Acceleration Structures for Ray Tracing

Uniform Grids

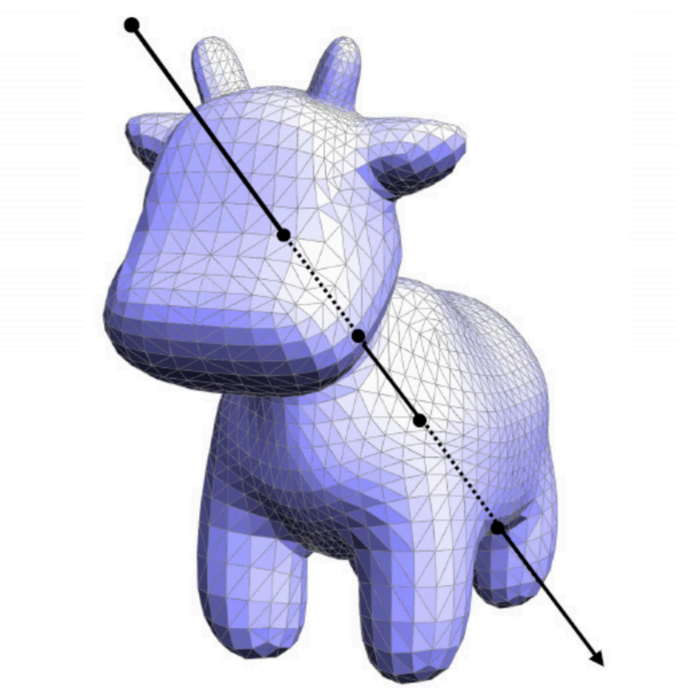
Yuehao Wang; Apr 21, 2021

Recall the routine of ray-tracing

- Generate rays from the camera
- **Detect the interaction between rays and objects**
- Sample reflected rays until meeting light sources
- Tracing the radiance back to the camera according to the rendering equation.

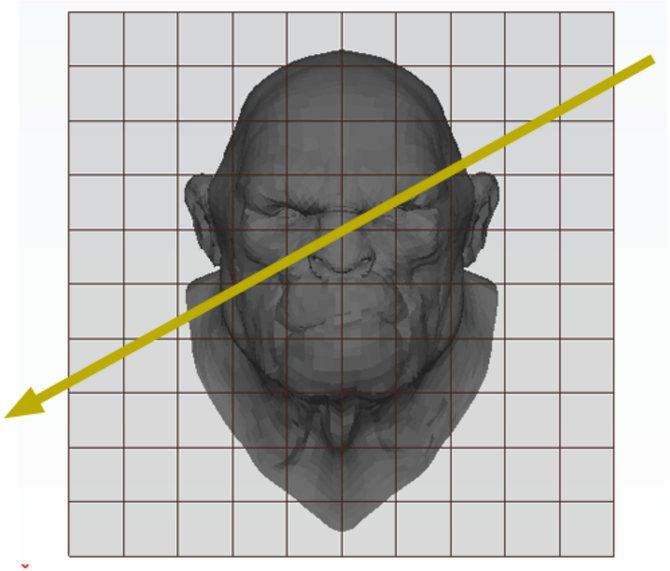
Ray-mesh intersection

- In general, meshes are composed of triangles.
- Basically, detect the interaction between each triangle and the ray.



