Sami Emre Erdogan
2019280513

Assignment 6 Illumination effect

This assignment was referenced from our tutorial 9_2 and was built on top of it. So, there can be similarities between them. We will have to use our object to load our eight.uniform.obj as we did for our assignment 1 so we had to make sure that we were calculating the normal vectors for the triangles within our object respect to our x y z coordinates. For an example of a triangle of ABC

```
std::vector<GLfloat> vertices = {};
for (int index = 0; index < vertices_obj.size(); index+=3) {
    GLfloat a_x = vertices_obj[index + 1].x - vertices_obj[index].x;
    GLfloat a_y = vertices_obj[index + 1].y - vertices_obj[index].y;
    GLfloat a_z = vertices_obj[index + 1].z - vertices_obj[index].z;

    GLfloat b_x = vertices_obj[index + 2].x - vertices_obj[index].x;
    GLfloat b_y = vertices_obj[index + 2].y - vertices_obj[index].y;
    GLfloat b_z = vertices_obj[index + 2].z - vertices_obj[index].z;

    GLfloat n_x = a_y * b_z - a_z * b_y;
    GLfloat n_y = a_z * b_x - a_x * b_z;
    GLfloat n_z = a_x * b_y - a_y * b_x;

    for (int index1 = 0; index1 < 3; index1++) {
        vertices.push_back(vertices_obj[index + index1].x);
        vertices.push_back(vertices_obj[index + index1].y);
        vertices.push_back(vertices_obj[index + index1].z);
        vertices.push_back(n_x) ,vertices.push_back(n_y) , vertices.push_back(n_z);
```

The lightning has been calculated from our material.frag.glsl referenced from our 9_2 tutorial. Also, the lighting effect is also been calculated by referencing from our tutorial which has the components of ambient diffuse specular.

```
glm::vec3 lightColor;
lightColor.x = sin(glfwGetTime() * 2.0f);
lightColor.y = sin(glfwGetTime() * 0.7f);
lightColor.z = sin(glfwGetTime() * 1.3f);
glm::vec3 diffuseColor = lightColor   * glm::vec3( scalar: 0.5f); // decrease the influence
glm::vec3 ambientColor = diffuseColor * glm::vec3( scalar: 0.2f); // low influence
GLint lightAmbientLoc = glGetUniformLocation(lightingShader.Program, "light.ambient");
GLint lightDiffuseLoc = glGetUniformLocation(lightingShader.Program, "light.diffuse");
GLint lightSpecularLoc = glGetUniformLocation(lightingShader.Program, "light.specular");
glUniform3f(lightAmbientLoc, ambientColor.x, ambientColor.y, ambientColor.z);
glUniform3f(lightDiffuseLoc, diffuseColor.x, diffuseColor.y, diffuseColor.z);
glUniform3f(lightSpecularLoc, 1.0f, 1.0f, 1.0f);

// material properties
GLint matAmbientLoc = glGetUniformLocation(lightingShader.Program, "material.ambient");
GLint matDiffuseLoc = glGetUniformLocation(lightingShader.Program, "material.diffuse");
GLint matSpecularLoc = glGetUniformLocation(lightingShader.Program, "material.specular");
GLint matShineLoc = glGetUniformLocation(lightingShader.Program, "material.shininess");

glUniform3f(matAmbientLoc, 1.0f, 0.5f, 0.31f);
glUniform3f(matDiffuseLoc, 1.0f, 0.5f, 0.31f);
glUniform3f(matSpecularLoc, 0.5f, 0.5f, 0.5f);
glUniform1f(matShineLoc, 32.0f);
```

Sami Emre Erdogan
2019280513

Calculation of normal vector's vertex surrounding triangles for smooth shading.

```
for (int index = 0; index < surrounding_indexes.size(); index++) {
    smooth_norms.push_back(glm::vec3( _x: 0,  _y: 0,  _z: 0));
}

for (int index = 0; index < surrounding_indexes.size(); index++) {
    glm::vec3 norm( _x: 0, _y: 0, _z: 0);
    for (int i = 0; i < surrounding_indexes[index].size(); i++) {
        norm.x += vertices[surrounding_indexes[index][i] * 3 * 6 + 3];
        norm.y += vertices[surrounding_indexes[index][i] * 3 * 6 + 4];
        norm.z += vertices[surrounding_indexes[index][i] * 3 * 6 + 5];
    }
    norm /= surrounding_indexes[index].size();
    smooth_norms[index] = norm;
}
for (int index = 0; index < vertices.size(); index += 6) {
    vertices_smooth.push_back(vertices[index]);
    vertices_smooth.push_back(vertices[index + 1]);
    vertices_smooth.push_back(vertices[index + 2]);
    vertices_smooth.push_back(smooth_norms[vertexIndices[index / 6] - 1].x);
    vertices_smooth.push_back(smooth_norms[vertexIndices[index / 6] - 1].y);
    vertices_smooth.push_back(smooth_norms[vertexIndices[index / 6] - 1].z);
```

Finally, you can press The key M for mode change. W,A,S,D,R and F for control.

```
void do_movement()
{
    // Camera controls
    if (keys[GLFW_KEY_W])
        camera.ProcessKeyboard( direction: FORWARD, deltaTime);
    if (keys[GLFW_KEY_S])
        camera.ProcessKeyboard( direction: BACKWARD, deltaTime);
    if (keys[GLFW_KEY_A])
        camera.ProcessKeyboard( direction: LEFT, deltaTime);
    if (keys[GLFW_KEY_D])
        camera.ProcessKeyboard( direction: RIGHT, deltaTime);
    if (keys[GLFW_KEY_R])
        camera.ProcessKeyboard( direction: UP, deltaTime);
    if (keys[GLFW_KEY_F])
        camera.ProcessKeyboard( direction: DOWN, deltaTime);
```

```
if (key == GLFW_KEY_M && action == GLFW_PRESS)
    mode_normal = !mode_normal;
if (key >= 0 && key < 1024)
{
    if (action == GLFW_PRESS)
        keys[key] = true;
    else if (action == GLFW_RELEASE)
        keys[key] = false;
```