INF1761 Projeto 2 - Computação Gráfica

Lucas Angel Larios Prado - 2020723

Paulo de Tarso Fernandes - 2011077

OBS: Ao rodar o código, mudar path do vertex.glsl e fragment.glsl

```
ShaderPtr shd = Shader::Make(light,"world");
shd->AttachVertexShader("D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\shaders\\default\\vertex.glsl");
shd->AttachFragmentShader("D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\shaders\\default\\fragment.glsl");
shd->Link();



ShaderPtr shd_clip = Shader::Make(light,"world");
shd_clip->AttachVertexShader("D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\shaders\\clipplane\\vertex.glsl");
shd_clip->AttachFragmentShader("D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\shaders\\clipplane\\fragment.glsl
shd_clip->Link();



ShaderPtr shd_reflect = Shader::Make(light, "world");
shd_reflect->AttachVertexShader("D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\shaders\\sky_reflect\\vertex.gls
shd_reflect->AttachFragmentShader("D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\shaders\\sky_reflect\\fragment
shd_reflect->Link();
```

- **SkyBox e reflexão com textura cúbica, Efeito Fog e Plano de Corte :**

A aplicação possui três shaders: o "shd" passado no nó root, que lida com o skybox e a neblina; o "shd_reflect", passado no nó filho do root, responsável por aplicar a reflexão em uma esfera flutuante no espaço; e, por último, o "shd_clip", que é passado em outro nó filho do root, também representando uma esfera no entanto partida ao meio.

fragment shader “shd” (trecho) do código para implementação do skybox e efeito fog:

```
    color = texture(sky, f.pos.xyz);
    float fog = exp(-pow(fdensity * distance(f.pos,cpos), 2));
    color = fog * color + (1.0 - fog) * fcolor;
```

Fragment Shader para reflexão com textura cúbica.

```
#version 410

uniform samplerCube sky;

in data {
```

```
  vec3 normal;
  vec3 view;
} f;

out vec4 color;

void main (void)
{
    vec3 normal = normalize(f.normal);
    vec3 view = normalize(f.view);
    vec3 dir = reflect (-view, normal) * vec3 (1, 1, -1);
    color = 0.9f * texture (sky, dir);
}
```

Vertex Shader do plano de corte.

```
#version 410

layout(location = 0) in vec4 pos;
layout(location = 1) in vec3 normal;
layout(location = 3) in vec2 texcoord;

uniform vec4 lpos;  // light pos in lighting space
uniform vec4 cpos;  // camera pos in lighting space
uniform mat4 Mv;
uniform mat4 Mn;
uniform mat4 Mvp;
uniform vec4 clipplane[1];
out float gl_ClipDistance[1];
out data {
  vec3 normal;
  vec3 view;
  vec3 light;
  vec2 texcoord;
} v;

void main (void)
{
  vec3 p = vec3(Mv*pos);
  vec4 p2 = Mv*pos;
  if (lpos.w == 0)
    v.light = normalize(vec3(lpos));
  else
    v.light = normalize(vec3(lpos)-p);
  v.view = normalize(vec3(cpos)-p);
  v.normal = normalize(vec3(Mn*vec4(normal,0.0f)));
  v.texcoord = texcoord;
  gl_Position = Mvp*pos;
  gl_ClipDistance[0] = dot(clipplane[0],p2);
}
```

Criação do skybox, clipplane e das variáveis fdensity e fcolor, responsáveis pelo efeito de neblina, no arquivo principal da aplicação:

```
SkyBoxPtr skybox = SkyBox::Make();

ClipPlanePtr clipplane = ClipPlane::Make("clipplane", 0.0f, -1.0f, 0.0f, 0.0f);

auto fcolor = Variable<glm::vec4>::Make("fcolor", glm::vec4(1.0f, 1.0f, 1.0f, 1.0f));
auto fdensity = Variable<float>::Make("fdensity", 0.200f);
```

Grafo de cena:

```
NodePtr root = Node::Make(
    shd,
    {fdensity,fcolor},
    {
```

```
        Node::Make({sky_tex},{skybox}),
        Node::Make(shd_clip,trf1,{clipplane,white,sphereTex},{sphere}),
        Node::Make(shd_reflect,trf2,{sphere}),

    }
);
```

