Università
della
Svizzera
italiana

Faculty
of Informatics

Computer Graphics

Bonus assignment: Spatial Data Structures

Let's render some high-quality meshes! The scene above has over 3 million triangles. In order to render it, you need to have a good acceleration structure, such as an axis-aligned bounding box hierarchy or a kd-tree. In this assignment, you will implement one and render the above scene. The basic version of the assignment considers implementing only simple bounding boxes. When implementing the acceleration structure you can limit it to handle only triangles. The planes can be rendered separately.

Scene

With the assignment, you are given three meshes coming from the Stanford repository[1]. The scene should contain:

- four planes visible in the image above, the same as in previous assignments,
- Armadillo model translated by $T_a = (-4.0, -3.0, 10)^T$,
- Bunny model translated by $T_b = (0.0, -3.0, 8.0)^T$,
- Lucy model translated by $T_l = (4.0, -3.0, 10)^T$,
- three point lights at positions $l_1 = (0, 26, 5)^T$, $l_2 = (0, 1, 12)^T$, $l_3 = (0, 5, 1)^T$, also the same as in the previous assignments.

The meshes do not have normal vectors defined, and smooth shading is not required. All the elements of the scene can be rendered with default diffuse material, as in the image above. When reproducing the scene, only the geometrical information is important. The final colors and shades won't be a subject of grading, but the ray tracer must render correct shadows. The meshes are provided in two versions with different triangle counts.

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

Version A (basic) [3 points]

Implement a simple bounding box for each mesh and demonstrate the obtained speed-up in a short report provided with the submission. In this task, you can render only the smaller version of the meshes. For large meshes, you will need a data structure that enables raytracing in sub-linear time with respect to the number of triangles (see version B).

Version B (full) [15 points]

Implement a spatial data structure that enables rendering meshes with millions of triangles. Demonstrate the capabilities of your raytracer using the scene described above with large meshes and report your timings. You can use different sets of meshes to demonstrate the timings for different numbers of triangles. Use the provided meshes to construct a plot that relates the number of triangles in the scene with the execution time of your raytracer and observe a sub-linear relation.
Apart from the book that was mentioned as a reference during the first lecture, you can use the following chapter of the online PBR-book for a detailed description of two acceleration structures described during the lecture: `https://pbr-book.org/3ed-2018/Primitives_and_Intersection_Acceleration#`

Small speed competition [up to 3 points]

Every submission that renders the image presented above with large meshes will be considered for a speed competition. For this, we request that your code renders the image without any modifications using a single thread in resolution 2048×1536 pixels. You should not use any external libraries, and you should be the author of all the code. Solutions not fulfilling the above requirements won't be considered for the competition. The colors in the image are not important, but the geometrical information must be accurate to verify the correctness of your code. Also, your ray tracer should render correct shadows. We will compile your solution using a standard C++ compiler with -O3 optimization using the following command `g++ main.cpp --std=c++11 -O3`. We will establish a time limit for your applications to render the scene. For reference, rendering the above image on a MacBook Pro 2022 took approximately 10 seconds. Note that your raytracer needs to be capable of rendering only this type of scene, e.g., planes, meshes, diffuse material, and shadows. Other functionalities can be disabled to increase efficiency.
The authors of the fastest execution will receive 3 extra points. For the second result, we will give 2 extra points, and for the third, we will give one point.

Submission

Your submission **must** contain only one ZIP-file containing:

- a short report in PDF format containing the authors of the solution, description of what was implemented, timings, plots, and description of any problems and solutions.
- one `result.ppm` file generated by the code you submit,
- a folder named `code` which contains all the files required to run your code.

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.