2021 캡스톤 디자인 최종 발표

Team 6. Lit

Kookmin Univ. Software Dept. **Yang Kyowon**

Kookmin Univ. Software Dept. **Kwak Sangyeol**

Motivation : Let there be light!

- We **love to know and implement** how **light** is working!!!

Unreal Engine 4

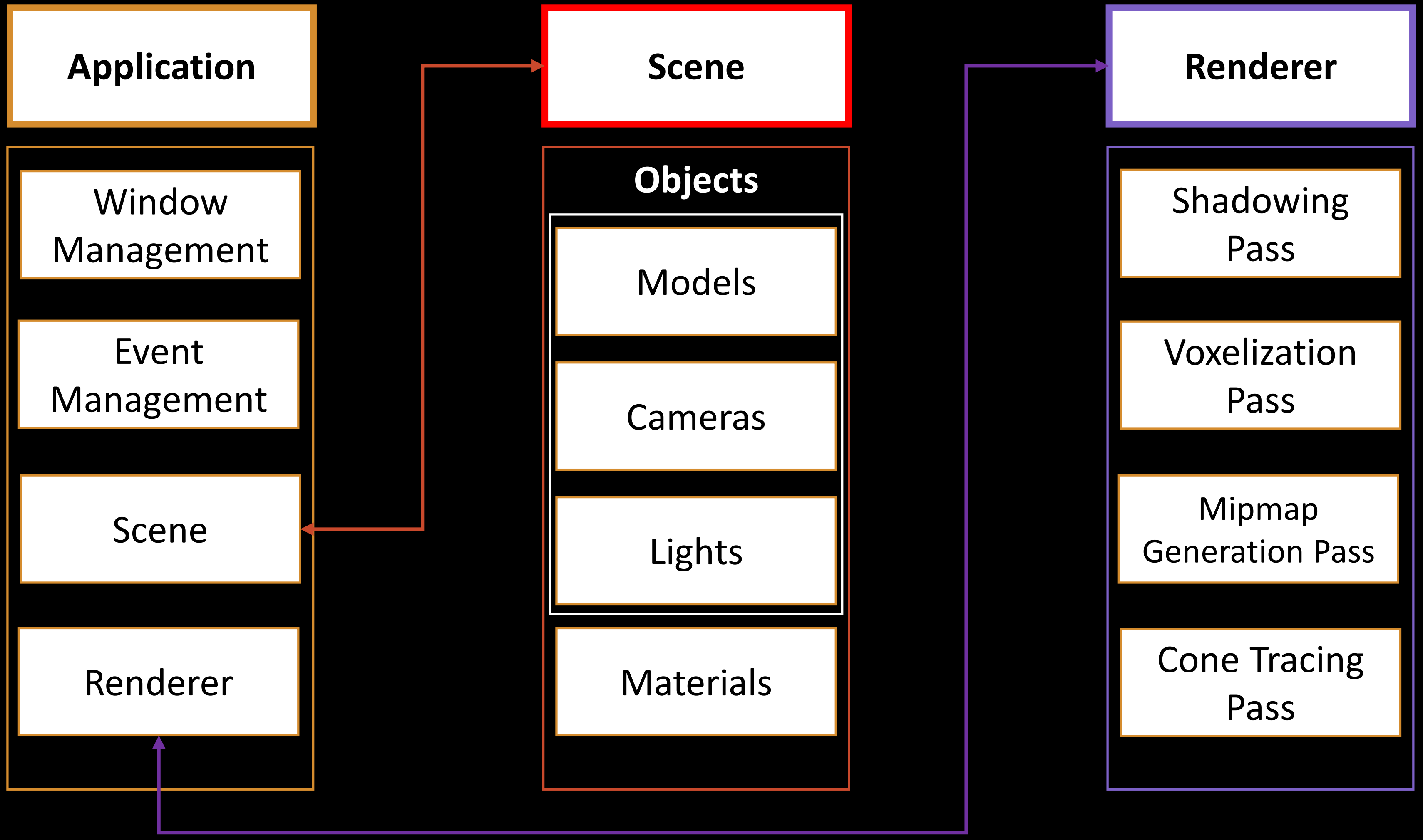
Unity

NVIDIA RTX

Lit : Goal

- Lets implement **renderer** to synthesize image with **global illumination effects** in a **real time**!