Aprimoramos ainda a classe encarregada pela a textura cúbica. Inicialmente, a classe fornecida apenas aceitava uma imagem, que era um cubo desmontado. No entanto, aprimoramos a solução para não só aceitar essa imagem, mas também para aceitar seis imagens independentes, com as quais é possível construir o skybox.

texcube.cpp(sobrecarga do construtor):

```
TexCube::TexCube(const std::string& varname, std::vector<std::string> faces)
{


    glGenTextures(1, &m_tex);
    glBindTexture(GL_TEXTURE_CUBE_MAP, m_tex);
    for (unsigned int i = 0; i < faces.size(); i++)
    {
        ImagePtr img = Image::Make(faces[i], false);
        glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X + i, 0, GL_RGB, img->GetWidth(), img->GetHeight(), 0, img->GetNChannels() == 3 ? GL_RGB
    }
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);


}
```

Ao testarmos a criação do skybox por meio de seis imagens independentes, observamos que o skybox ficava invertido. Para resolver esse problema, modificamos o arquivo image.cpp e adicionamos um parâmetro de flip, pois anteriormente no código fornecido, stbi_set_flip_vertically_on_load estava sempre definido como true.

Image.cpp

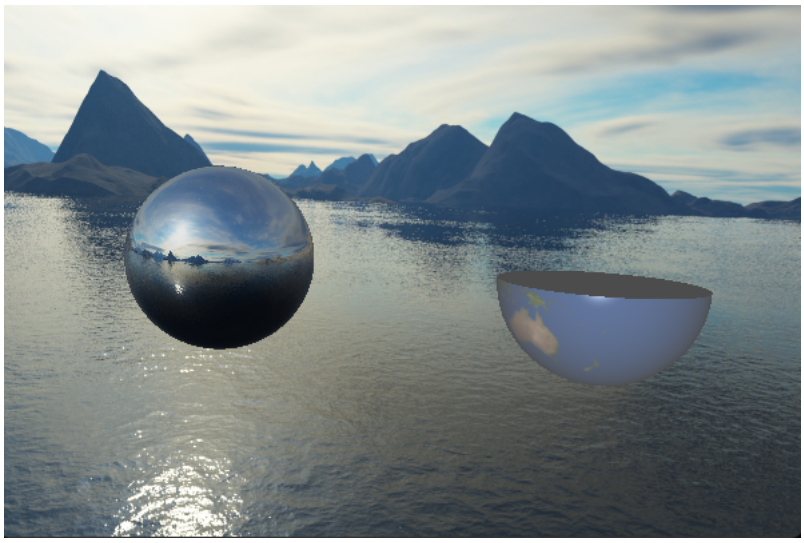
```
Image::Image(const std::string& filename, bool flip)
{

    stbi_set_flip_vertically_on_load(flip);

    m_data = stbi_load(filename.c_str(), &m_width, &m_height, &m_nchannels, 0);
    if (!m_data) {
        std::cerr << "Could not load image: " << filename << std::endl;
        exit(1);
    }
}
```

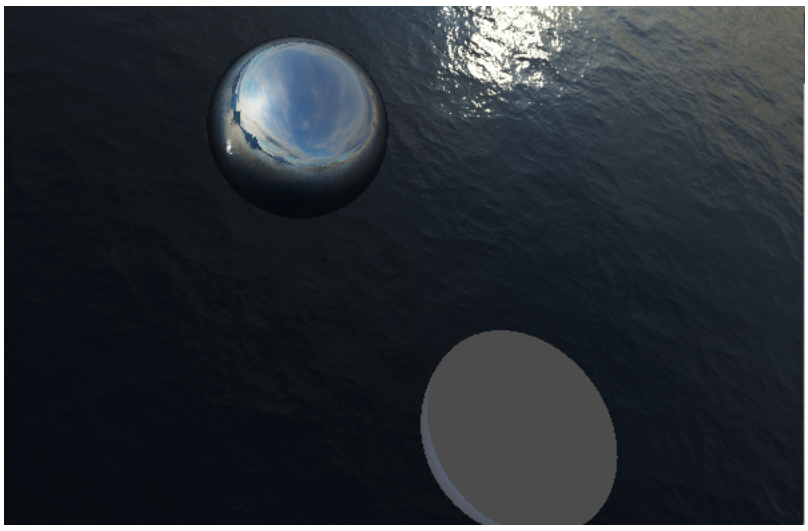
Criando textura cúbica com 6 imagens:

```
std::vector<std::string> faces = {
    "D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\images\\skyboxImages\\right.jpg",
    "D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\images\\skyboxImages\\left.jpg",
    "D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\images\\skyboxImages\\top.jpg",
    "D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\images\\skyboxImages\\bottom.jpg",
    "D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\images\\skyboxImages\\front.jpg",
    "D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\images\\skyboxImages\\back.jpg"
};
  //AppearancePtr sky_tex = TexCube::Make("sky", "D:\\Faculdade\\7 periodo\\computação gráfica\\tarefas\\tarefa_2_1\\files\\images\\skybox.
  AppearancePtr sky_tex = TexCube::Make("sky",faces);
```

