Für Abgabegespräch

Task 4

Normals

Würfel: Für jeden Vertex (Ecke) 3 Normals.

Kugel: Top und Bottom Normals und für jeden anderen Vertex = position - mitte

Zylinder: Top und Bottom Normals + Kreissegmente top bottom und Side Faces (2 + 2n)

Normals VBO machen auf location 1 und binden.

Shader

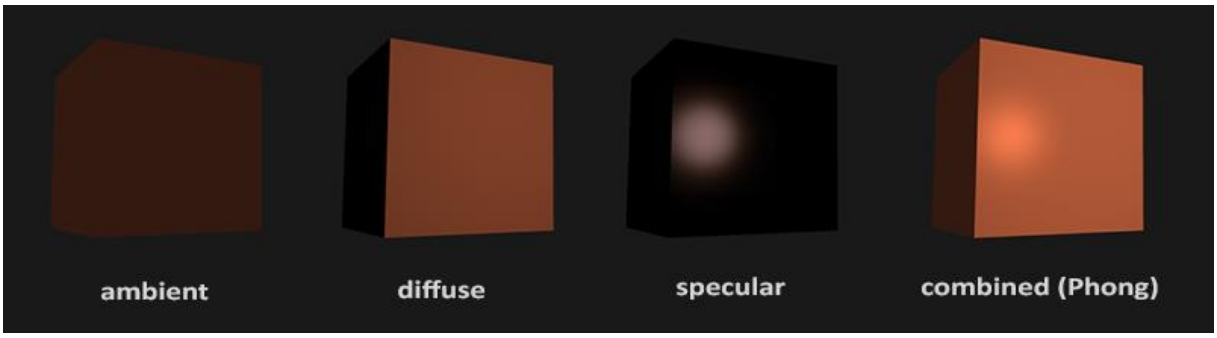
Phong: interpoliert die Farbe über jedes Fragment/Pixel
Gouraud: interpoliert die Farbe über jedes Polygon

Vertex Shader (Phong):

- normal kommt dazu auf location 1

```
vert.normal_world = normal * mat3(transpose(inverse(modelMatrix)));
```

Um die Translation einer Matrix zu entfernen, muss man einfach die **4 Reihe und Zeile entfernen**.
Um das Skalieren fixen, muss man die Matrix **transponieren und invertieren.**



Ambient:

Für alle Lichter gleich: ka mal ambient konstante

Directional Light:

<u>Diffus:</u>

Wir brauchen die Lichtrichtung: die wird invertiert, damit wir den **Winkel** mit dem **Skalarprodukt** berechnen können. **Clamp**: macht Minus Zahlen zu 0ern, Werte dürfen nur zwischen 0 und 1 liegen.

<u>Specular:</u>

Wir brauchen die **View Direction** und die **Reflection Direction** (mit reflect), da wird der **Winkel** berechnet mit dem **Skalarprodukt** und das dann **hoch alpha**.

Zu den beiden dann noch jeweils die konstante berechnen.

Nur Diffus und Ambient werden mit der Objektfarbe mal genommen! Und nur Specular und Diffus mit der Lichtfarbe! Damit man auf das richtige Ergebnis kommt!

Point Light:

Statt der Lichtrichtung wird hier die Lichtposition benötigt und daraus wird dann die Lichtrichtung berechnet.

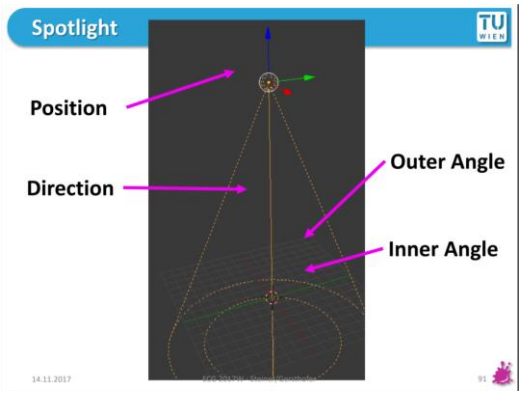
Ansonsten sind Diffus und Specular gleich.

attenuation = Abschwächung

Da wird die Distanz benötigt von Licht zu Fragment um dann das Licht abzuschwächen je weiter das Fragment weg ist. (Formel war gegeben)

Spot Light:

Wie Point Light nur mit Winkeln und Intensität des Lichts.



Dann werden alle Werte in der Main deklariert bevor ein Objekt gezeichnet wird.

Task 5

Textur laden:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
```

= Minification Filter: linear (hohe Frequenzen verursachen Artefakte während dem Sampling)

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

= Magnification Filter: linear (super sampling)

Mipmap:

- filtern und downsamplen
- macht eine Mip-Map Bildpyramide von Level 0 bis zu einen 1x1 Bild

Shader:

Vertex:

- neuer VBO mit location 2.

- sampler2D: 2D Grafik

- texture(colorTexture, vert.uv_world).rgb;

UV-Koordinaten

Würfel:

für jede Normal, die in die gleiche richtung geht jeweils die uv koordinaten übergeben

Kugel:

Finding UV on a sphere [edit]

For any point P on the sphere, calculate $\hat{d}$, that being the unit vector from P to the sphere's origin.

Assuming that the sphere's poles are aligned with the Y axis, UV coordinates in the range $[0, 1]$ can then be calculated as follows:

$$u = 0.5 + \frac{\arctan2(d_z, d_x)}{2\pi}$$

$$v = 0.5 - \frac{\arcsin(d_y)}{\pi}$$

2x für top und bottom und dann für jede normal mit der Formel.

Zylinder: Top und Bottom und circle segments ausrechnen und side faces nur u wie bei der Kugel, v bleibt immer gleich (entweder 0 oder 1)