USI Rendering Competition

Federico Pallotti & Volodomyr Karpenko

Computer Graphics 2021 - Università della Svizzera Italiana

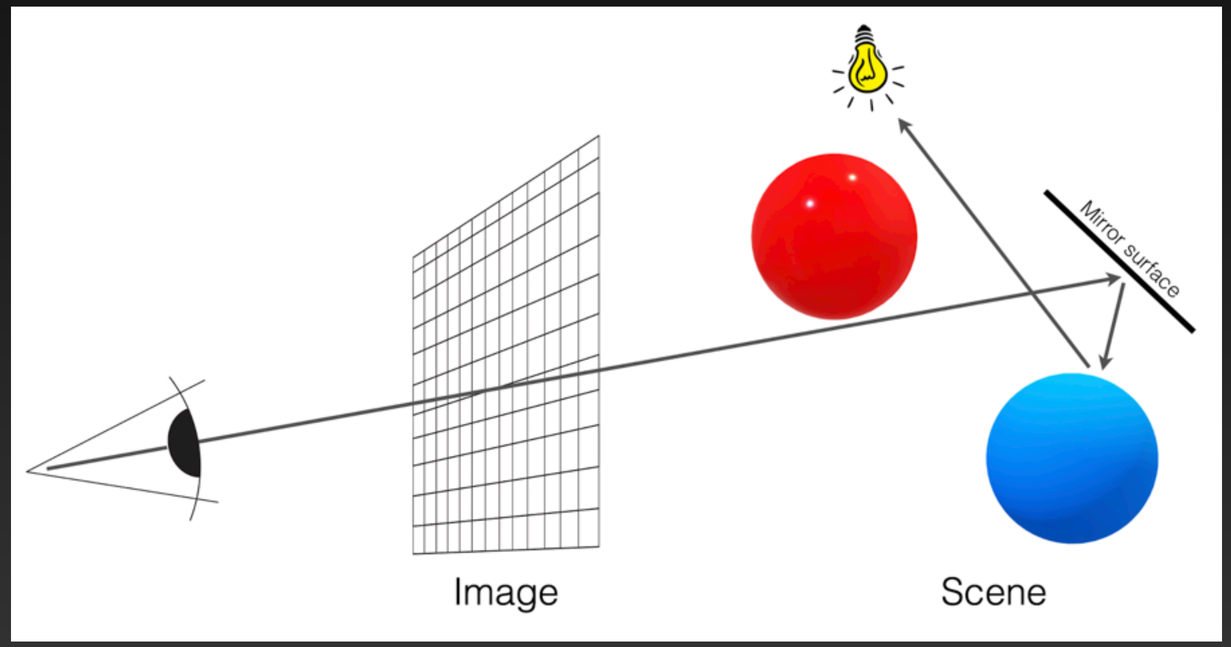
What is a Ray tracer?

Ray tracing

- Used for offline rendering of realistic images and for real time purpose like in video games.
- Based on the actual physical behaviour of the light rays.
- Computationally really expensive.

Basic Ray tracer

- Shoot rays from the camera/eye to the scene
- One ray for each pixel
- Save the color value of the hit object only if the ray “rebounds” on the surface and then hit a light source.
- Assign the color to that pixel



GOALS

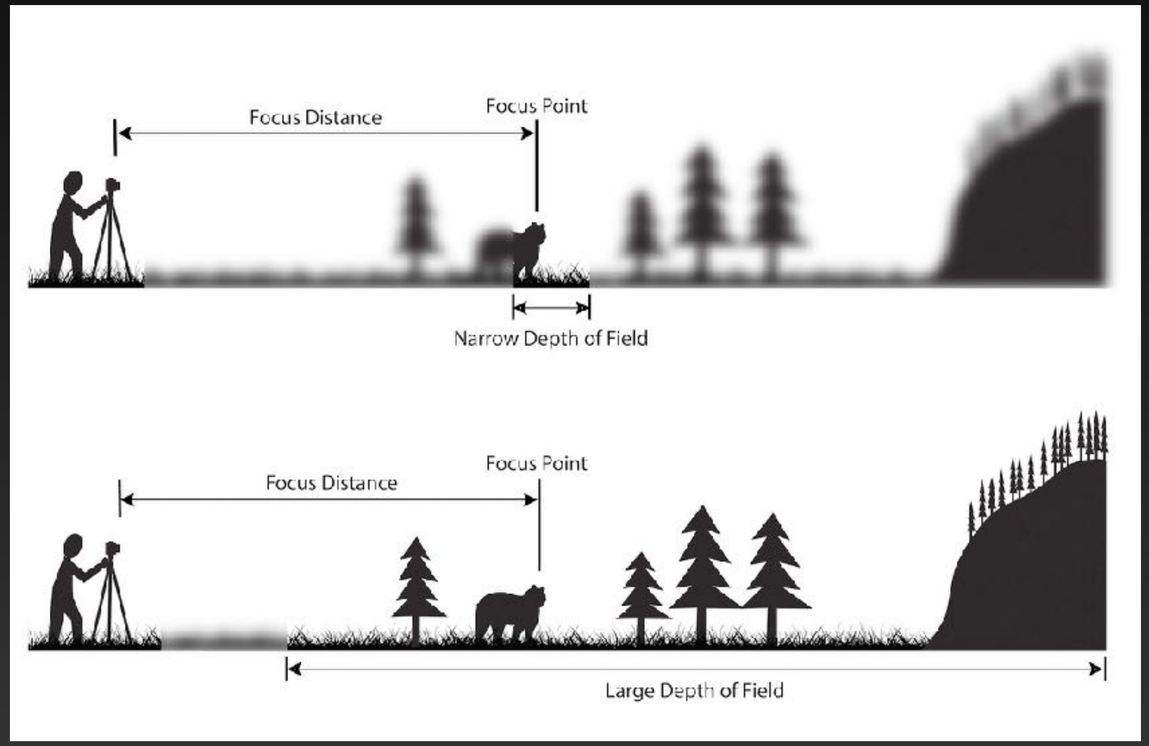
- Depth of Field
- Antialiasing
- Soft Shadows
- Kd Tree for mesh rendering
- Perlin Noise

DOF

What’s behind it?

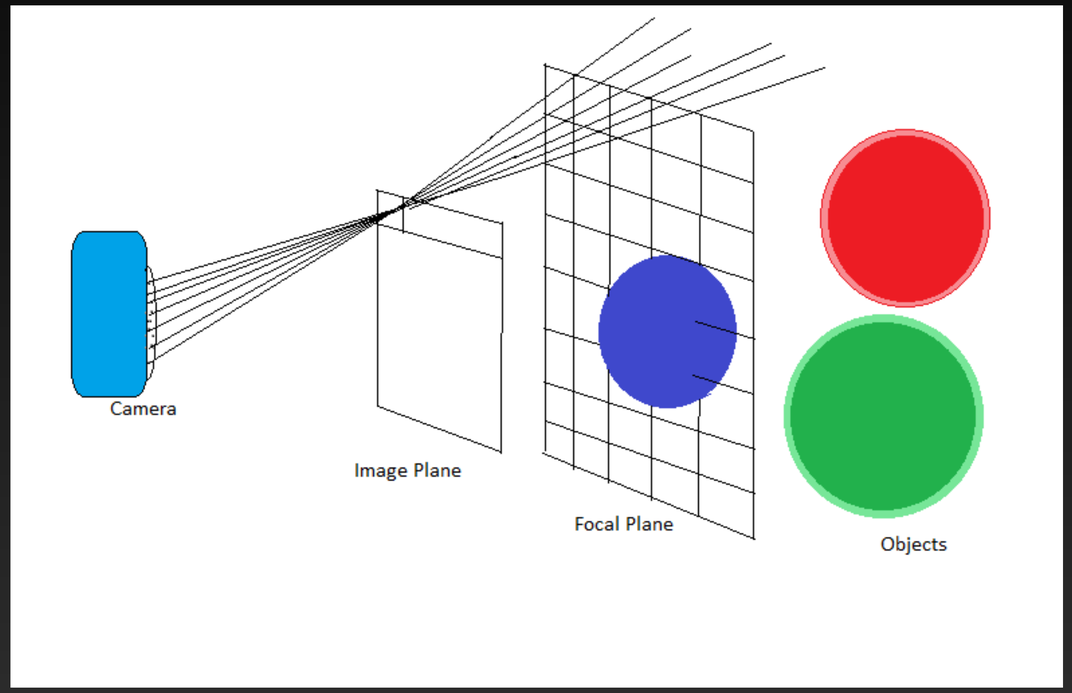
DOF

- Range of distance from the Camera/Eye where objects are on focus.
- Due to the physical behaviour of the lenses used in cameras.

