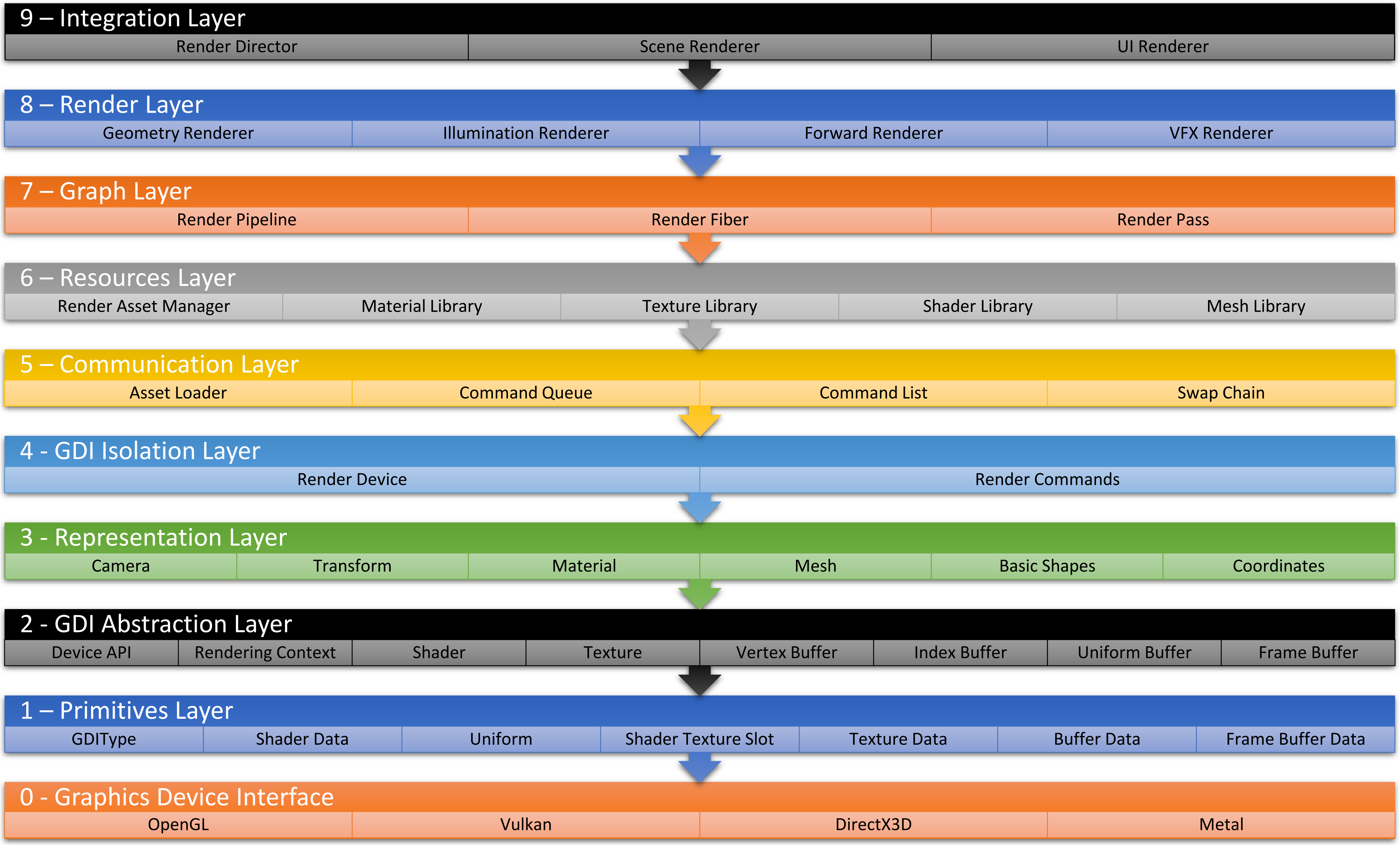
Rendering Architecture of Fools Engine

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

$$
\begin{aligned}
&\mathrm{L}_o(\mathrm{p}, \omega_o) = \\
&= \sum_i \left(k_d \frac{c}{\pi} + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)} \right) L_i\ (n \cdot \omega_i) = \\
&= \sum_i \left(k_d \frac{c}{\pi} L_i\ (n \cdot \omega_i) + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)} L_i\ (n \cdot \omega_i) \right) = \\
&= \sum_i \left(k_d \frac{c}{\pi} L_i\ (n \cdot \omega_i) + \frac{DFG}{4(\omega_o \cdot n)} L_i \right) = \\
&= \sum_i k_d \frac{c}{\pi} L_i\ (n \cdot \omega_i) + \sum_i \frac{DFG}{4(\omega_o \cdot n)} L_i \\
&\mathrm{L}_d(\mathrm{p}, \omega_o) = \sum_i k_d \frac{c}{\pi} L_i\ (n \cdot \omega_i) \\
&\mathrm{L}_s(\mathrm{p}, \omega_o) = \sum_i \frac{DFG}{4(\omega_o \cdot n)} L_i \\
&\mathrm{L}_o(\mathrm{p}, \omega_o) = L_d(\mathrm{p}, \omega_o) + L_s(\mathrm{p}, \omega_o)
\end{aligned}
$$

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

$$\mathrm{L}_d(\mathrm{p},\omega_o) = \sum_i k_d \frac{c}{\pi} L_i\,(n\cdot\omega_i) = \frac{c}{\pi}\sum_i k_d L_i(n\cdot\omega_i) =$$
$$= \frac{c}{\pi}\sum_i (1-k_s)(1-m)L_i(n\cdot\omega_i) =$$
$$= (1-m)\frac{c}{\pi}\sum_i (1-k_s)L_i(n\cdot\omega_i) \approx$$
$$\approx (1-m)\frac{c}{\pi}\sum_i \big(1-(F_0+(1-F_0)P_i)\big)L_i(n\cdot\omega_i) =$$
