Tehtävä 4. Blinn-phong valaisu

Tehtävässä muokataan tehtävän 3 pohjalta uusi scene, joka rendaa teepannun ruudulle käyttäen blinn-phong -valaisumallia.

Ensin, kopioi tehtävän 3 scene uudeksi sceneksi ja laita se käynnistymään se ohjelman alussa. Nimeksi uudelle voisi antaa vaikkapa BlinnPhongScene tms.

Varjostinkoodit

Fragment -shader koodi (nimeä esim. Blinn-phong.fs):

```
// default to medium precision
precision mediump float;

// OpenGL ES require that precision is defined for a fragment shader
// Define some useful macros
#define saturate(x) clamp( x, 0.0, 1.0 )

// Struct for material
struct MATERIAL
{
    vec4  vAmbient;
    vec4  vDiffuse;
    vec4  vSpecular;
};

// Uniforms
uniform MATERIAL g_Material;

// Varyings
varying vec3     g_vViewVecES;
varying vec3     g_vNormalES;
varying vec3     g_vLightVecES;

void main()
{
    // Normalize per-pixel vectors
    vec3 vNormal = normalize( g_vNormalES );
    vec3 vLight  = normalize( g_vLightVecES );
    vec3 vView   = normalize( g_vViewVecES );
    vec3 vHalf   = normalize( vLight + vView );

    float NdotL = saturate( dot(vNormal,vLight) );
    float NdotH = saturate( dot(vNormal,vHalf) );

    // Compute the lighting in eye-space
    float fDiffuse  = NdotL;

    float fSpecular = pow( NdotH, g_Material.vSpecular.w );
    float SelfShadow = 4.0 * fDiffuse;

    // Combine lighting with the material properties
    gl_FragColor.rgba  = vec4(0.15,0.15, 0.15, 0.15) * g_Material.vAmbient;
    gl_FragColor.rgba += g_Material.vDiffuse * fDiffuse;
    gl_FragColor.rgb  += SelfShadow * vec3(0.15, 0.15, 0.15) * g_Material.vSpecular.xyz * fSpecular;
}
```

Vertex shader koodi (nimi esim Blinn-phong.vs):

```
// Uniforms
uniform mat4    g_matView;
uniform mat4    g_matModelView;
uniform mat4    g_matModelViewProj;
uniform mat4    g_matNormal;
uniform vec3    g_lightPos;

// Attributes
attribute vec4  g_vPositionOS;
attribute vec3  g_vNormalOS;

// Varyings
varying vec3    g_vNormalES;
varying vec3    g_vViewVecES;
varying vec3    g_vLightVecES;

void main()
{
    vec4 vPositionES = g_matModelView     * g_vPositionOS;
    vec4 vPositionCS = g_matModelViewProj * g_vPositionOS;
    vec3 vLightPosES = (g_matView * vec4(g_lightPos,1.0)).xyx;

    // Transform object-space normals to eye-space
    vec3 vNormalES = (g_matNormal * vec4(g_vNormalOS,0.0)).xyz;

    // Pass everything off to the fragment shader
    gl_Position  = vPositionCS;
    g_vNormalES  = normalize(vNormalES.xyz);
    g_vViewVecES = normalize(-vPositionES.xyz);
    g_vLightVecES = normalize(vLightPosES - vPositionES.xyz);
}
```