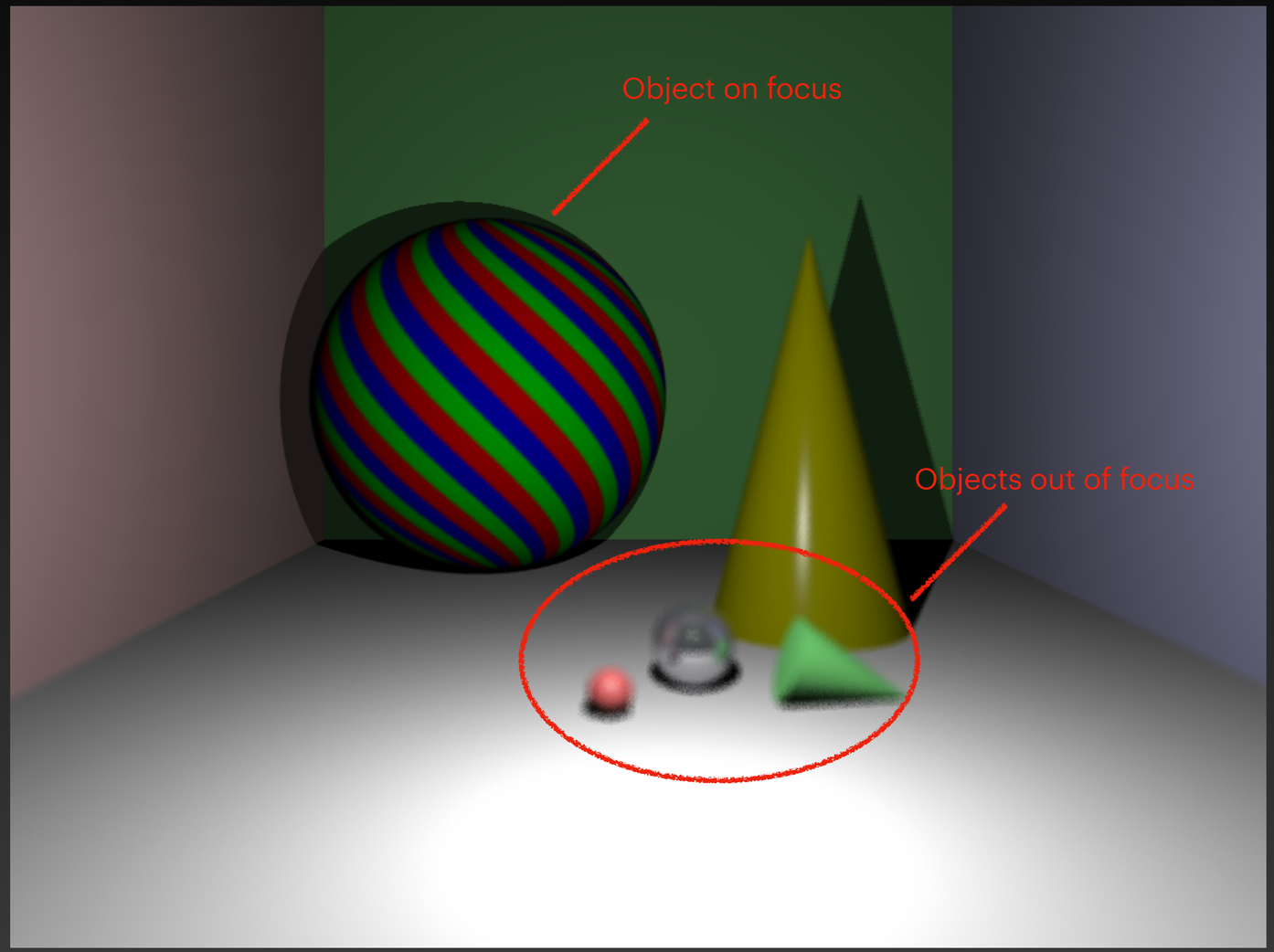


DOF in Ray tracing

- From the POV of a Ray tracer is like shooting more rays from more random small distances from the original Camera/Eye position.
- All the rays converge at a given distance, called **“Focal Distance”**, given as an input parameter to the Ray tracer.

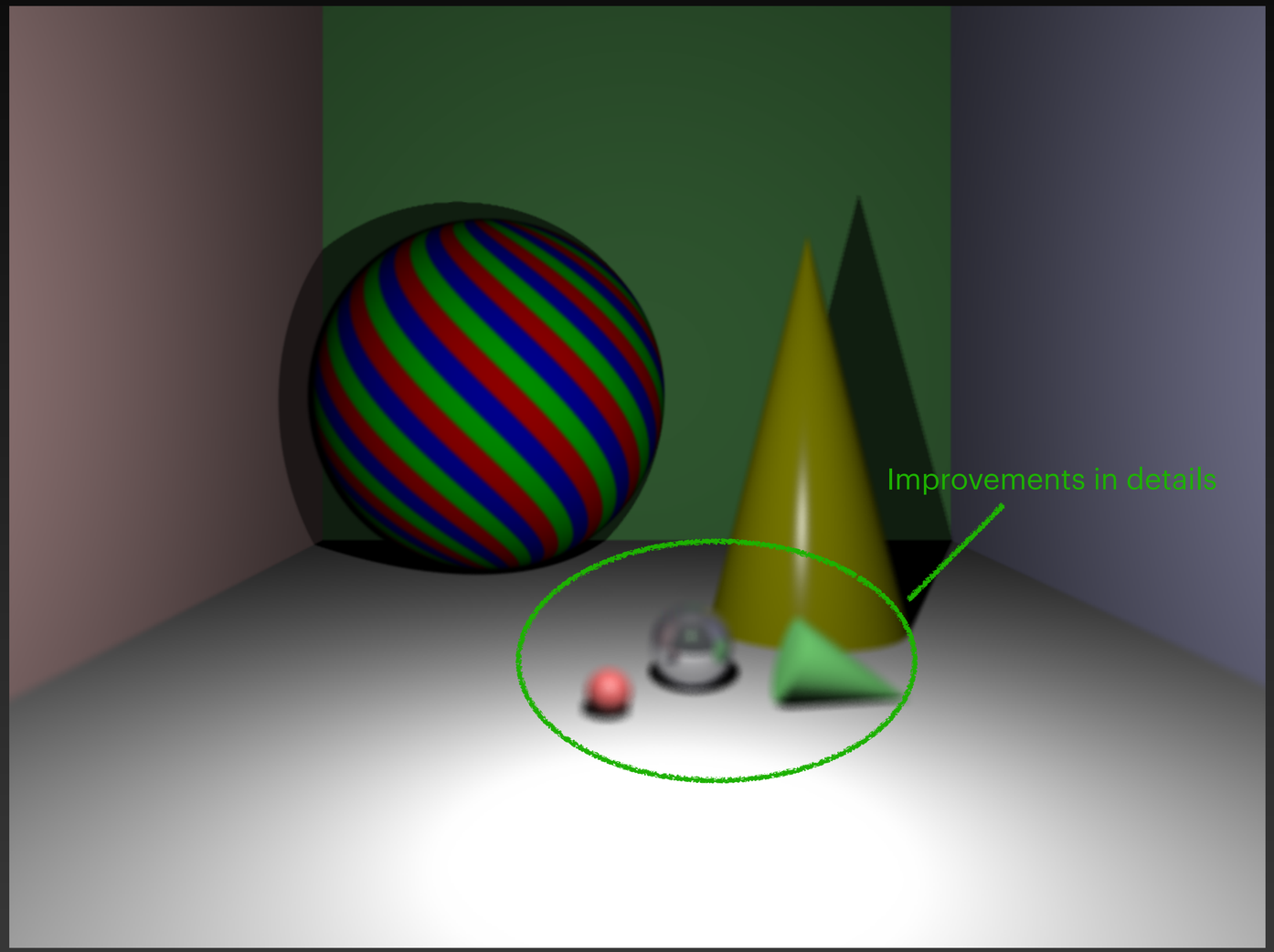


Our results:



