Mireia Espuga 206319
Aina Vendrell 205975
Anna Domènech 204798

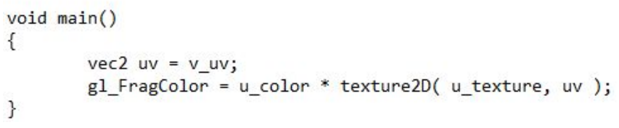
Advanced Visualization

Lab 1 - Introduction to ACG

The main purpose of this lab is to understand and implement the basic concepts of advanced computer graphics. In order to achieve this, we have implemented the following points:

Texture Shader

In this section we have to render a node with textures without illumination, in order to do that we have implemented the following texture shader:



```
void main()
{
	vec2 uv = v_uv;
	gl_FragColor = u_color * texture2D( u_texture, uv );
}
```

In order to render the node we will consider both the assigned texture and the color of the given object. To do that we use the texture2D function which returns a texel, i.e. the (color) value of the texture for the given coordinates, and then we combine the texel with the preassigned color.

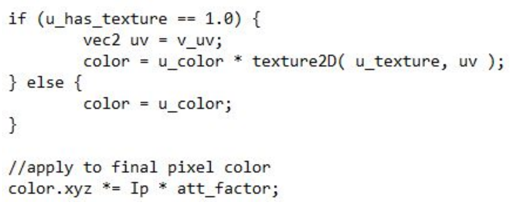
View *Annex 1.1 Texture Shader* to see its renderization.

Phong Material

In this section we have to implement the phong equation for illumination, we have considered the properties of the object material and its texture to compute the interaction that the object has with the lights. In the shader[1] have implemented the Phong's equation:

$$I = k_a i_a + k_d (N \cdot L) i_i + k_s (R \cdot V)^n i_i$$

In order to take into account the lights and the texture we have added the following lines:

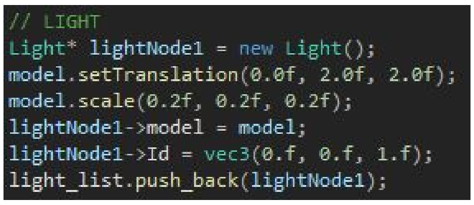


```
if (u_has_texture == 1.0) {
	vec2 uv = v_uv;
	color = u_color * texture2D( u_texture, uv );
} else {
	color = u_color;
}

//apply to final pixel color
color.xyz *= Ip * att_factor;
```

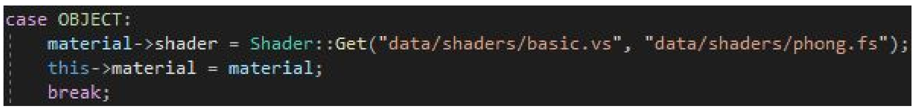
Where att_factor is the attenuation factor that takes into account the light intensity depending on the distance with the object.

In order to render multiple lights we have used the multipass method, where we repeatedly render the object several times but with a different light every time. We then accumulate the illumination using the blending option. In the case of the skybox and the reflective material we don't apply this method. Notice that we have created a new class called **light**, to be able to modify its properties with the imGUI. To do so, we have added a new drop-down level and thus modified the main.cpp file.



```
// LIGHT
Light* lightNode1 = new Light();
model.setTranslation(0.0f, 2.0f, 2.0f);
model.scale(0.2f, 0.2f, 0.2f);
lightNode1->model = model;
lightNode1->Id = vec3(0.f, 0.f, 1.f);
light_list.push_back(lightNode1);
```

In the case we want to render a node with the phong shader it's initialized in the following way:

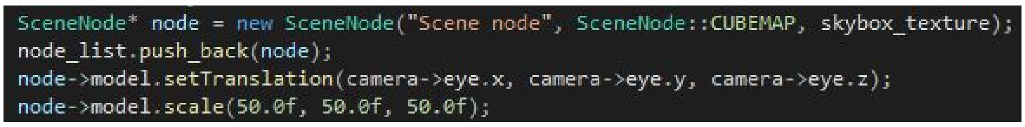


```
case OBJECT:
    material->shader = Shader::Get("data/shaders/basic.vs", "data/shaders/phong.fs");
    this->material = material;
    break;
```

View *Annex 1.2 Phong* to see its renderization.

