State-of-the-Art Computer-Generated Imagery

CSU44052 Computer Graphics

Binh-Son Hua

Objectives

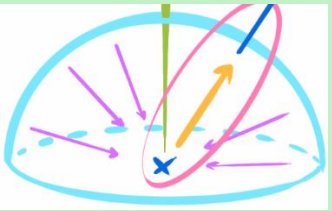
- Path tracing
- State-of-the-art computer graphics

Reflectance Equation

- The *reflectance equation* relates reflected radiance to incoming radiance that is scattered according to the surface's BRDF:

$$\underset{\textit{(at point x, in reflected direction } \omega_r\textit{)}}{\text{Reflected radiance}} = \int_{\Omega} \text{BRDF} \; * \; \underset{\textit{(at point x, from incoming direction } \omega_i\textit{)}}{\text{Incident radiance}}$$

- Ω = domain of integration
  - *i.e., the set of all possible incoming directions*
  - *Hemisphere if surface is opaque, otherwise a sphere*

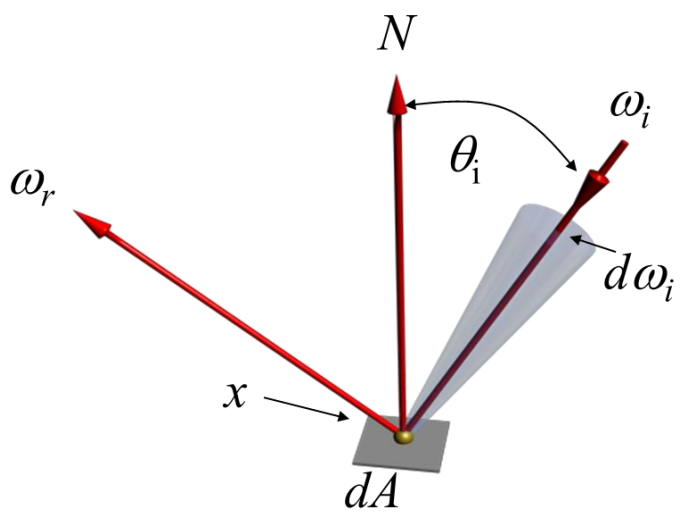


- Incident radiance, weighted with the BRDF, is integrated over the hemisphere to compute the outgoing radiance. This expresses energy balance

Reflectance Equation

- The *reflectance equation* relates reflected radiance to incoming radiance that is scattered according to the surface's BRDF:

$$\underbrace{L_r(x,\omega_r)}_{\text{Reflected radiance}} = \int_{\Omega} \underbrace{f_r(x,\omega_i,\omega_r)}_{\text{BRDF}} \underbrace{L_i(x,\omega_i)}_{\text{Incident radiance}} \cos\theta_i \, d\omega_i$$



Ω = domain of integration

$(\theta_i, \phi_i) = \omega_i$ = incoming light vector

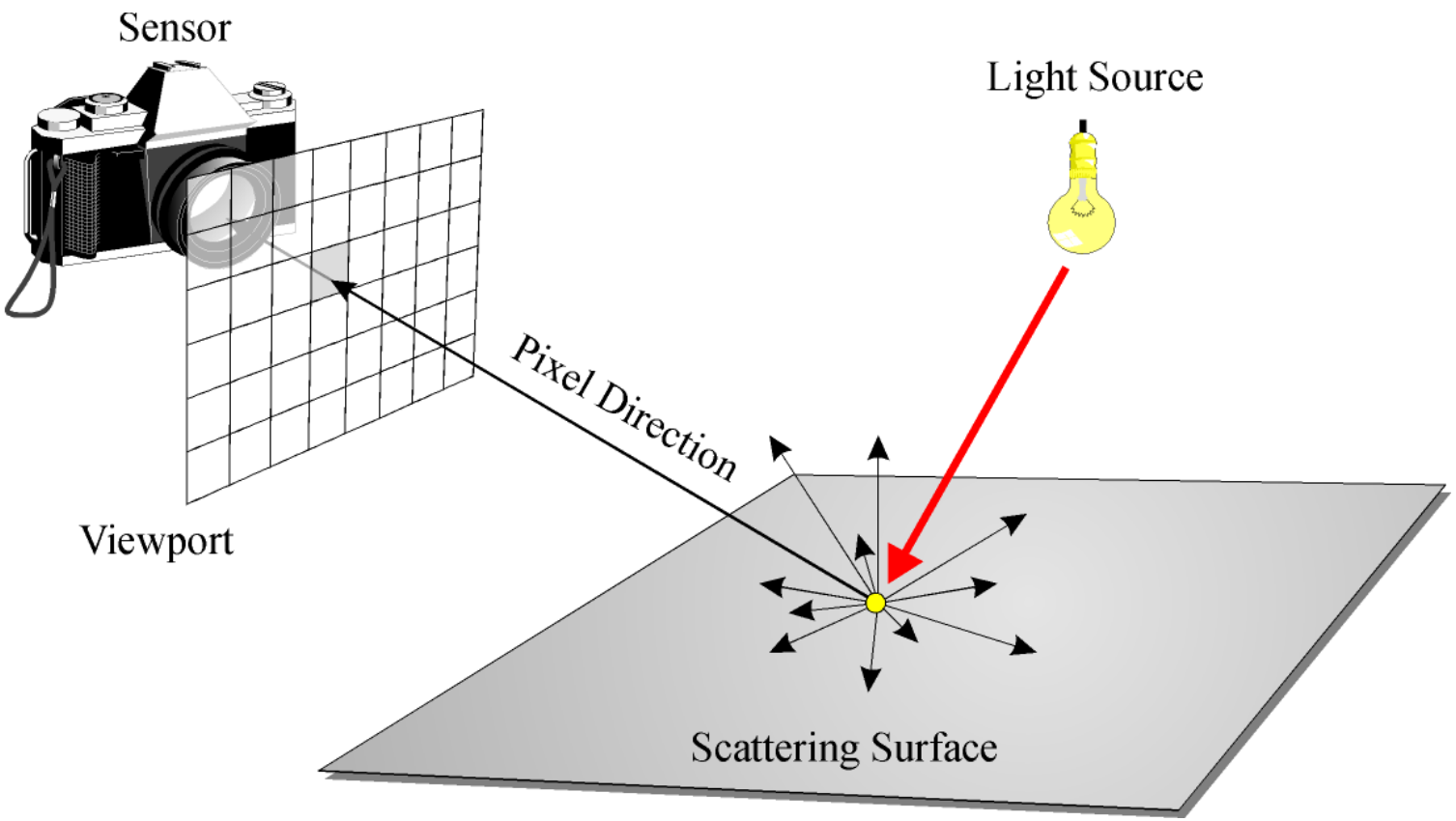
Rendering Equation

- The radiance equation includes a *self-emitted term* to account for light sources.
- This is the most important equation in rendering theory.
- We solve for $L_r(x, \omega_r)$ at each visible point in the scene.

$$L_r(x,\omega_r) = \underbrace{L_e(x,\omega_r)}_{\text{Self emitted radiance}} + \int_{\Omega} f_r(x,\omega_i,\omega_r) L_i(x,\omega_i) \cos\theta_i \, d\omega_i$$

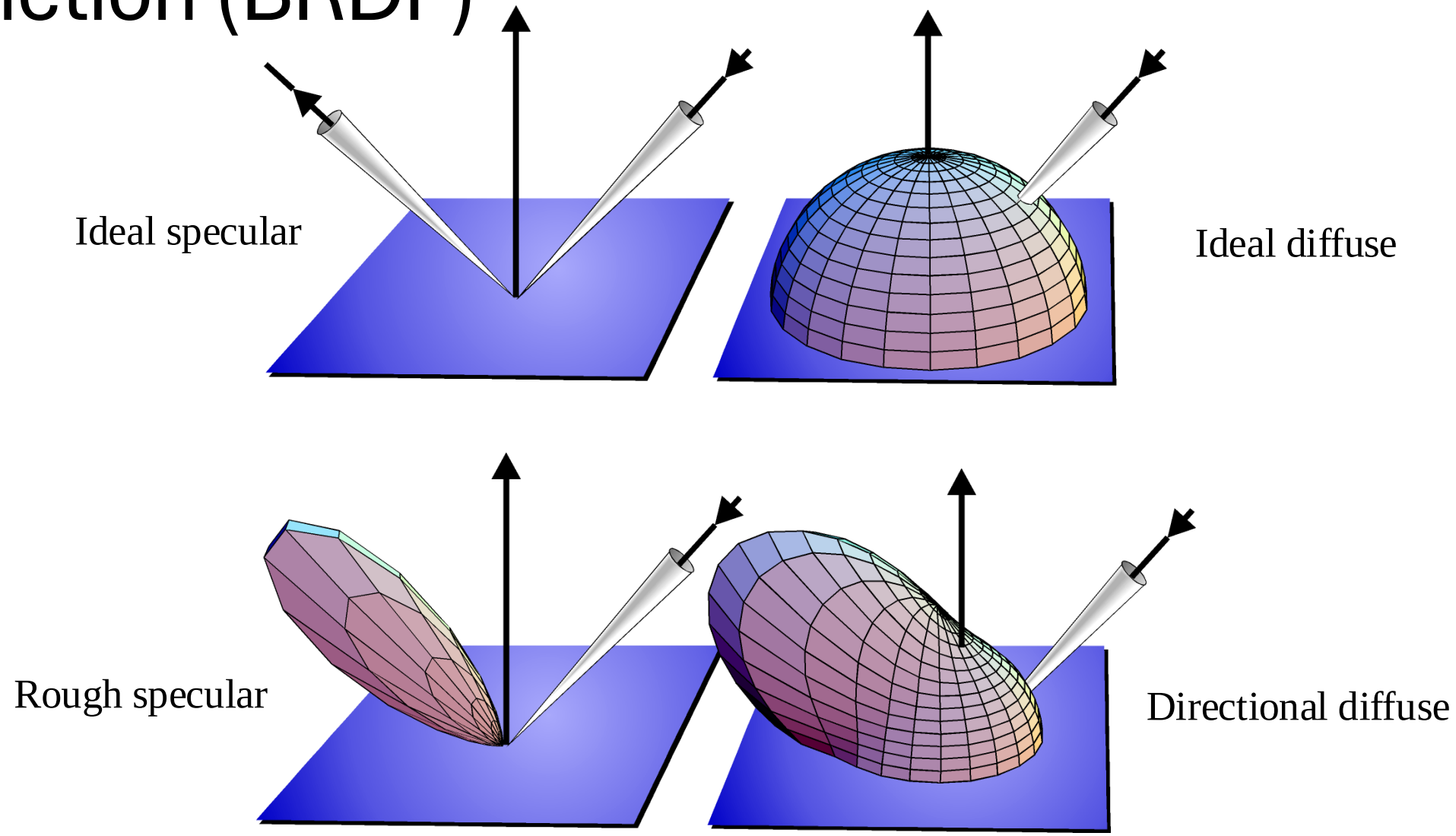
(non zero for light sources)

(Linear Fredholm Integral of the 2nd kind)



- An image is formed when light energy is scattered by surfaces in the scene towards the viewport
  - (or emitted directly towards the viewport)

Bi-directional Reflectance Distribution Function (BRDF)