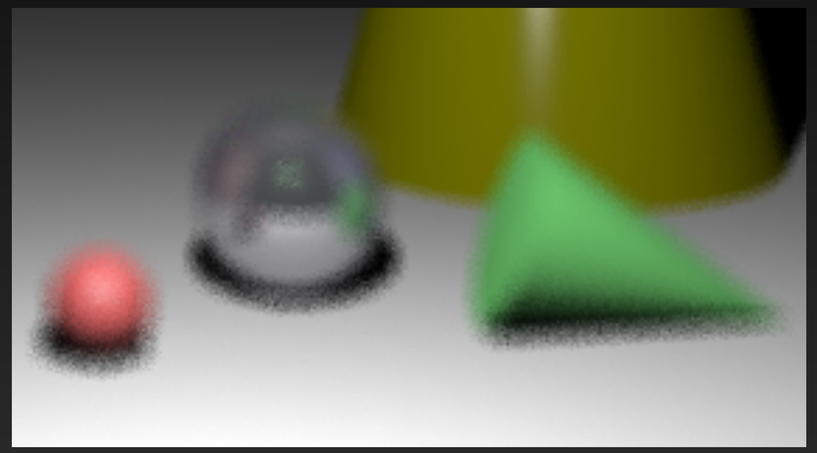
- Focal distance = 30
- Aperture 0.3
- Number of samples = 50

Our results:

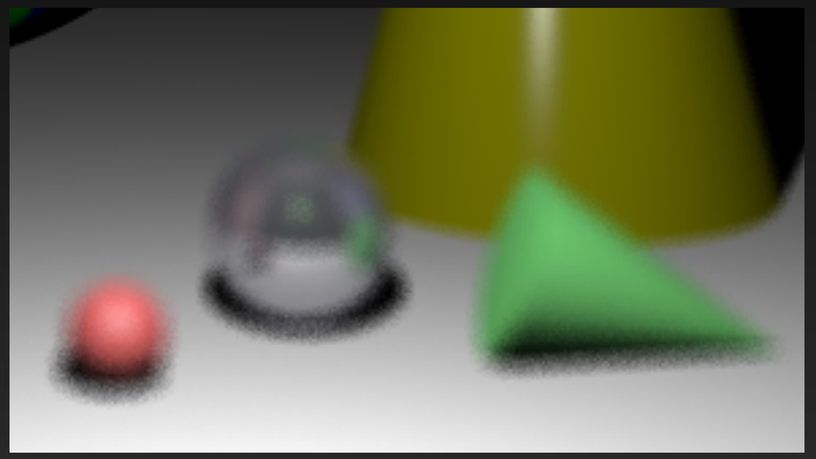


- Focal distance = 30
- Aperture 0.3
- Number of samples = 500

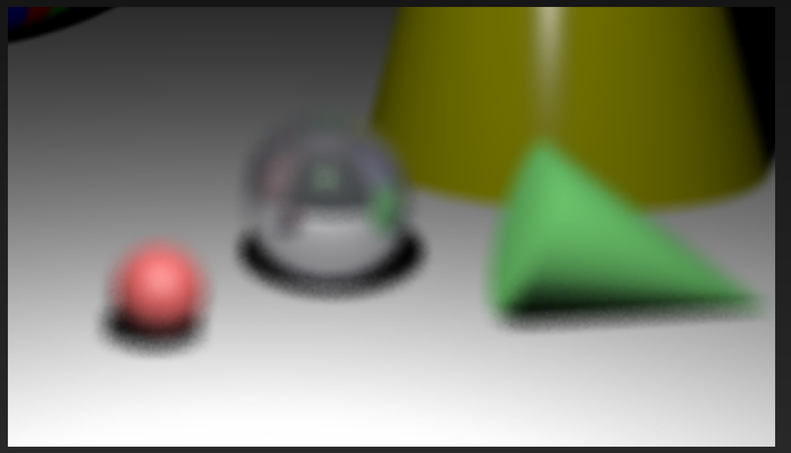
Our results:



Sampling = 50



Sampling = 100



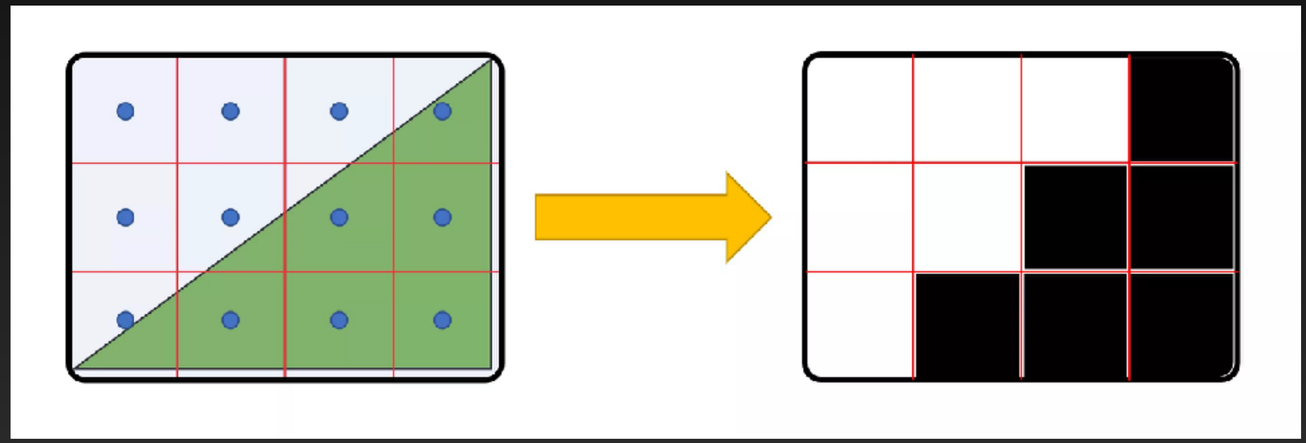
Sampling = 500

Antialiasing

What’s behind it?

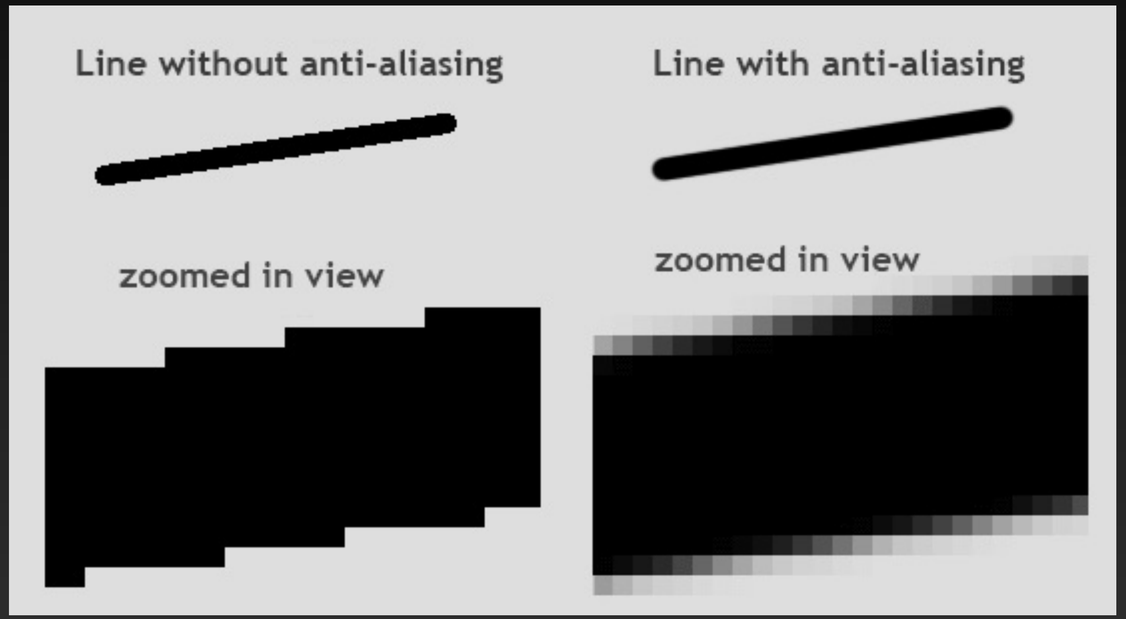
Aliasing

- Sampling approximation of the pixels.
- Borders of objects look noisy



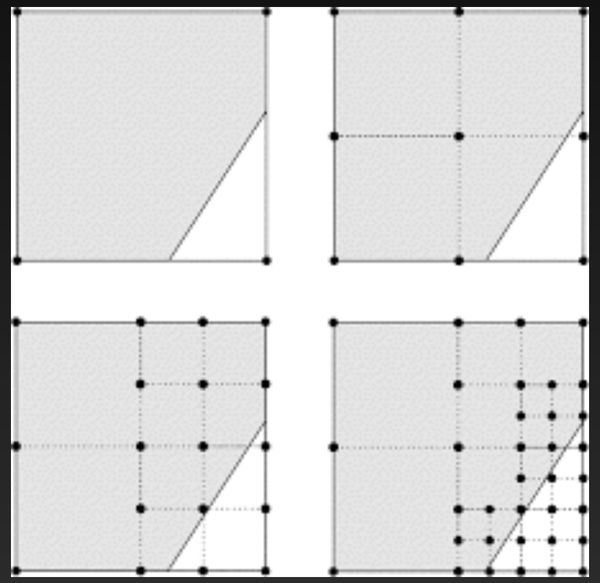
Antialiasing

- The solution for Aliasing in rendering is to assign a color value to a pixel based on an average of values of more than one sampling.
- In this way we “smooth” the color value of one pixel based on the neighbour pixels values.

