Tehtävä 2 – Shaderin tekeminen

- Kopioi FullScreenQuadScene ja nimeä se uudelleen esim. FullScreenQuadShaderScene
- Kopioi käyttämäsi vertex shader tiedostoon SimpleGraphicsTest\assets\FullScreenQuad.vs ja fragment shader tiedostoon SimpleGraphicsTest\ assets\PlasmaEffect.fs. Tiedostot kopioituvat automaattisesti oikeaan paikkaan buildin yhteydessä.
- Käytä moottorin Shader-luokkaa OpenGL id:n sijasta:

Lisää luokan jäsen:

```
core::Ref<graphics::Shader> m_shader;
```

Lataa shader konstruktorissa:

```
int numAttributes = sizeof(attributes) / sizeof(FRM_SHADER_ATTRIBUTE);
m_shader = new graphics::Shader("assets/FullScreenQuad.vs", "assets/ PlasmaEffect.fs", attributes, numAttributes);
```

Shaderin piirrossa onnistuu Shader::bind -metodilla:

```
m_shader->bind();
```

- Poista vanha shader -koodi skenestä ja testaa, että sovellus toimii, kuten aiemminkin

Tehtävä 2 - Materiaalin määrittäminen

- Tee uusi tiedosto esim. MyMaterials.h ja cpp tänne tulee tarvittavat tietorakenteet materiaaleille:
- Lisää struct SharedShaderValues, jolla on yksi float jäsen nieltään "totalTime"
- tee luokka "GlobalShaderUniforms", joka käyttää kantaluokkana "graphics::ShaderUniforms" -luokkaa.
  - Konstruktorin parametrit:
    - graphics::Shader* shader
    - const SharedShaderValues* shaderShaderValues = 0
- Kantaluokkalle pitää välittää shader -konstruktorissa
- Shader values tallennetaan luokan jäsenmuuttujaan:
  - const SharedShaderValues* m_shaderShaderValues;
- Tee tyhjä toteutus metodeille ja varmista, että ohjelma kääntyy:
  - virtual void getUniformLocations(graphics::Shader* shader)
  - virtual void bind(graphics::Shader* shader)

Tehtävä 2 - Shared arvojen käyttäminen

- Lisää Esimerkki scenen luokkaan seuraavat luokan jäsenet:
  - SharedShaderValues m_sharedValues;
  - core::Ref<graphics::ShaderUniforms> > m_material;
  - Luo niihin m_material -muuttujeen luokasta GlobalShaderUniforms instanssi esimerkkiskenen konstruktorissa
  - m_sharedValues.totalTime -pitää alustaa arvoon 0.0 ja päivittää sitä updatessa
- Anna luotu shader SharedShaderValues -objektille konstruktorissa
- Kokeile, että softa kääntyy ja toimii, kuten ennenkin

Tehtävä 2 - Shaderin bindaaminen ShaderUniform -luokan avulla

- Lisää koodia GlobalShaderUniforms::bind -metodiin
  - shader->bind();
- Muuta esimerkkiskenen koodia niin, että se ei enää kutsu suoraan shader-luokan bindiä, vaan sen sijaan kutsutaan m_material->bind();
- Kokeile, että softa toimii, kuten ennenkin

Tehtävä 2 - Uniformien asettaminen materiaalista

- Lisää luokkaan GlobalShaderUniforms jäsen: GLint m_id;
- Lisää koodia uniformien lokaatioiden hakemiseksi GlobalShaderUniforms::getUniformLocations-metodiin:

```
m_id = glGetUniformLocation(shader->getProgram(),  "totalTime");
```

- Lisää koodia myös bind-metodiin, joka asettaa shaderille uniformien arvon (tämä koodi tulee sen shader bindin jälkeen):

```
if( m_shaderShaderValues  )
{
glUniform1f( m_id, m_shaderShaderValues->totalTime);
}
```

- Poista vastaava vanha koodi scenestä ja kokeile, että toimii ok