GlobalShaderUniforms -muutokset

Edellisessä tehtävässä vertex shaderille välitettiin vain yksi matriisi. Tämän tehtävän shaderit vaativat useamman matriisin, jotka esilasketaan C++ koodissa ja välitetään sitten vertex-shaderille. Esilasketut matriisit c++ koodissa auttavat optimoimaan matriisikertolaskut pois, jotka muuten jouduttaisiin tekemään vertex-shaderissä. Muokkaa SharedShaderValues ja GlobalShaderUniforms rakenteita niin, että shadereille lähetetään model view proj -matriisin lisäksi muutama muu matriisi . Alla olevan kuvan mukaisesti:

```
// Shared "global values", which is used by each shader program.
// Typically this consists of different matrix values, light
// related values etc. which might be needed by each shader.
// Usage: Create one object of type SharedShaderValues to the
// scene and pass it for each shader as uniforms to be used.
struct SharedShaderValues : public core::Object
{
    slmath::mat4 matModel;              // Model matrix (object world matrix)
    slmath::mat4 matView;               // View matrix. (inverse of camera world matrix)
    slmath::mat4 matProj;               // Projision matrix of the camera
    slmath::mat4 matModelView;          // Model view matrix. Used to transform position vertices to camera space.
    slmath::mat4 matNormal;             // Model view matrix. Used to transform normal/binormal/tangent vertices to camera space.
    slmath::mat4 matModelViewProj;      // Model view projection matrix. Used to transform position vertices to clip space.
    slmath::vec3 lightPos;              // World poition of point light.
    slmath::vec3 camPos;                // World position of camera.
};
```

GlobalShaderUniforms -luokka pitää muokata vastaamaan SharedShaderValues-struktin kanssa vastaavanlaiseksi, jotta kaikki arvot välitetään shaderille. Alla siihen koodia:

```
// Class for uniforms, which are used by each shader. Sets values
// of SharedShaderValues-object to shader program in bind.
class GlobalShaderUniforms : public graphics::ShaderUniforms
{
public:
    GlobalShaderUniforms(graphics::Shader* shader, const SharedShaderValues* shaderShaderValues = 0)
        : ShaderUniforms(shader)
        , m_shaderShaderValues(shaderShaderValues)
    {
    }

    virtual ~GlobalShaderUniforms()
    {
    }

    virtual void getUniformLocations(graphics::Shader* shader)
    {
        // Get uniform locations
        m_ids[0] = glGetUniformLocation(shader->getProgram(),   "g_matModel");
        m_ids[1] = glGetUniformLocation(shader->getProgram(),   "g_matView");
        m_ids[2] = glGetUniformLocation(shader->getProgram(),   "g_matProj");
        m_ids[3] = glGetUniformLocation(shader->getProgram(),   "g_matModelView");
        m_ids[4] = glGetUniformLocation(shader->getProgram(),   "g_matNormal");
        m_ids[5] = glGetUniformLocation(shader->getProgram(),   "g_matModelViewProj");
        m_ids[6] = glGetUniformLocation(shader->getProgram(),   "g_lightPos");
        m_ids[7] = glGetUniformLocation(shader->getProgram(),   "g_camPos");
    }


    virtual void bind(graphics::Shader* shader)
    {
        shader->bind();
        if( m_shaderShaderValues )
        {
            glUniformMatrix4fv( m_ids[0], 1, GL_FALSE, &m_shaderShaderValues->matModel[0][0]);
            glUniformMatrix4fv( m_ids[1], 1, GL_FALSE, &m_shaderShaderValues->matView[0][0]);
            glUniformMatrix4fv( m_ids[2], 1, GL_FALSE, &m_shaderShaderValues->matProj[0][0]);
            glUniformMatrix4fv( m_ids[3], 1, GL_FALSE, &m_shaderShaderValues->matModelView[0][0]);
            glUniformMatrix4fv( m_ids[4], 1, GL_FALSE, &m_shaderShaderValues->matNormal[0][0]);
            glUniformMatrix4fv( m_ids[5], 1, GL_FALSE, &m_shaderShaderValues->matModelViewProj[0][0]);

            glUniform3f( m_ids[6], m_shaderShaderValues->lightPos.x, m_shaderShaderValues->lightPos.y, m_shaderShaderValues->lightPos.z);
            glUniform3f( m_ids[7], m_shaderShaderValues->camPos.x, m_shaderShaderValues->camPos.y, m_shaderShaderValues->camPos.z);
        }
    }
private:
    const SharedShaderValues*   m_shaderShaderValues;
    GLint                       m_ids[8];
};
```