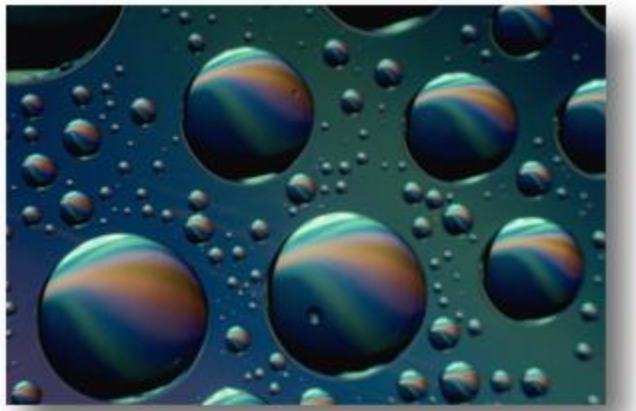
Complex Surface Scattering

Multi-layered Surfaces

Coloured Glass

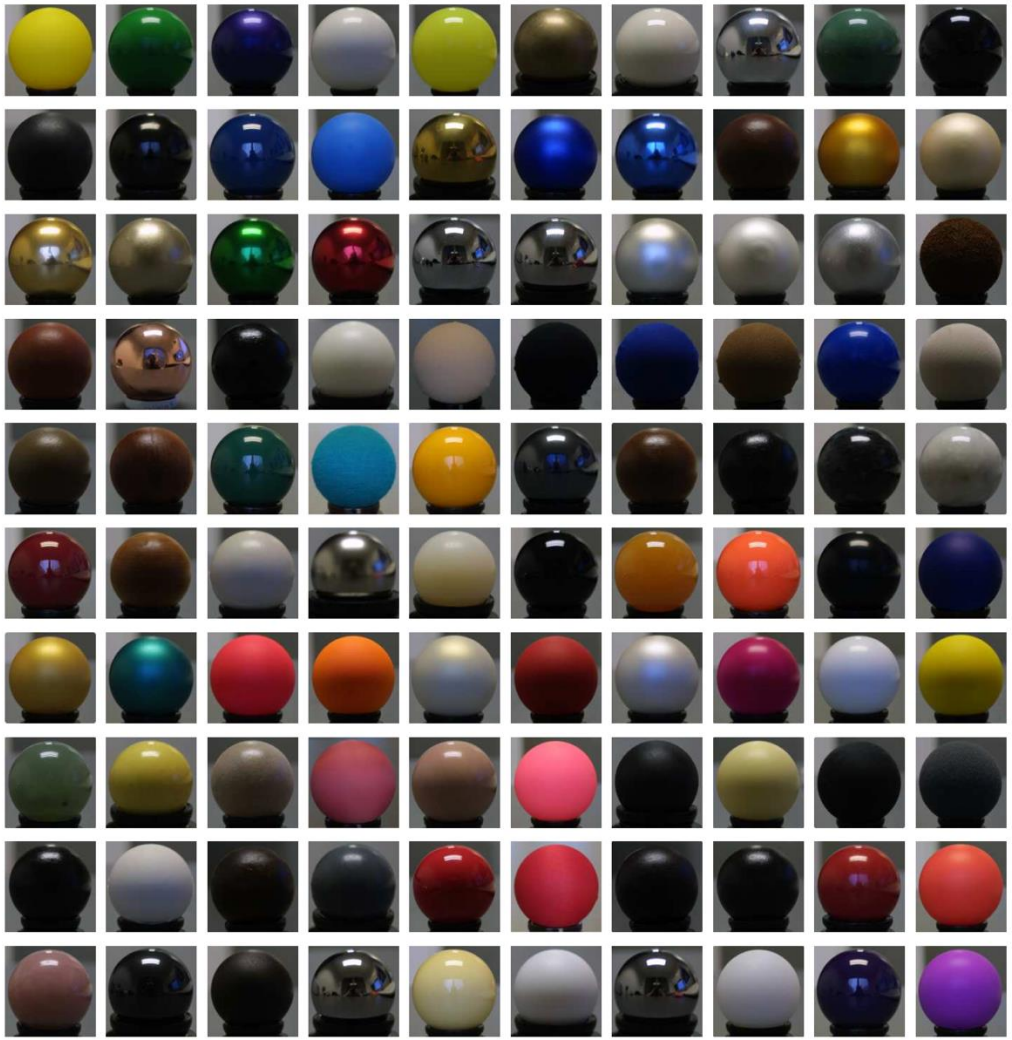
Human Skin

Stone



Thin Film

Tarnished Metal



MERL dataset

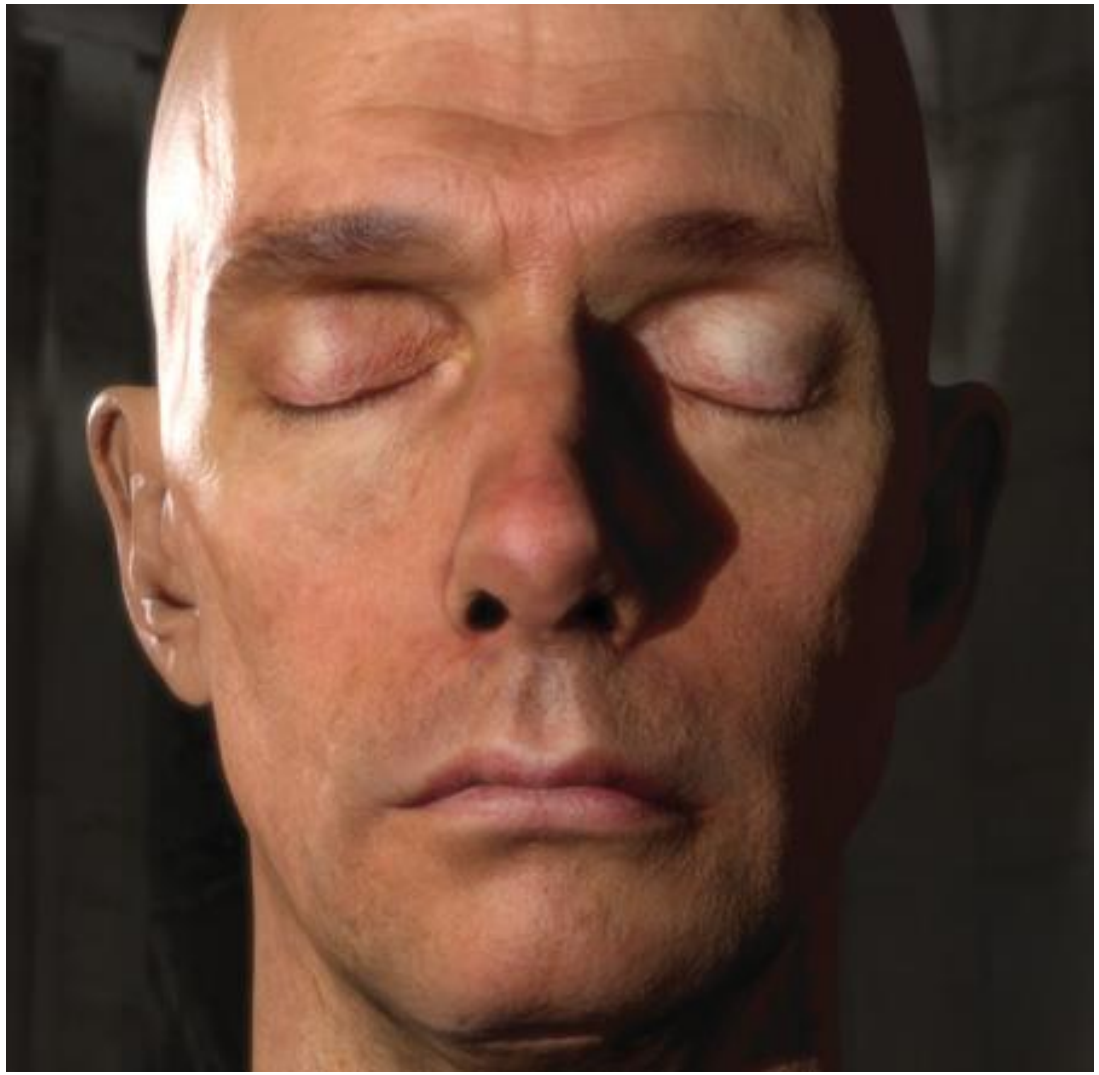
Teapot rendered using nickel, hematite, gold paint, and pink fabric.

Matusik et al., A Data-Driven Reflectance Model, SIGGRAPH 2003

Subsurface Scattering

- This occurs when light enters a translucent object and exits at different points

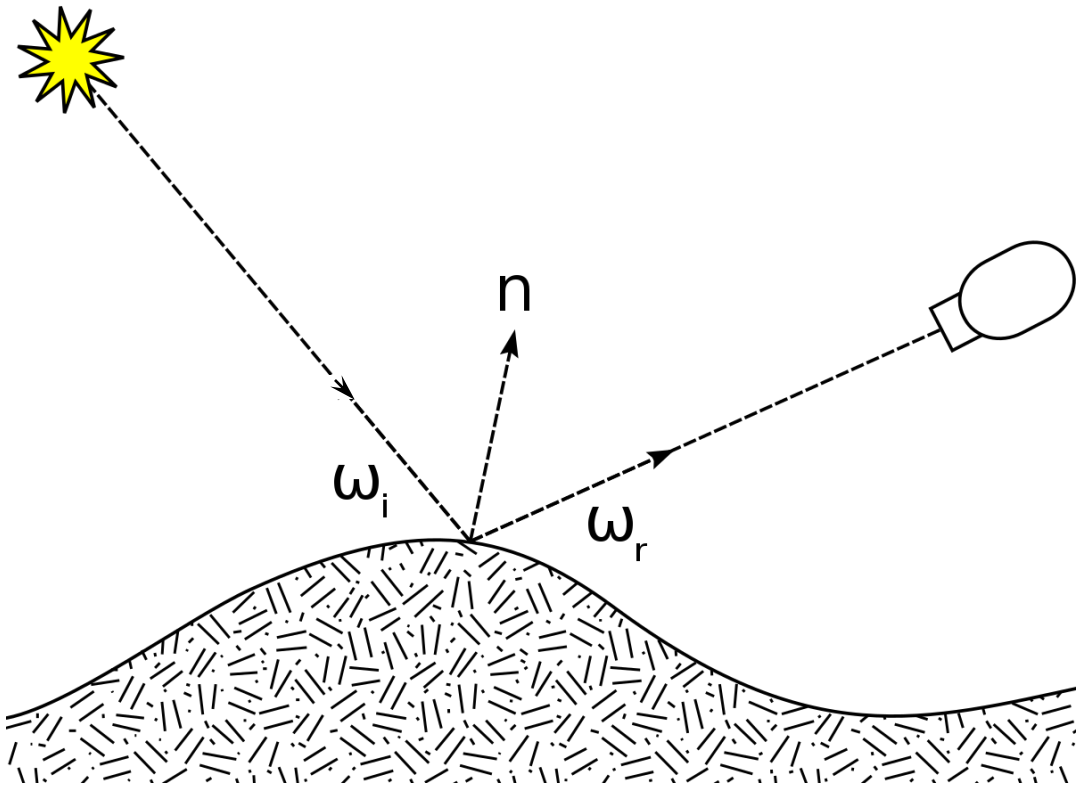
- Examples: skin, marbles.



Subsurface scattering, NVIDIA

BRDF

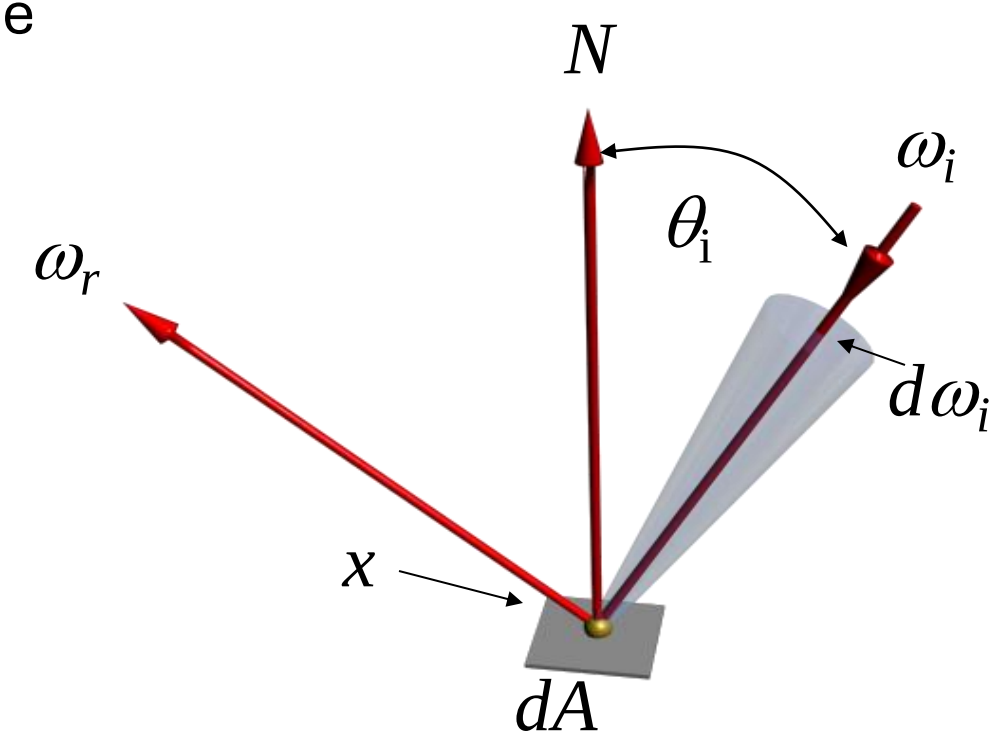
- The BRDF is a probability distribution function.
- It describes the probability that an incoming ray of light is scattered in a particular outgoing direction.
- Usually, we want to know what is being scattered towards the viewpoint (i.e., the camera)



BRDF

- The BRDF describes <u>reflected</u> radiance given incoming radiance (irradiance)
  - L_r = Radiance
  - E_i = Irradiance

- Represented by a high-dimensional function:
  - position = x
  - incoming direction = $\omega_i = (\theta_i, \phi_i)$
  - reflected direction = $\omega_r = (\theta_r, \phi_r)$
  - wavelength = λ



$$f_r(x, \omega_i, \omega_r) = \frac{dL_r(x, \omega_r)}{E_i(x, \omega_i)\cos\theta_i \, d\omega_i}$$

Common BRDFs

- Lambertian BRDF:

$$f(\omega_o, \omega_i) = \frac{\rho}{\pi}$$

- Phong BRDF:

$$f(\omega_o, \omega_i) = f_d(\omega_o, \omega_i) + f_s(\omega_o, \omega_i) = \rho_d \frac{1}{\pi} + \rho_s \frac{n+2}{2\pi} \cos^n(\omega_i, \omega_r)$$

- Other BRDFs: microfacet models, layered materials, measured BRDFs.

