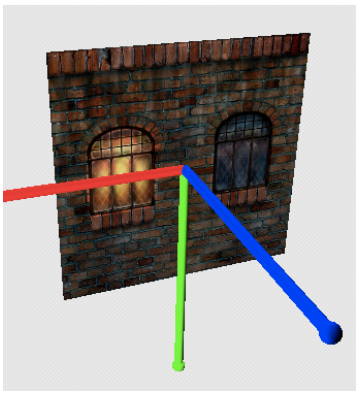
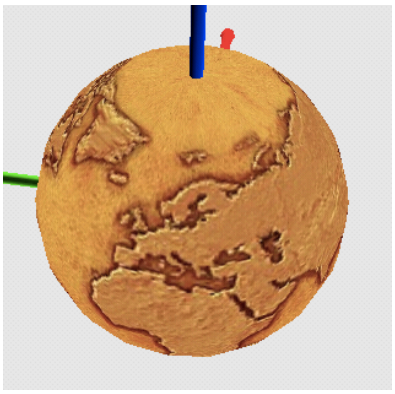
WP Einführung in die Computergrafik

WS 2013/2014, Hochschule für Angewandte Wissenschaften (HAW), Hamburg
Prof. Dr. Philipp Jenke, Lutz Behnke

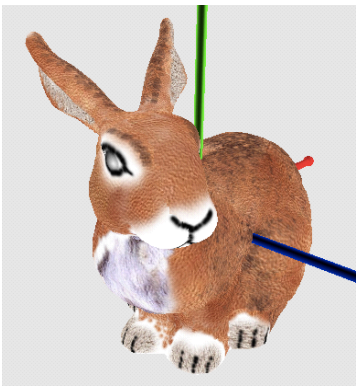
Aufgabenblatt 5 - Texturen und Shader



Texturiertes Quadrat.



Texturierte Kugel.



Texturiertes Oberflächennetz.

Aufgabe 1: Texturierung in Java3D

In dieser Aufgabe erweitern Sie die bestehende Funktionalität zur Darstellung von Dreiecksnetzen. Neben einer einfachen Oberflächenfarbe sollen Ihre Netze mit jeweils einer Textur belegt werden können.

a) Erweiterung der Datenstruktur `Triangle`
Die Texturkoordinaten werden, genau wie die Vertices, für jedes Dreiecksnetz in einer Liste verwaltet. Mit jedem Dreieck werden drei Texturkoordinaten verknüpft, je eine pro Eckpunkt. Jeder Eckpunkt eines Dreiecks benötigt daher einen Index zur Referenzierung der benötigten Texturkoordinaten. Erweitern Sie die Datenstruktur `Triangle` entsprechend.

b) Erweiterung von `ITriangleMesh` und `TriangleMesh`
Erweitern Sie das Interface und damit die Klasse zur Repräsentierung von Dreiecksnetzen um die notwendigen Felder und Methoden zur Verwaltung der Liste von Texturkoordinaten. Die einzelnen Koordinaten sollen über der Klasse `javax.vecmath.TexCoord3f` repräsentiert werden. Die Texturkoordinaten sind nur zweidimensional, daher setzen Sie die dritte Koordinate in der Datenstruktur auf `0`.

c) Erstellen eines Shape3D-Objekts
Überarbeiten Sie nun Ihre Funktionalität zum Generieren eines `Shape3D`-Objektes, das in den Szenengraph gehängt werden kann. Insbesondere müssen Sie pro Vertex die richtigen Texturkoordinaten setzen.

d) Erstellen einer Texture-Appearance
Als letzten Schritt müssen Sie nun noch eine zu einer Textur gehörige `Appearance` erzeugen - analog zu den einfachen `Appearances` zum Setzen von Farben. Implementieren Sie die notwendige Funktionalität in der beigefügten Klasse `AppearenceHelper` in den Methoden `createTextureAppearance` und `createTexture`. Die Stellen, an denen Sie Änderungen vornehmen müssen, sind mit

```
// Your code here
```

markiert. Folgendes Vorgehen ist generell notwendig:

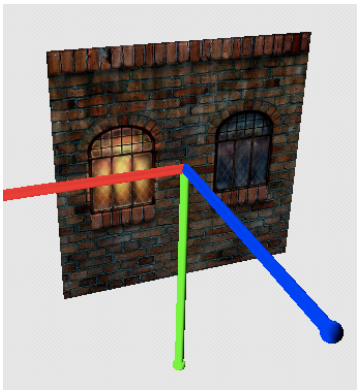
- Erzeugen einer `Appearance`
- Einlesen eines Bildes (der Textur) als `java.awt.Image`
- Erzeugen eines Textur-Objektes mit

```
Texture tex = new TextureLoader(image,
      TextureLoader.BY_REFERENCE | TextureLoader.Y_UP, null).getTexture();
```

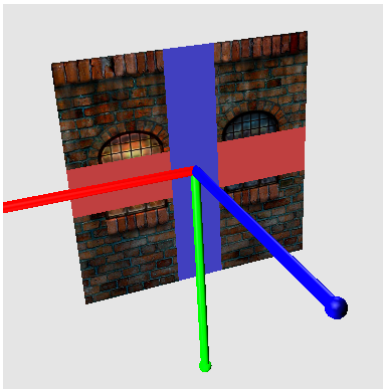
Weisen Sie nun die erzeugte `Appearance` dem `Shape3D` zu und hängen Sie sie in den Szenengraphen.

e) Erzeugen von texturierten Dreiecksnetzen
Legen Sie jetzt mindestens zwei texturierte Dreicksnetze mit unterschiedlichen Texturen an. Wie die Dreiecksnetze aufgebaut sind, ist Ihnen freigestellt.

Aufgabe 2: Shader



Original Shader.



Modifizierte Shader.

In der beigefügten Klasse `AppearanceHelper` finden Sie eine Methode zum Erzeugen einer `Appearance` mit Shadern mit folgendes Signatur:

```
public static Appearance createShaderAppearance(String vertexShaderFilename,
        String fragmentShaderFilename, String textureFilename)
```

Als Parameter übergeben Sie je einen Dateinamen für eine Vertex- und einen Fragmentshader, sowie den Dateinamen einer Textur. Es werden außerdem ein Vertex- (*vertex_shader_texture.glsl*) und ein Fragmentshader (*fragment_shader_texture.glsl*) zum Verwenden von Texturen mitgeliefert. Verändern Sie den Fragmentshader so, dass für Texturkoordinaten (`u`, `v`) folgende Pixelfarben erzeugt werden (siehe Abbildung)

- falls `0.4 < u < 0.6: (0.25, 0.25, 0.75)`
- falls `0.4 < v < 0.6: (0.75, 0.25, 0.25)`
- sonst: Texturfarbe