Rendering Architecture
of Fools Engine

Work in progress

9 – Integration Layer

Render Director	Scene Renderer	UI Renderer

8 – Render Layer

Geometry Renderer	Illumination Renderer	Forward Renderer	VFX Renderer

7 – Graph Layer

Render Pipeline	Render Fiber	Render Pass

6 – Resources Layer

Render Asset Manager	Material Library	Texture Library	Shader Library	Mesh Library

5 – Communication Layer

Asset Loader	Command Queue	Command List	Swap Chain

4 - GDI Isolation Layer

Render Device	Render Commands

3 - Representation Layer

Camera	Transform	Material	Mesh	Basic Shapes	Coordinates

2 - GDI Abstraction Layer

Device API	Rendering Context	Shader	Texture	Vertex Buffer	Index Buffer	Uniform Buffer	Frame Buffer

1 – Primitives Layer

GDIType	Shader Data	Uniform	Shader Texture Slot	Texture Data	Buffer Data	Frame Buffer Data

0 - Graphics Device Interface

OpenGL	Vulkan	DirectX3D	Metal

Redeferred rendering – Experimental concept

Let's split diffused and specular contributions of the rendering equation

$$L_o(p, \omega_o) =$$

$$= \sum_i \left(k_d \frac{c}{\pi} + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)} \right) L_i \, (n \cdot \omega_i) =$$

$$= \sum_i \left(k_d \frac{c}{\pi} L_i \, (n \cdot \omega_i) + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)} L_i \, (n \cdot \omega_i) \right) =$$

$$= \sum_i \left(k_d \frac{c}{\pi} L_i \, (n \cdot \omega_i) + \frac{DFG}{4(\omega_o \cdot n)} L_i \right) =$$

$$= \sum_i k_d \frac{c}{\pi} L_i \, (n \cdot \omega_i) + \sum_i \frac{DFG}{4(\omega_o \cdot n)} L_i$$

$$L_d(p, \omega_o) = \sum_i k_d \frac{c}{\pi} L_i \, (n \cdot \omega_i)$$

$$L_s(p, \omega_o) = \sum_i \frac{DFG}{4(\omega_o \cdot n)} L_i$$

$$L_o(p, \omega_o) = L_d(p, \omega_o) + L_s(p, \omega_o)$$

We're going to store them in dedicated variables while calculating them in a per light loop.

Then we are going to add them back together outside the loop.

Diffused

$$L_d(\mathrm{p}, \omega_o) = \sum_i k_d \frac{c}{\pi} L_i (n \cdot \omega_i) = \frac{c}{\pi} \sum_i k_d L_i (n \cdot \omega_i) =$$

$$= \frac{c}{\pi} \sum_i (1 - k_s)(1 - m) L_i (n \cdot \omega_i) =$$

$$= (1 - m) \frac{c}{\pi} \sum_i (1 - k_s) L_i (n \cdot \omega_i) \approx$$

$$\approx (1 - m) \frac{c}{\pi} \sum_i \left(1 - (F_0 + (1 - F_0)P_i)\right) L_i (n \cdot \omega_i) =$$

$$= (1 - m) \frac{c}{\pi} \sum_i (1 - F_0 - (1 - F_0)P_i) L_i (n \cdot \omega_i) =$$

$$= (1 - m) \frac{c}{\pi} \sum_i \left((1 - F_0) - (1 - F_0)P_i\right) L_i (n \cdot \omega_i) =$$

$$= (1 - m) \frac{c}{\pi} \sum_i \left((1 - F_0)(1 - P_i)\right) L_i (n \cdot \omega_i) =$$

$$= (1 - m) \frac{c}{\pi} (1 - F_0) \sum_i (1 - P_i) L_i (n \cdot \omega_i)$$

$$L_{d_1} = \sum_i (1 - P_i) L_i (n \cdot \omega_i)$$

$$L_d(\mathrm{p}, \omega_o) \approx (1 - m) \frac{c}{\pi} (1 - F_0) L_{d_1}$$

Specular

$$L_s(\mathrm{p}, \omega_o) = \sum_i \frac{FDG}{4(\omega_o \cdot n)} L_i = \frac{1}{4(\omega_o \cdot n)} \sum_i DGL_i F \approx$$

$$\approx \frac{1}{4(\omega_o \cdot n)} \sum_i DGL_i (F_0 + (1 - F_0)P_i) =$$

$$= \frac{1}{4(\omega_o \cdot n)} \sum_i (DGL_i F_0 + DGL_i P_i (1 - F_0)) =$$

$$= \frac{1}{4(\omega_o \cdot n)} \left(\sum_i DGL_i F_0 + \sum_i DGL_i P_i (1 - F_0) \right) =$$

$$= \frac{1}{4(\omega_o \cdot n)} \left(F_0 \sum_i DGL_i + (1 - F_0) \sum_i DGL_i P_i \right)$$

$$L_{s_1} = \sum_i DGL_i$$

$$L_{s_2} = \sum_i DGL_i P_i$$

$$L_s(\mathrm{p}, \omega_o) \approx \frac{1}{4(\omega_o \cdot n)} \left(F_0 L_{s_1} + (1 - F_0) L_{s_2} \right)$$