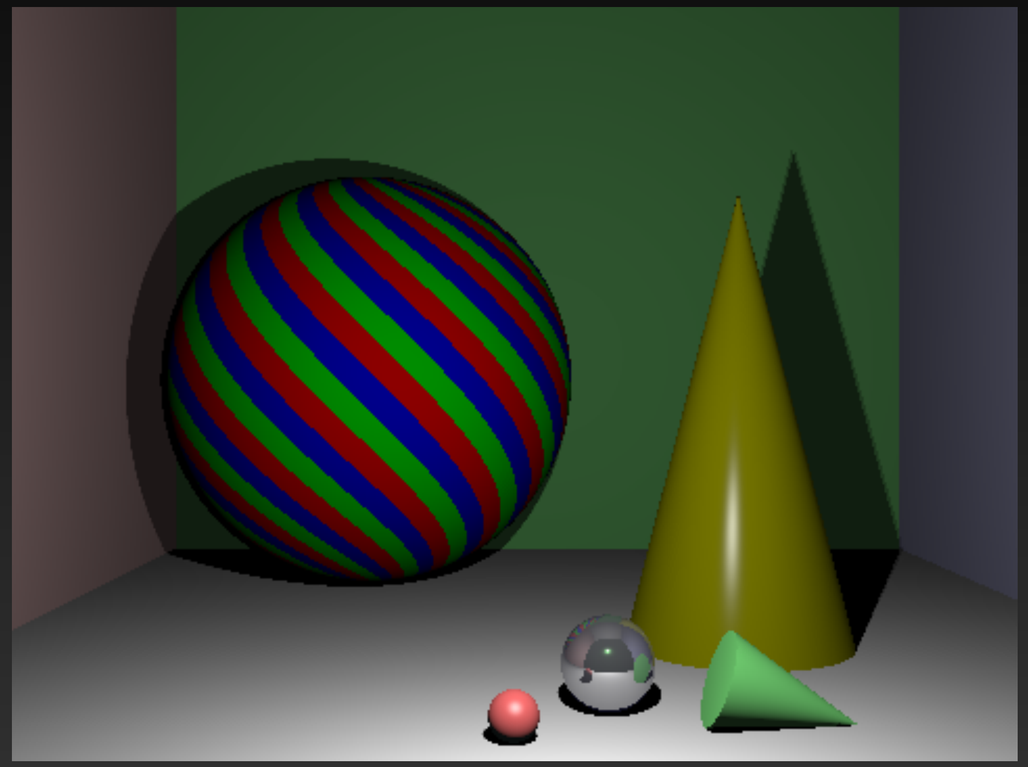


Antialiasing in Ray tracing

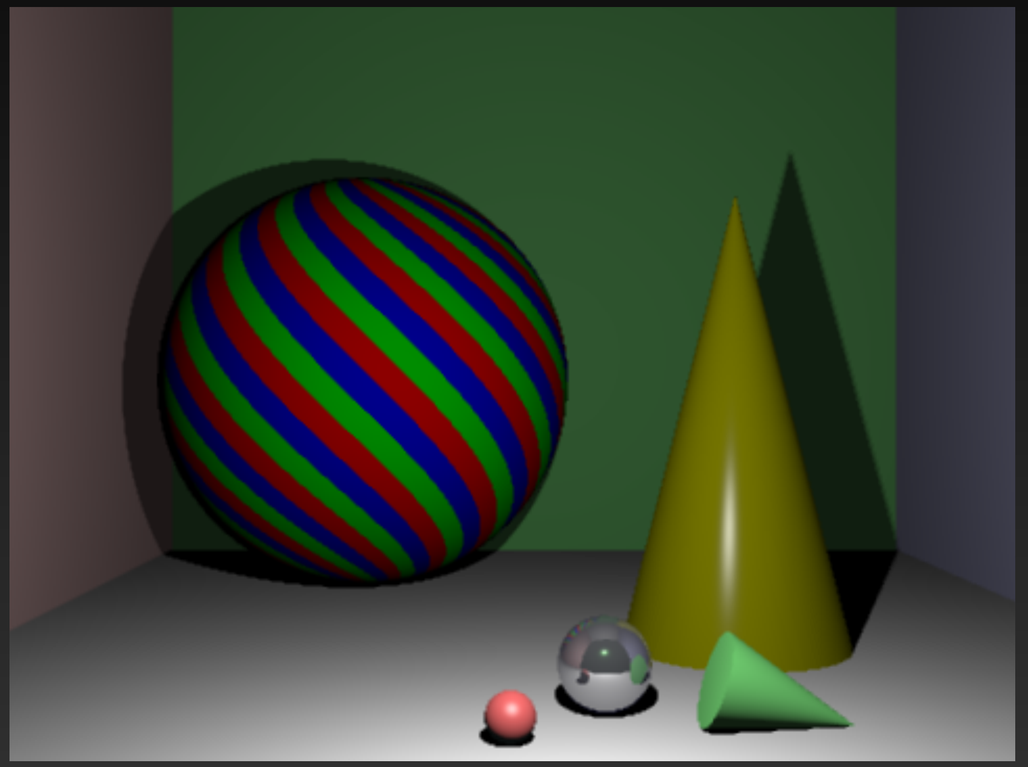
- Shooting more rays from the same position (Camera/Eye), to different points of the scene at a given distance from the original destination point.
- We implemented the shooting of 4 rays instead of one, for each pixel



Our results:

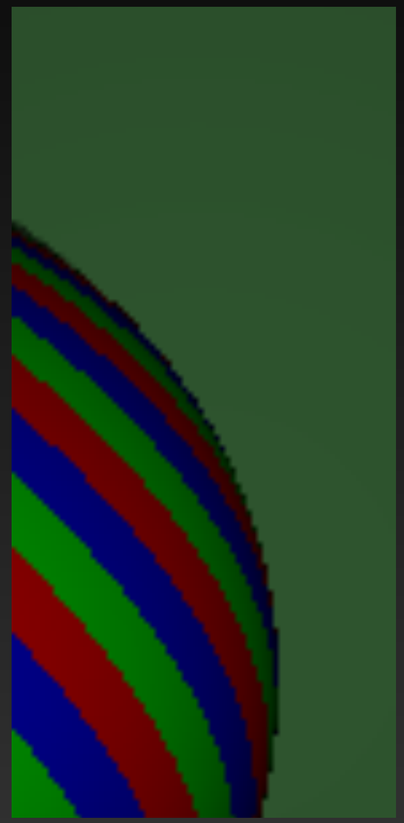


BEFORE

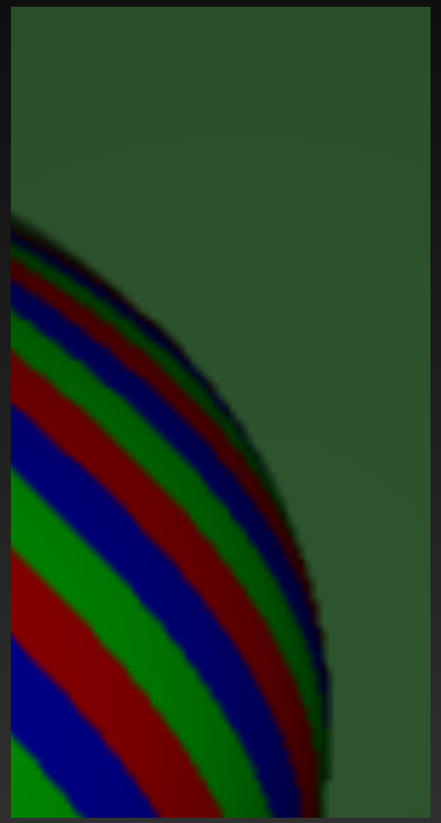


AFTER

Our results:

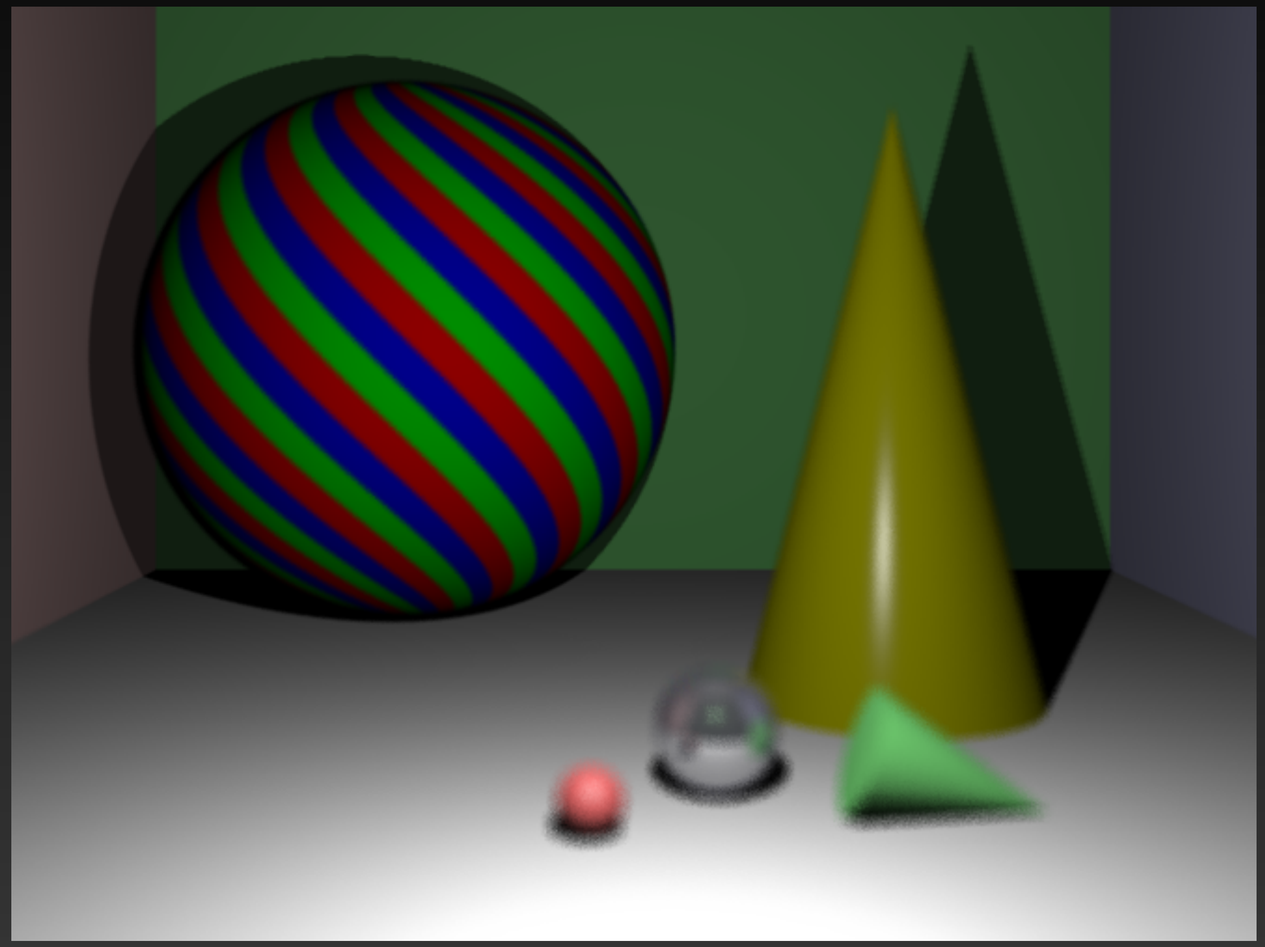


BEFORE



AFTER

Our results:



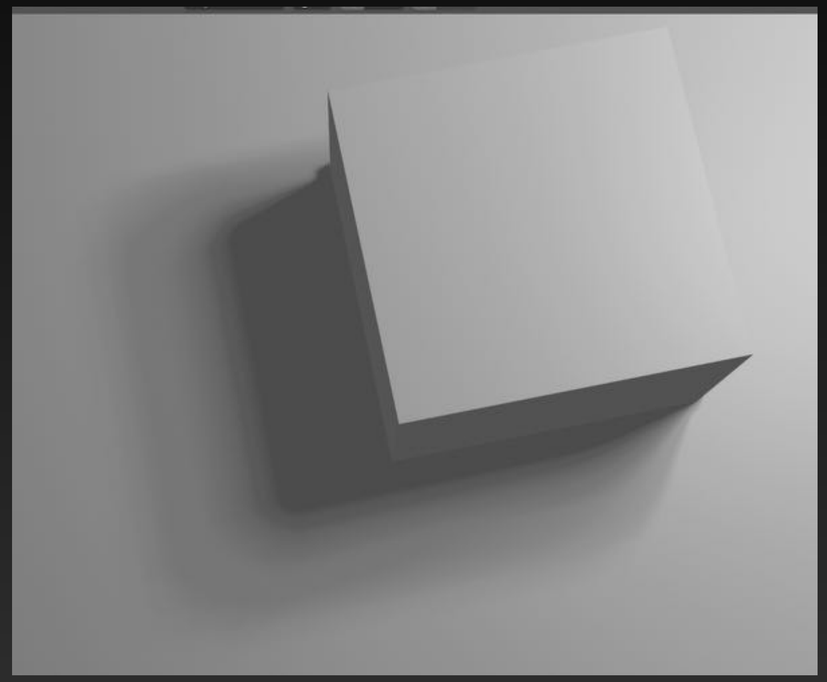
DOF + ANTIALIASING

Soft Shadows

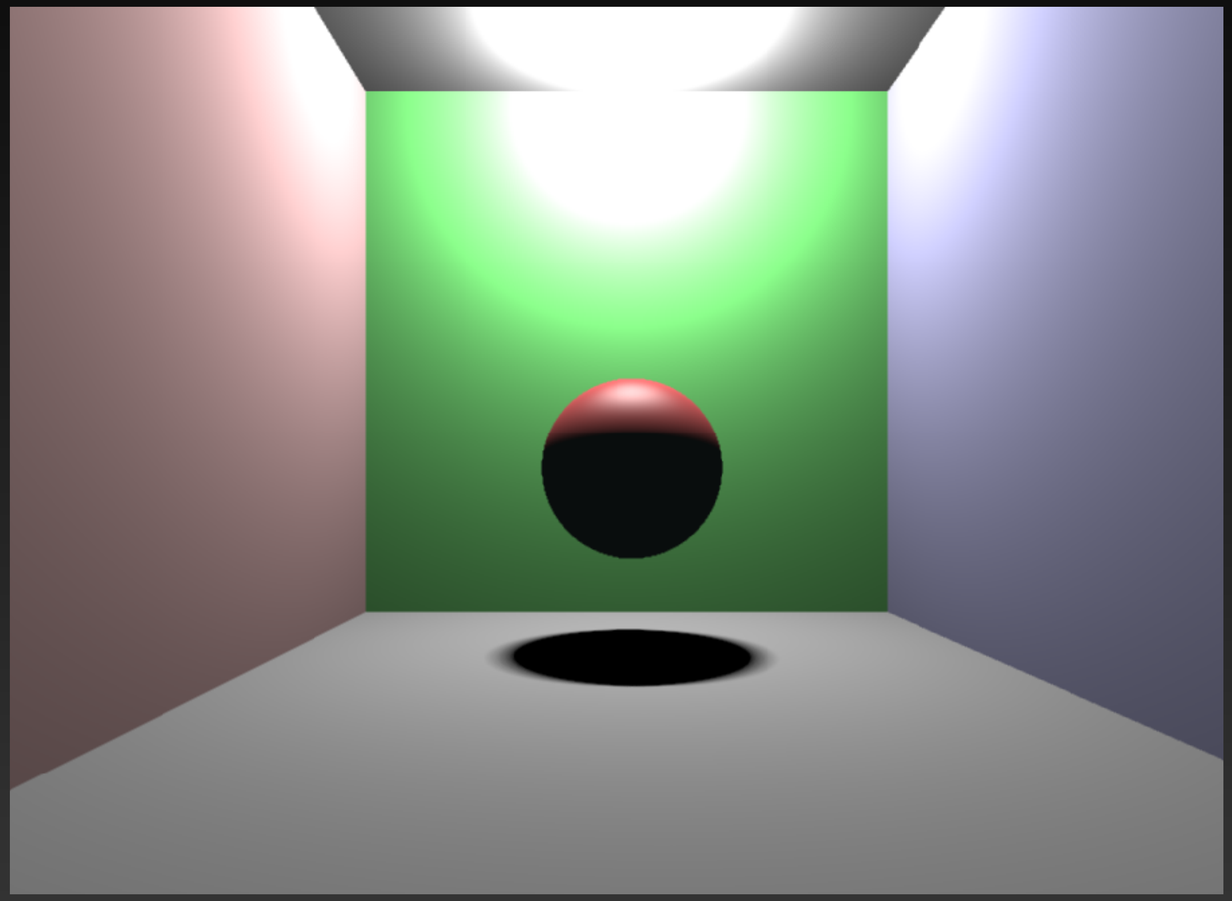
What’s behind it?