How to solve the rendering equation?

$$L_r(x, \omega_r) = L_e(x, \omega_r) + \int_{\Omega} f_r(x, \omega_i, \omega_r) L_i(x, \omega_i) \cos\theta_i \, d\omega_i$$

- How to generalize ray tracing to a full light transport simulation?
- How to handle different types of light sources, BRDFs, and surfaces?

Monte Carlo Integration

- The rendering equation has an integral that has no general, closed-form solution.

- Multiple approximate solutions: ray tracing, radiosity, but these methods assume simplified light transport, e.g., Lambertian surfaces, transparent surfaces, no indirect illumination, etc.

- Monte Carlo rendering provides an estimate to the rendering equation and therefore capture full light transport effects.

Monte Carlo Integration

$$I = \int f(x)dx$$

- Monte Carlo algorithms are algorithms based on the use of random numbers to estimate an integral, giving the right answer **on average**.

$$\langle I \rangle = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

- $\langle I \rangle$ is an estimator of the true integral value I. We can prove $\mathbb{E}[\langle I \rangle] = I$.

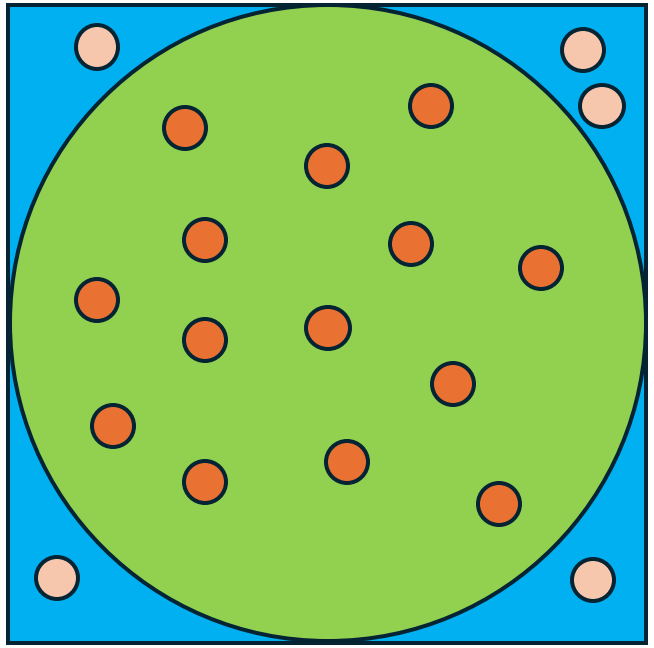
Exercise: Estimate area of a unit circle

- Use Monte Carlo estimation to find the area of a circle with unit diameter (radius = 0.5):

$$I = \iint_0^1 f(x, y)dxdy$$

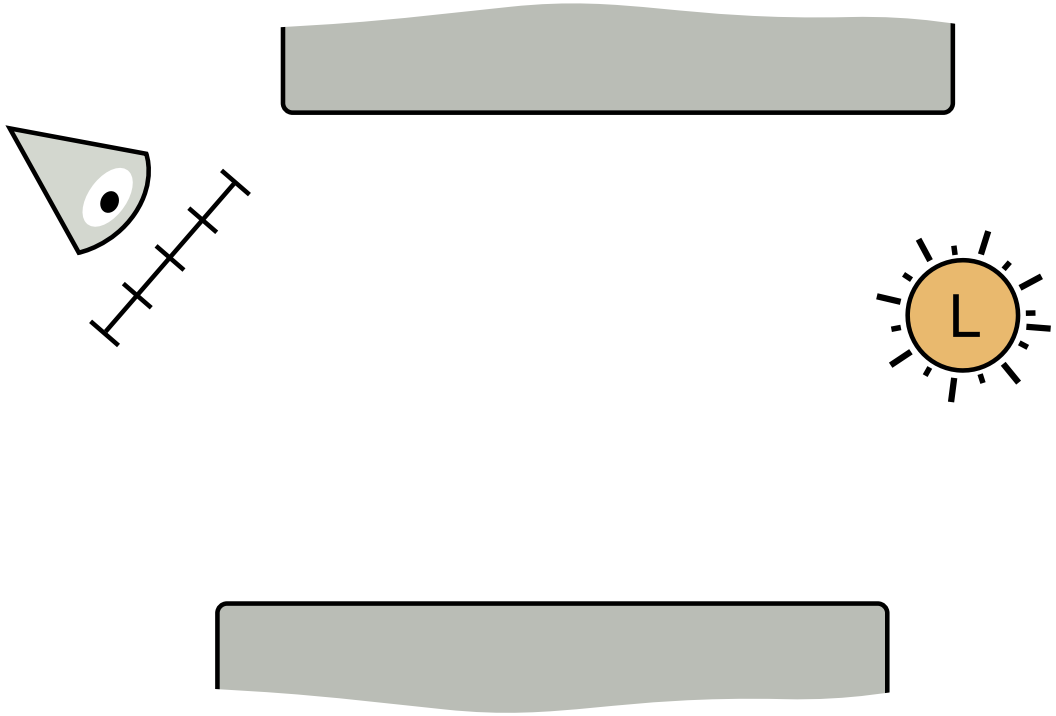
- $f(x, y)$ indicates points inside a circle and returns 1 when $x^2 + y^2 \leq 1$ else returns 0.
- We estimate the integral by taking uniform random samples in the square domain: $0 \leq x, y \leq 1$
- Probability of a point $p(x, y) = 1$ (unit square)
- Estimated area is number of points inside circle divided by total number of sampled points in the square.

$$I = \frac{1}{N}\sum_{i=1}^{N} f(x_i, y_i)$$



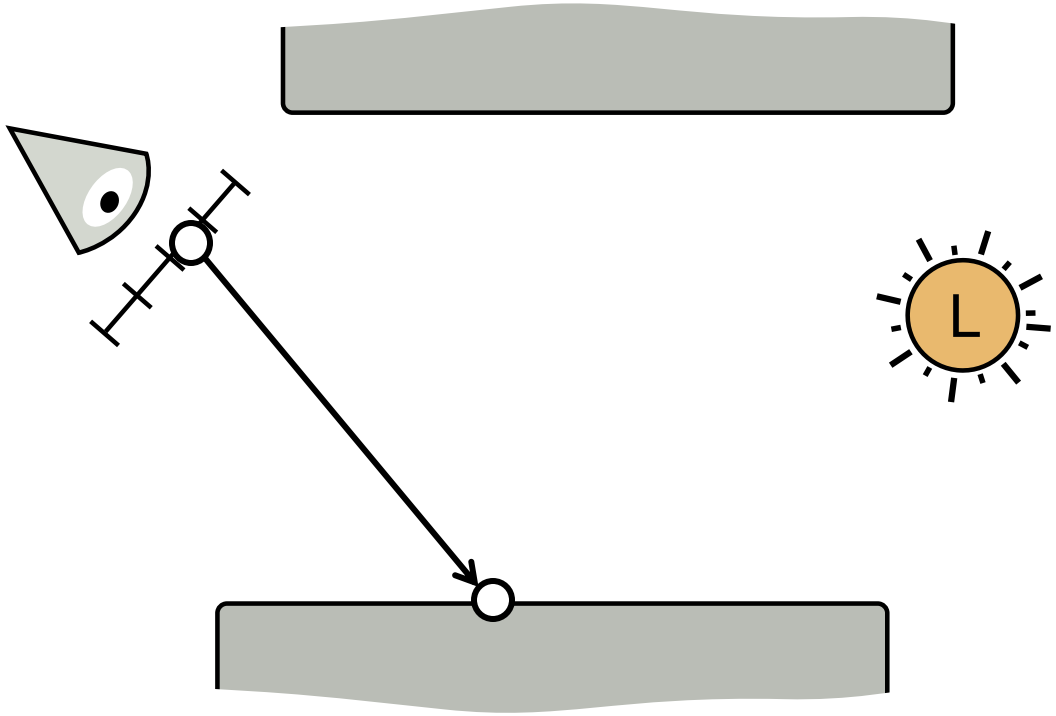
Path Tracing

- A general light transport algorithm based on unbiased Monte Carlo estimation of the rendering integral.



