Mireia Espuga 206319
Aina Vendrell 205975
Anna Domènech 204798

ADVANCED VISUALIZATION
LAB 2: PBR & HDR Rendering

The main purpose of this lab is to improve our light reflection model using Physically Based Rendering to compute the direct lighting and add the reflections by means of image based lighting. In order to achieve this, we have pre-defined two types of objects:

- CUBEMAP: Used to render the environments. This type of object uses the textureCube.fs shader.
- OBJECT: Used to render the different objects. Explanation below.

Before explaining how we have implemented the new shader, let us show the structs:

```
struct SceneVectors{
        vec3 V;
        vec3 L;
        vec3 N;
        vec3 H;
        vec3 R;
        float NoL;
        float NoV;
        float NoH;
        float LoH;
} sceneVectors;
```

This struct is used to save all vectors information

```
struct MatProps{
        float ambient_occlusion;
        float opacity;
        float metalness;
        float roughness;
        vec4 emissive;
        vec4 color;
} matProps;
```

This struct is used to set the material properties.

IBL and PBR shader

Our shader sigue the following steps to render the scene:
First, it sets the material properties of the voxel to render,i.e, fill MatProps' struct. Therefore, uses **getPixelColor()** function to compute IBL and PBR. When we use getPixelColor(), first we compute the specular F0 and the diffuse color using the metalness to do linear interpolation:

	cDiff	**f0**
Conductors (metalness = 1)	vec3(0.0)	color
Dielectrics (metalness = 0)	color	vec3(0.04)

```
//specular F0 (conductors -> base color, dielectrics -> vec3(0.04))
vec3 f0 = (1.0 - matProps.metalness) * vec3(0.04) +  matProps.metalness *  matProps.color.rgb ;

//diffuse cDiff (conductors -> vec3(0.0), dielectrics -> base color)
vec3 diffuseColor = (1.0 - matProps.metalness) *  matProps.color.rgb;
```

Then, the computeDirect() and computeIBL() functions are called. computeDirect() is the result of computing the following formula:

$$f_{pfacet}(\mathbf{l},\mathbf{v}) = \frac{F(\mathbf{l},\mathbf{h})G(\mathbf{l},\mathbf{v},\mathbf{h})D(\mathbf{h})}{4(\mathbf{n}\cdot\mathbf{l})(\mathbf{n}\cdot\mathbf{v})}$$

Where F(l,h), G(l,v,h) and D(h) are:

$$F(l,n) \simeq F_0 + (1-F_0)(1-(l\cdot n))^5$$

$$G(\mathbf{l},\mathbf{v},\mathbf{h}) = G_1(\mathbf{l})G_1(\mathbf{v}) \quad \text{where} \quad G_1(\mathbf{v}) = \frac{(\mathbf{n}\cdot\mathbf{v})}{(\mathbf{n}\cdot\mathbf{v})(1-k)+k} \quad \text{and} \quad k = \frac{(roughness+1)^2}{8}$$

$$D(\mathbf{m}) = \frac{\alpha^2}{\pi((\mathbf{n}\cdot\mathbf{m})^2(\alpha^2-1)+1)^2}$$

Once the direct lighting is computed the computeIBL() function is used to compute the indirect lighting. In order to correctly apply IBL, we have used HDR[1] environments to capture the illumination of the scene and be able to use the environment map as a radiance map. Note that depending on the roughness of the texture the material reflectivity will change, as more reflective better we will see the reflection of the environment. Note

[1] Includes 5 levels of blurred versions of the original environment, each one corresponds to a given level of reflectivity.

that the specular term is computed using 2D LUT texture to extract its value by taking into account the roughness and the N dot L values at a given point.
Once we have IBL and PBR we apply tone mapping and before sending it to the screen gamma correction is applied to texture color channels.

Extra maps

We've added the following interactive (Imgui) extra maps:

- ➢ **Normal texture:** Used to give details to the surface of our 3d meshes through the *perturbNormal(N, V, uv, normal_map)* function which gives us the detailed normal (vec3 normal_map) we'll use for our computations.
- ➢ **Emissive texture:** Used to add light emitted from the material itself as a source of light. In this lab we don't consider points lights or glow effects, so we treat the emission as a color.
- ➢ **Opacity texture**: Used to make transparent areas of the material. From the texture we extract a greyscale float(0 to 1) which we will assign to the alpha channel of the color texture. Note that adding this channel does not affect the light computations.
- ➢ **Ambient occlusion texture:** Used to simulate mesh self occlusions with ambient light. From the texture map we extract a grayscale float (0 to 1) which we apply (multiply) to IBL light only.
- ➢ **Displacement texture:** Used to give even more detail to the mesh. From the texture we extract a grayscale float (0 to 1) that contains the height of every pixel mapped. With this value we'll slightly change the position of our vertex to compute V and L vectors.

