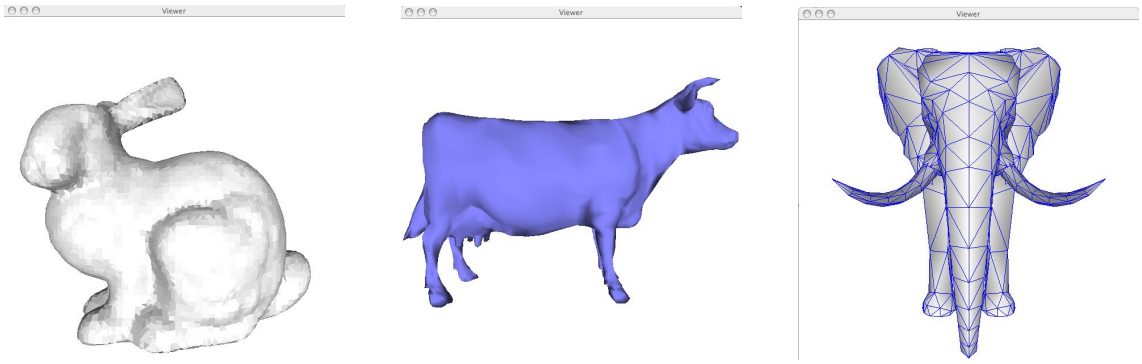
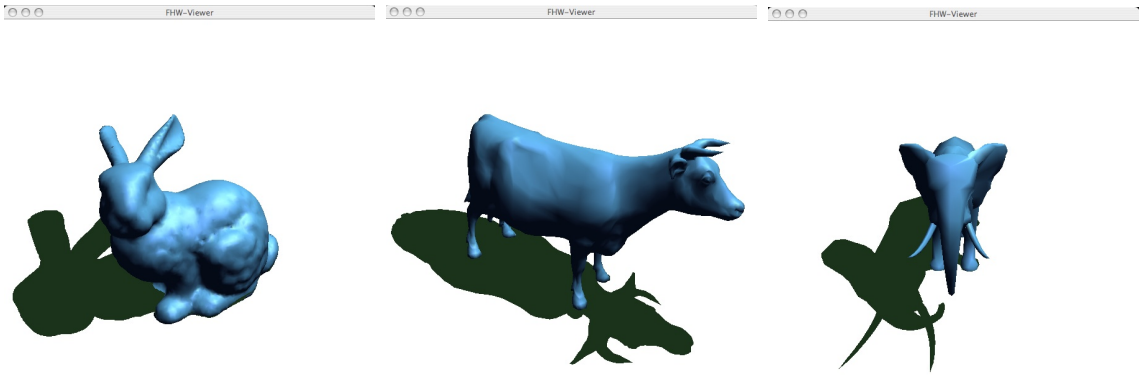
Übungsblatt 8

Aufgabe 1. Erweitern Sie Ihren OpenGL-Viewer aus Aufgabe 1 von Übungsblatt 7 so, dass die eingelesenen Modelle beleuchtet sowie mit überlagertem *Wireframe* dargestellt werden können. Experimentieren Sie dabei mit

- *Flat-* und *Gourad-Shading*
- unterschiedlichen *Positionen* für die Lichtquelle
- *mehreren* Lichtquellen
- unterschiedlichen *Licht-* und *Material-Eigenschaften*

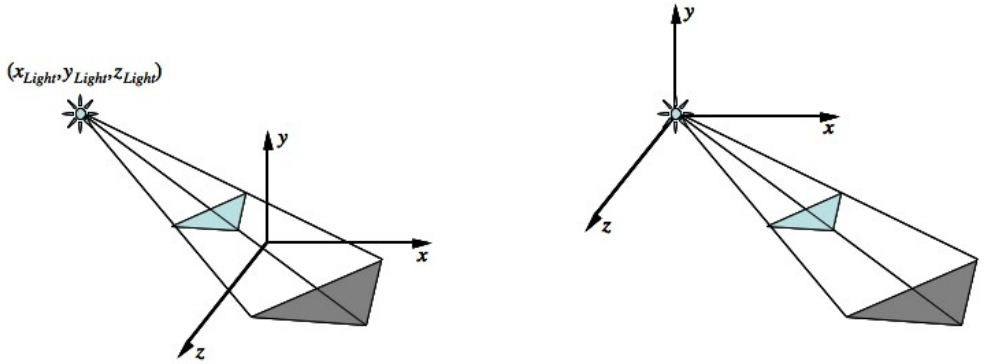


Aufgabe 2. Erweitern Sie Ihren OpenGL-Viewer aus Aufgabe 1 so, dass die eingelesenen Modelle zusätzlich einen (einfachen) Schatten werfen.



Bei einem lokalen Beleuchtungsmodell, wie OpenGL es realisiert, kann ein einfacher Schatten (siehe obige Abbildungen), wie nachfolgend beschrieben, durch eine zusätzliche projektive Abbildung des darzustellenden Objekts erzeugt werden.

Wir betrachten einen Schatten, der durch *eine einzige* Punktlichtquelle erzeugt wird und nehmen der Einfachheit halber an, dass er auf die Ebene $y = 0$ fällt. Dieser Schatten entsteht durch (Zentral-)Projektion des Objekts auf die Ebene $y = 0$. Das Projektionszentrum ist die Position der Lichtquelle. Projiziert man nun die Polygone des Objekts in einem Koordinatensystem, in dem die Lichtquelle der Ursprung ist, erhält man die Eckpunkte der Polygone des Schattens in diesem Licht-Koordinatensytem (siehe Abbildung). Diese Eckpunkte müssen anschließend in das Welt-Koordinatensystem zurücktransformiert werden.



Als konkretes Beispiel sei die *Position der Lichtquelle* $(x_{Light}, y_{Light}, z_{Light})$. Man verschiebt diese nun mittels $\mathbf{t} = (-x_{Light}, -y_{Light}, -z_{Light})$ in den Ursprung und führt anschließend die Projektion $\mathbf{P}$ mit dem Ursprung als Projektionszentrum durch. Nach der Rücktranslation mittels $\mathbf{t} = (x_{Light}, y_{Light}, z_{Light})$ erhält man die Eckpunkte der Polygone des Schattens im Weltkoordinatensystem. Die Transformationen können zum Beispiel wie folgt mit Hilfe von OpenGL durchgeführt werden:

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{-y_{Light}} & 0 & 0 \end{pmatrix}$$

```
# projection matrix
p = [1.0, 0, 0, 0, 0, 1.0, 0, -1.0/yLight, 0, 0, 1.0, 0, 0, 0, 0, 0]

glColor3fv(modelColor)
glCallList(modelList)                       # render object normally

glMatrixMode(GL_MODELVIEW)                  # use modelview matrix
glPushMatrix()                              # save state
glTranslatef(xLight, yLight, zLight)        # translate back
glMultMatrixf(p)                            # project object
glTranslatef(-xLight, -yLight, -zLight) # move light to origin
glColor3fv(shadowColor)
glCallList(modelList)                       # render object again
glPopMatrix()                               # restore state
```