$$= (1-m)\frac{c}{\pi}\sum_i (1-F_0-(1-F_0)P_i)L_i(n\cdot\omega_i) =$$
$$= (1-m)\frac{c}{\pi}\sum_i ((1-F_0)+(F_0-1)P_i)L_i(n\cdot\omega_i) =$$
$$= (1-m)\frac{c}{\pi}\sum_i \big((1-F_0)L_i(n\cdot\omega_i)+(F_0-1)P_iL_i(n\cdot\omega_i)\big) =$$
$$= (1-m)\frac{c}{\pi}\left(\sum_i (1-F_0)L_i(n\cdot\omega_i)+\sum_i (F_0-1)P_iL_i(n\cdot\omega_i)\right) =$$
$$= (1-m)\frac{c}{\pi}\left((1-F_0)\sum_i L_i(n\cdot\omega_i)+(F_0-1)\sum_i P_iL_i(n\cdot\omega_i)\right)$$

$$\mathrm{L}_{d_1} = \sum_i L_i(n\cdot\omega_i)$$
$$\mathrm{L}_{d_2} = \sum_i P_iL_i(n\cdot\omega_i)$$
$$\mathrm{L}_d(\mathrm{p},\omega_o) \approx (1-m)\frac{c}{\pi}\big((1-F_0)\mathrm{L}_{d_1}+(F_0-1)\mathrm{L}_{d_2}\big)$$

Specular

$$\mathrm{L}_s(\mathrm{p},\omega_o) = \sum_i \frac{FDG}{4(\omega_o\cdot n)}L_i = \frac{1}{4(\omega_o\cdot n)}\sum_i DGL_iF \approx$$
$$\approx \frac{1}{4(\omega_o\cdot n)}\sum_i DGL_i(F_0+(1-F_0)P_i) =$$
$$= \frac{1}{4(\omega_o\cdot n)}\sum_i (DGL_iF_0+DGL_iP_i(1-F_0)) =$$
$$= \frac{1}{4(\omega_o\cdot n)}\left(\sum_i DGL_iF_0+\sum_i DGL_iP_i(1-F_0)\right) =$$
$$= \frac{1}{4(\omega_o\cdot n)}\left(F_0\sum_i DGL_i+(1-F_0)\sum_i DGL_iP_i\right)$$

$$\mathrm{L}_{s_1} = \sum_i DGL_i$$
$$\mathrm{L}_{s_2} = \sum_i DGL_iP_i$$
$$\mathrm{L}_s(\mathrm{p},\omega_o) \approx \frac{1}{4(\omega_o\cdot n)}\big(F_0\mathrm{L}_{s_1}+(1-F_0)\mathrm{L}_{s_2}\big)$$