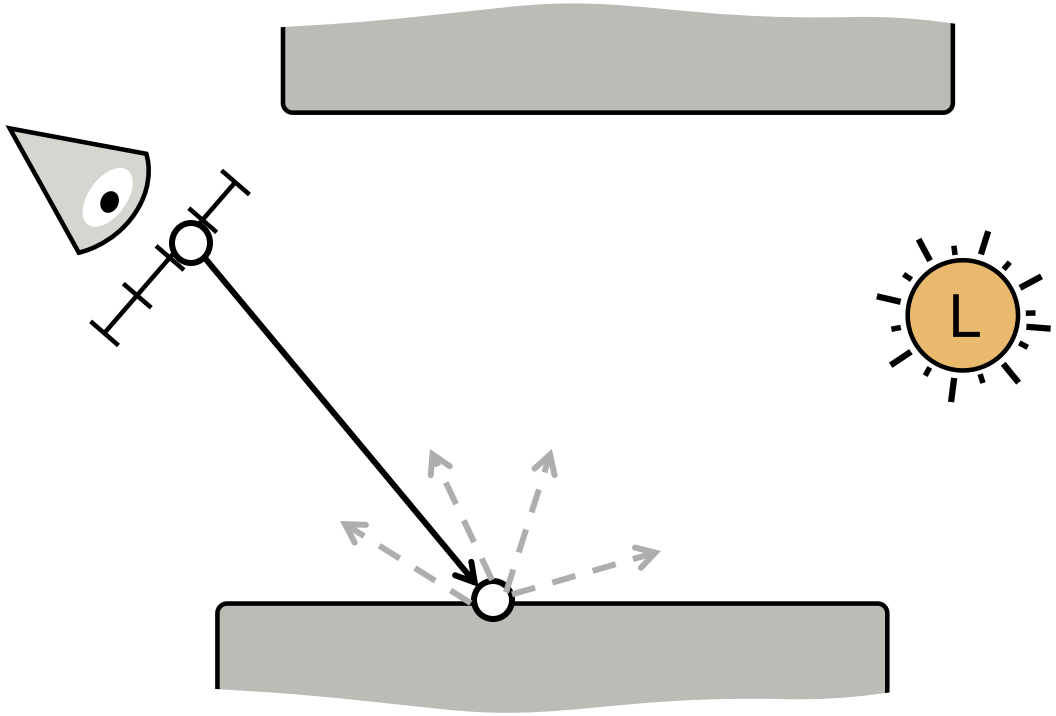
Path Tracing

- Direct illumination



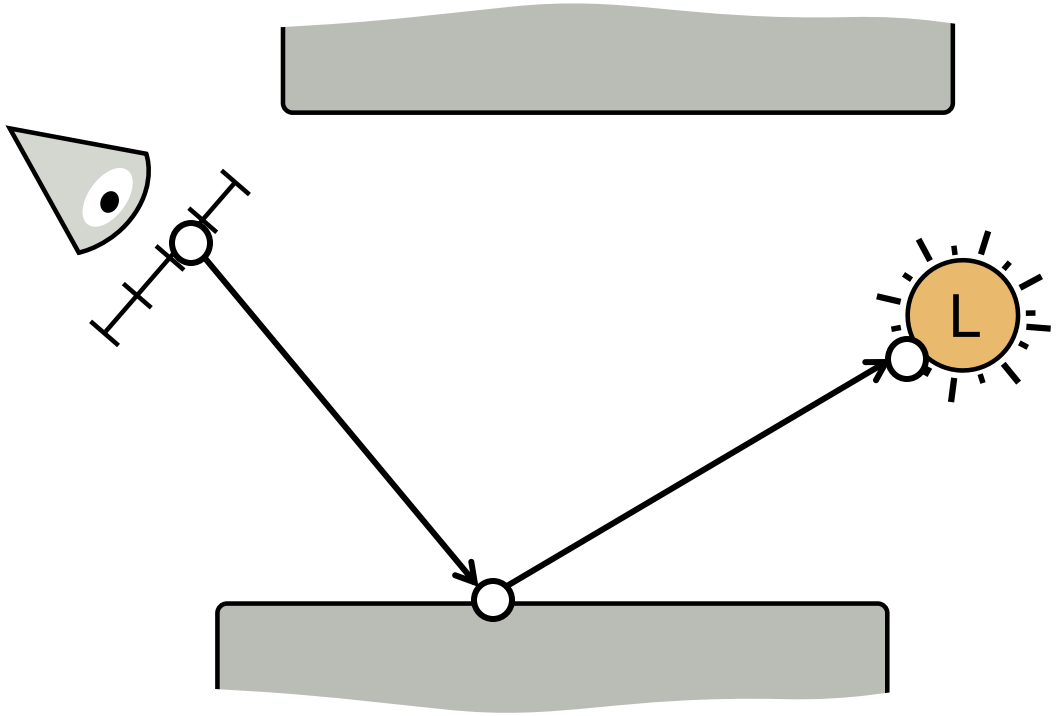
Path Tracing

- Direct illumination



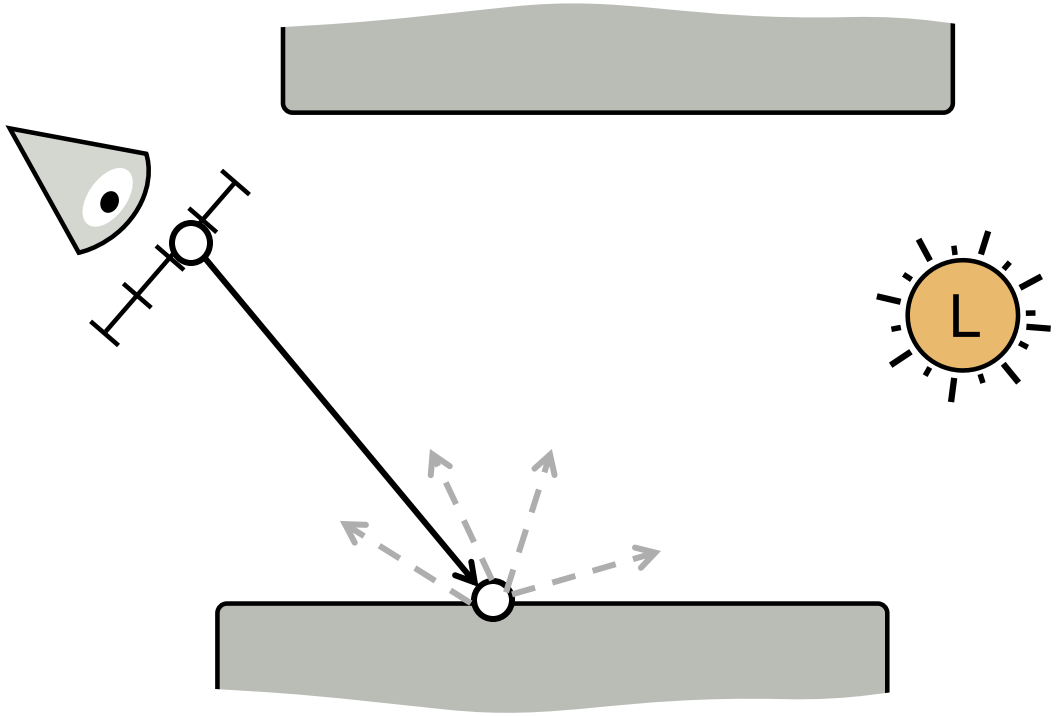
Path Tracing

- Direct illumination



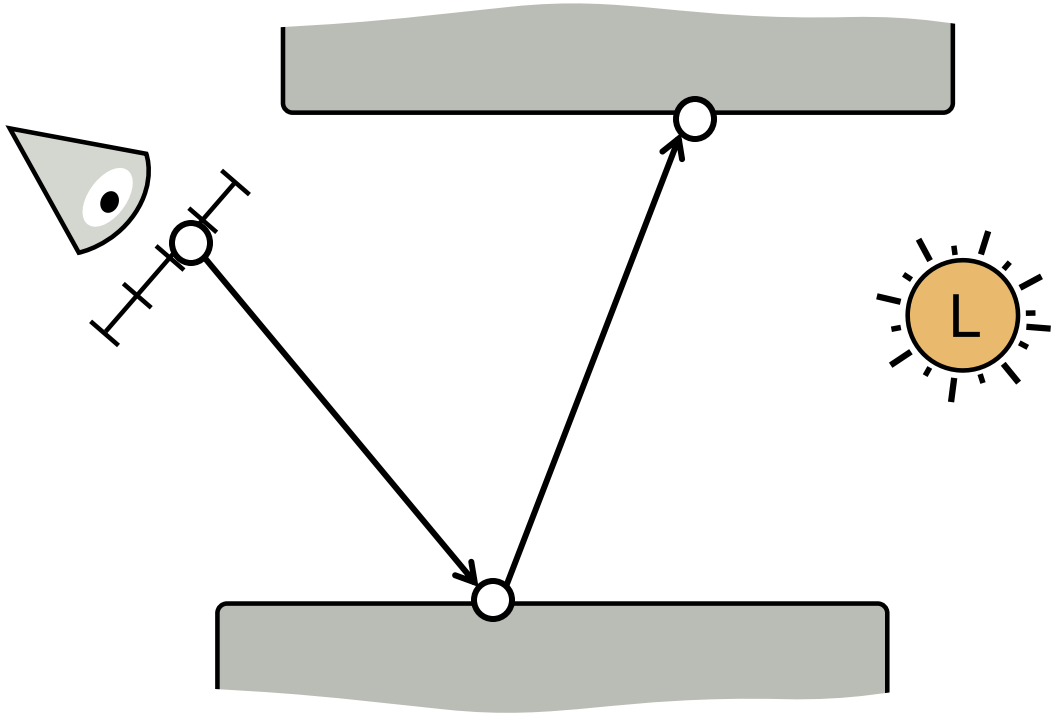
Path Tracing

- Direct illumination



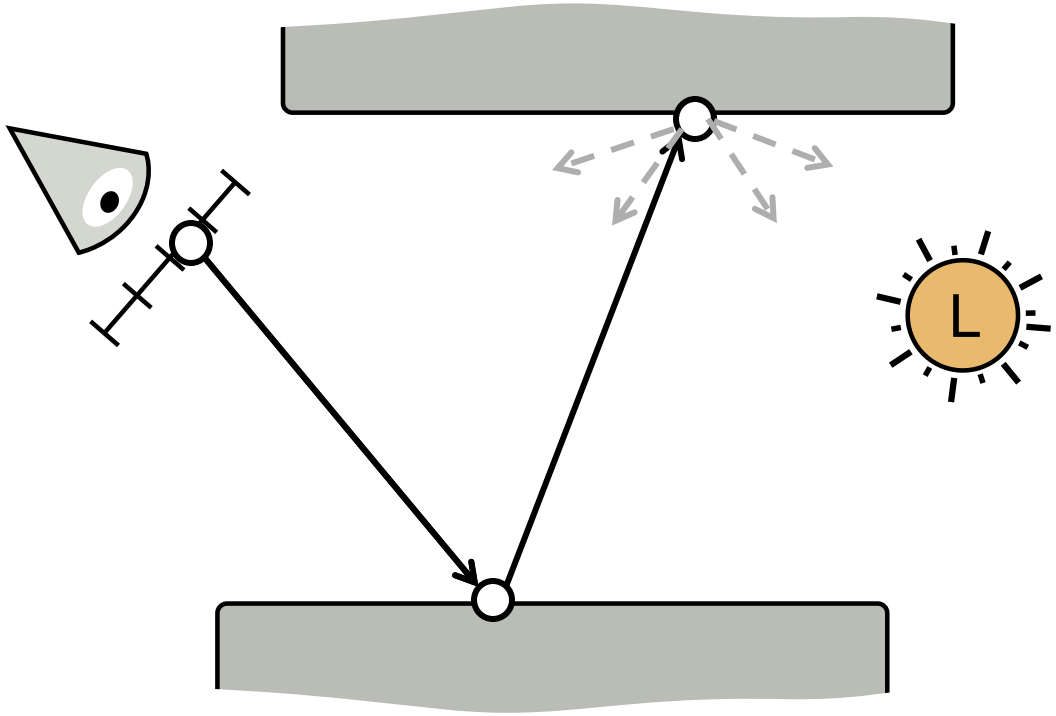
Path Tracing

- Second bounce illumination



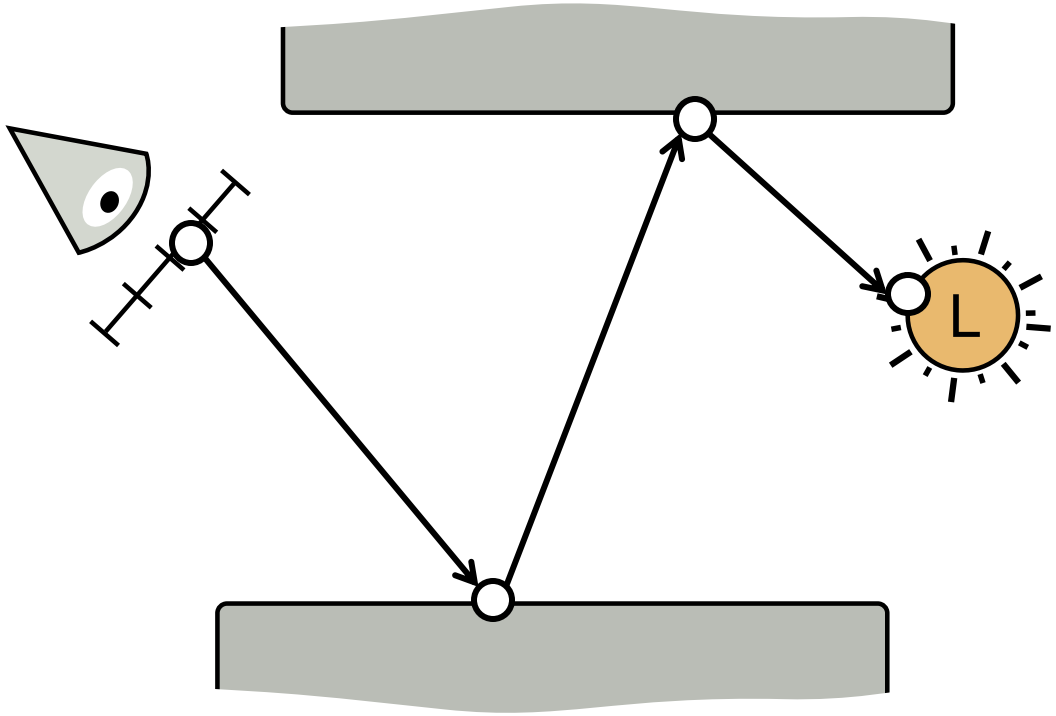
Path Tracing

- Second bounce illumination



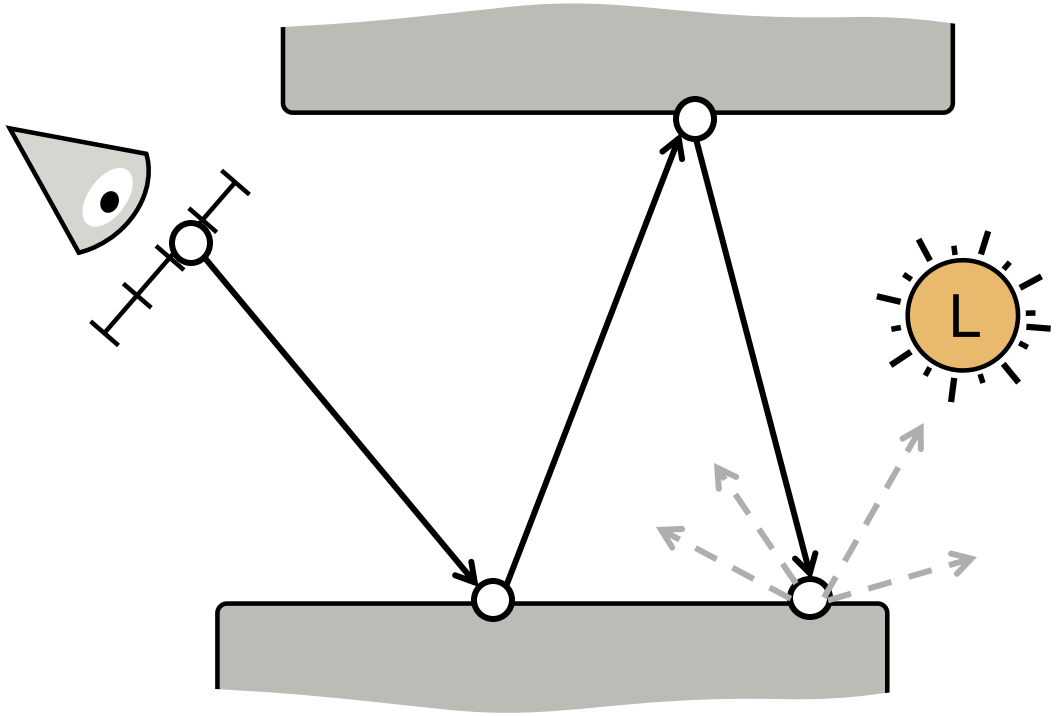
Path Tracing

- Second bounce illumination



Path Tracing

- Third bounce illumination …

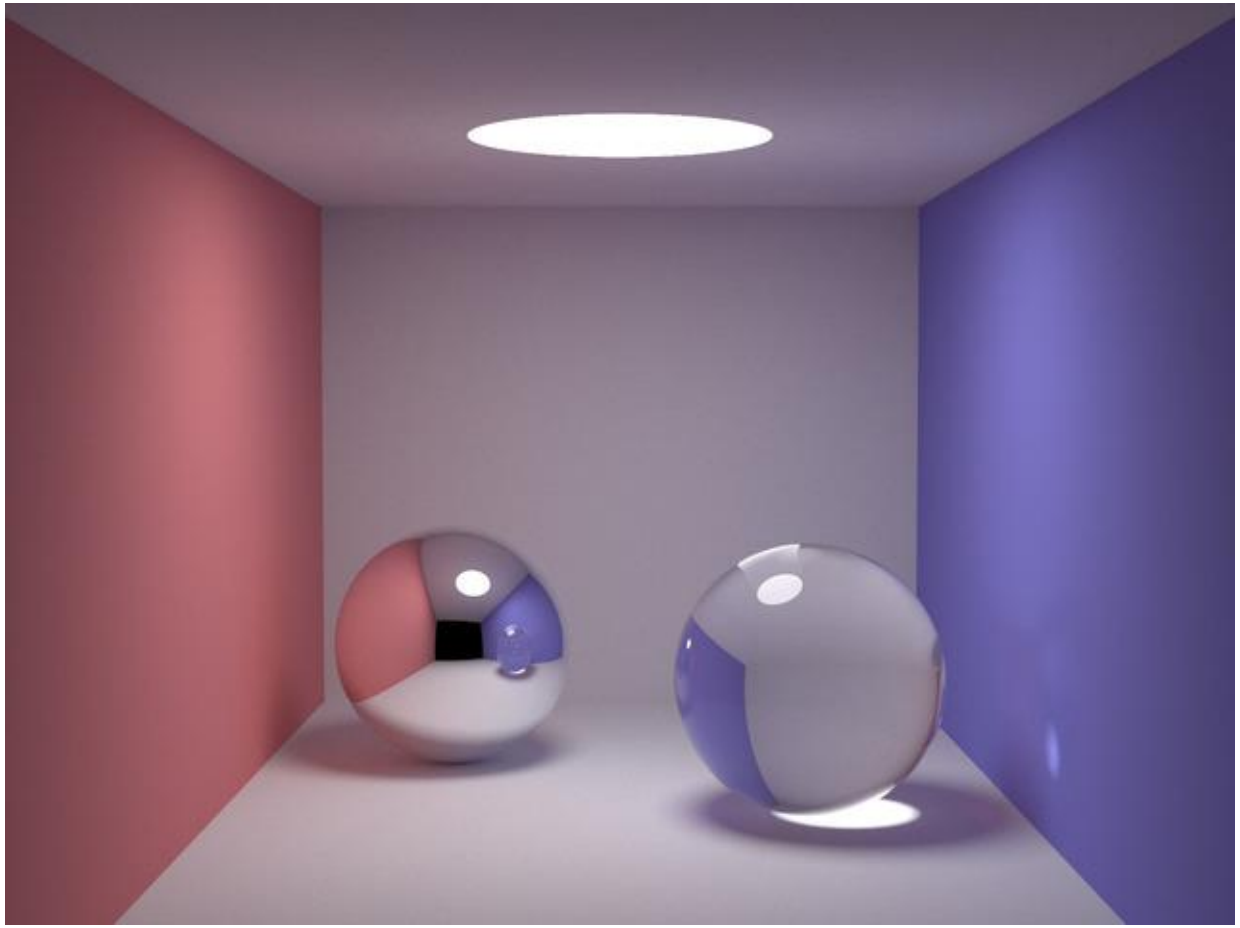


Path Tracing: Brute-force Recursion

- **Step 0**: Start by selecting a pixel. Trace a ray from the camera origin to the pixel
- **Step 1**: Determine the intersection of the ray with the scene objects.
  If the ray hits the light source, return the light intensity. Else go to step 2.
- **Step 2**: At the intersection, randomly select a direction and create a new ray from the intersection.
  Get the radiance of the new ray (recursion to Step 1) and compute the reflected radiance at the intersection.

Path Tracing in 99 Lines of C++ Code

- A simple demo on path tracing implementation
- Ray-sphere intersection
- Lambertian walls
- Mirror spheres
- Transparent spheres
- Global illumination
- Soft shadows



https://www.kevinbeason.com/smallpt/

State-of-the-art Computer Graphics

State-of-the-art Computer Graphics