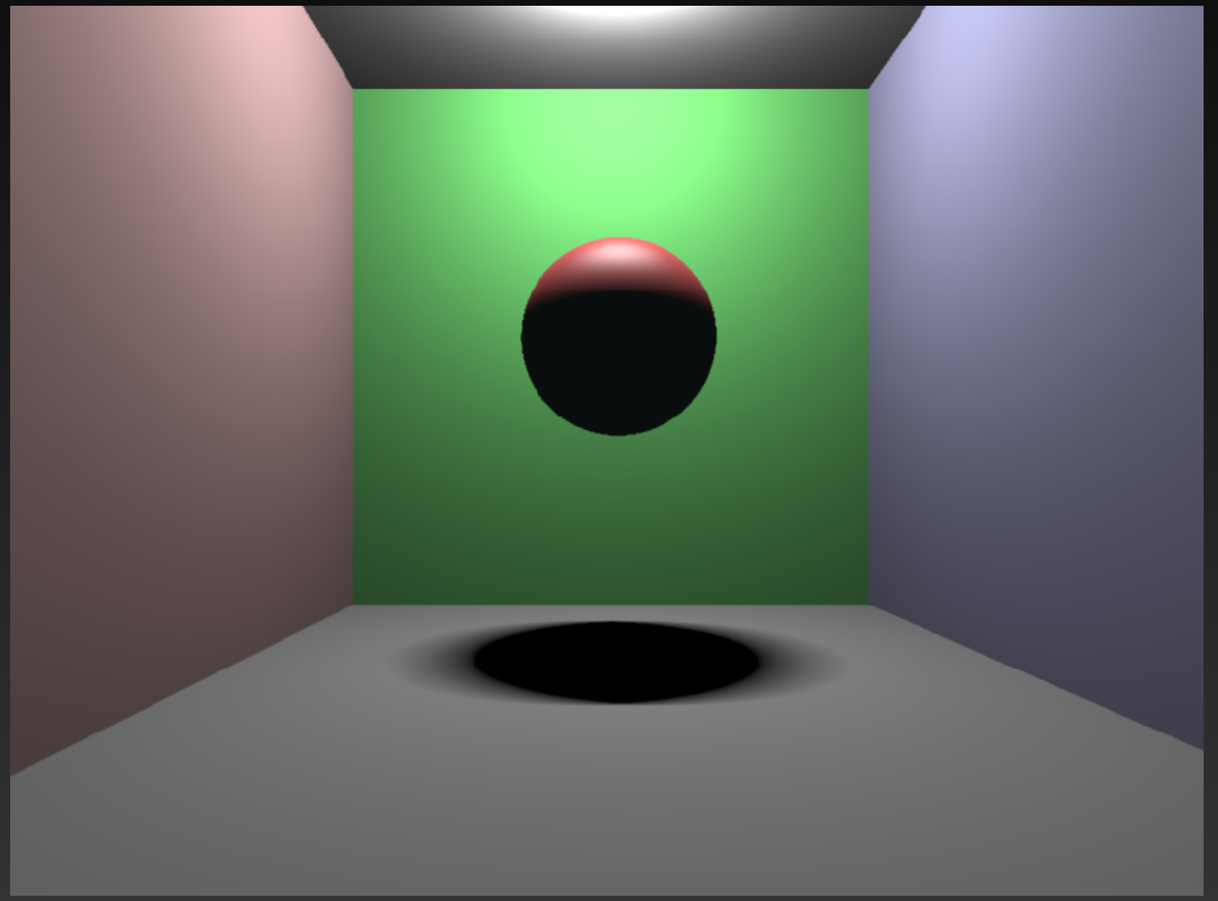
Soft shadows

- Using an “area light source” instead of a point light source we should obtain is a smoother shadow, result of the sum of more point light sources.
- From the POV of a Ray tracer this means using more point sources.
- for each point of the scene to render, assign a shadow value as the average of the incoming light sources.



Our results:





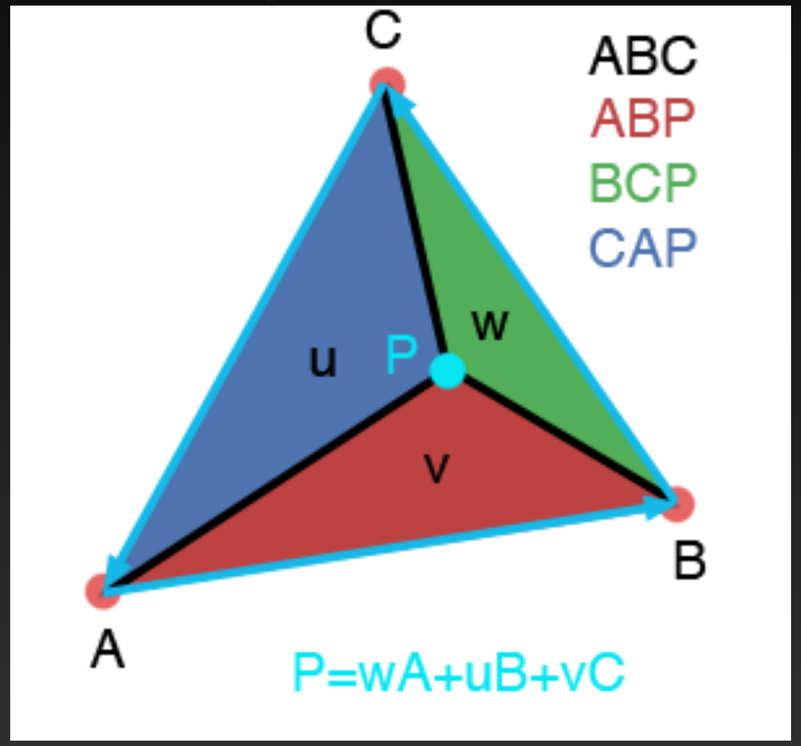
600 light sources (30 clusters with a light every 18 degrees)

KD Tree for Mesh Rendering

What’s behind it?

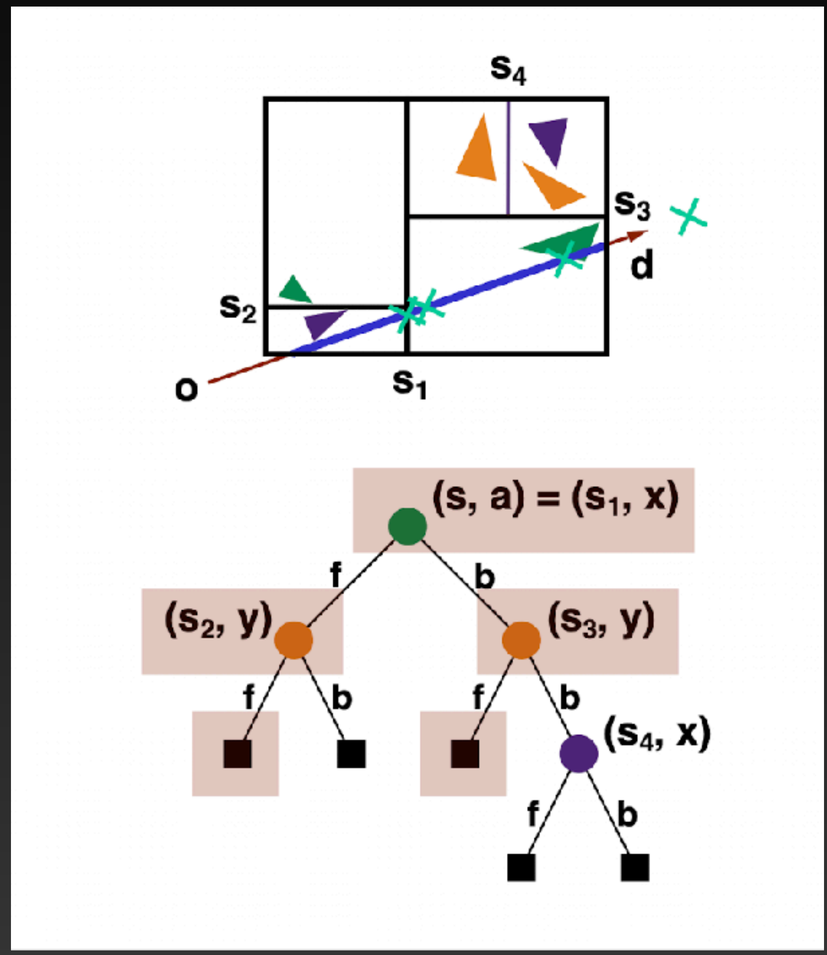
Mesh rendering

- First we render a mesh using barycentric coordinates, in order to see if the ray hits the triangle or not.