Overview of the Framework



Framework – Low Level APIs

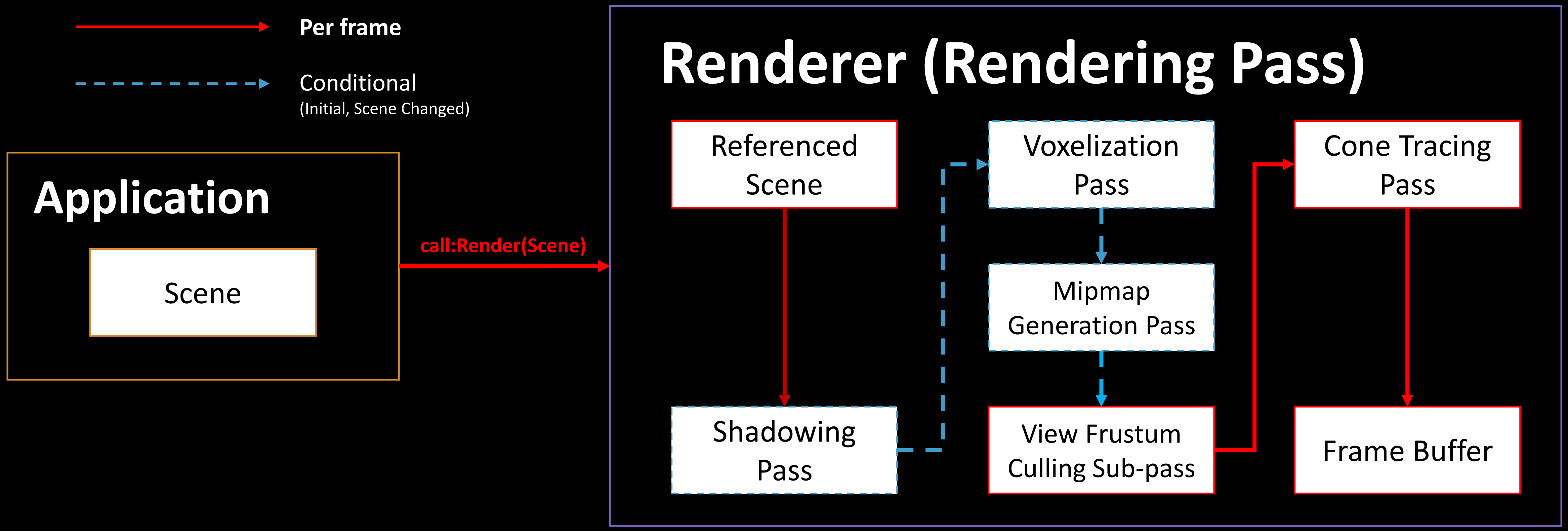
Application	GLFW
Graphics	OpenGL®
Mathematics	glm