Fresnel-Schlick approximation:

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

Recombination

$$\mathrm{L}_{d_1} = \sum_i L_i(n \cdot \omega_i)$$

$$\mathrm{L}_{d_2} = \sum_i P_i L_i(n \cdot \omega_i) \qquad P_i = \left(1 - (h_i \cdot v)\right)^5$$

$$\mathrm{L}_{s_1} = \sum_i DGL_i$$

$$\mathrm{L}_{s_2} = \sum_i DGL_iP_i$$

$$\mathrm{L}_d(\mathrm{p}, \omega_o) \approx (1 - m)\frac{c}{\pi}\left((1 - F_0)\mathrm{L}_{d_1} + (F_0 - 1)\mathrm{L}_{d_2}\right)$$

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

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

We are going to calculate $\mathrm{L}_{s_1}, \mathrm{L}_{s_2}, \mathrm{L}_{d_1}, \mathrm{L}_{d_2}$ 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).

Redeffered rendering

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

$$\mathrm{L}_{d_1} = \sum_i L_i(n \cdot \omega_i)$$

$$\mathrm{L}_{d_2} = \sum_i P_i L_i(n \cdot \omega_i)$$

$$\mathrm{L}_{s_1} = \sum_i DGL_i$$

$$\mathrm{L}_{s_2} = \sum_i DGL_i P_i$$

$$\mathrm{L}_d(\mathrm{p}, \omega_o) \approx (1 - m)\frac{c}{\pi}\left((1 - F_0)\mathrm{L}_{d_1} + (F_0 - 1)\mathrm{L}_{d_2}\right)$$

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

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

Lets split shading into two stages.

First stage will produce L_{d_1}, L_{d_2}, 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 (to be later used by final deferred lighting composition pass combining it albedo and metalness).

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. Also, redeferred rendering can be used analogously with some 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) – this will require visual confirmation about quality.

Second stage does not have to iterate over lights, so it will be used to refine results at full resolution. It will fully cover artifacts of reduced shading rate from first pass, hopefully 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.

Memory throughput while accessing high number of textures in second stage can become a significant bottleneck, but certain remedies can also be taken into consideration.

Redeffered rendering - considerations

In case memory throughput becomes significant bottleneck:

A. Calculating $\frac{1}{F_0}\left(F_0 \mathrm{L}_{s_1} + (1-F_0)\mathrm{L}_{s_2}\right)$ and $\frac{1}{F_0}\left((1-F_0)\mathrm{L}_{d_1} + (F_0-1)\mathrm{L}_{d_2}\right)$ in first pass, storing results as a 2 colors instead of 4 and multiplying back by F_0 in second pass (F_0 is different now, because is taken from 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

