Series Title: *Simulating Reality: Physics, Math, and the Art of Rendering*

Lecture 1 – Light, Physics & Why We Render

Goal: Motivate rendering through physics, history, and first-principle math.
Duration: ~2 hrs

Learning flow

1. *Hook:* real photo vs render — what's missing?
2. Why we render → "when we can't record, we simulate."
3. Huygens → Fermat → Snell (derive via stationary optical path).
4. Fresnel reflectance → Schlick's approximation.
5. "How GPUs fake physics every frame."
6. Intro to basic ray tracing concept.
7. *Keriso note:* shaders simulate these principles even in 2D.

Assignment 1:

- Lorem Ipsum

Blog:

- Full derivation of Snell's Law and Fresnel equations.
- Visuals: light paths, stationary vs non-stationary routes.

Lecture 2 – Rays, Monte Carlo & Path Tracing

Goal: Unite physics + probability → realistic light transport.
Duration: ~2 hrs

Learning flow

1. Equation of a ray, surface intersection recap.
2. Rendering Equation (Kajiya) → Monte Carlo approximation.
3. Expected-value derivation & law of large numbers.
4. Variance, importance sampling, convergence.
5. Concept of path tracing (recursive integration).

6. Demo: noisy → smooth image as samples increase.
7. *Keriso note:* sampling logic parallels particle systems and lighting noise in 2D games.

Assignment 2:

- Lorem Ipsum

Blog:

- Derivation of rendering equation & Monte Carlo integration.
- Visuals: integrand sampling & variance plots.

Mini Project A (between Lecture 2 & 3)

Goal: Build intuition for light-geometry interaction before heavy math.
Task:

- Implement a minimal CPU path tracer (diffuse only) or visualize random ray bounces in Godot 2D.
- Record convergence with sample count.
  Deliverable: short demo + one-paragraph reflection on what "Monte Carlo realism" felt like.

Lecture 3 – Geometry, Light & Performance

Goal: Combine geometry math with lighting and show performance implications.
Math focus: vector algebra, matrix transforms, BVH concepts.

Learning flow

1. Vectors, normals, and coordinate transforms (world→view→camera).
2. Ray-sphere and ray-triangle intersection formulas.
3. Combine intersections + lighting to form first full render.
4. BVH math — AABB intersection and recursive partitioning.
5. GPU parallelism basics (thread groups, SIMD).
6. *Keriso note:* same math drives sprite placement and collision in 2D engine.

Assignment 3:

- Lorem Ipsum

Blog:

- Detailed BVH intersection derivation & complexity analysis.

Lecture 4 – Surfaces, Materials & Shaders

Goal: Model surface response mathematically and implement via shader logic.
Math focus: BRDF integrals, Lambertian & specular models.

Learning flow

1. Energy conservation and BRDF definition.
2. Lambertian diffuse derivation from radiance integral.
3. Phong and Blinn-Phong equations.
4. Microfacet model concepts (GGX overview).
5. Tone mapping and exposure math (Reinhard curve).
6. Shaders in Godot (2D lighting, normal maps).
7. *Keriso note:* integrate a custom lighting shader for Keriso's 2D pipeline.

Assignment 4:

- Lorem Ipsum

Blog:

- Normalization of BRDFs and Lambertian derivation.

Lecture 5 – Color, Perception & Realism

Goal: Connect physics of light to human vision and art direction.
Math focus: color spaces, gamma, chromaticity coords, perception laws.

Learning flow

1. Spectrum → tristimulus integration → RGB space.
2. CIE XYZ to sRGB transform matrix.
3. Gamma correction and tone mapping functions.
4. Perception laws (Weber–Fechner, contrast adaptation).
5. Realism vs artistic style → case studies (PBR vs stylized).
6. *Keriso note:* color grading and LUTs in 2D scene composition.

Assignment 5:

- Lorem Ipsum

Blog:

- Math behind gamma correction and chromatic adaptation.

Lecture 6 – Engines & The Art of Illusion

Goal: Integrate physics, math, and perception into engine architecture and creative direction.

Learning flow

1. Recap: light → geometry → material → color → perception.
2. Engine overview: render loop, ECS, update loop, resource loading.
3. Optimization and parallel processing.
4. Realism perception vs art direction (believability over accuracy).
5. Future paths: real-time GI, neural rendering, simulation research.
6. Open discussion + Keriso task division.

Capstone Project:
Integrate one studied concept (shader, lighting, sampling) into Keriso and write a devlog explaining the math behind it.

Blog: *"Engineering Illusions: When Physics Feels Like Art."*

Pedagogical Principles

- Every lecture < 2 hrs with a visible demo midway.
- Deep math handled conceptually in class, formally derived in blog.
- Regular cross-reference to Keriso for relevance.
- Mini projects and assignments for layered retention.
- Constant show-think-code loop to keep attention alive.