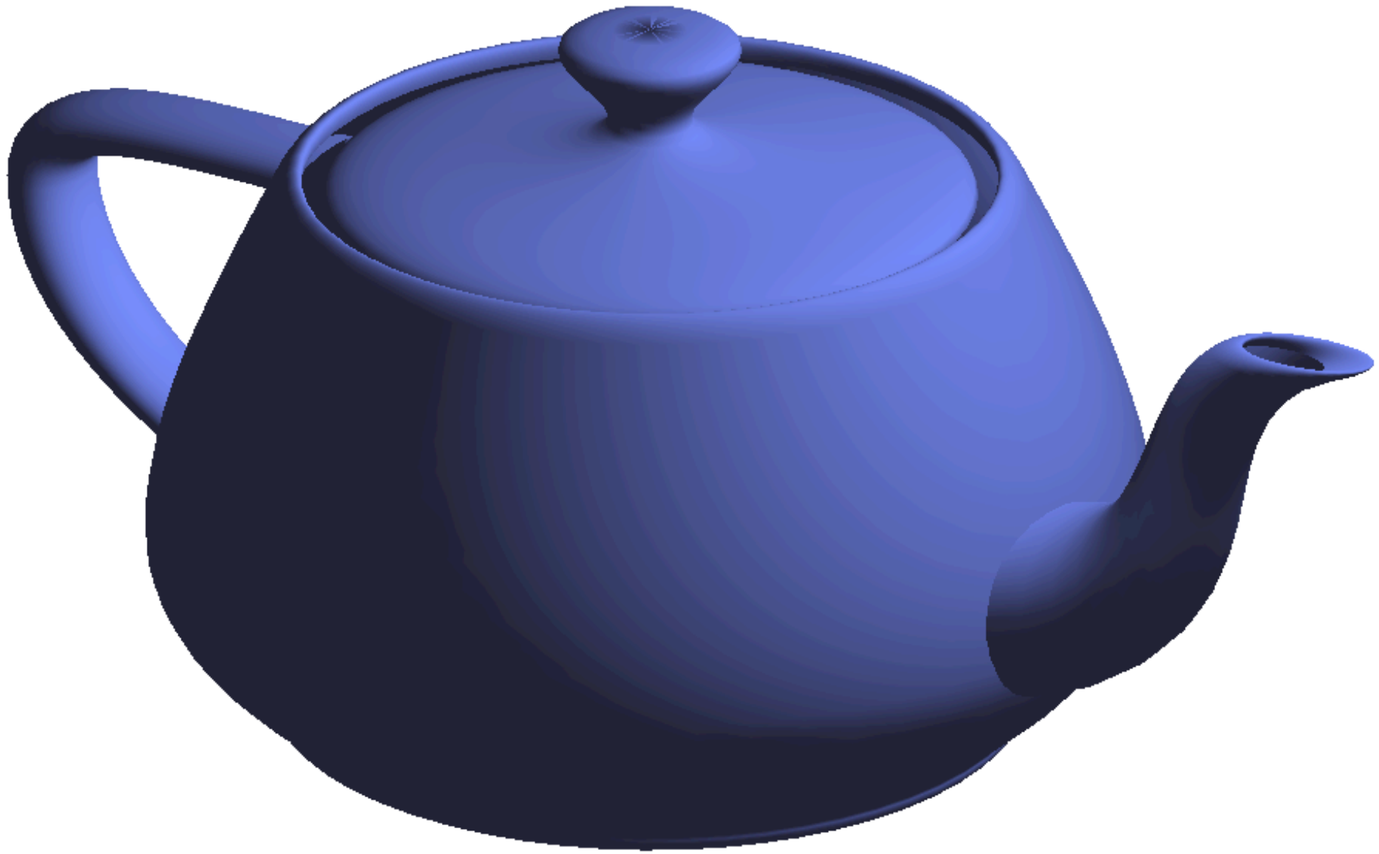
Spekulares Licht



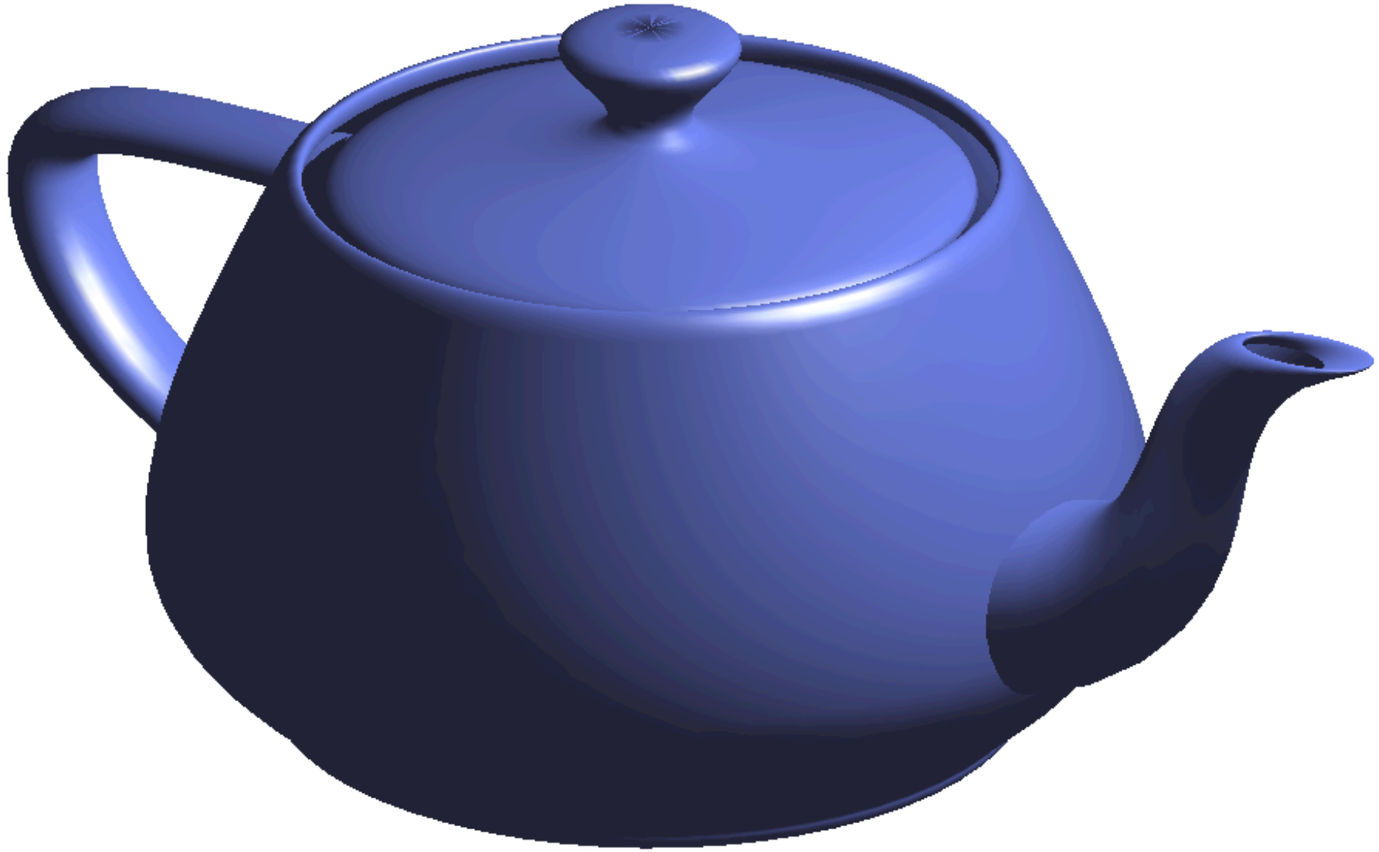
Ambient



Ambient + Diffus

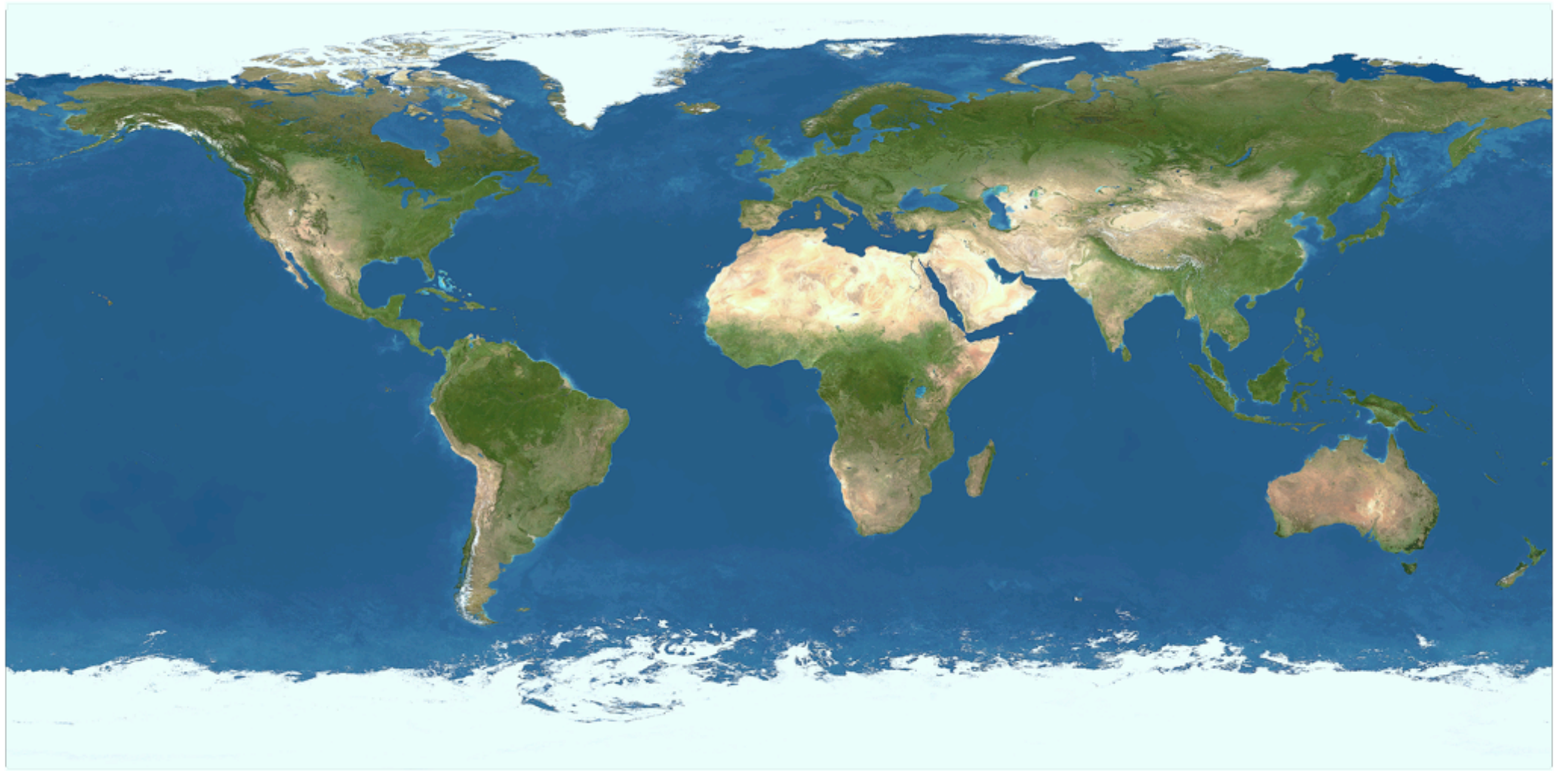