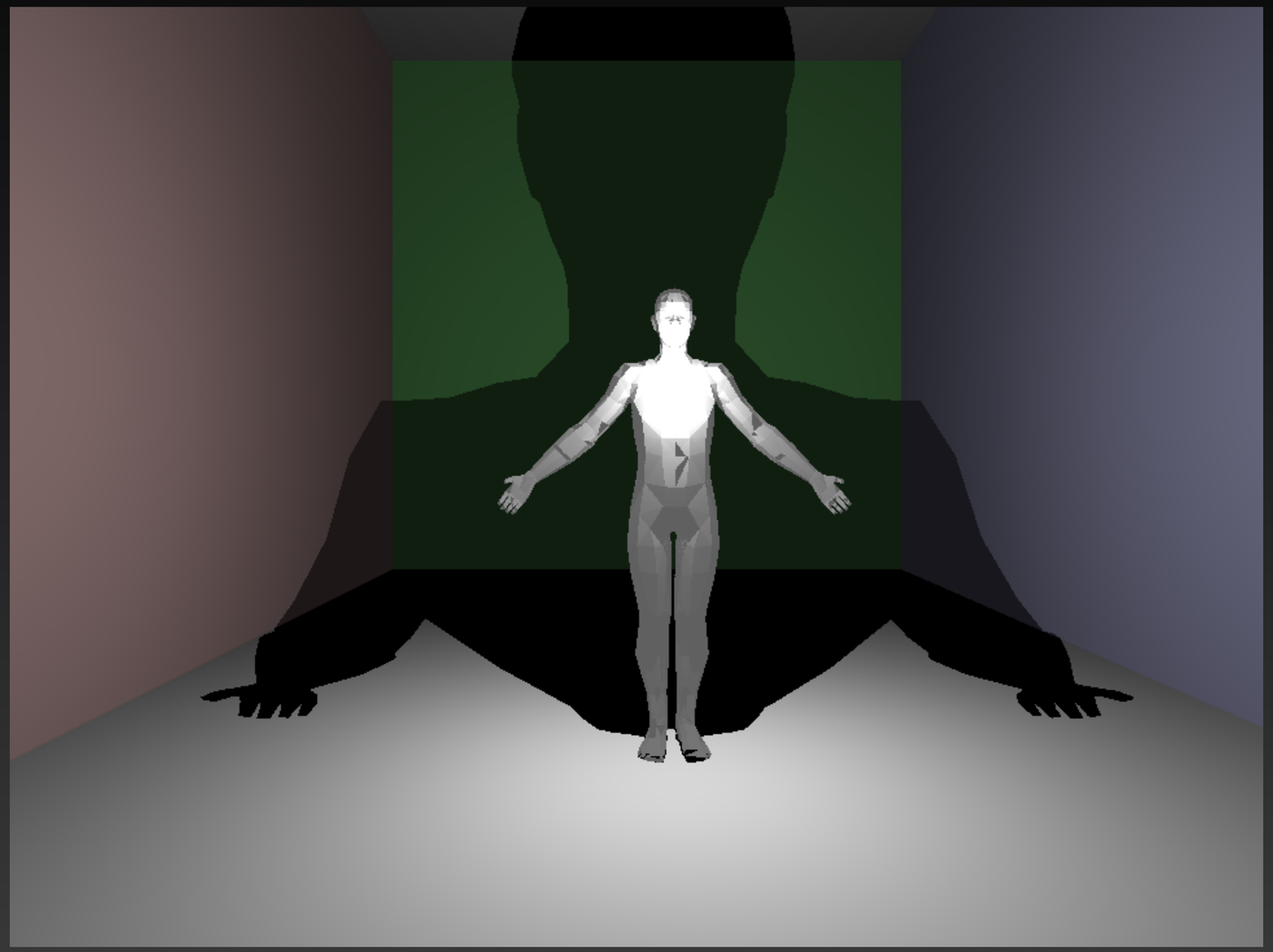


Kd tree implementation

- Then to optimise the storing of each mesh we implement a kd tree.
- In this way we obtain an retrieval algorithm of complexity o(logn) instead of o(n)



Our results:

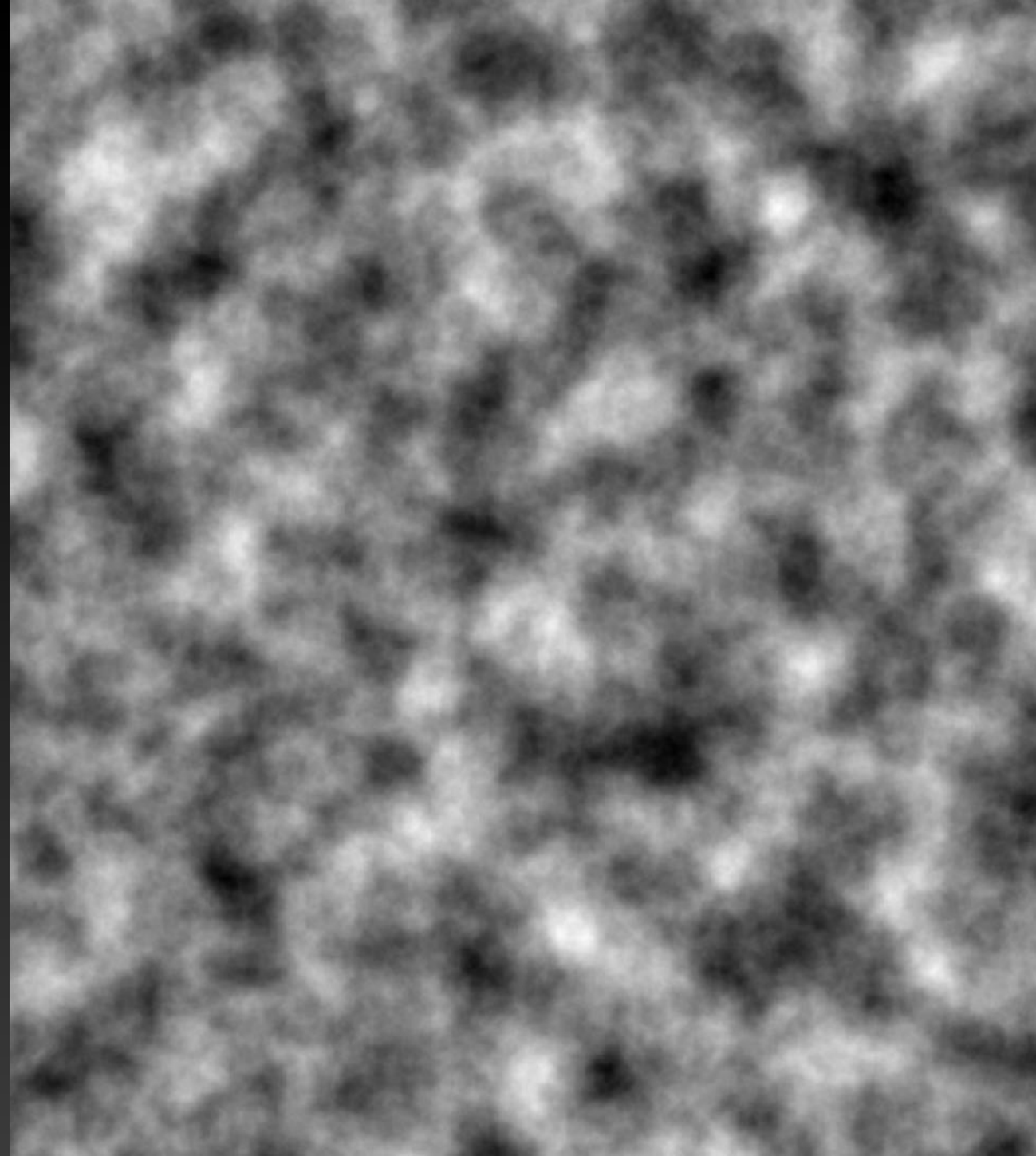


Perlin Noise

What’s behind it?

