# Tehtävä 5. Tekstuurin lisääminen

Tehtävässä muokataan tehtävän 5 pohjalta uusi scene, joka rendaa teepannun ruudulle käyttäen blinn-phong –valaisumallia ja teksturointia. Tee joko kokonaan uusi skene ja materiaali tai muokkaa tehtävän 4 skeneä ja materiaalia. Suositeltavaa on tehdä uusi scene ja materiaali tai lisätä toinen teepannun piirtäminen ruudulle toisen materiaalin avulla.

## Oleelliset muutokset varjostinkoodeihin

Kopioi fragment shader koodi Blinn-phong.fs ja nimeä esim. Blinn-phong-textured.fs. Kopioi ja nimeä vastaavasti myös vertex shader –tiedosto.

Oleelliset muutokset fragment shaderiin:

- lisää uusi sampler2D-tyyppinen uniform ”uniform sampler2D s_diffuseMap;”
- lisää uusi “varying” tekstuurikoordinaatteille esim. “varying vec2 g_texCoordsOS”
    - Nimeäminen ”OS” tarkoittaa, että tekstuurikoordinaatit ovat “Object Spacessa”, eli objektin koordinaatistossa (ei relevanssia vielä tässä vaiheessa)
- fragment shader funktiossa, käytä tekstuurista poimittua väriarvoa ambient ja diffuse värine laskennassa. Koodi menee jokseenkin tähän tyyliin:
    - vec4 texelColor = texture2D(s_diffuseMap, g_texCoordOS);
    - gl_FragColor += g_Material.ambientColor * texelColor;
    - gl_FragColor += g_Material.diffuseColor * texelColor;

Oleelliset muutokset vertex shaderiin:

- Lisää uusi attribute: “attribute vec2 g_vTextureCoords;”
- Lisää uusi varying “varying vec2 g_texCoordsOS;”
- vertex shader funktiossa välitä tekstuurikoordinaatit attribute-taulukosta fragment shaderille menevälle varying muuttujalle tyyliin:
    - g_texCoordsOS = g_ vTextureCoords;

## Uusi Strukti teksturoidulle materiaalille

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

```
class SimpleMaterialWithTextureUniforms : public SimpleMaterialUniforms
{
public:
    core::Ref<graphics::Texture> diffuseMap;

    SimpleMaterialWithTextureUniforms(graphics::Shader* shader, SharedShaderValues* sharedValues)
        : SimpleMaterialUniforms(shader,sharedValues)
    {
    }

    virtual ~SimpleMaterialWithTextureUniforms()
    {
    }

    virtual void getUniformLocations(graphics::Shader* shader)
    {
        SimpleMaterialUniforms::getUniformLocations(shader);

        m_diffuseMapLoc = glGetUniformLocation(shader->getProgram(), "s_diffuseMap");
    }


    virtual void bind(graphics::Shader* shader)
    {
        SimpleMaterialUniforms::bind(shader);
        // Bind diffuse texture to taxture sampler unit # 0
        glActiveTexture ( GL_TEXTURE0 + 0);
        glBindTexture ( GL_TEXTURE_2D, diffuseMap->getTextureId() );

        // Set sampler unit 0 to be used as sampler for diffuse map uniform.
        glUniform1i( m_diffuseMapLoc, 0 );
    }

private:
    GLint m_diffuseMapLoc;
};
```

## Tekstuurin lataaminen

graphics::Image* img = graphics::Image::lodFromTGA(“minun_tekstuuri.tga”);

graphics::Texture* texture = new graphics::Texture();

texture->setData(img);