SkyBox

In this section we have to render a skybox, i.e, render an environment from a cubemap. To do that we have created a new node:

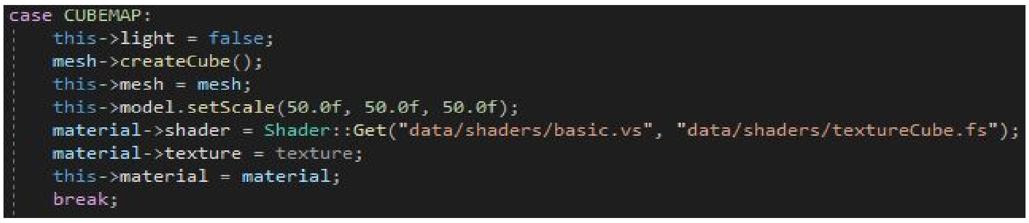


```
SceneNode* node = new SceneNode("Scene node", SceneNode::CUBEMAP, skybox_texture);
node_list.push_back(node);
node->model.setTranslation(camera->eye.x, camera->eye.y, camera->eye.z);
node->model.scale(50.0f, 50.0f, 50.0f);
```

In order to match the skybox's view to the camera view, we set a translation in the model of our skybox's node that are the camera coordinates. This way when we move, we see the respective side of the cubemap and we cannot exit

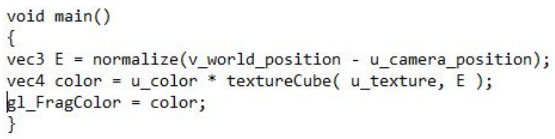
[1] Notice that in this shader we have unified all possible combinations: texture + phong, texture, phong.

nor go through it. In SceneNode we have implemented a switch that depending on the object that we want to create uses the suitable shader, texture and functions:



```
case CUBEMAP:
    this->light = false;
    mesh->createCube();
    this->mesh = mesh;
    this->model.setScale(50.0f, 50.0f, 50.0f);
    material->shader = Shader::Get("data/shaders/basic.vs", "data/shaders/textureCube.fs");
    material->texture = texture;
    this->material = material;
    break;
```

In the textureCube shader we consider the view vector to assign the textels to the correct pixel by means of textureCube() function.

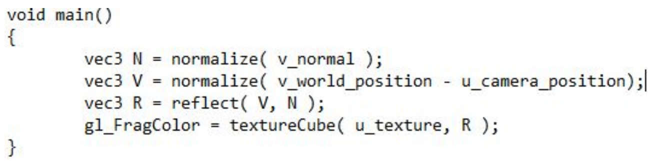


```
void main()
{
vec3 E = normalize(v_world_position - u_camera_position);
vec4 color = u_color * textureCube( u_texture, E );
gl_FragColor = color;
}
```

View *Annex 1.3 Reflective material and SkyBox* to see its renderization.

Reflective Material

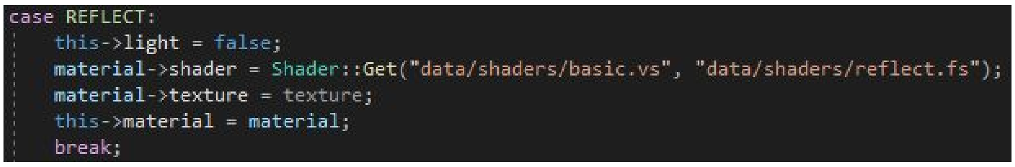
In this section we have to render the mesh as a mirror reflecting the cubemap environment,i.e, implement a new shader where only the cubemap is considered as the light emisor.



```
void main()
{
        vec3 N = normalize( v_normal );
        vec3 V = normalize( v_world_position - u_camera_position);
        vec3 R = reflect( V, N );
        gl_FragColor = textureCube( u_texture, R );
}
```

Since the object is a mirror it only has specular reflection and thus we only consider the view vector and the normal in each pixel to assign the correct textel.

For the case of reflective object the sceneNode uses the following presets:



```
case REFLECT:
    this->light = false;
    material->shader = Shader::Get("data/shaders/basic.vs", "data/shaders/reflect.fs");
    material->texture = texture;
    this->material = material;
    break;
```

View *Annex 1.3 Reflective material and SkyBox* to see its renderization.