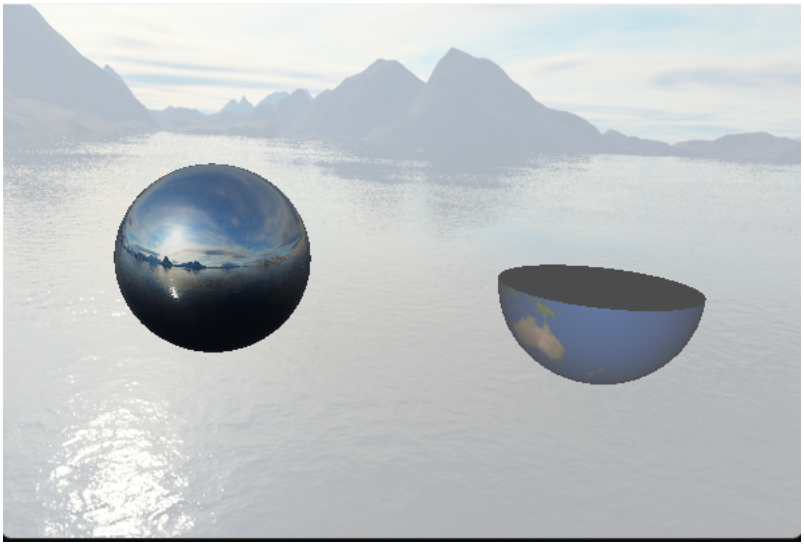
Resultados:

Skybox, reflexão e plano de corte (sem efeito FOG - densidade=0.0f):

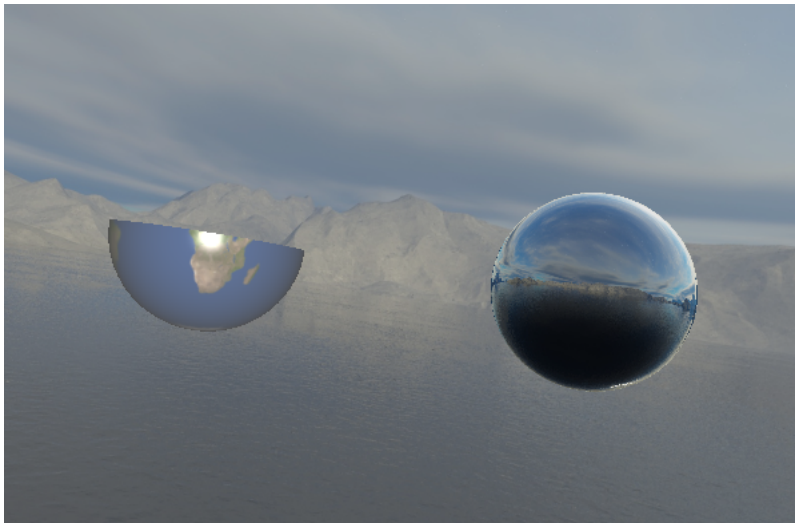




Skybox, reflexão e plano de corte (COM efeito FOG - densidade=0.2f):



Skybox, reflexão e plano de corte (COM efeito FOG - densidade=0.1f):



OBS: Será enviado dois videos, com e sem efeito fog(densidade 0.2f apenas para melhor vizualização).