Platforms

Features

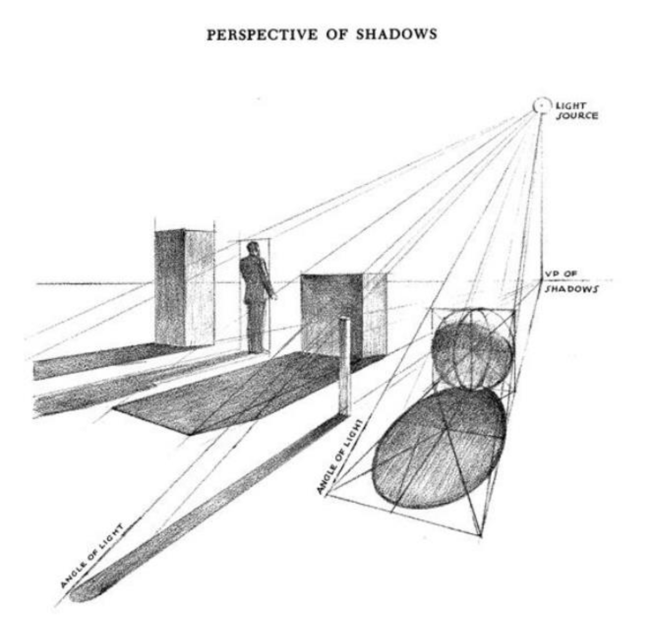
- **Real-time Global Illumination Effects (Voxel Cone Tracing)**
- Scene Management
  - Objects
  - Cameras
  - Lights
- Camera Path Animator
- Physically Based Workflow
- View Frustum Culling

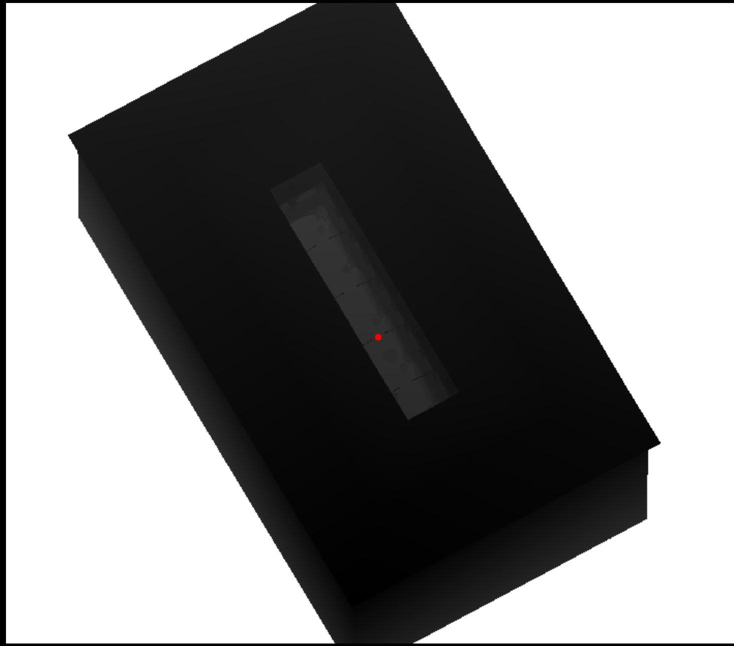
Overview of the Renderer



Shadowing Pass

- Render Depth Map(Shadow Map) from Light Source's view





Shadow Map (Depth Map from Light Source)

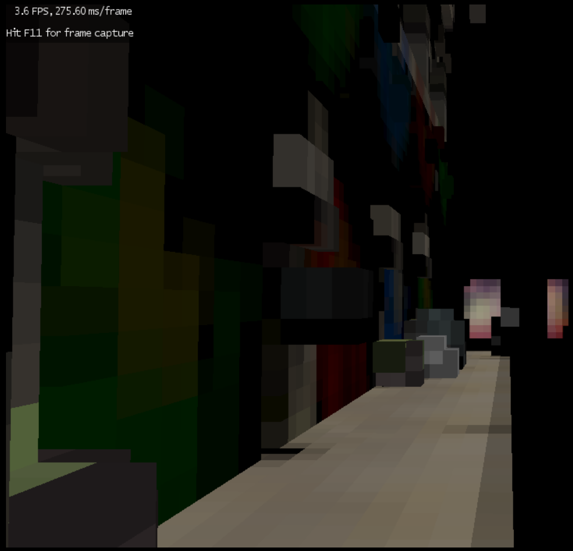
Shadowing Pass

- Render Depth Map(Shadow Map) from Light Source's view

Local Illumination with Shadows (Sponza Scene)