ImGui

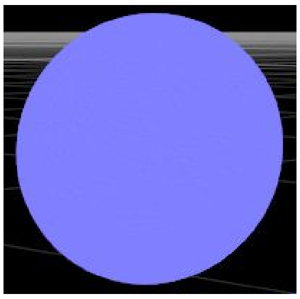
In ImGui we can find 4 main sections:

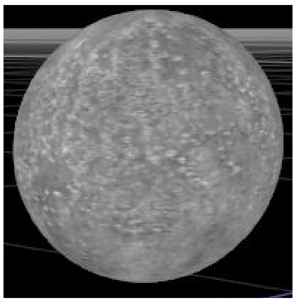
- **Global**: Here you find a dropdown where you can change the environment of the scene.
- **Camera**: In this section you find the type of camera you want, and also the controls to modify the position, fov, near and far of the camera.
- **Entities**: This section displays the different nodes and their characteristics. Depending on the type of entity we find unique options
    - Scene node (Cubemap): There is an enable/disable option that will hide the cubemap
    - Reflective node: There is an enable/disable option and the model modification option
    - Object node: There is an enable/disable option, the model modification option, the material modification (you find examples or a custom material) and the enable/disable light option.
    - Basic node: This is an extension of the object node but with the possibility of texture changes.
- **Light**: There is an enable/disable option, a show option to hide the mesh but still have light, model modification, the light reach distance modification option and light properties modification option.

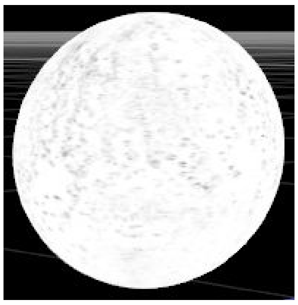
View *Annex 1.4 ImGui* to see its renderization.

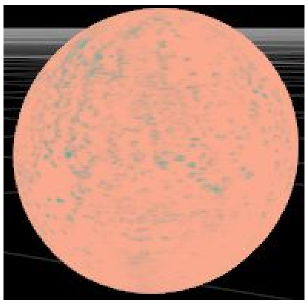
Annex 1: Images

Annex 1.1: Texture Shader


Normal texture

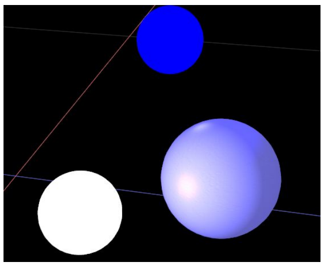

Roughness texture

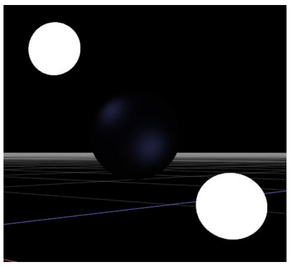

Metalness texture

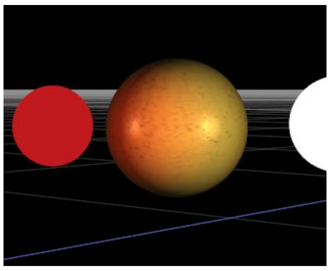

Color texture

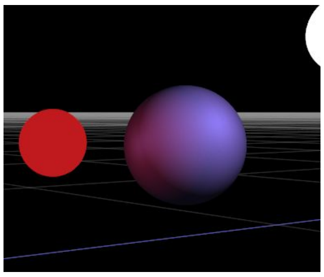
Annex 1.2: Phong Shader

With our Phong Shader we are able to combine different object materials and textures to see the illumination behavior:

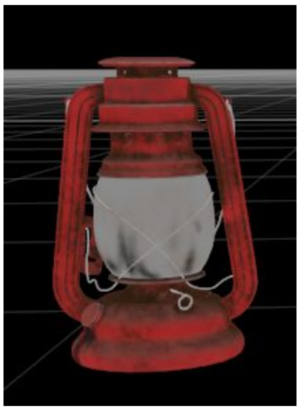

Generic material and normal texture

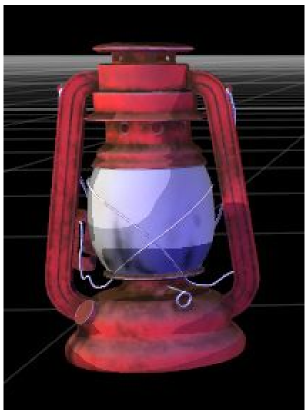

Blackrubber material and normal texture

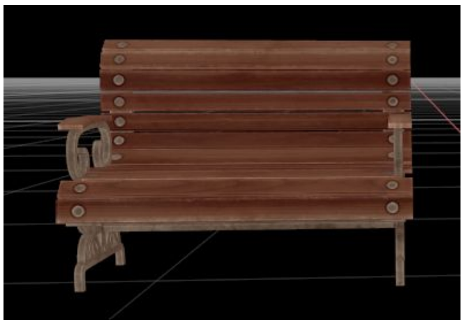

Gold material and metalness texture

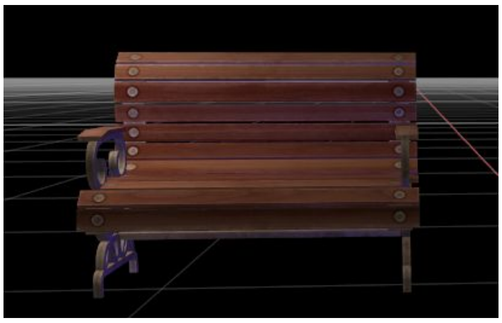

Pearl material and normal texture

Other meshes:


Lintern mesh and its texture without considering light.

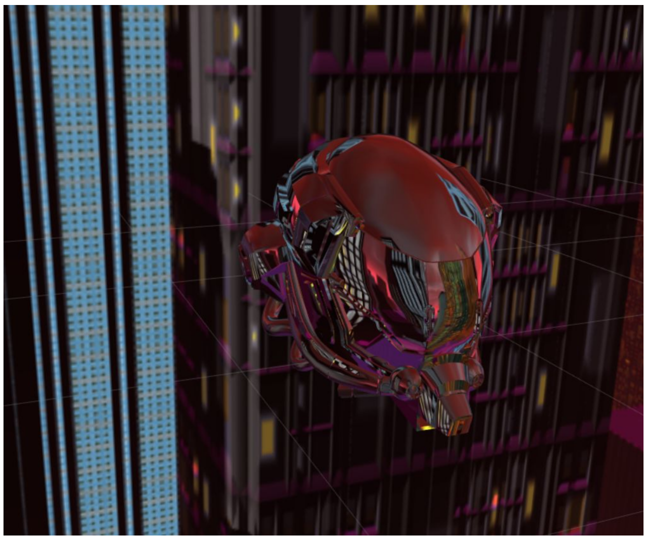

Lintern mesh and its texture considering light.


Bench mesh and its texture without considering light.


Bench mesh and its texture considering light.

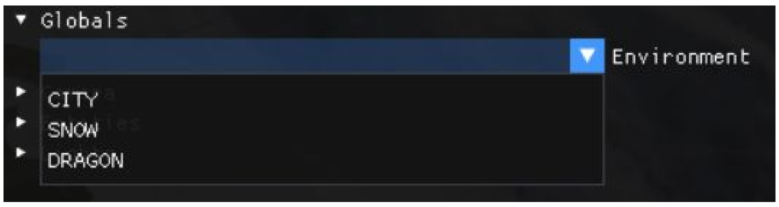
Annex 1.3: Reflective material and Skybox

Helmet with reflective material and snow Skybox.

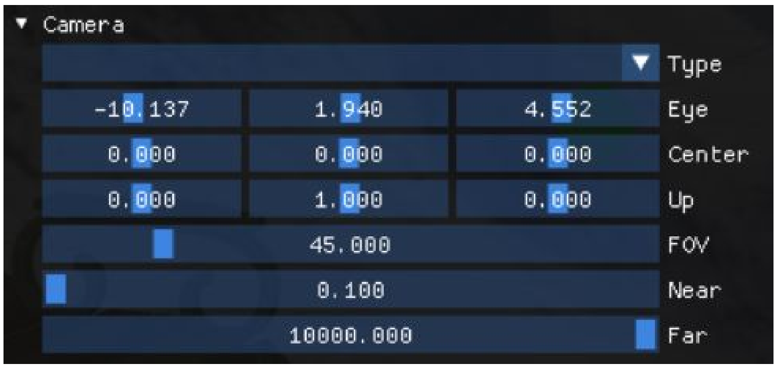

Helmet with reflective material and city Skybox.

Helmet with reflective material and dragon Skybox.