Jaettujen arvojen laskeminen scenen update -metodissa (tämä pitää siis tehdä ennen materiaalin bindiä):

```
// Calculate needed stuff for m_sharedValues
m_sharedValues->matModel = m_matModel;
m_sharedValues->matView = m_matView;
m_sharedValues->matProj = m_matProjection;

slmath::mat4 matModelView = m_matView * m_matModel;
slmath::mat4 matModelViewProj = m_matProjection * matModelView;
slmath::mat4 matNormal = slmath::transpose(slmath::inverse(matModelView));

m_sharedValues->matModelView = matModelView;
m_sharedValues->matNormal = matNormal;
m_sharedValues->matModelViewProj = matModelViewProj;

m_sharedValues->lightPos = slmath::vec3(0.0, 70.0f, 70.0f);
m_sharedValues->camPos  = slmath::vec3(0.0, 70.0f, 70.0f);
```

Struktin tekeminen materiaalille

Tee luokka SimpleMaterialUniforms. Luokka sisältää muutaman väriarvon materiaalin ambient, diffuse ja specular-komponenteille. Luokka toimii vastaavasti, kuin GlobalShaderUniforms, mutta erona on se, että se sisältää myös GlobalShaderUniforms-luokan instanssin ja käyttää sitä asettamaan ns. globaalit ja jaetut shaderin uniformien arvot. Koodia alla:

```
// Simple material. Consists of material ambient, diffuse and specular colors.
class SimpleMaterialUniforms : public graphics::ShaderUniforms
{
public:
    slmath::vec4  vAmbient; /// Ambient color of the material (rgba)
    slmath::vec4  vDiffuse; /// Ambient color of the material (rgba)
    slmath::vec4  vSpecular; /// Specular color of the material (rgb). Specular exponent (a)

public:
    SimpleMaterialUniforms(graphics::Shader* shader, SharedShaderValues* sharedValues = 0)
        : ShaderUniforms(shader)
        , m_globalShaderUniforms( new GlobalShaderUniforms(shader, sharedValues) )
    {
    }

    virtual ~SimpleMaterialUniforms()
    {
    }

    virtual void getUniformLocations(graphics::Shader* shader)
    {
        m_globalShaderUniforms->getUniformLocations(shader);
        m_materialAmbientLoc    = glGetUniformLocation(shader->getProgram(), "g_Material.vAmbient");
        m_materialDiffuseLoc    = glGetUniformLocation(shader->getProgram(), "g_Material.vDiffuse");
        m_materialSpecularLoc   = glGetUniformLocation(shader->getProgram(), "g_Material.vSpecular");
    }

    virtual void bind(graphics::Shader* shader)
    {
        shader->bind();
        m_globalShaderUniforms->bind(shader);
        glUniform4fv(m_materialAmbientLoc,  1, &vAmbient.x );
        glUniform4fv(m_materialDiffuseLoc,  1, &vDiffuse.x );
        glUniform4fv(m_materialSpecularLoc, 1, &vSpecular.x );
    }

private:
    core::Ref<GlobalShaderUniforms> m_globalShaderUniforms;
    GLint                           m_materialAmbientLoc;
    GLint                           m_materialDiffuseLoc;
    GLint                           m_materialSpecularLoc;
};
```