Voxelization Pass

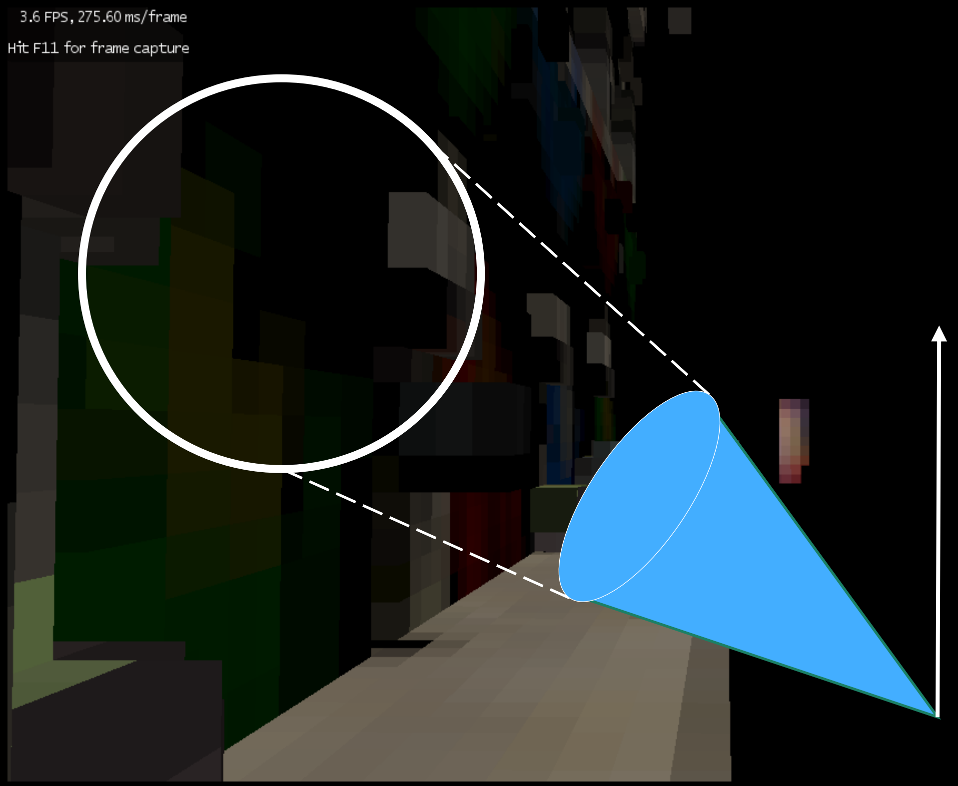
- Voxelize entire scene objects then inject radiance to 3D Texture



Voxelize Sponza Scene (Lambertian Diffuse with Shadows)
< Our renderer used 512^3 RGBA8 3D Texture >