View *Annex 1.1 Extra maps* 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

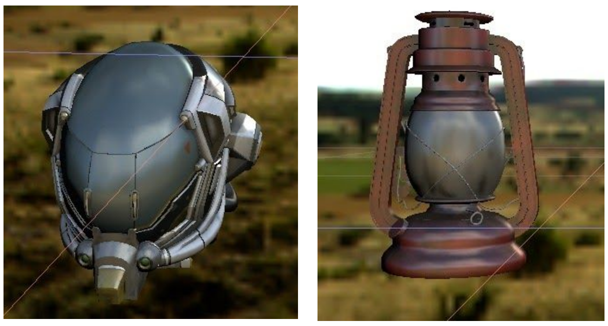
- ➢ **Scene node (Cubemap):** There is an enable/disable option that will hide the cubemap
- ➢ **Object:** There is an enable/disable option and the model modification option. The 3 objects we find in the scene share the following material textures:
    - ▪ RGB selector for color
    - ▪ enable/disable of albedo, ambient occlusion, normals, roughness and metalness textures
    - ▪ slider of roughness and metalness value

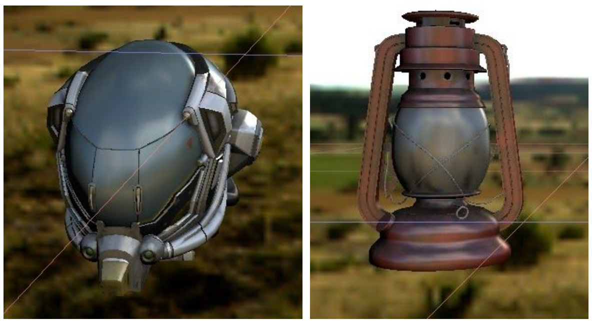
  We have 3 objects with unique options
    - ○ Helmet: This node has an extra enable/disable option for the emissive texture
    - ○ Lantern: This node has an extra enable/disable option for the opacity texture
    - ○ Sphere: This node has an extra enable/disable option for the displacement texture. And since it's an extension of the object node has the possibility of 3 texture changes found in the "Textures" dropdown.

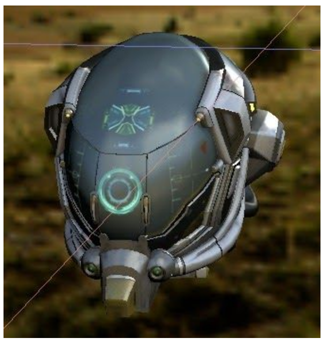
Light: There is an enable/disable option, a show option to hide the mesh but still have light, model modification, the light reach distance modification option and light properties modification option.
View *Annex 1.3 ImGui* to see its renderization.

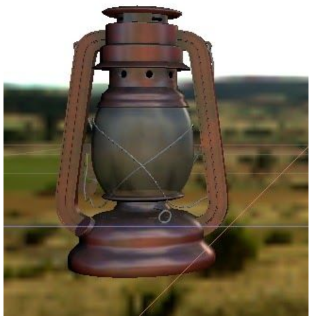
Annex 1: Images

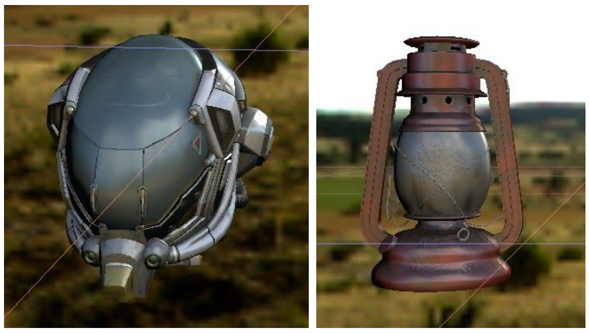
Annex 1.1 : Texture maps (+Extra maps)

