Computer Graphics Fundamentals and Applications
Rendering Assignment

María José Rueda Montes.
Computational Colour and Spectral Imaging Master.

INTRODUCTION

Rendering is an important part of computer graphics, which consists of transforming the description of a scene, i.e. the code, into a three-dimensional image. The aim of this project is to explore and understand the generation of realistic images through the rendering of a scene, that in this case, will be a table whose legs are made of wood and plate made of glass, also including the camera, and lighting. As a rendering method *pbrt* will be used, a *physically based* method based on the *ray-tracing* algorithm.

METHODOLOGY

1. *Modeling and Geometry*

The first step in creating a realistic object is to model the shape it will have in order to be able to generate the geometry. For this purpose, blender is used, as is showed in the Fig. 1.

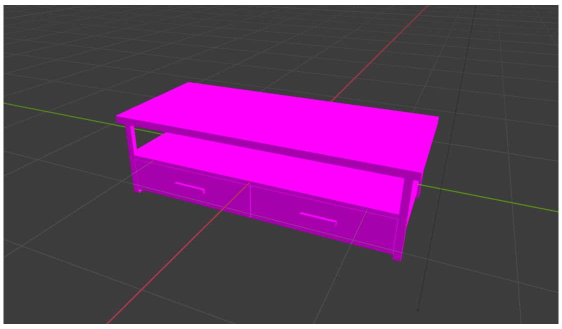
Figure 1. Coffee table modeled in Blender (Free3D, 2019)

It is important to separate the object into as many parts as different materials are to be applied, which on this occasion will be two, glass and wood.

After modelling the object with the same blender program, the ".obj" file containing the geometry data is generated, which describe the scene geometry. To convert the ".obj" file into ".pbrt", pbrt-v3 has a converter from the Wavefront OBJ format, using the following path:

```
$ obj2pbrt scene.obj scene.pbrt
```

In this way, the two pbrt files corresponding to the geometry of the plate and the table support are obtained.

2. *Option block state*

In the option block, scene-wide global options may be set, such as camera type and position, sampler definition, and information about the scene to be generated.

The *Camera* defines a new system of coordinates with its origin at the camera's location. Then it is included pbrtLookAt(), used to transform the camera location and place the object in center of the scene, and which coordinates are: LookAt *eye_x eye_y eye_z look_x look_y look_z up_x up_y up_z*.

```
LookAt -25 25 25  # eye
      0.4 -0.3 0  # look at point
      0 1 0       # up vector
Camera "perspective" "float fov"
[22]
```

In this code, a perspective camera mode is set with a 22 degree field of view.

Continuing with the code, appears the *Sampler*, which generates the samples for the image, time, lens, and Monte Carlo integration.

```
Sampler "sobol" "integer
pixelsamples" [ 64 ]
```

SobolSampler is selected since it is generally one most effective samplers for Monte Carlo integration. In this instance with 64 samples per pixel.

The algorithm that records the radiance reaching the film generated by the interaction of the light with the objects in the scene is implemented by the *Integrator*.

```
Integrator "volpath" "integer
maxdepth" [ 65 ]
```

It is fixed a “volpath” integrator with 65 as maximum length of the light-carrying path sampled.

The implementation of ImageFilm uses the instance *Filter* class to filter sample values to compute final pixel values.

```
PixelFilter "triangle" "float
xwidth" [ 1.000000 ] "float ywidth"
[ 1.000000 ]
```

“xwitdth” correspond with the width of the filter in the x direction, and “ywith” is the width of the filter in the y direction.

Finally, the *Film* is set, that specifies the characteristics of the image and the “image” in input files.

```
Film "image" "string filename"
["C:/Users/gusil/Desktop/Graphics_F
inal/mytable.png"]
      "integer xresolution" [1024]
"integer yresolution" [1024]
```

The “xresolution” and “yresolution” describe the number of pixels in the “x” and “y” axis respectively.

3. *World block state*

This block is delimited by `WorldBegin` and `WorldEnd`, within which it is specified the scene information, such as the light source, materials, textures and the geometry of the objects used in the scene.

The *Light Source* is described by LightSource directive, and this cast the light in the scene which is not associated without any geometry.

```
LightSource "infinite" "string
mapname" ["textures/marmol3.png"]
```

There are several types of lights that pbrt provides, for this scene "infinite" light is used, which represents a light source located at infinity and which directs light in all directions. On the other hand, “mapname” is the environment map used for the infinite area light. The Fig. 2 is the image used like “mapname” for this project.