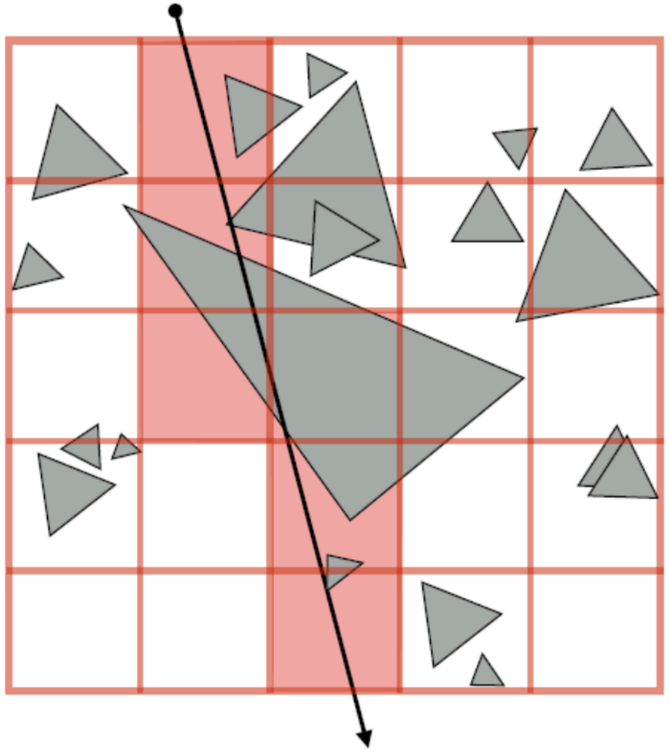
Why acceleration structures

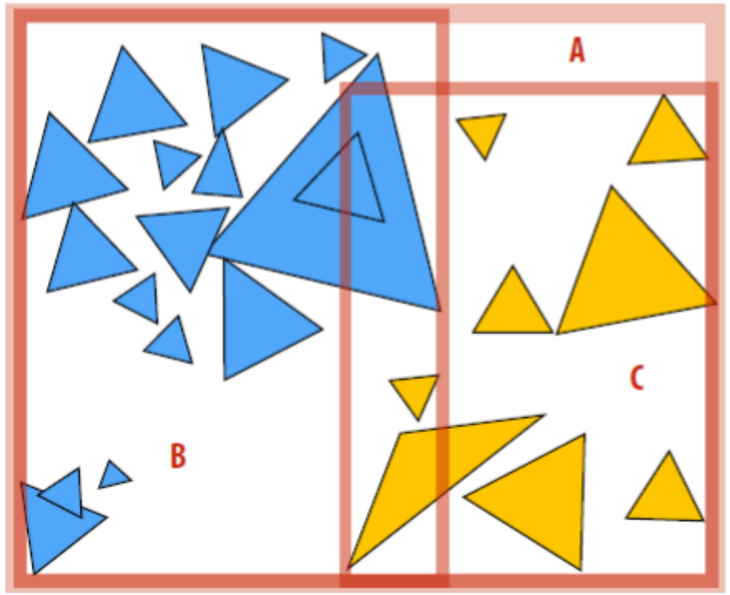
- A mesh may have too many triangles.
- However, the ray may just go through few triangles.
- Acceleration structure is used to left out some triangles.