- A short introduction to modern methods in computer graphics
- Modern software and tools used in the graphics industry.
- Latest research: generative AI x 3D graphics.

Direct3D 12, Metal, Vulkan

- New APIs for platform-specific implementation of 3D graphics: Windows and Macs.

- Vulkan is a cross-platform graphics API. Initially referred to as the next OpenGL.

WebGL / Three.js

- Graphics in the web browser.
- **WebGL** for running a subset of OpenGL in the browser!
- **Three.js** is a wrapper library for easy use of WebGL.

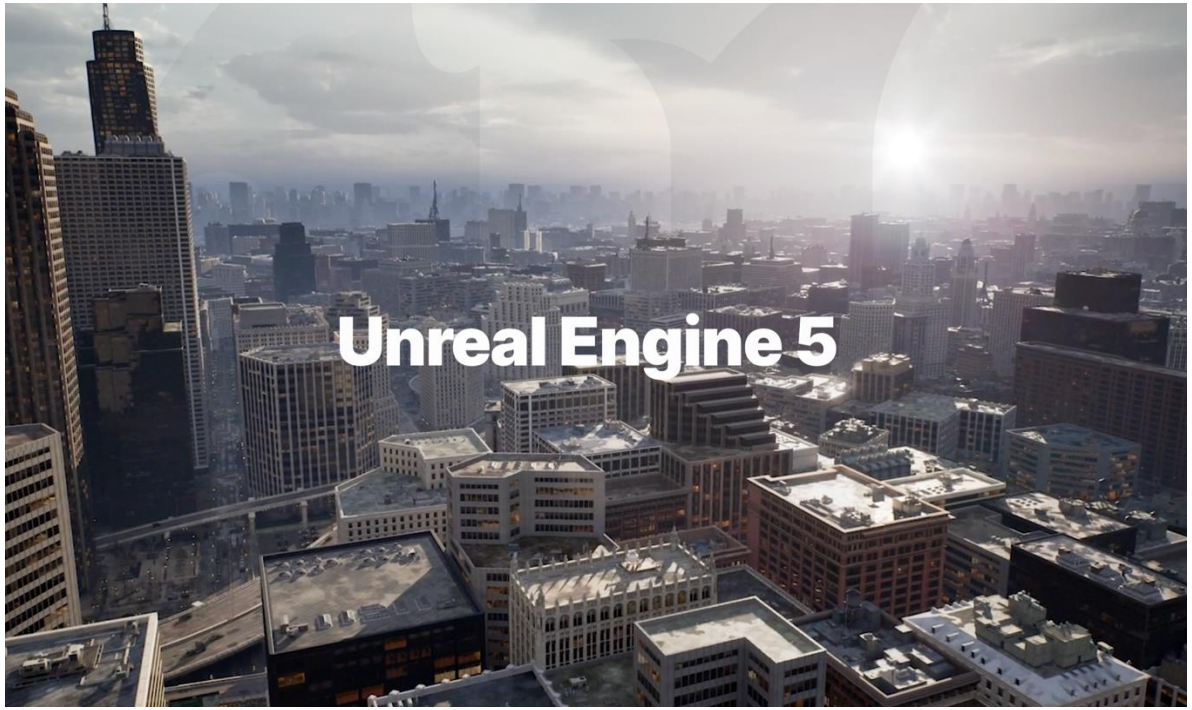
https://threejs.org/examples/#webgl_animation_keyframes

WebGPU

- Successor to WebGL.
- Directly use system's native graphics API (Direct3D 12, Metal, Vulkan).
- Similar graphics pipeline concepts:
  vertex buffer, vertex shader, fragment shader.
- More at:
  https://webgpu.github.io/webgpu-samples/
  https://github.com/gpuweb/gpuweb (spec)

Game Engines

- Software libraries for developing real-time games
- Heavily based on graphics: 2D and 3D.
- Include other libraries for physics, audios, controls, networks.



Physics Engine

- Real-time physics simulation
- Discrete and continuous collision detection
- Rigid body simulation
- Soft body simulation

Havok is founded in 1998 by
Hugh Reynolds and Steven Collins from
Trinity College Dublin!

Generative AI

A blue jay standing on a large basket of rainbow macarons.

Credit: ProlificDreamer

Generative AI and 3D Graphics

- Automatically generate a 3D model by inputing “an ice cream sundae”.