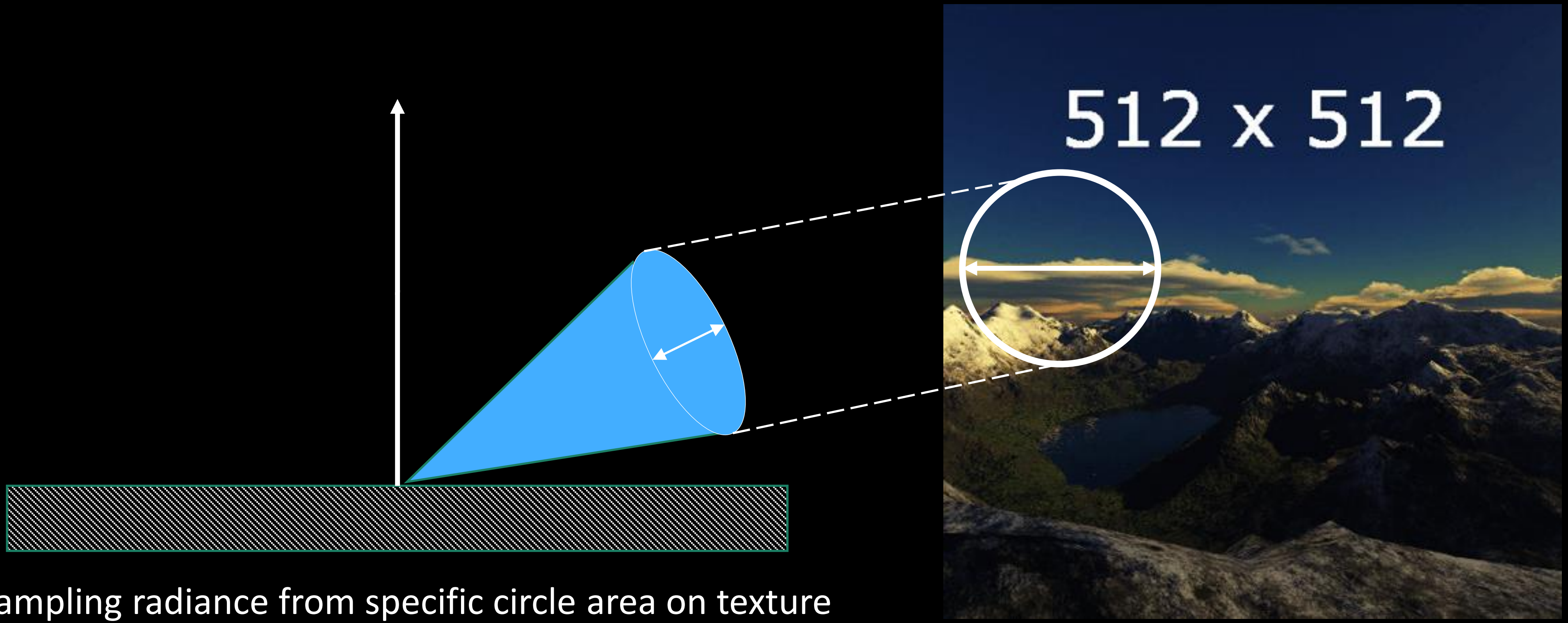
Mipmap Generation Pass

- Sampling radiance from 3d texture using Cone tracing



Mipmap Generation Pass

- Cone tracing with slice of 3D texture (=2D texture)



Sampling radiance from specific circle area on texture

Mipmap Generation Pass

- Mipmap : 기본 텍스처를 연속적으로 다운 샘플링 시킨 텍스처들의 집합

Mipmap Generation Pass

- Approximate sampling process using Mipmap!

$$SampleMipmap(3D\ Texture, Coords = Center\ of\ Circle, LOD = Diameter\ of\ Circle)$$

Mipmap Generation Pass

- Generate 3D texture's mipmap for cone sampling!

