Rasterization-based rendering

Rasterizaton-based rendering

- Rasterization-based rendering is quite old method extensively used and researched still today (and will stay with us for some time as we will see).

- Techniques used to produce an image with this method were developed 1960-1980*

- Rasterization is method used by GPUs to produce images. Rasterization-based techniques for image generation are so fundametnal that they are deeply integrated within the GPU hardware arhitecture.

- GPUs changed a lot from their beginnings but foundamental methods they implement for generating images haven't changed much. GPUs changed in efficiency and flexibility with some extensions. The pipeline and methods for generating images fundamentally stayed the same.

Rasterization → another solution to visibility problem

- When we introduced rendering, we divided it in two main tasks: visibility and shading.

- In raytracing-based rendering, we have seen that visibility can be divided into camera rays visibility (which objects are visible from camera) and secondary rays visibility (which surfaces are visible to each other during light transport in shading phase).

- Rasterization is another method for computing visibility. This method is extremely efficient for finding objects that are visible from camera and not very good when it comes to visibility needed for light transport.

  – Rasterization and ray-tracing should produce the same images until the point the shading is applied

- Calculation of visibiliy using rasterizer as it was in raytracing-based rendering, relies on geometrical techniques

Practical note: using rasterization-based renderer

- As discussed, rasterization-based rendering technique is deeply integrated in GPU hardware. Opposed to raytracing-based rendering which can be often found purely implemented on CPU.

- Rasterization-based rendering can also be completely implemented on CPU. For learning purposes this is useful to understand all the aspects of it. There are some special use-cases which also benefit from CPU implementation of rasterization based-rendering*.

- Almost all professional software which uses rasterization-based rendering is using GPU hardware implementation which can be further programmed using graphical APIs such as OpenGL, Vulkan, DirectX, Metal, etc.

- In these lectures, we will have two passes on the topic:
  - One pass over concepts of rasterization-based rendering which should give the idea how can it be implemented.
  - One pass over GPU pipeline and graphical API

* Cases when objects that are rendered are smaller than pixel. Those are advanced topics and are related to point rendering (https://www.cg.tuwien.ac.at/research/publications/2022/SCHUETZ-2022-PCC/) or micropolygon rendering (https://docs.unrealengine.com/5.0/en-US/nanite-virtualized-geometry-in-unreal-engine/ or https://graphics.pixar.com/library/Reyes/)

Concepts of rasterization 1

- Reminder: ray-tracing-based rendering starts from image plane, generates rays which are traced into scene and tested for intersections. This method is call **image centric** and has two loops:
    - Outer loop that iterates over all pixels (and generates ray)
    - Inner loop that iterates over all objects in 3D scene (and tests for intersection)

-

Rasterization on GPU 1