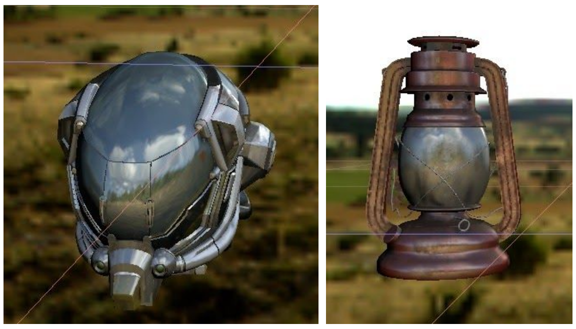

albedo texture

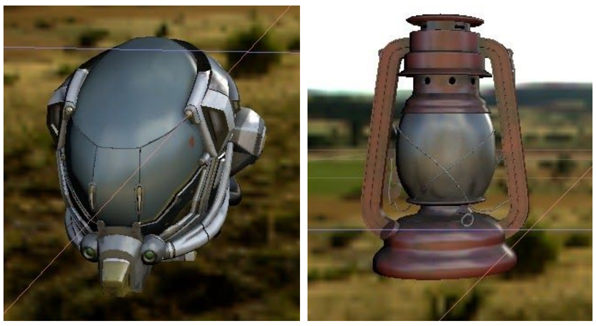

albedo + ambient occlusion textures

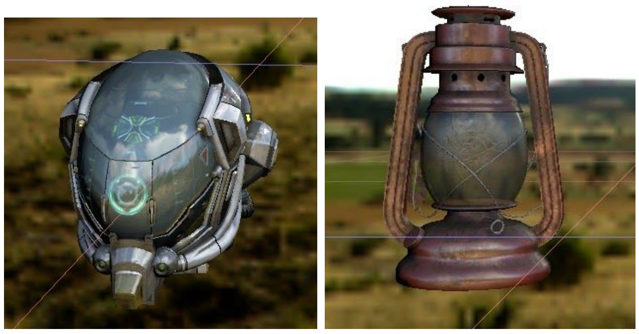

albedo + emissive textures

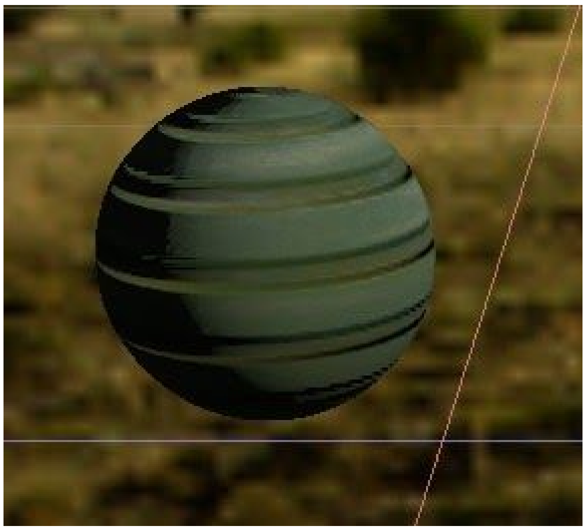

albedo + opacity textures


albedo + normal textures

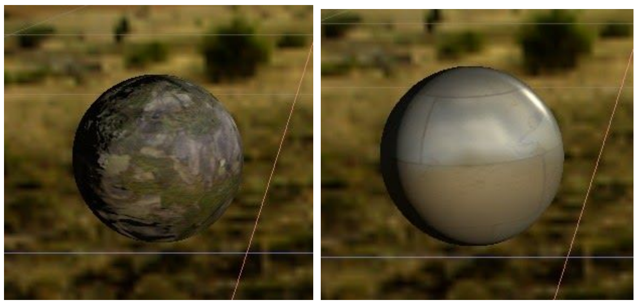

albedo + roughness textures


albedo + metalness textures


all textures

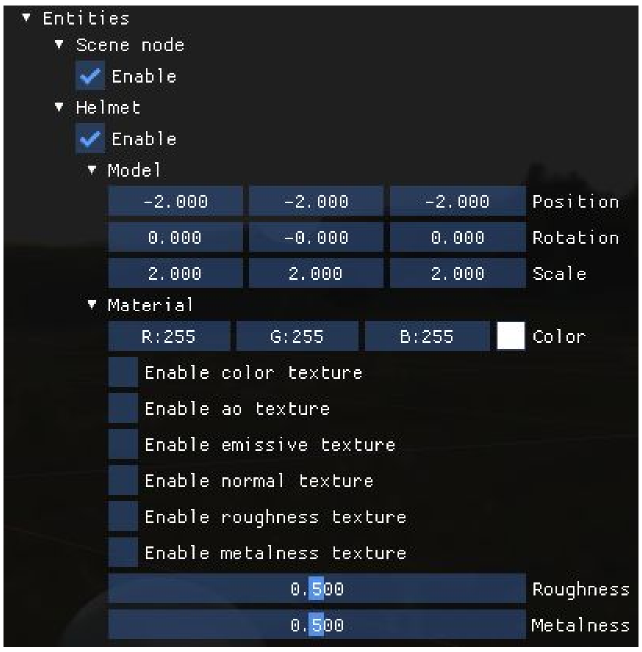


albedo + metalness + roughness + ao + normal + displacement textures

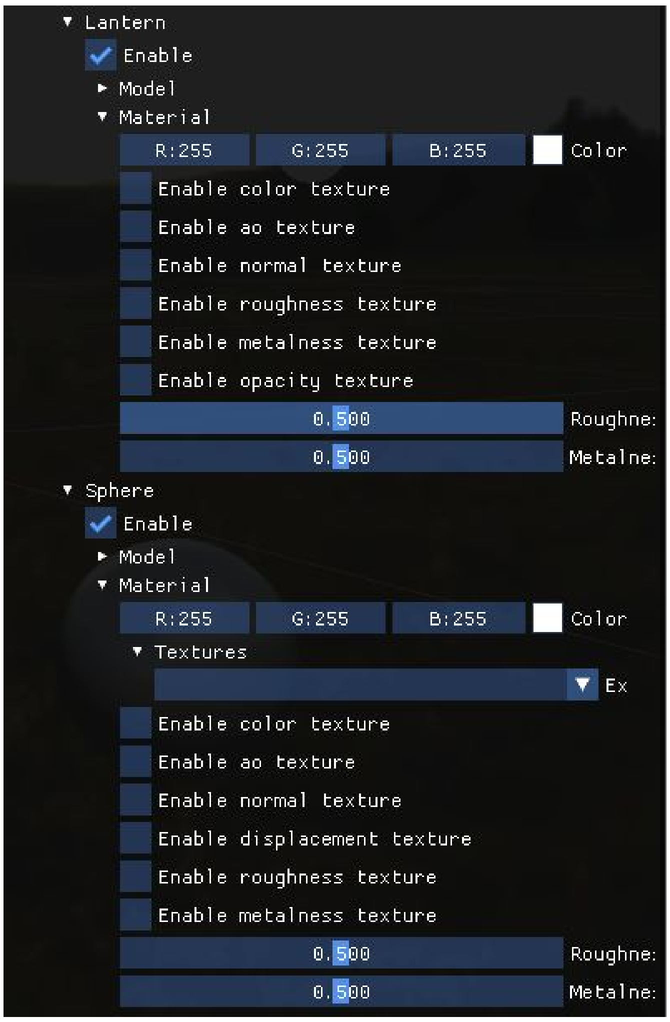


all textures with other types of presets

Annex 1.2 : Imgui



```
▾ Entities
  ▾ Scene node
    ✓ Enable
  ▾ Helmet
    ✓ Enable
    ▾ Model
      -2.000   -2.000   -2.000   Position
      0.000    -0.000   0.000    Rotation
      2.000    2.000    2.000    Scale
    ▾ Material
      R:255  G:255  B:255  Color
      Enable color texture
      Enable ao texture
      Enable emissive texture
      Enable normal texture
      Enable roughness texture
      Enable metalness texture
      0.500  Roughness
      0.500  Metalness
```



```
▾ Lantern
  ✓ Enable
  ▸ Model
  ▾ Material
    R:255  G:255  B:255  Color
    Enable color texture
    Enable ao texture
    Enable normal texture
    Enable roughness texture
    Enable metalness texture
    Enable opacity texture
    0.500  Roughne:
    0.500  Metalne:
▾ Sphere
  ✓ Enable
  ▸ Model
  ▾ Material
    R:255  G:255  B:255  Color
    ▾ Textures
      ▼ Ex
    Enable color texture
    Enable ao texture
    Enable normal texture
    Enable displacement texture
    Enable roughness texture
    Enable metalness texture
    0.500  Roughne:
    0.500  Metalne:
```