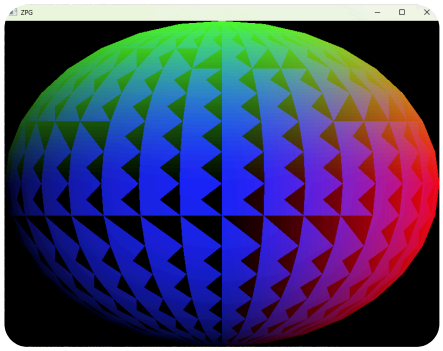
LMS Moodle VŠB-TUO: 460-2021/03 Základy počítačové grafiky (2024/2025 ZS): Sekce: Cvičení 3

Složitější modely

Podívejte se na použití složitějších modelů (Models.zip), které mají více parametrů. Připravíme si modely s pozici a normálou.

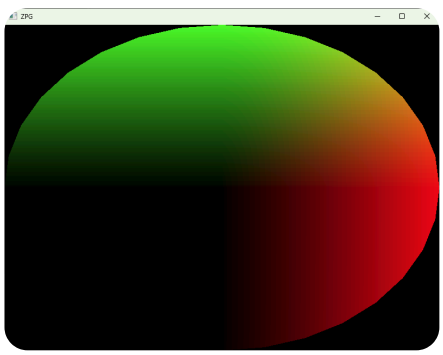
```
const a[] ={
   -.5f, -.5f, .5f,  0, 0, 1,
   -.5f, .5f, .5f,  0, 0, 1,
    .5f, .5f, .5f,  0, 0, 1,
    .5f, -.5f, .5f,  0, 0, 1 };

// přidejte další atribut
glEnableVertexAttribArray(0);
glEnableVertexAttribArray(1);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (GLvoid*)0);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), (GLvoid*)(3 * sizeof(float)));
```

Můžeme zobrazit třeba kouli, kde normálu použijeme jako barvu.



Pokud zapnete z-buffer (blíže na některé z dalších přednášek), tak uvidíme vše správně.

```
glEnable(GL_DEPTH_TEST);//Do depth comparisons and update the depth buffer.
while (!glfwWindowShouldClose(window)) {
  ...
```



Transformace

Vytvoříme si matici 4x4 (transformaci) pomocí knihovny GLM (popřípadě sami).

```
glm::mat4 M = glm::mat4(1.0f); // construct identity matrix

M = glm::rotate(glm::mat4(1.0f),angle,glm::vec3(0.0f, 1.0f, 0.0f));
M = glm::rotate(M, angle, glm::vec3(1.0f, 0.0f, 0.0f));
M = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, myView));
M = glm::scale(glm::mat4(1.0f), glm::vec3(0.5f));
```

Upravíme vertex shader tak, že si uvnitř vytvoříme uniformní proměnnou, kam budeme naši matici posílat. A následně každý vrchol touto proměnnou (matici) vynásobíme ve vertex shaderu. Pozor na správné násobení (nasobení matic není komutativní).

```
//Vertex shader
...
uniform mat4 modelMatrix;
...
main{
...
```

Přenásobíme každý vrchol ve vertex shaderu naší transformační matici (už víme proč je 4x4).

```
gl_Position = modelMatrix * vec4 (vp, 1.0);
```

Zjistíme pozici naši uniformní proměnné v shader programu (přidejte test na -1, kterou vrátí pokud proměnnou nenašel)

```
GLint idModelTransform = glGetUniformLocation(programID, "modelMatrix");
```

Pošlete naši matici do shaderu

```
//Render

glUseProgram(programID);

glUniformMatrix4fv(idModelTransform, 1, GL_FALSE, &M[0][0]);

//location, count, transpose, *value
```

Vytvořte více těles s různými transformacemi a různými shadery.
Zjistěte jak vypadá souřadný systém v OpenGL (osy X,Y a Z).
Na transformace můžete použít návrhový vzor kompozit.

Výsledek odešlete na kelvin.