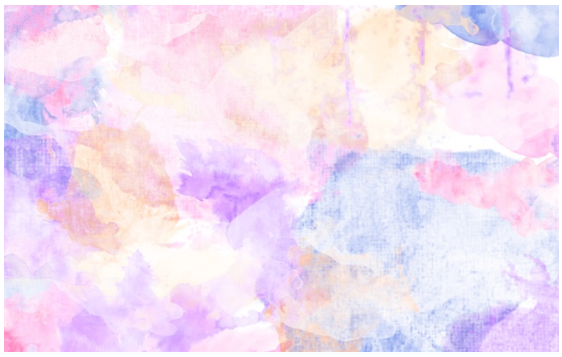
Figure 2. Marble background used in Light Source (FondosMil, 2020)

In the last part of the code that is provided, the *Materials* are implemented. Materials indicate the light scattering properties of surfaces in the scene. Depending on the material, different parameters are included. To include your own materials, first define the texture and include the image corresponding to the texture that will be applied to the object. Then, the values that indicate how the light interacts with the material that is defined are specified. And the last step is to apply the material and texture to the geometry of the object that had already been obtained after the modeling. For this reason, the pbrt file is included that contains the geometry of the structure to which will be apply the material described.

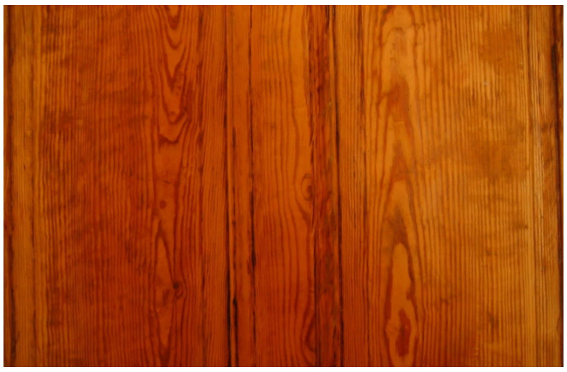
Figure 3. Wood texture for the bottom part of the table (Bitterli, 2020)

```
#Wood material
Texture "mywood" "spectrum"
"imagemap"
     "string filename"
["textures/wood5.tga"]
     "bool trilinear" ["true"]
MakeNamedMaterial "mywood"
     "string type" ["substrate"]
     "rgb Ks" [0.040000 0.040000
0.040000]
     "texture Kd" ["mywood"]
     "float uroughness" [0.100000]
     "float vroughness" [0.100000]
     "bool remaproughness"
["false"]
AttributeBegin
NamedMaterial "mywood"
     Scale 5.0 5.0 5.0
     Include "Bottompart.pbrt"
AttributeEnd
```

In the code, for the bottom of the table, the material "substrate" with a wood texture ("mywood") is applied. Since “bool trilinear” is true, perform trilinear interpolation when looking up pixel values. Regarding the material parameters, “Ks” indicates the coefficient of specular reflection; “Kd” is the coefficient of diffuse reflection; “uroughness” the roughness of the surface in the u direction; “vroughness” the roughness of the surface in the v direction; and “remaproughness”, that if it is true, not like in this image, the roughness values are remapped to microfacet distribution function parameter values that range from near-perfect-specular at 0 to very rough at 1. Moreover, “Scale” included before the geometry of the material, allows to expand the object 5.0 in the x, y, z axis.

For the plate part of the table, a glass material is defined, in which the “index” and “remaproughness” parameters are utilized. “Index” defines the refraction inside the object, and usually is fixed in 1.5, whereas “remaproughness” has the same effect as above, being false for this material.

RENDERING RESULT

The end result is a fairly realistic and physically correct image, as it is shown in the Fig. 4. The materials of the object reflect the light as it would behave in the real world, obtaining a virtual result very similar to the same object in real life after only a few minutes of rendering.

REFERENCES

[1] Pharr, M., Jakob, W., & Humphreys, G. (2016). *Physically based rendering: From theory to implementation*. Morgan Kaufmann.

[2] (2015-2019). Coffee Table Free 3D models download. *Free3D*. https://free3d.com/3d-model/coffee-table-71234.html

[3] (2020). 10949. *FondosMil*. https://fondosmil.com/fondo/10949.png

[4] Benedikt Bitterli (2020). Modern Hall by NewSee2I035. *Benedikt Bitterli.* https://benedikt-bitterli.me/resources/

[5] Benedikt Bitterli (2020). Utah Teapot (full) by Benedikt Bitterli .*Benedikt Bitterli.* https://benedikt-bitterli.me/resources/

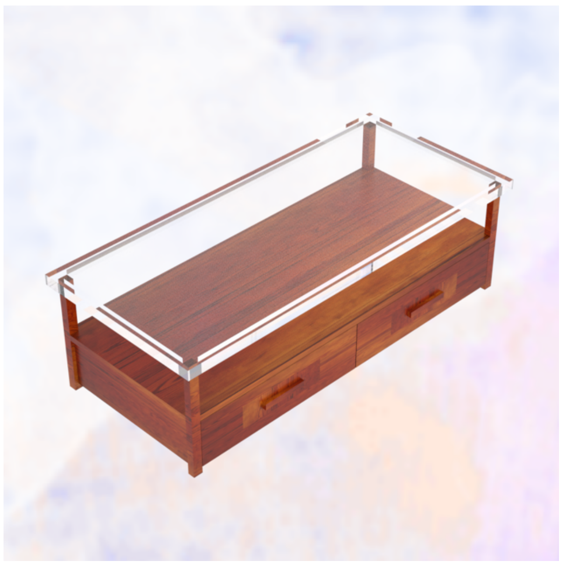

Figure 4. Final rendering result using pbrt