Fresnel-Schlick approximation:

$$k_s = F = F_0 + (1 - F_0)\left(1 - (h_i \cdot v)\right)^5$$

$$P_i = \left(1 - (h_i \cdot v)\right)^5$$

$$k_s = F = F_0 + (1 - F_0)P_i$$

Recombination

$$L_{d_1} = \sum_i (1 - P_i) L_i (n \cdot \omega_i)$$

$$L_{s_1} = \sum_i DGL_i \qquad\qquad P_i = \left(1 - (h_i \cdot v)\right)^5$$

$$L_{s_2} = \sum_i DGL_i P_i$$

$$L_d(p, \omega_o) \approx (1 - m) \frac{c}{\pi} (1 - F_0) L_{d_1}$$

$$L_s(p, \omega_o) \approx \frac{1}{4(\omega_o \cdot n)} \left(F_0 L_{s_1} + (1 - F_0) L_{s_2}\right)$$

$$L_o(p, \omega_o) = L_d(p, \omega_o) + L_s(p, \omega_o)$$

We are going to calculate $L_{s_1}, L_{s_2}, L_{d_1}$ inside the lights loop
We are going to calculate L_s, L_d and finally L_o outside the loop

A portion of calculations is now pushed out of the loop and performed only once, instead of on a per light basis.

Metalness and albedo are now not accessed inside the loop at all, what reduces memory congestion.

Depending on chosen approximations of normal distribution function and geometry function additional elements could be pulled out of the loop (e.g. $\frac{\alpha^2}{\pi}$ can be pulled out of Trowbridge-Reitz GGX approximation of NDF) although not fully.

Redeffered rendering

$$P_i = \left(1 - (h_i \cdot v)\right)^5$$

$$L_{d_1} = \sum_i (1 - P_i) L_i (n \cdot \omega_i)$$

$$L_{s_1} = \sum_i DGL_i$$

$$L_{s_2} = \sum_i DGL_i P_i$$

$$L_d(\mathrm{p}, \omega_o) \approx (1 - m)\frac{c}{\pi}(1 - F_0)L_{d_1}$$

$$L_s(\mathrm{p}, \omega_o) \approx \frac{1}{4(\omega_o \cdot n)}\left(F_0 L_{s_1} + (1 - F_0)L_{s_2}\right)$$

$$L_o(\mathrm{p}, \omega_o) = L_d(\mathrm{p}, \omega_o) + L_s(\mathrm{p}, \omega_o)$$

Lets split shading into two stages.

First stage will produce L_{d_1}, L_{s_1}, L_{s_2} values and store them in a dedicated buffer – I-buffer.
Second stage will use those values to calculate L_o.

I-buffer retains fundamental characteristic of light to be cumulative. This allows first step to be split into multiple passes using specialized shaders for different types of lights and shadows. Redeferred rendering can be used analogously with most of the indirect illumination techniques too and their results can be accumulated to the same I-buffer with a common second stage to be applied.

First stage depends only on lights and geometry (from PBR point of view roughness is a simplified representation of subpixel geometry, not a property of a substance). The variability of output buffer is expected to be low (imagine white world with colorful lights). That makes this stage a good candidate for shading rate reducing technique (lower overall resolution of the buffer / VRS / Deferred Adaptive Compute Shading / etc.). We can safely apply reduced shading rate everywhere unless there is an illumination discontinuity (hard shadow / geometry edge) from a particular illumination source – this will require visual confirmation about quality.

Second stage does not have to iterate over illumination sources, so it will be used to refine results by integrating them with materials at full resolution. It will fully cover artifacts of reduced shading rate from first pass, making them completely imperceptible. If roughness is pulled out of Trowbridge-Reitz GGX approximation of NDF, then we have all parameters of a given pixel from G-buffer taken into consideration during refining second pass.

Redeffered rendering - considerations

$$P_i = \left(1 - (h_i \cdot v)\right)^5$$

$$L_{d_1} = \sum_i (1 - P_i) L_i (n \cdot \omega_i)$$

$$L_{s_1} = \sum_i DGL_i$$

$$L_{s_2} = \sum_i DGL_i P_i$$

In case memory throughput becomes significant bottleneck:

A. Calculating $\frac{1}{F_0}\left(F_0 L_{s_1} + (1 - F_0)L_{s_2}\right)$ in first pass, storing results as a 1 color instead of 2 and multiplying back by F_0 in second pass (F_0 is different now, because is taken from a different sampling point – full resolution second pass vs reduced shading rate in first pass).

B. Precalculating $(\omega_o \cdot n)$ in geometry pass – no need for normal and depth in second pass

C. Smart combined packing of I-Buffer and G-Buffer

$$L_d(\mathrm{p}, \omega_o) \approx (1 - m)\frac{c}{\pi}(1 - F_0)L_{d_1}$$

$$L_s(\mathrm{p}, \omega_o) \approx \frac{1}{4(\omega_o \cdot n)}\left(F_0 L_{s_1} + (1 - F_0)L_{s_2}\right)$$

$$L_o(\mathrm{p}, \omega_o) = L_d(\mathrm{p}, \omega_o) + L_s(\mathrm{p}, \omega_o)$$

In case of light bleeding over edges and aliased shadow edges due to greater aliasing in I-Buffer then in G-Buffer:

A. Using VRS with dedicated shading rate attachments per illumination aspect (direct, shadows, reflections, indirect, etc.)

B. Using different shading rates per illumination aspect