Computergrafik
Übungsblatt 8

Aufgabe 1 Blinn-Phong Lighting

Bauen Sie mittels CMake die Visual Studio Solution cogra.sln und öffnen Sie diese anschließend. In Visual Studio, editieren Sie im Projekt LightingAndShadingStub die Dateien LightingAndShading.cpp, den Fragment-Shader shaders/Lighting.frag.glsl und den Vertex Shader shaders/Lighting.vert.glsl.

Um die 3D Modelle zu kopieren, bauen Sie bitte händisch das Projekt CopyData.

Setzen Sie das Projekt LightingAndShadingStub als Startup-Project.

(a) Implementieren Sie das Blinn-Phong-Lighting Model wie in der Vorlesung besprochen.

In der Methode `void onDraw()`, übergeben Sie dem Vertex-Shader `m_lightingShaderSmooth` folgende Uniformvariablen:

- `u_mv`: Die Matrix, welche die Vertex Position in View-Coordinates transformiert. In LightingAndShading.cpp, bekommen Sie diese Matrix über `glm::mat4x4 mv = m_examinerController.getTransformationMatrix();`
- `u_mvp`: Die Matrix, welche die Vertex Position in Clip-Coordinates transformiert. In LightingAndShading.cpp, bekommen Sie diese Matrix über `glm::mat4x4 mvp = m_camera.getMatrix() * mv`.
- `u_mvInvT`: Die Inverse-Transponierte von `u_mv`. Im namespace `glm::` finden Sie dazu geeignete Methoden.

Nutzen Sie nun die Matrizen im Vertex-Shader um die per-vertex Input-Attribute `inPosition` und `inNormal` geeignet zu transformieren und setzen Sie folgende per-vertex Output-Attribute:

- `gl_Position` Die per-vertex Position in Clip Coordinates
- `out vec3 esNormal` Der per-vertex Normalen Vektor in View Coordinates
- `out vec3 esPosition`: Der per-vertex Positions Vektor in View Coordinates

Diese Variablen werden vom Rasterizer interpoliert und an den Fragment-Shader weitergereicht und liegen dort in den Variablen `in vec3 esPosition`, `in vec3 esNormal`, und `in vec3 esColor` vor. Nutzen Sie diese um das Blinn-Phong Lighting zu realisieren.

Dazu benötigen Sie noch die uniform Variablen `uniform vec3 u_ambient`, `uniform vec3 u_diffuse`, `uniform vec3 u_specular`, `uniform float u_exponent`, `uniform vec3 u_lightDirection`, welche in der Methode `void onDraw()` bereits geeignet gesetzt wurden.

(b) Implementieren Sie ein Variante der Normalen-Vektor Berechnung.

Setzen Sie dazu das flag `const bool useOwnNormals = true;` und vervollständigen Sie die Methode `computeNormals(uint32_t const * const indexBuffer, const size_t nTriangles, glm::vec3 const * const positions, const size_t nVertices, glm::vec3 * const normals)`.