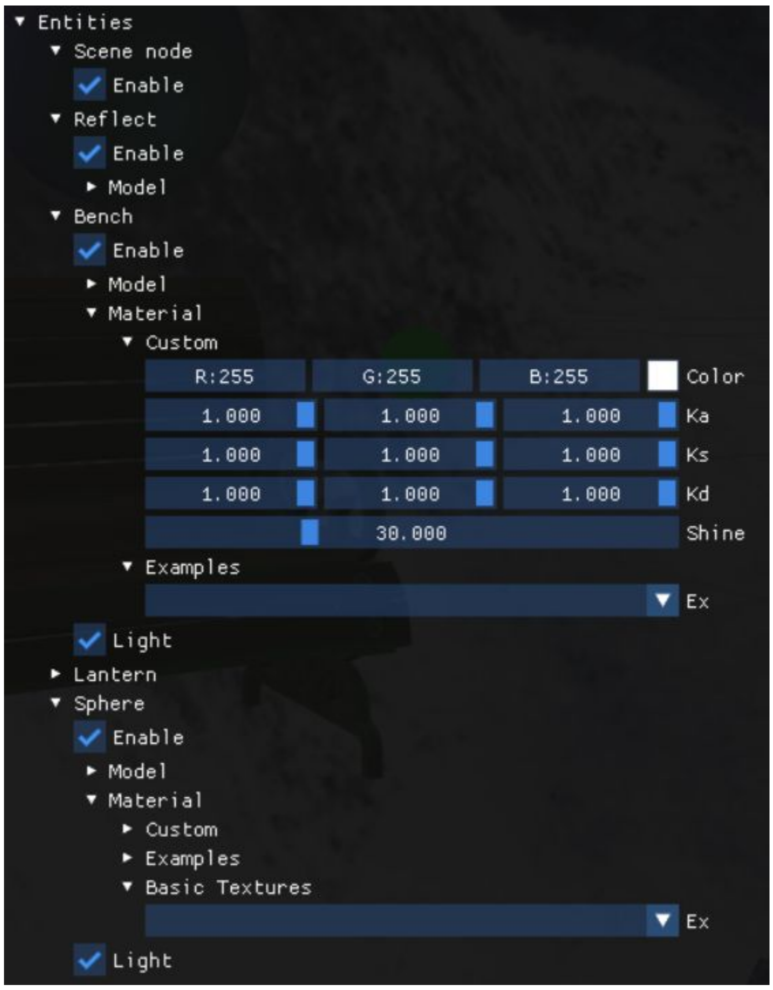
Annex 1.4: ImGui



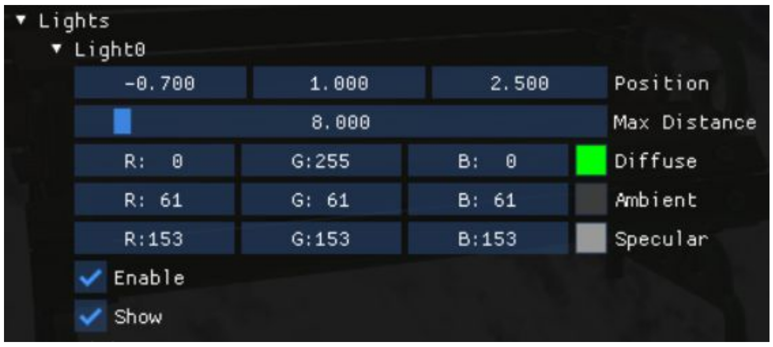
```
▼ Globals
                                    Environment
  CITY
  SNOW
  DRAGON
```



```
▼ Camera
                                         Type
  -10.137      1.940       4.552         Eye
    0.000      0.000       0.000         Center
    0.000      1.000       0.000         Up
              45.000                     FOV
               0.100                     Near
           10000.000                     Far
```



```
▼ Entities
  ▼ Scene node
    ✓ Enable
  ▼ Reflect
    ✓ Enable
    ► Model
  ▼ Bench
    ✓ Enable
    ► Model
    ▼ Material
      ▼ Custom
        R:255     G:255     B:255     Color
        1.000     1.000     1.000     Ka
        1.000     1.000     1.000     Ks
        1.000     1.000     1.000     Kd
              30.000                  Shine
      ▼ Examples
                                  ▼   Ex
    ✓ Light
  ► Lantern
  ▼ Sphere
    ✓ Enable
    ► Model
    ▼ Material
      ► Custom
      ► Examples
      ▼ Basic Textures
                                  ▼   Ex
    ✓ Light
```



```
▼ Lights
  ▼ Light0
    -0.700    1.000     2.500     Position
              8.000               Max Distance
    R:  0     G:255     B:  0     Diffuse
    R: 61     G: 61     B: 61     Ambient
    R:153     G:153     B:153     Specular
    ✓ Enable
    ✓ Show
```