Texture Mapping

Canhua Huang & Fan Yang

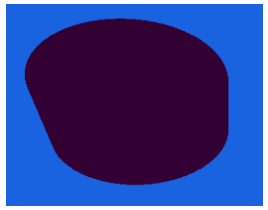
Texture Mapping

The first step of texture mapping is reading the object file and material file. Then we store these data in VBO.

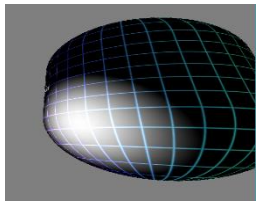
```
if (tag == "newmtl") {
    line_stream >> name;
} else if (tag == "Ka") {
    line_stream >> this->Ka[0];
    line_stream >> this->Ka[1];
    line_stream >> this->Ka[2];
} else if (tag == "Kd") {
    line_stream >> this->Kd[0];
    line_stream >> this->Kd[1];
    line_stream >> this->Kd[2];
} else if (tag == "Ks") {
    line_stream >> this->Ks[0];
    line_stream >> this->Ks[1];
    line_stream >> this->Ks[2];
} else if (tag == "Ns") {
    line_stream >> this->Ns;
} else if (tag == "Tr" || tag == "d") {
    line_stream >> this->Tr;
} else if (tag == "map_Ka") {
    this->map_Ka.append(dir);
    string tmp;
    line_stream >> tmp;
    this->map_Ka.append(tmp);
} else if (tag == "map_Kd") {
    this->map_Kd.append(dir);
    string tmp;
    line_stream >> tmp;
    this->map_Kd.append(tmp);
} else if (tag == "map_Ks") {
    this->map_Ks.append(dir);
    string tmp;
    line_stream >> tmp;
    this->map_Ka.append(tmp);
} else if (tag == "map_Ns") {
    this->map_Ns.append(dir);
    string tmp;
    line_stream >> tmp;
    this->map_Ka.append(tmp);
}
```

```
glGenBuffers(1, &buffer);
glBindBuffer(GL_ARRAY_BUFFER, buffer);
glBufferData(GL_ARRAY_BUFFER, points_size + normals_size + textures_size,
             NULL, GL_DYNAMIC_DRAW);
glBufferSubData(GL_ARRAY_BUFFER, 0, points_size, this->points.data());
glBufferSubData(GL_ARRAY_BUFFER, points_size,
             normals_size, this->normals.data());
glBufferSubData(GL_ARRAY_BUFFER, textures_offst,
             textures_size, this->texture_vs.data());
```

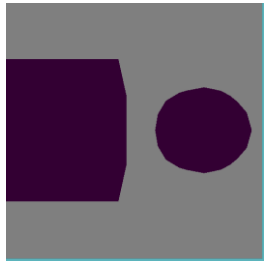
According to the data in material file, we can render the object and map the texture on the object.

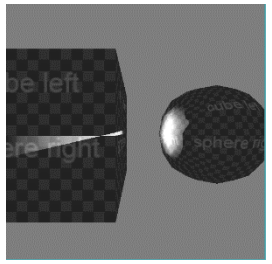


Original object



Texture Mapping





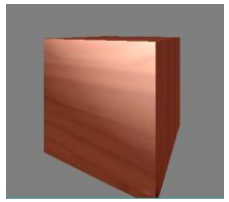
* The light source is in the cube

Introduction

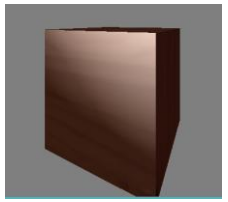
There are 3 parts in our project. The main part of our project is texture mapping, we can map different textures on multiple objects by the object files and material files; The second part of the project is shading, we can interactively edit the lights of the objects we used in texture mapping; In the third part of the project, we add some texture functions to our project so that we can combine the color in the texture map with original color of the object.

Shading

Based on Phong Illumination Model, for each light source and each color component, the Phong model can be written as $I = k_d L_d \, l \cdot n + k_s \, L_s \, (v \cdot r)^{\alpha} + k_a \, L_a$. The three coefficients in the Phong Illumination model are ambient, diffuse and specular. By chaninge them, we can get different redering color.

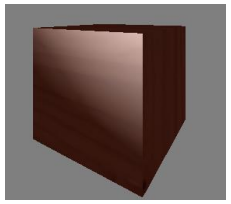


Ambient++



Ambient--

Diffuse++



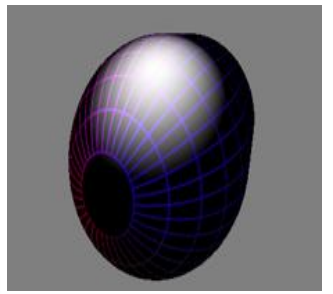
Diffuse--

Texture Function

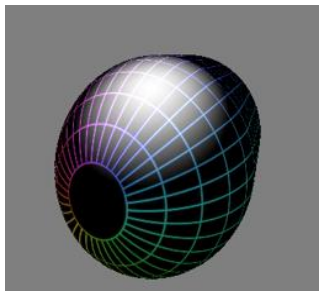
Before adding texture function, the values in the texture map have been used directly as colors to be painted on the surface being rendered. But we also can use the values in the texture map to modulate the color in which the surface would be rendered without texturing or to combine the color in the texture map with the original color of the surface.

Based on the internal format GL_RGBA, we use two ways to approach: Replace and Modulate.

For the replace function, we change $C = C_s$ and $A = A_s$; for the modulate function we change $C = C_f \, C_s$ and $A = A_f \, A_s$



Modulate



Replace

Oregon State University OSU