Ambient + Diffus + Spekular

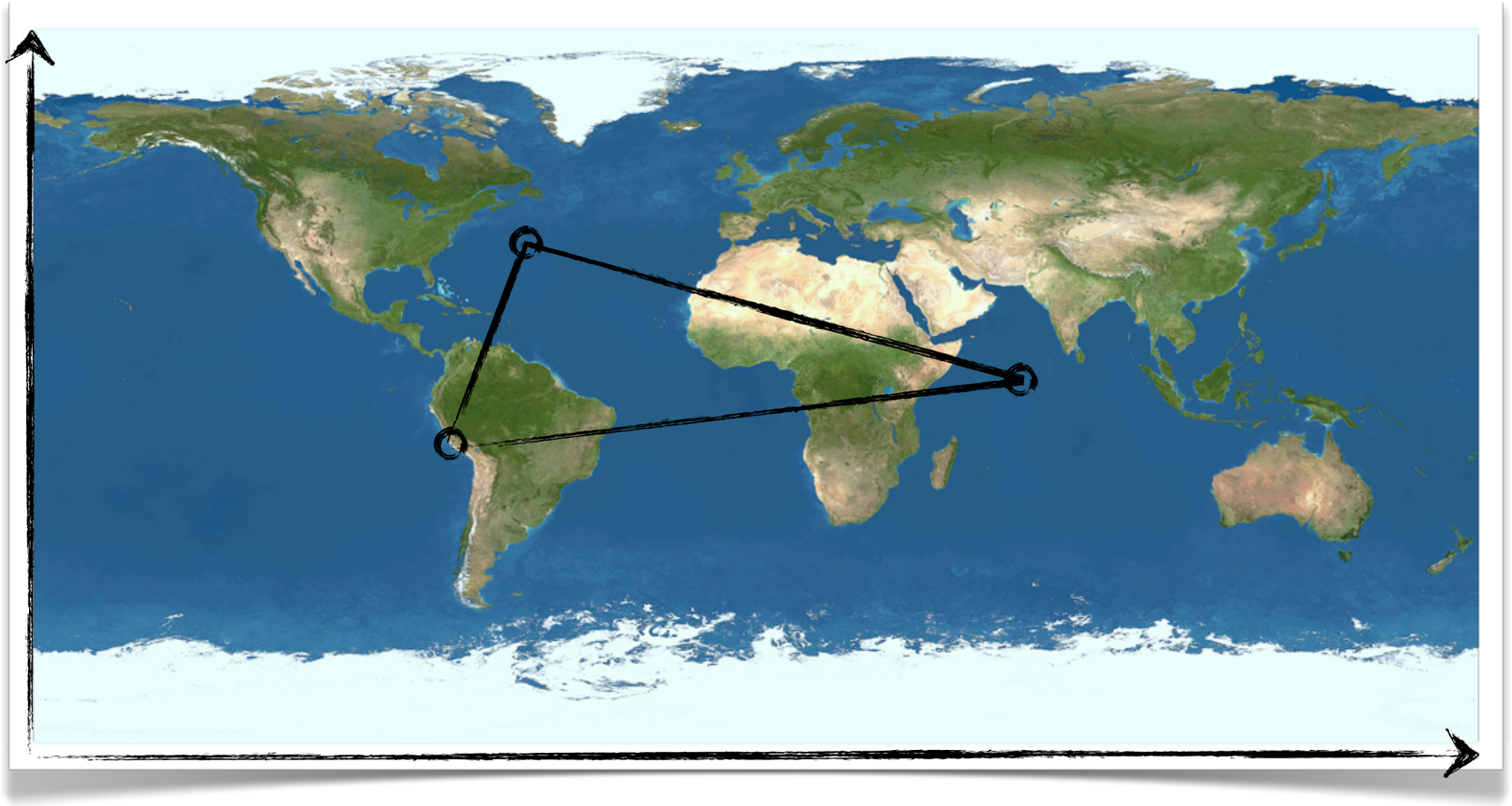


Texturen

Beispieltextur



Texturkoordinaten



Texturkoordinaten

```
GLuint texture;
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, GL_RGBA, GL_FLOAT, pixels);
...
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, texture);
...
glTexCoord2f(u, v);
glVertex3f(x, y, z);
```

Tiefenspeicher

Z-Buffer

- Berechnung des vordersten, sichtbaren Objekts
- Bestimmung des Tiefenwerts für jedes einzelne Pixel
- Auflösung ist einzustellen
- Einstellung von Near- und Far-Plane

Fragen?

Vielen Dank