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

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

B. Using different shading rates per illumination aspect

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

$$\mathrm{L}_{d_1} = \sum_i L_i(n \cdot \omega_i)$$

$$\mathrm{L}_{d_2} = \sum_i P_i L_i(n \cdot \omega_i)$$

$$\mathrm{L}_{s_1} = \sum_i DGL_i$$

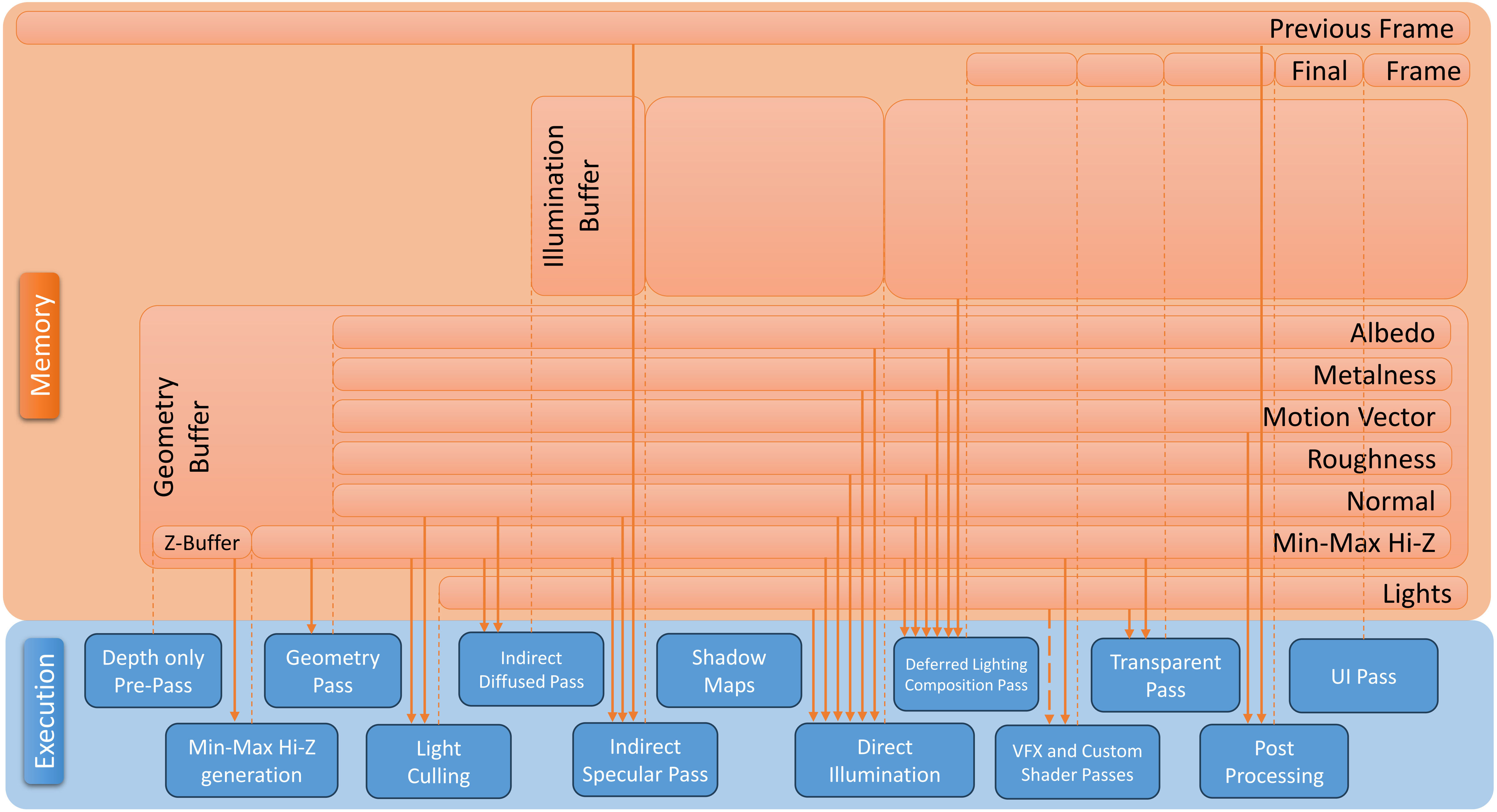
$$\mathrm{L}_{s_2} = \sum_i DGL_iP_i$$

$$\mathrm{L}_d(\mathrm{p},\omega_o) \approx (1-m)\frac{c}{\pi}\left((1-F_0)\mathrm{L}_{d_1} + (F_0-1)\mathrm{L}_{d_2}\right)$$

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

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

	R	G	B	A
32 bits	albedo R	albedo G	albedo B	metalness
32 bits	Roughness		Normal-View Angle	
32 bits	Normal X	Normal Y		material ID
32 bits	Motion vector X	Motion vector Y		free
128 bits				


Previous Frame
Final
Frame
Memory
Illumination Buffer
Geometry Buffer
Albedo
Metalness
Motion Vector
Roughness
Normal
Z-Buffer
Min-Max Hi-Z
Lights
Execution
Depth only Pre-Pass
Geometry Pass
Indirect Diffused Pass
Shadow Maps
Deferred Lighting Composition Pass
Transparent Pass
UI Pass
Min-Max Hi-Z generation
Light Culling
Indirect Specular Pass
Direct Illumination
VFX and Custom Shader Passes
Post Processing