Text-to-3D Generation

A flower in a hand crafted vase

Tran et al., DiverseDream: Diverse Text-to-3D Synthesis with Augmented Text Embedding, ECCV 2024

Text-to-3D Generation with References

A 3D model of an adorable cottage with a thatched roof

A birthday cake rendered in 3D

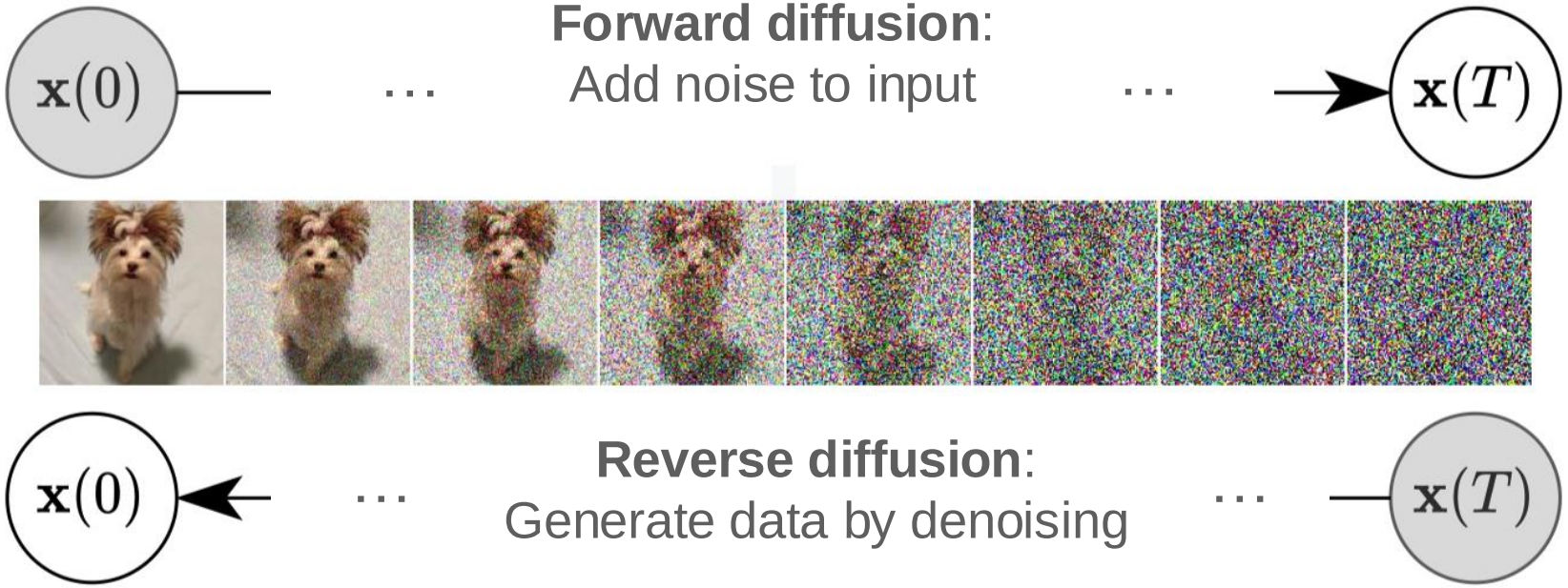
DiverseDream, ECCV 2024

“Flying dragon”

Image-to-3D Generation

Voleti et al., SV3D: Novel Multi-view Synthesis and 3D Generation from a Single Image using Latent Video Diffusion
https://sv3d.github.io/ and https://huggingface.co/stabilityai/sv3d ECCV 2024

Diffusion Models



- Forward diffusion:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$$

- Reverse diffusion:

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

Denoising U-Net

Training by minimizing $\mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0),\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I}),t\sim\mathcal{U}(1,T)}\left[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon}_{\mathbf{x}_t}, t)||^2\right]$

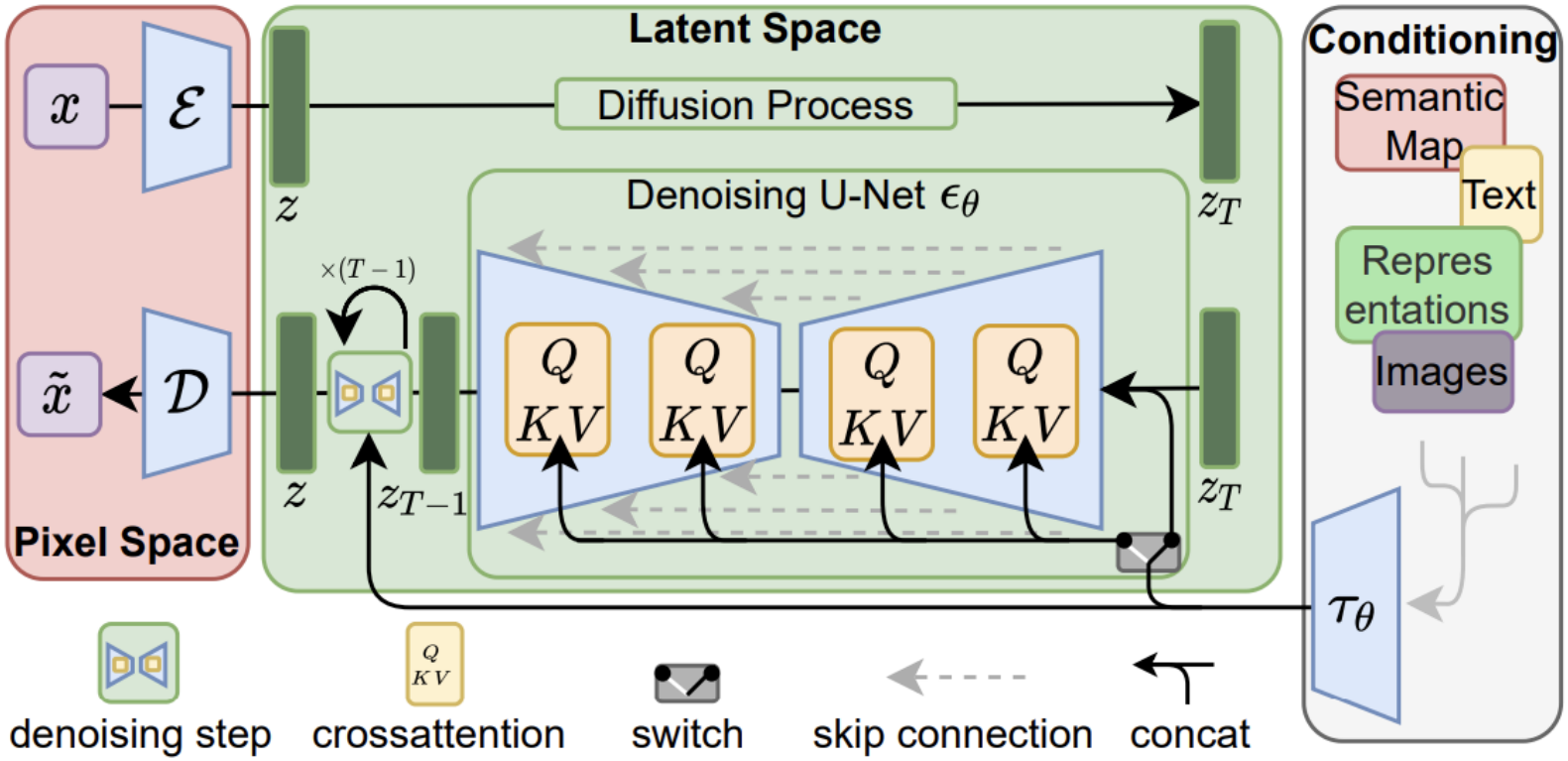
Noise prediction network

Sohl-Dickstein et al., 2015;
Ho et al., 2020; Song et al., 2021

Latent Diffusion

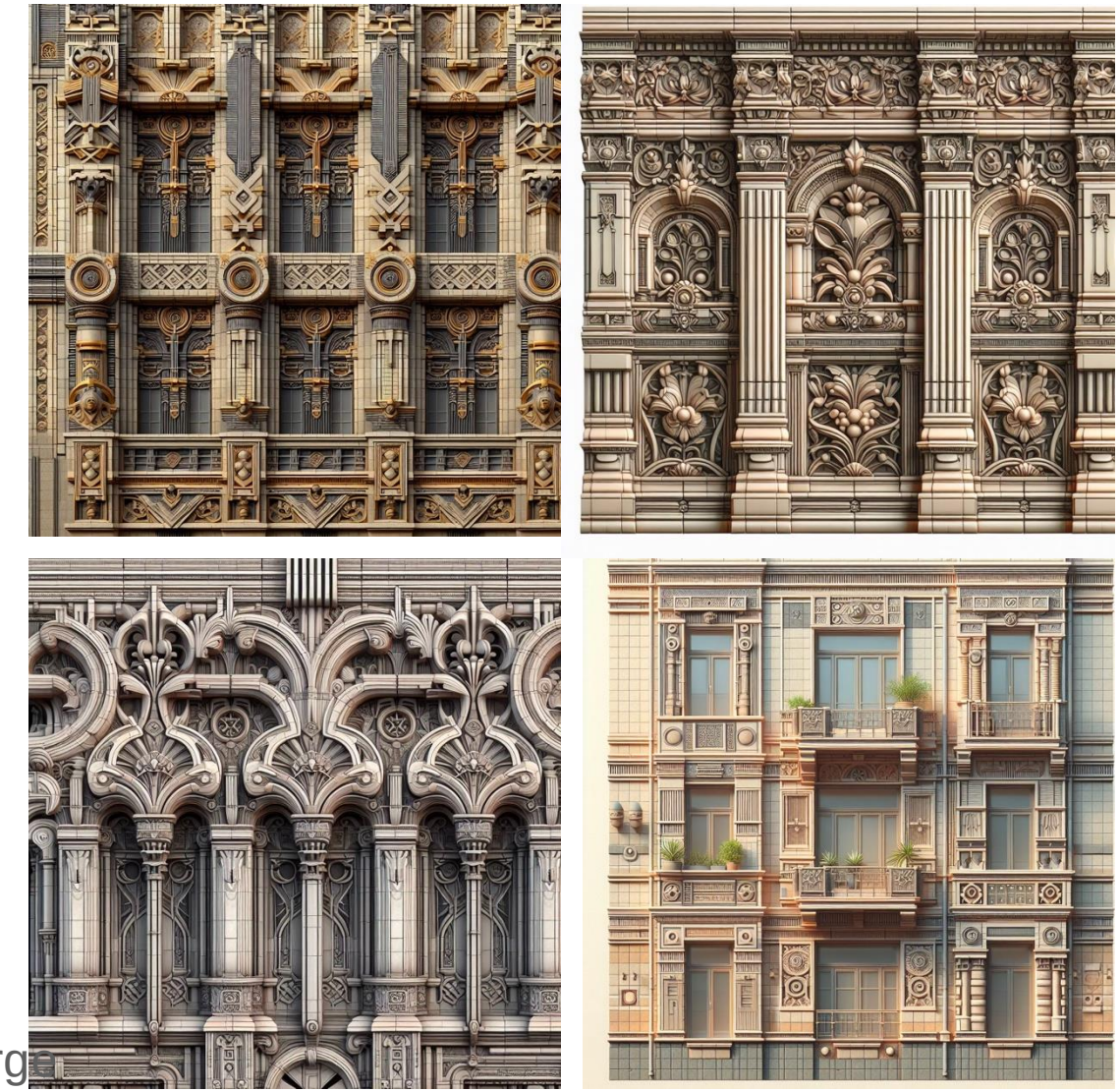
(Stable Diffusion)

- Shift away from diffusion defined in pixel space.
- Diffusion in a latent space defined by an encoder and a decoder network.
- Cross-attention mechanism added to the denoising U-Net.



