# APAL Project Proposal:
# 2D Global Illumination

Austin J. Maddison
austin.mad@student.mahidol.edu
Mahidol University International College

## 1 Problem Description

One of the most critical components of realistic rendering is **Indirect Illumination**, also known as **Global Illumination (GI)**. Offline renderers typically solve for the radiance of visible surfaces in a scene using Monte Carlo Raytracing[1], iterating until the image converges to a noise-free result. This level of computation is infeasible for real-time rendering budget of ≤26ms per frame without specialized optimizations because of ray-tracing's long running time to converge.

## 2 Literature Review

Over the past decade, real-time GI solutions have relied on secondary scene representations, probe-based methods, or pre-baked lighting for static environments. These techniques generally fall into the following four categories:

(1) **Lightmaps**
   - Store ray-traced lighting data in texture maps.
   - Baked in advance for static scenes.

(2) **Tracing with Alternative Scene Representations**
   - Uses efficient structures such as:
     - Discrete Signed Distance Fields
     - Voxels
     - Proxies (low LOD proxies of LOD0 scene meshes)

(3) **Screen-Space Methods**
   - Raytrace against information in the G-buffer (Depth, Albedo, Normal Buffer).

(4) **Irradiance Probes**
   - Stores a coarse irradiance field.
   - Can be precomputed or updated dynamically.

Additional modern techniques that try to bring Monte Carlo Raytracing into realtime include:

- **Machine Learning (ML)**
  - Traces a low number of rays and infers missing details through denoising and upscaling.
- **Caching & Reuse**
  - **Temporal Accumulation & Reprojection**: Uses previous frames to refine current results.
  - **ReSTIR**: Reuses and resamples importance-traced rays for efficient global illumination.

### Irradiance Probes

Among these techniques, **Irradiance Probes**[2] have been one of the most favorable approaches due to their:

- Simplicity
- Scalabilty

However, they have notable **limitations**:

- They are usually have a coarse resolution because they are placed sparsly which limts their ability to capture fine details in exchange for performance.

For non static scenes increasing probe density significantly raises computational costs since the number of required rays scales accordingly, so its a bit of a conundrum. Thus, finding an optimal balance remains a challenge in real-time GI solutions.

### Radiance Cascades

In 2021, Alexander Sannikov, a programmer at Grinding Gear Games, developed a new GI solution called **radiance cascades**[3] while working on Path Of Exile 2. This solution draws inspiration from irradiance probes and the relationship between linear and angular resolution required to accurately capture an object's shape within a given area. Radiance cascades demonstrate better fidelity than irradiance probes while having a similar performance.

Radiance cascades utilize a hierarchy of cascading probes arranged in a grid. Unlike traditional (like Irradiance Probes) methods that shoot rays in random directions to measure the radiance at a point in the scene, radiance cascades employ a more structured approach. At each successive level in the cascade, the probes become more sparse but have higher angular resolution. This means that as you move up the levels, there are fewer probes, but they can capture more detailed angular information about the light in the scene. This hierarchical structure allows for efficient and accurate global illumination by balancing the density and resolution of probes.

Notable **limitations**:

- Since of it's complexity, implementing the solution to 3D world-space hasn't been well explored or documented.

## 3 Application

I aim to develop a **GPU benchmarking program**. This app will allow users to load simple 2D scene, and material using bitmap. The application will then intepret the image to render the scene using **three different GI solutions in screen-space** for benchmarking purposes.

(1) **Reference Raytracer** *(non-deterministic).*
(2) ***Irradiance Probes*** *(non-deterministic).*
(3) ***Radiance Cascades*** *(deterministic).*

I chose 2D as it reduces implementation time while still being able to apreciate and draw observatiosn from the different algorithms.

I did a preliminary test of using glsl compute shaders as I never used them before. Basic monte carlo raytracer, no colors, all surfaces is a light source, no bounce. I am using chapters 30 to 32 of the textbook "Computer Graphics: Principles and Practice"[4] as my main reference.

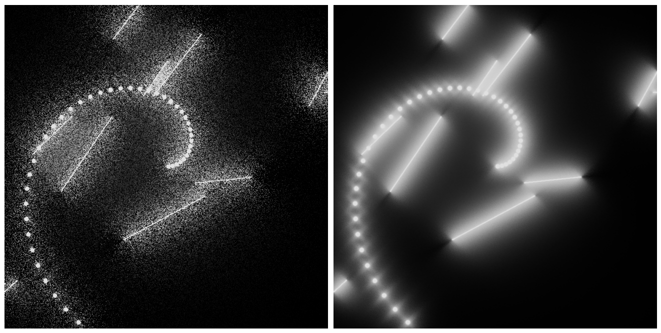

**Figure 1: The left image is rendered with 1 sample, while the right image uses 1024 samples.**

The hash function that I found from a blog[5] used to generate random directions for every new iteration seems to favor certain directions for some reason.

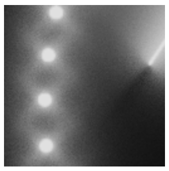

**Figure 2: Star shaped accumalation, side effect of the bias in directions that the hash function has.**

I don't exactly know how to generate better uniform random numbers on the GPU and make it fast, maybe this is an area I should explore, ... idk?

## Theory

For each GI solution I will discuss,

(1) **how they work**
(2) **time complexity**

How each suceeding GI solution (Monte-Carlo Raytracing → Irradiance Probes → Radience Cascades) was motivated from the last.

## Practice

For each GI soltuion I will compare thier performance and fidelity using the outputs from the benchmarking application...

(1) **running time**
(2) **image output**

## References

[1] James T. Kajiya, *The rendering equation*, SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, Pages 143 - 150, https://doi.org/10.1145/15922.15902, Published: 31 August 1986.

[2] Carsten Dachsbacher, *Cascaded Light Propagation Volumes for Real-Time Indirect Illumination*, Crytek GmbH, VISUS / University Stuttgart, 2010.

[3] Alexander Sannikov, *Radiance Cascades: A Novel Approach to Calculating Global Illumination [WIP]*, Grinding Gear Games. https://github.com/Raikiri/RadianceCascadesPaper/blob/main/out_latexmk2/RadianceCascades.pdf

[4] John F. Hughes, Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley, *Computer Graphics: Principles and Practice*, Third Edition, Addison-Wesley Professional, 2014.

[5] Blackle Mori, *Useful Functions for Shader Live Coding*, Suricrasia Online, https://suricrasia.online/blog/shader-functions/, Published: 11 April 2020.

Last modified Monday 3rd March, 2025