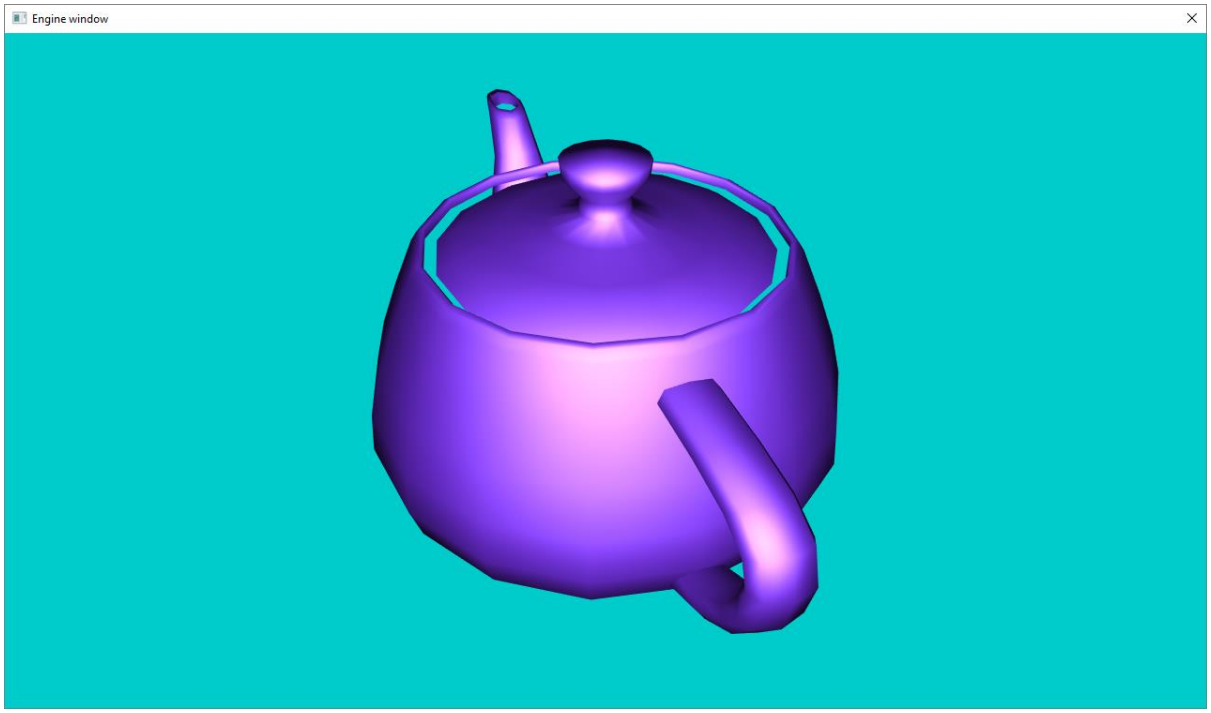
SimpleMaterialUniformssin luonti ja alustuskoodi korvaa vastaavan koodin edellisestä tehtävästä. SharedValues annetaan tässä tapauksessa SimpleMaterialUniforms konstruktorissa. SimpleMaterialUniforms -luokan instanssille annetaan sopivat värit luonnin jälkeen:

```
core::Ref<graphics::Shader> shader =
    new graphics::Shader("assets/Blinn-Phong.vs", "assets/Blinn-Phong.fs",
    attributes, sizeof(attributes) / sizeof(FRM_SHADER_ATTRIBUTE));

SimpleMaterialUniforms* simpleMaterialUniforms = new SimpleMaterialUniforms(shader,m_sharedValues);

// Material values for mesh
simpleMaterialUniforms->vAmbient    = slmath::vec4(0.5f, 0.2f, 1.0f, 1.0f);
simpleMaterialUniforms->vDiffuse    = slmath::vec4(0.5f, 0.2f, 1.0f, 1.0f);
simpleMaterialUniforms->vSpecular   = slmath::vec4(1.0f, 1.0f, 1.0f, 5.0f);
m_material = simpleMaterialUniforms;
```

Lopputulos



Lisätehtäviä

- Kokeile rendata toinen teepannu ruudulle erivärisellä materiaalilla.
- Muuttele kameran paikkaa ja orientaatiota ja tutkaile, miten pyöritykset ja liikutukset vaikuttaa näkymään
- Kokeile valon position muuttamista