Demo: https://huggingface.co/spaces/stabilityai/stable-diffusion

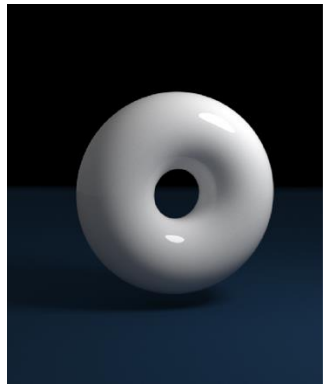
Text-to-Image Generation



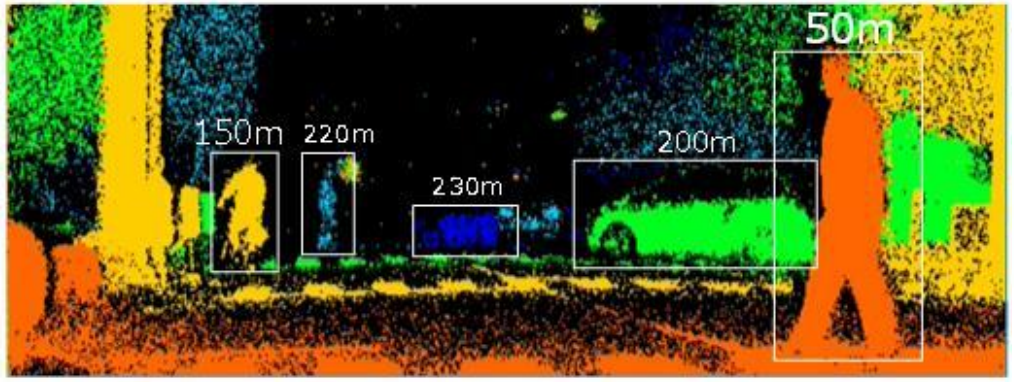
A photo of a large, grinning, twelve-legged cat with a large bushy tail and a hollow body that serves as a bus, with windows and seats covered with fur (Studio Ghibli)

"A texture map for building facade"

3D Shape Representation

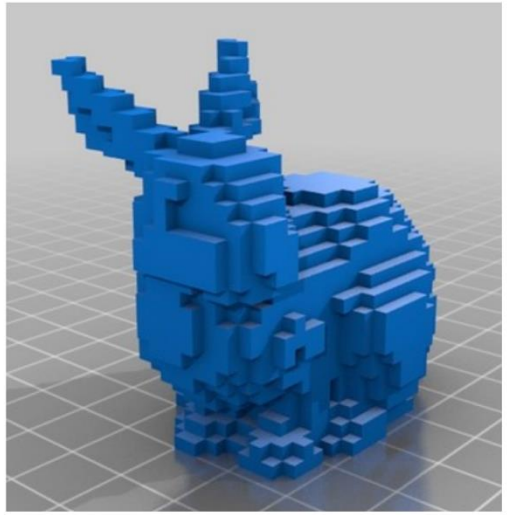


Parametric

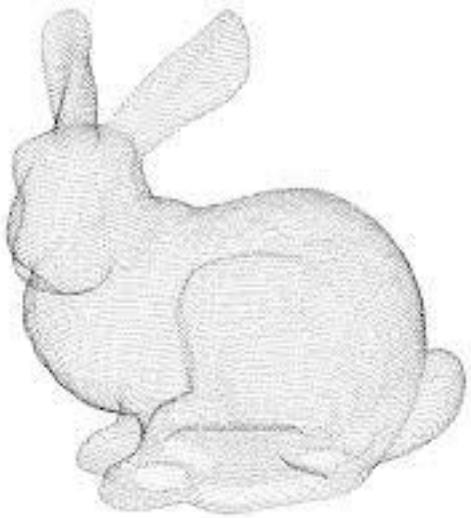


Depth

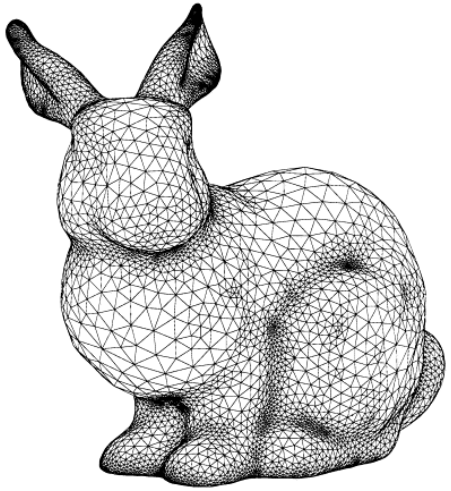
Multiple images



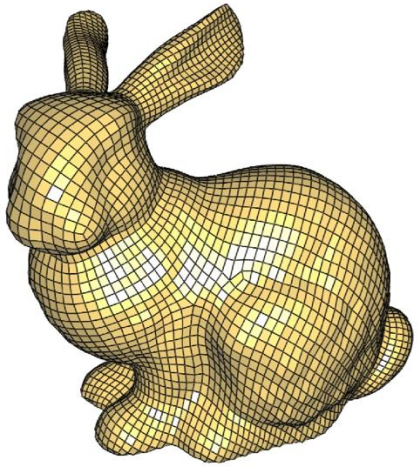
Voxel



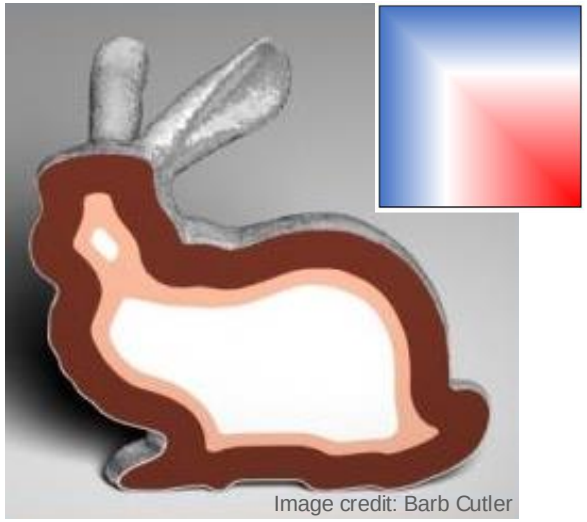
Point cloud



Triangle mesh

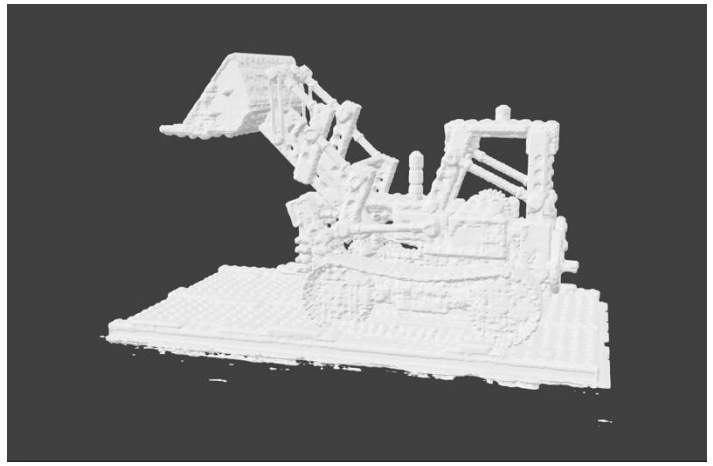


Quad mesh



Implicit surface

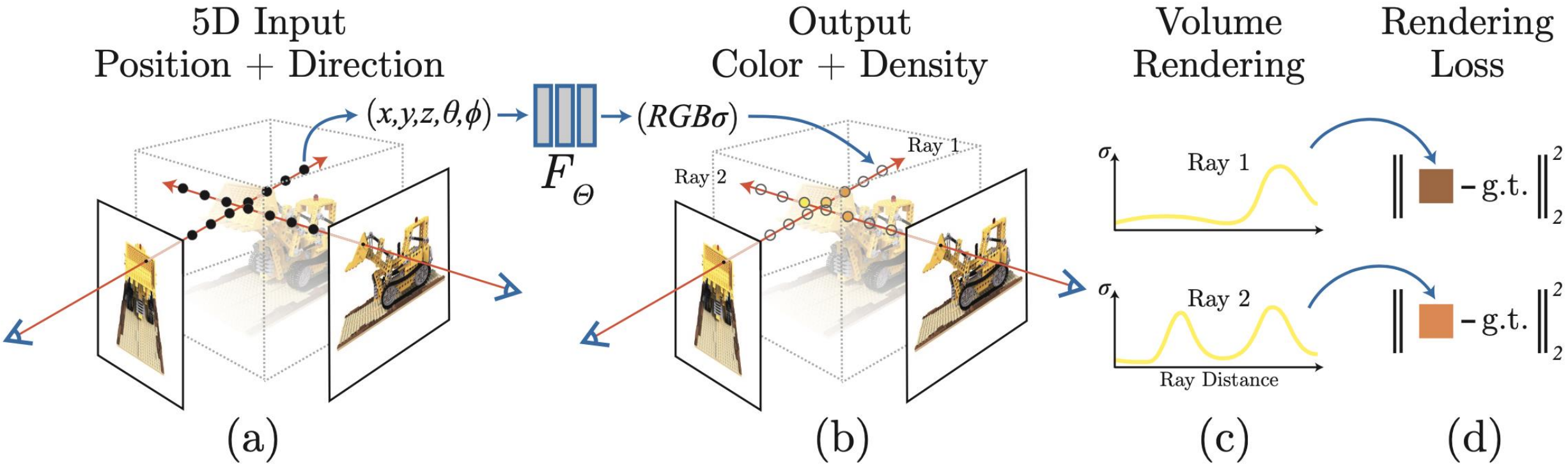
Neural Radiance Fields



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis
Mildenhall et al., ECCV 2020

Neural Radiance Field

- Learn a neural network that predicts a radiance field by mapping (x, y, z, theta, phi) to color and opacity. Render the radiance field to images.



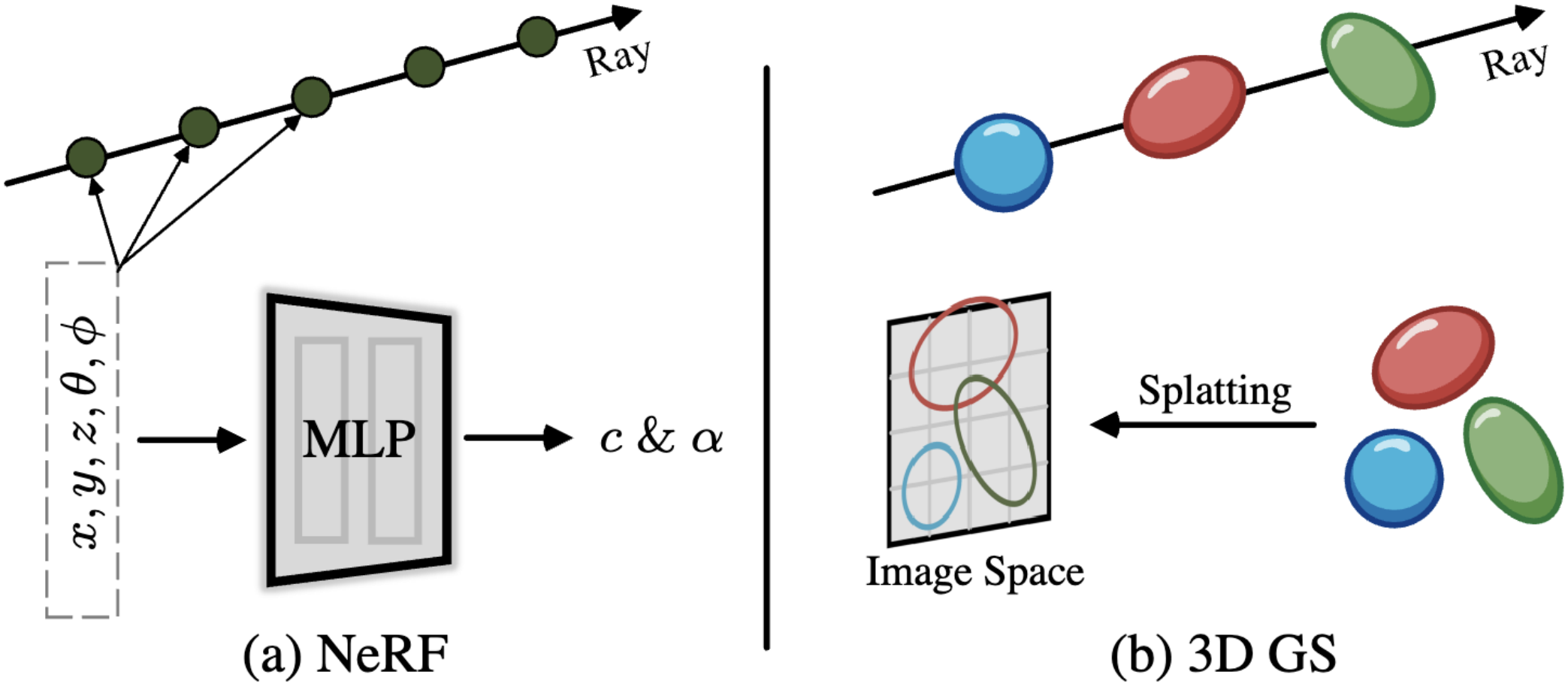
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis
Mildenhall et al., ECCV 2020

