Real time Global Illumination using irradiance probes.

Table of Contents:

Objectives & Primary Goals:

The main objective of my project is to implement Real time Global illumination using DirectX 12 with use of raytraced irradiance probes based on Nvidia paper (Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields)

Achieved the following goals:

- Deferred Renderer with multipoint lighting
- Implement simple GPU raytracer in compute shader.
- Implement DDGI (dynamic diffuse global illumination
- Display the infamous Sponza Base Scene

Literature Survey:

Lightmaps:

- Precompute lighting over the entire surface of the model.
- Store it inside a texture.

Light probes

Precompute lighting inside a volume of space, rather than a texture.

Cached irradiance volumes

- irradiance volumes populated across the scene, that contains baked reflection cube maps (done offline)
- When rendering, simply look up into the volume to find out incoming lighting.
- Ray-Traced Global Illumination – Path Tracing.
- GPU based pathtracer with denoise function.

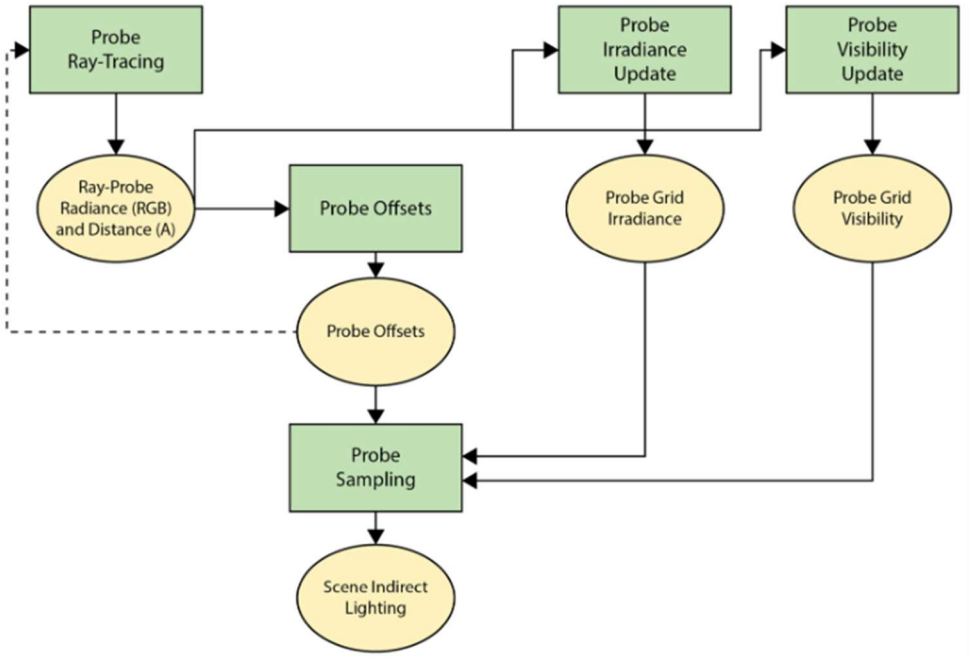
Real time Global Illumination using irradiance probes.

Main Issues from previous approaches:

- Apart from the path tracer, all others are baked.
- Hence for the dynamic light, these become inaccurate and useless.
- Path tracer is pretty slow for it to be fully real time even with modern RT compute graphic cards and heavily relies on denoising function.

Approach:

Dynamic Diffuse Global illumination



Approach: Dynamic Diffuse Global Illumination

Explanation:

Dynamic diffuse global illumination (DDGI) is a technique to render realistic indirect lighting in scenes with dynamic objects and lighting.

- Perform ray tracing for each probe and calculate the radiance and distance.
- Update the irradiance of all probes with the radiance calculated while applying some hysteresis.
- Update the visibility data of all probes with the distance calculated in the ray tracing pass, again with some hysteresis.
- Calculate the per-probe offset position using the ray tracing distance.

Real time Global Illumination using irradiance probes.

- Calculate indirect lighting by reading the updated irradiance, visibility, and probe offsets.

Architecture

I used Dx12 as the Rendering Api. The architecture flow is as follows,

- Deferred renderer
- Deferred Shadow generation
- Raytrace irradiance probes
- Direct lighting pass
- Indirect lighting
- Tonemapping

Deferred renderer:

- The deferred renderer renders the position, color and normal data as three different render textures.
- The deferred shadow generation generates shadow map in a different texture.

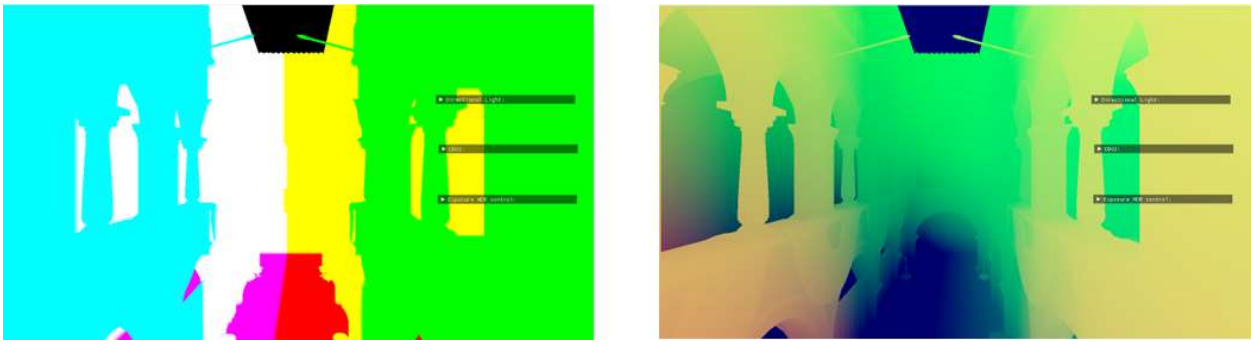


Figure: Deferred Renderer

Raytracing radiance probes:

- Created a software based raytracer written in compute shader.
- Created a grid of 16 * 16 * 16 probes.
- Run a raytracing computer shader on the probes.
- Used spherical Fibonacci method to determine ray direction towards each probe.
- Raycast from all the primitive's position which are in the position texture of g buffer, with the color value from the color texture of g buffer.

Real time Global Illumination using irradiance probes.

- Store this in an octahedral map for each probe.

Figure: Raytracing radiance probes

Rendering:

- We do a direct lighting pass that would just render the colors with direct light.
- Now for the indirect Lighting pass, we use the computed radiance stored in the cubemap, we sample 4 nearest grids.
- Combining this with the shadow pass, we generate an HDR texture.

Figure: Rendering

Future Work:

- Add ray traced reflections to the renderer.
- Add a dynamic exposure tone mapper.