Tehtävä 6. Cube Reflection

Tehtävässä muokataan tehtävän 6 pohjalta uusi scene, joka rendaa teepannun ruudulle käyttäen blinn-phong –valaisumallia ja teksturointia. Lisäksi teepannulle lisätään toiminnallisuus, joka lisää väriä teepannun pinnalle ympäröivästä cube tekstuurista. Tee joko uusi skene ja materiaali tai muokkaa tehtävän 5 skeneä ja materiaalia. Suositeltavaa on tehdä uusi scene ja materiaali tai lisätä toinen teepannun piirtäminen ruudulle toisen materiaalin avulla.

Cube tekstuurin lataaminen ja luominen

```
// Load cube images
eastl::string name = "BedroomCubeMap";

Engine::Ref<Engine::Image> cubeImageRefs[6];
cubeImageRefs[0] = Engine::Image::loadFromTGA("assets/" + name + "_RT.tga");
cubeImageRefs[1] = Engine::Image::loadFromTGA("assets/" + name + "_LF.tga");
cubeImageRefs[2] = Engine::Image::loadFromTGA("assets/" + name + "_DN.tga");
cubeImageRefs[3] = Engine::Image::loadFromTGA("assets/" + name + "_UP.tga");
cubeImageRefs[4] = Engine::Image::loadFromTGA("assets/" + name + "_FR.tga");
cubeImageRefs[5] = Engine::Image::loadFromTGA("assets/" + name + "_BK.tga");

Engine::Image* cubeImages[6];
for(int i=0; i<6; ++i )
    cubeImages[i] = cubeImageRefs[i].ptr();

// Create cube map and set data to it.
Engine::Ref<Engine::TextureCube> cubeMap = new Engine::TextureCube();
cubeMap->setData(cubeImages);
```

Uusi luokka cube mapped materiaalille

Tee luokka SimpleMaterialWIthTextureUniformsCube tms. Luokka käyttää kantaluokkana edellisessä tehtävässä totetutettua SimpleMaterialUniformsTextured-luokkaa ja laajentaa sen toimintaa niin, että se lisää cube tekstuurin asettamisen shaderille:

```
class SimpleMaterialWithTextureUniformsCube : public SimpleMaterialUniformsTextured
{
public:
    Engine::Ref<Engine::TextureCube> cubeMap;

    SimpleMaterialWithTextureUniformsCube(Engine::Shader* shader, SharedShaderValues* sharedValues)
        : SimpleMaterialUniformsTextured(shader, sharedValues)
    {
    }

    virtual ~SimpleMaterialWithTextureUniformsCube()
    {
    }

    virtual void getUniformLocations(Engine::Shader* shader)
    {
        SimpleMaterialUniformsTextured::getUniformLocations(shader);

        m_cubeMapLoc    = glGetUniformLocation(shader->getProgram(), "s_cubeMap");
    }

    virtual void bind(Engine::Shader* shader)
    {
        SimpleMaterialUniformsTextured::bind(shader);

        // Bind cube texture to texture sampler unit # 1
        glActiveTexture ( GL_TEXTURE0 + 1);
        glBindTexture ( GL_TEXTURE_CUBE_MAP, cubeMap->getTextureId() );

        // Set sampler unit 1 to be used as sampler for cube map uniform.
        glUniform1i( m_cubeMapLoc, 1 );

    }

private:
    GLint m_cubeMapLoc;
};
```

Oleelliset muutokset varjostinkoodeihin

Kopioi fragment shader koodi Blinn-phong.fs ja nimeä esim. Blinn-phong-textured-cube.fs. Vertex shaderiin ei tule muutoksia.

Oleelliset muutokset fragment shaderiin:

- lisää uusi samplerCube-tyyppinen uniform ”uniform samplerCube s_cubeMap;”
- Tekselin värin samplaaminen cube mapistä onnistuu koodilla:
    - vec3 vNormal = normalize(g_vNormalES);
    - vec3 vView = normalize(g_vViewVecES);
    - vec4 vEnvColor = textureCube(s_cubeMap, reflect(vView,vNormal));
        - "g_vNormalES" on kameran view-vektori (tulee varying muuttujassa vertex shaderiltä)
        - "g_vViewVecES" on pinnan normaalivektori (tulee varying muuttujassa vertex shaderiltä)
- Käytä tekstuurista poimittua väriarvoa lopullisen värin laskemiseen lisäämällä koodirivi: "gl_FragColor.rgba += vEnvColor * 0.1f;"

- Kerroin 0.1 on materiaalin "glossyness arvo", arvo 0.1 on ravistettu hihasta.

Lisätehtäviä

- Glossyness arvo on nyt kovakoodattuna suoraan fragment shaderiin. Muuta toimintaa niin, että se on C++ koodin puolelsta konfffattavissa sopivaan arvoon (Muuta material-strukteja)
- Glossyness -arvo voidaan hakea myös per-pixel -kohtaisesti esimerkiksi diffuusin tekstuurin alpha -kanavasta tai mahdollisesti jostakin muusta tekstuurista, jolloin saadaan objektin eri kohtiin eri tavoin heijastavia kohtia. Kokeile toteuttaa materiaali, jossa glossynes arvo poimitaan jostakin erillisestä tekstuurista (glossy map).
- Kokeile rendata Cube map "normaalina objektina" ruudulle (ns Skybox). Periaate perustuu siihen, että kameran view vektoria käytetään poimimaan ruudun taustaväri suoraan cubemapistä:
    - Rendaa full screen quad ruudulle käyttäen sopivaa shaderia ja cube mappiä.
    - Laske Vertex shaderissä kameran view-vektori.
    - Fragment shaderissä käytä view vektoria ja laita se parametrinä textureCube() funktiokutsun toiseen parametriin ja käytä saatua väriä fragmentin väriarvona
    - Tässä yksi artikkeli netistä, jossa tekniikka kuvattu tarkemmin: https://forum.qt.io/topic/38607/how-to-creating-a-skybox-using-samplercube-and-the-qopengltexture-class