Volume Rendering

$$\underbrace{C(\overbrace{\mathbf{r}}^{\text{ray}})}_{\text{color}} = \int_{t_n}^{t_f} T(t)\underbrace{\sigma(\mathbf{r}(t))}_{\text{density}}\underbrace{\mathbf{c}(\mathbf{r}(t), d)}_{\text{radiance}}dt$$

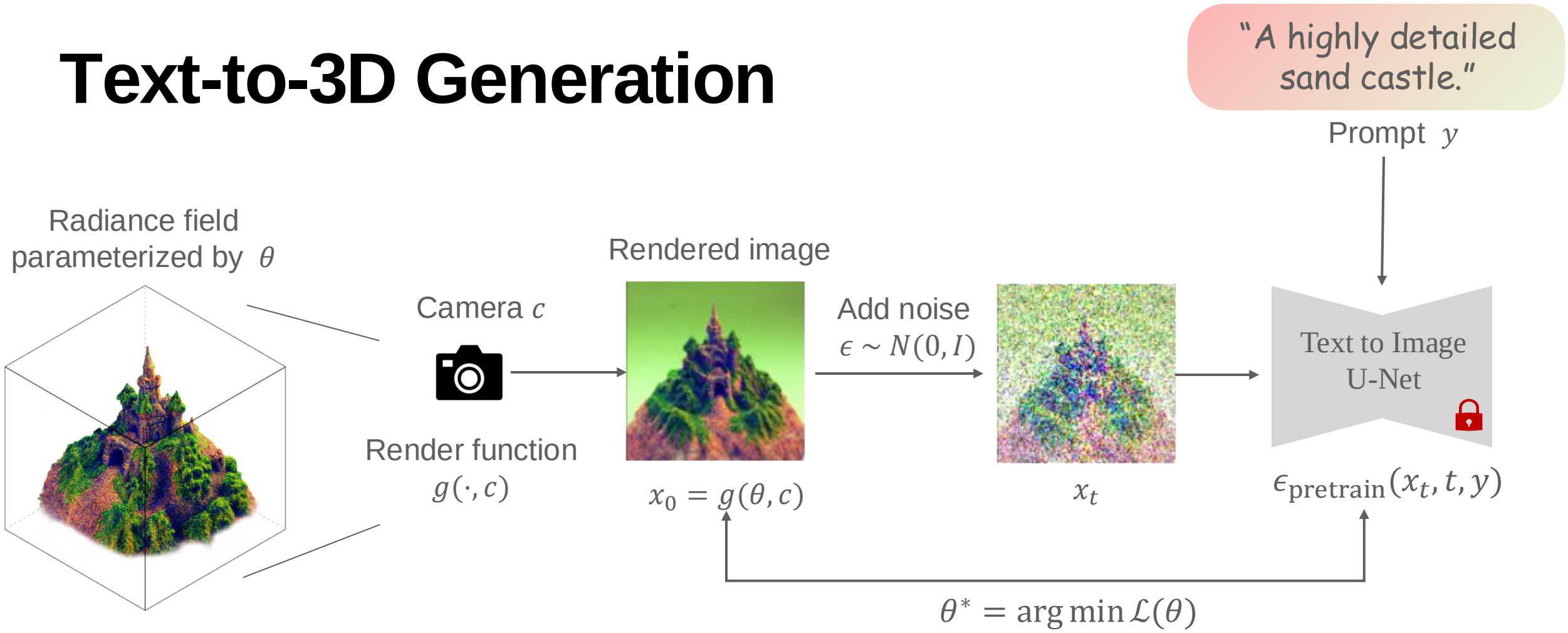
$$\underbrace{T(t)}_{\text{transmittance}} = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

3D Gaussian Splatting



Chen and Wang, A Survey on 3D Gaussian Splatting, 2024

Text-to-3D Generation



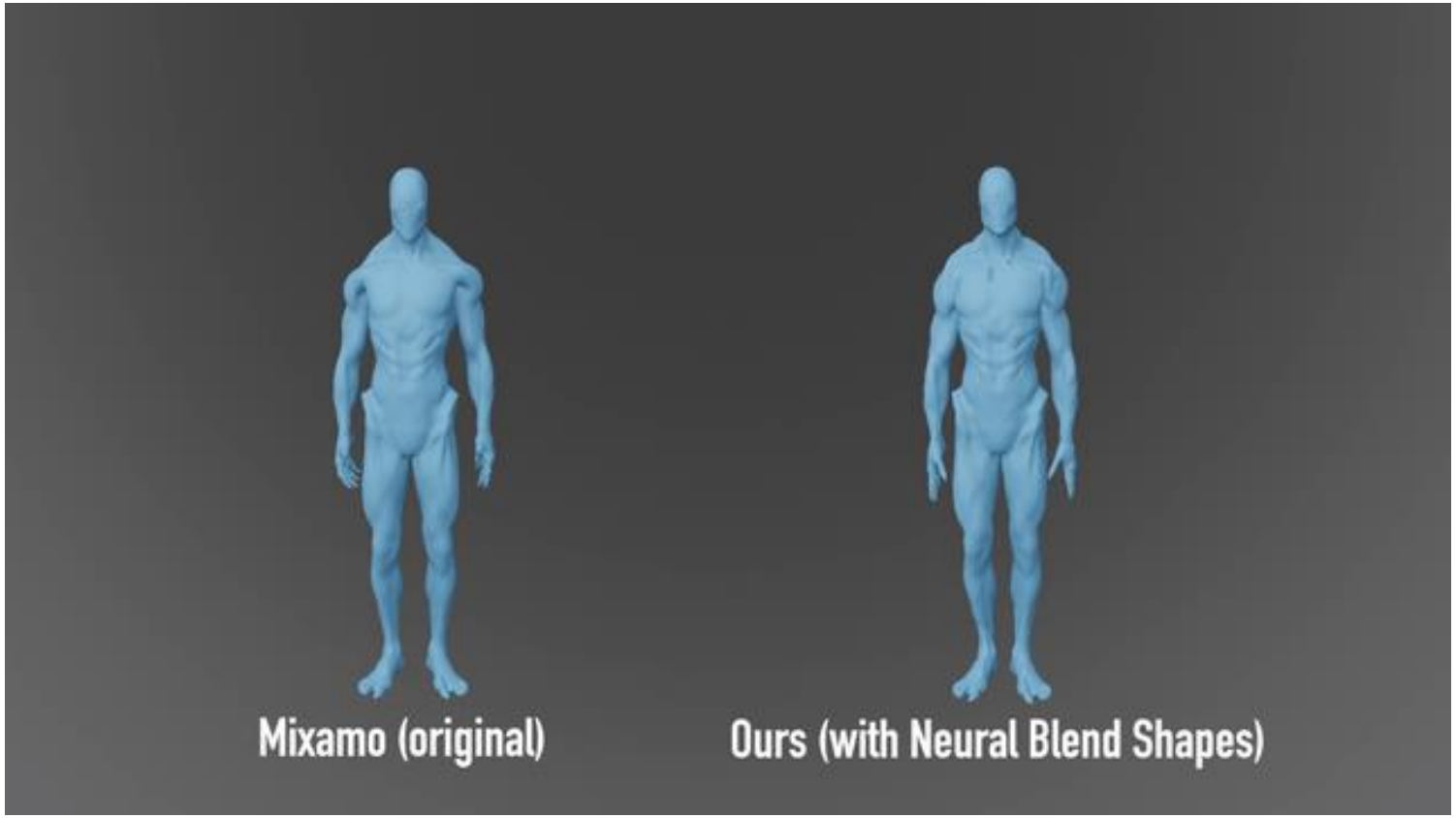
Score distillation

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\theta) \approx \mathbb{E}_{t,\boldsymbol{\epsilon},c}\left[\omega(t)(\boldsymbol{\epsilon}_{\text{pretrain}}(\boldsymbol{x}_t, t, y^c) - \boldsymbol{\epsilon})\frac{\partial \boldsymbol{g}(\theta, c)}{\partial \theta}\right]$$

Credit: ProlificDreamer

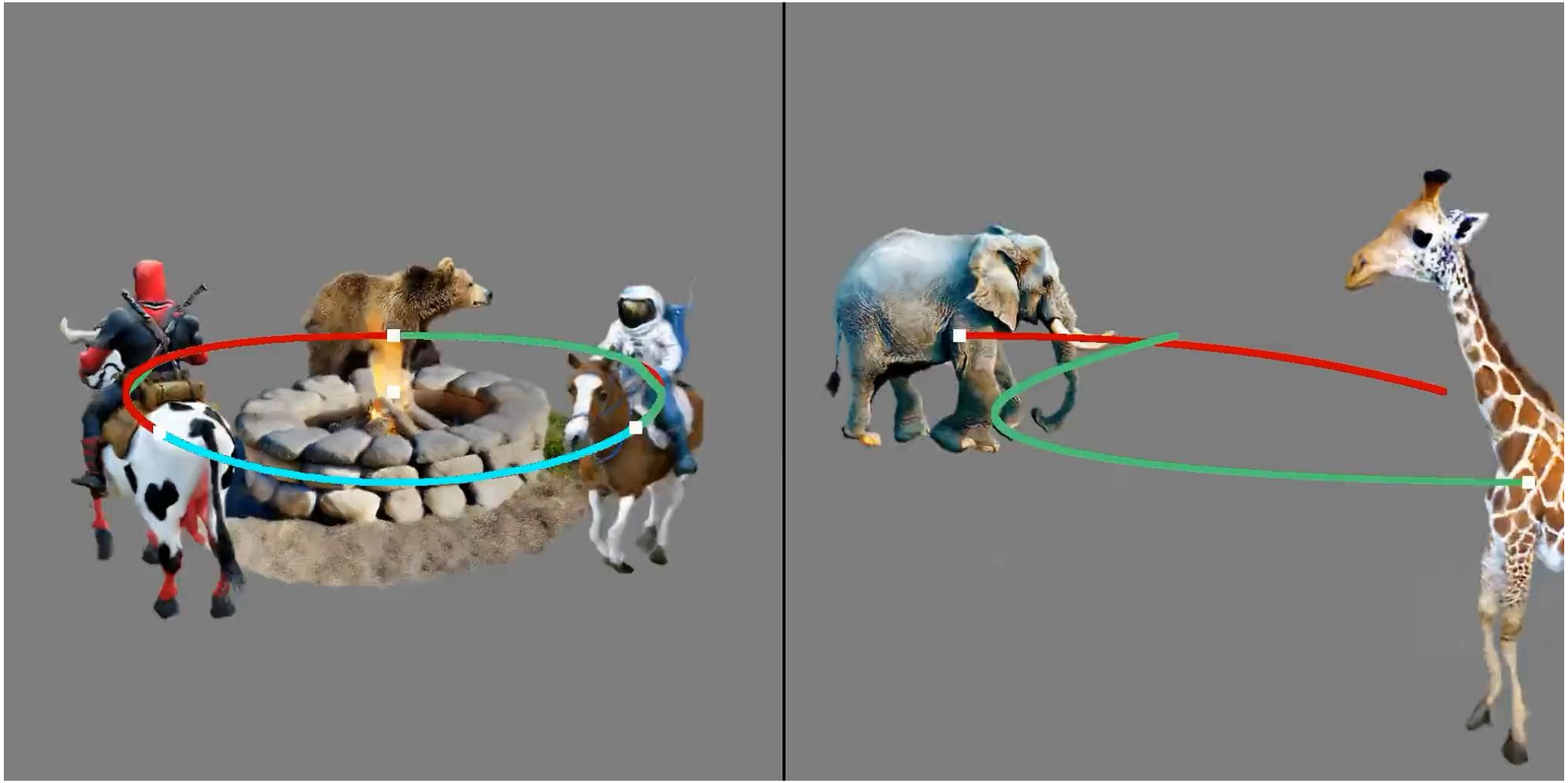
Neural Blend Shapes

- Automatic character rigging, skinning, and blend shapes generation



Learning Skeletal Articulations with Neural Blend Shapes, SIGGRAPH 2021

Text-to-4D Generation



Bahmani et al., TC4D: Trajectory-Conditioned Text-to-4D Generation, ECCV 2024

Module Recap

Modeling + Rendering + Animation!

OpenGL programming in C++

Potential next steps:

Advanced real-time computer graphics

Generative AI + computer graphics