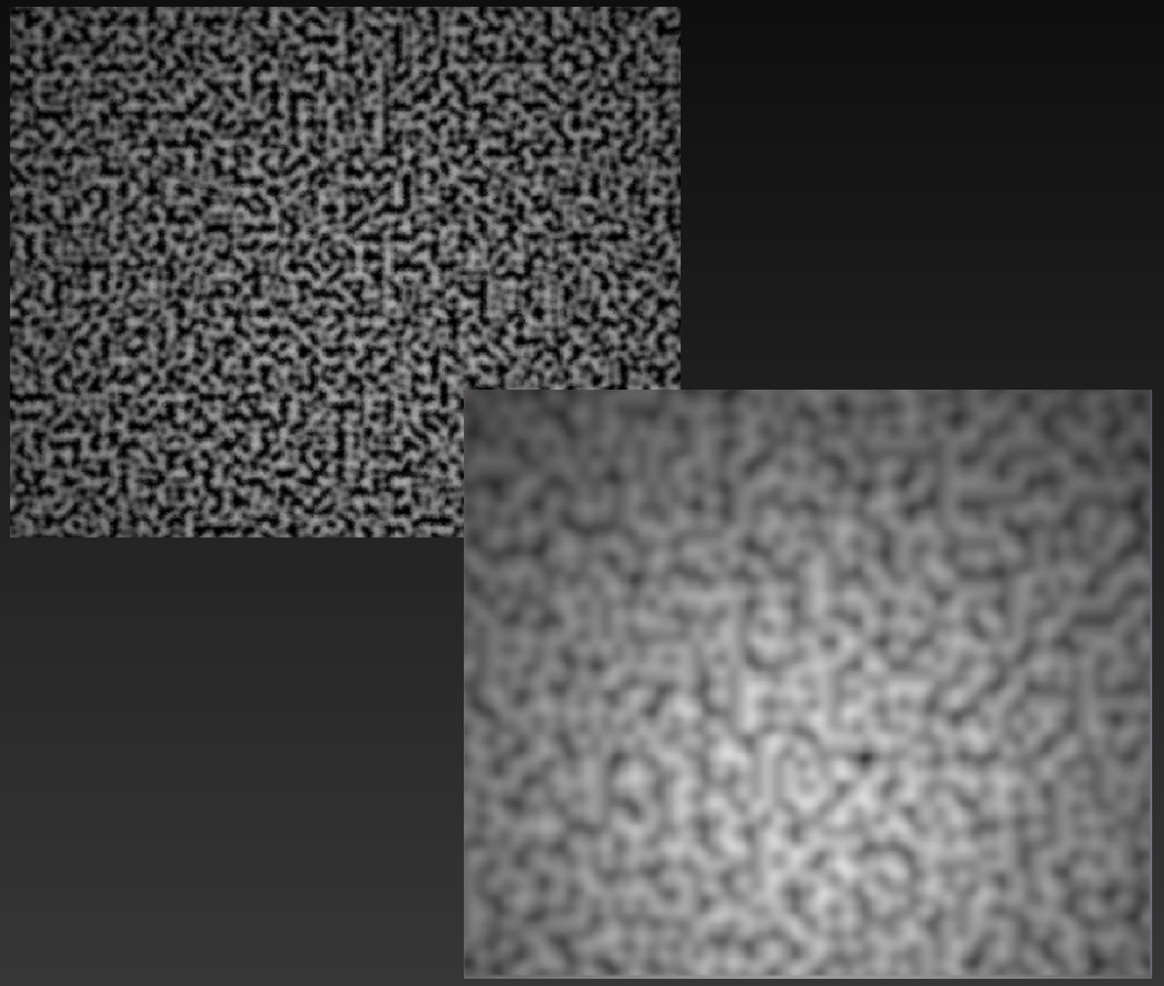
Perlin Noise

- Function used to recreate the noisy effect of some textures/ shapes in Nature like marble or water’s waves.
- The aim is to create a pattern in the randomness using this random value generation equation.

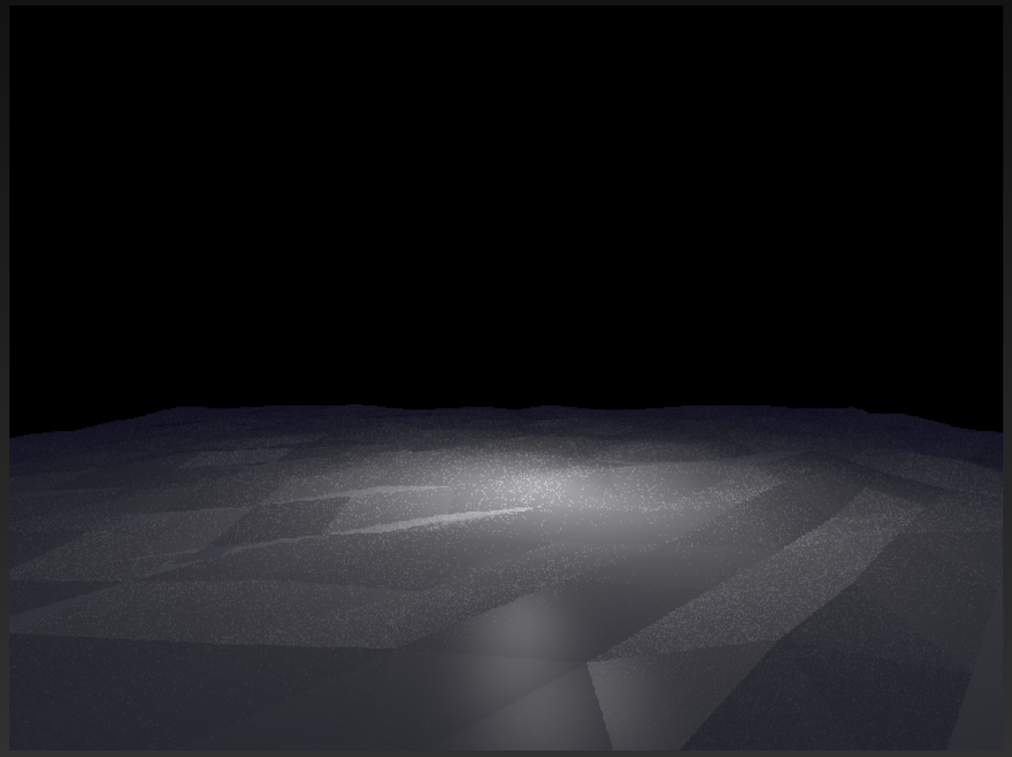


Our results:

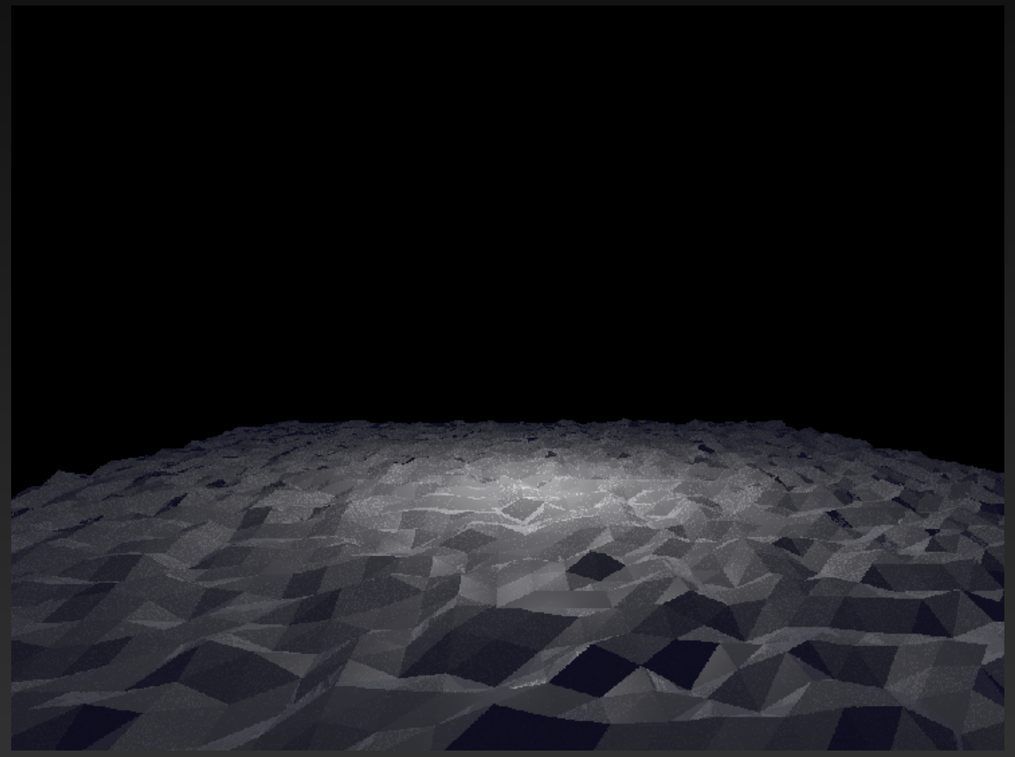
- First we can visualise the noise as 2D.
- This represents the height map that we can use to 3D model the textures in our scene.



Our results:



25x25 grid



50x50 grid

Thanks for the attention!