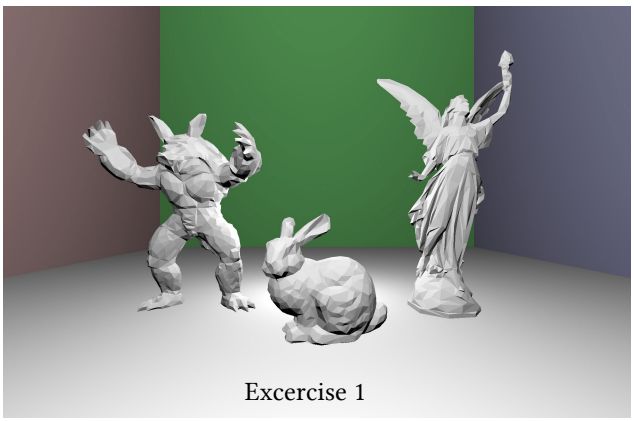
Università
della
Svizzera
italiana

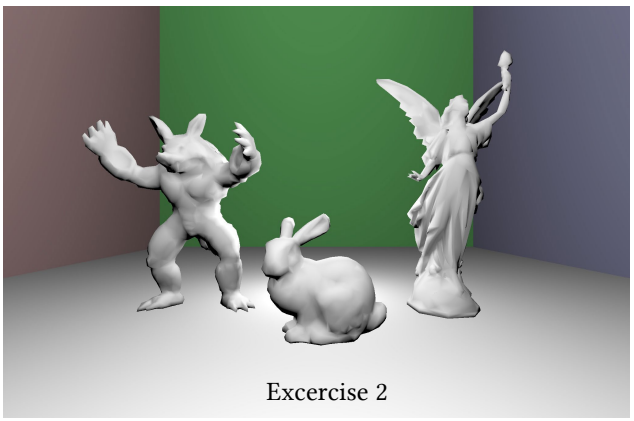
Faculty
of Informatics

Computer Graphics

Assignment 3: Meshes



Excercise 1



Excercise 2

Let's render some iconic models from computer graphics! In this assignment, you are asked to extend the raytracer so that it can render meshes. This will take the raytracer to the next level, as after successfully completing the assignment, you will be able to obtain meshes from the internet or model them using any modeling tool and include them in your scene.

Updated framework

The current template is a solution for the previous assignment. Whether you want to start directly with the provided template or continue from your solution to the previous assignment is up to you. In the latter case, you can copy the updates to the framework. There are no other updates to the framework; implement the required functionalities the way you want. **You are not allowed to use any external libraries or snippets of code downloaded from the internet**.

Meshes

You are given three meshes that you can use to demonstrate the new raytracer capabilities. They come from Stanford repository[1] but are decimated and adjusted to make your tasks slightly easier. Each mesh has two versions, with and without per-vertex normal vectors. The meshes with normal vectors are for smooth shading (see the next exercise). You are also invited to look for other meshes and try to render them. To successfully complete the exercises, you must demonstrate the results using at least one mesh. The arrangement of objects is not important as long as your final image demonstrates the implemented functionalities. The meshes are located at point $(0, 0, 0)$. Therefore, you will need to translate the objects away from the camera location.
Note that once you introduce meshes, your raytracer will become slow. For reference, the above images were rendered in 2048×1536 resolution, and it took approximately one hour to render each of them with our current raytracer. In the future, we will study how to make it faster. For now, we recommend reducing the resolution of the output image, especially for testing and placing objects.

[1] http://graphics.stanford.edu/data/3Dscanrep/

Exercise 1 [10 points]

Extend your raytracer to render meshes from OBJ[2] files. To this end, you must introduce a new object to the raytracer (triangle/mesh), implement a ray-triangle intersection, and implement a mesh loader. OBJ files are very versatile and can store a lot of different information. You only need to implement the functionality that you need for this assignment.

Exercise 2 [5 points]

Implement smooth normal interpolation, i.e., Phong shading, using barycentric coordinates and demonstrate the effect.

Bonus Exercise 3 [5 points]

Download an object from the internet or create your own using a modeling tool, such as Blender[3], and add it to your scene. Make sure you include your mesh file to your submission.

Submission

Your submission **must** contain one ZIP-file with:

- a readme file or a PDF document file with information about which exercises you solved, the authors of the solutions, and the explanation of encountered problems, if any,
- an image file, *result.ppm*, containing the final image you could render,
- a directory named *code* containing all the source code used to generate the image.

The source code, upon compilation, should generate the image identical to the submitted *result.ppm* file. Your code should compile by calling *g++ main.cpp*. The ZIP file name must be of a form *surname1_surname2.zip* for a team of two, and *surname.zip* for a single submission. Only one person from the team should submit the solution.

Solutions must be submitted via iCorsi by the indicated there deadline.

[2] https://en.wikipedia.org/wiki/Wavefront_.obj_file
[3] https://www.blender.org