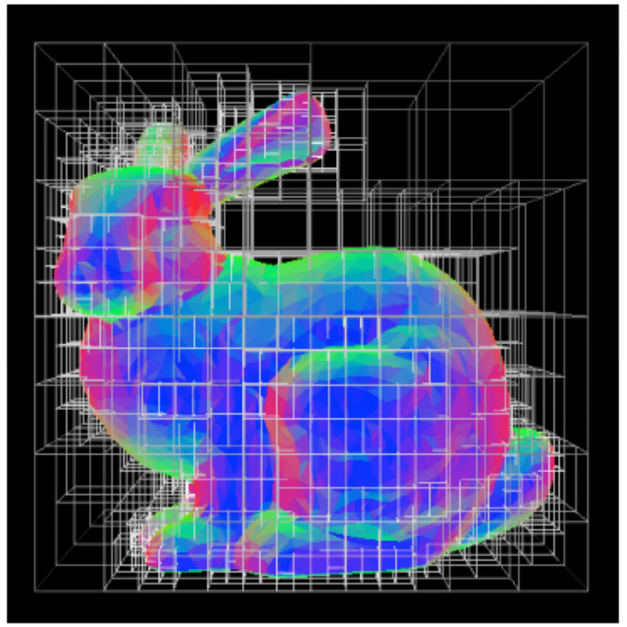


Acceleration structures

- Uniform grids
- Bounding volume hierarchies
- KD-tree





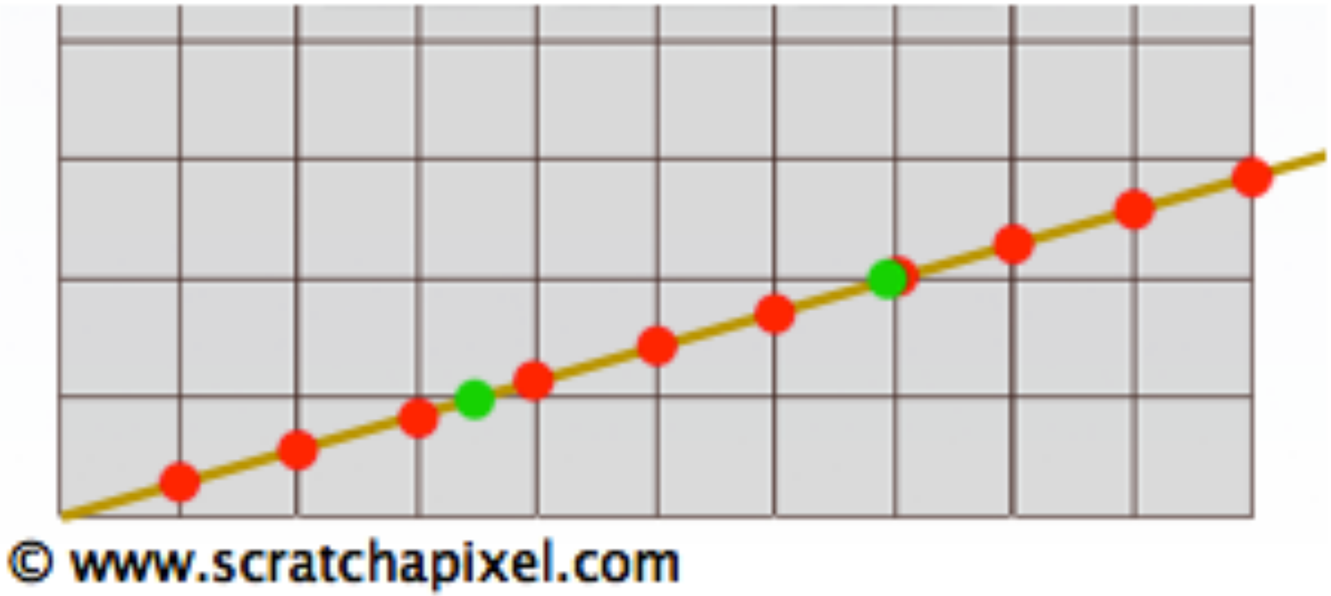


Uniform Grids

- A very basic structure.
- Divide bounding box to several grids (or cells)
- Each grid has a container of triangles.
- Construction:
  - If a triangle is intersected with a grid, store the triangle in the grid.
- Once the ray goes through a grid, just check the interaction of triangles inside the grid.

Uniform Grids

- Intersection between ray and grids
- Consider a 2D case:
  - Similar to the DDA algorithm
  - Find each pixel of the line



- More details: https://www.scratchapixel.com/lessons/advanced-rendering/introduction-acceleration-structure/grid

Construct the Grid

- Input: triangles and their bounding boxes
- Find the corresponding cell of the bounding box min and bounding box max

$$cell\ size = \frac{grid\ size}{grid\ resolution}\ min\ cell = \frac{triangle\ BBox\ min}{cell\ size}$$

$$max\ cell = \frac{triangle\ BBox\ max}{cell\ size}$$

- Add the triangle to from min cell to max cell

```
001 for (int z = cellMin.z; z <= cellmax.z; ++z)
002     for (int y = cellMin.y; y <= cellmax.y; ++y)
003         for (int x = cellMin.x; x <= cellmax.x; ++x)
004             cell[x][y][z].insert(triangle);
```