Mipmap Generation Pass

- Perform mipmap generation using Parallel Reduction Compute Shader

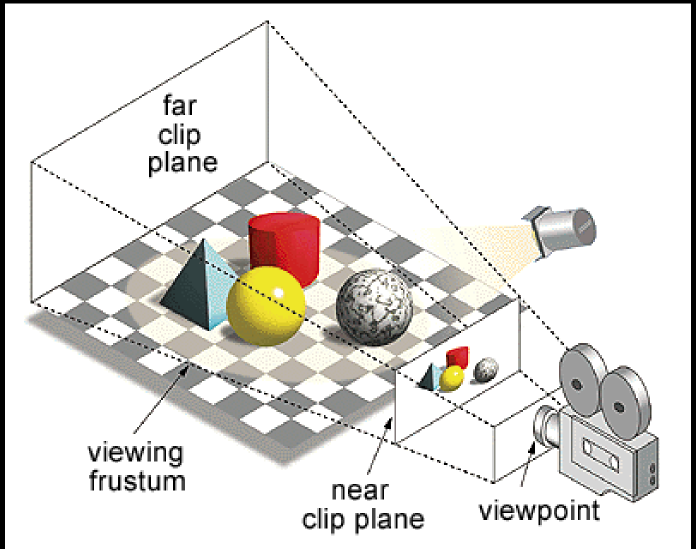
API Call	Count	Avg CPU ms	Σ CPU ms	Avg GPU ms	Σ GPU ms
glBindTexture()	1,874	<0.01	0.43	0.00	0.00
glActiveTexture()	1,872	<0.01	0.09	0.00	0.00
glUniform1i()	1,734	<0.01	0.19	0.00	0.00
glUniform1f()	838	<0.01	0.08	0.00	0.00
glBindVertexArray()	686	<0.01	0.42	0.00	0.00
glUniform3fv()	461	<0.01	0.06	0.00	0.00
glDrawElements()	343	<0.01	1.31	0.01	4.96
glUniform4fv()	252	<0.01	0.06	0.00	0.00
glUniformMatrix4fv()	48	<0.01	0.02	0.00	0.00
glEnable()	27	<0.01	<0.01	0.00	0.00
glBindFramebuffer()	4	<0.01	0.04	0.00	0.00
glDisable()	4	<0.01	<0.01	0.00	0.00
glUseProgram()	3	<0.01	0.01	0.00	0.00
glViewport()	3	<0.01	<0.01	0.00	0.00
glClearColor()	2	<0.01	<0.01	0.00	0.00
glClear()	2	0.04	0.09	0.01	0.02
glColorMask()	2	<0.01	<0.01	0.00	0.00
glCullFace()	1	<0.01	<0.01	0.00	0.00
glFrontFace()	1	<0.01	<0.01	0.00	0.00
glGetIntegerv()	1	<0.01	<0.01	0.00	0.00
glClearTexImage()	1	0.04	0.04	0.00	0.00
glBindImageTexture()	1	<0.01	<0.01	0.00	0.00
glGenerateTextureMipmap()	1	73.62	73.62	0.00	0.00
glBindRenderbuffer()	1	<0.01	<0.01	0.00	0.00
SwapBuffers()	1	0.54	0.54	0.00	0.00

API Call	Count	Avg CPU ms	Σ CPU ms	Avg GPU ms	Σ GPU ms
glBindTexture()	1,875	<0.01	0.44	0.00	0.00
glActiveTexture()	1,873	<0.01	0.08	0.00	0.00
glUniform1i()	1,739	<0.01	0.19	0.00	0.00
glUniform1f()	842	<0.01	0.07	0.00	0.00
glBindVertexArray()	686	<0.01	0.40	0.00	0.00
glUniform3fv()	461	<0.01	0.06	0.00	0.00
glDrawElements()	343	<0.01	1.30	<0.01	3.21
glUniform4fv()	252	<0.01	0.06	0.00	0.00
glUniformMatrix4fv()	48	<0.01	0.02	0.00	0.00
glEnable()	27	<0.01	<0.01	0.00	0.00
glBindImageTexture()	9	<0.01	<0.01	0.00	0.00
glUseProgram()	4	<0.01	0.01	0.00	0.00
glBindFramebuffer()	4	0.02	0.08	0.00	0.00
glDisable()	4	<0.01	<0.01	0.00	0.00
glDispatchCompute()	4	<0.01	0.02	0.19	0.78
glViewport()	3	<0.01	<0.01	0.00	0.00
glClearColor()	2	<0.01	<0.01	0.00	0.00
glClear()	2	0.03	0.05	<0.01	0.02
glColorMask()	2	<0.01	<0.01	0.00	0.00
glCullFace()	1	<0.01	<0.01	0.00	0.00
glFrontFace()	1	<0.01	<0.01	0.00	0.00
glGetIntegerv()	1	<0.01	<0.01	0.00	0.00
glClearTexImage()	1	0.04	0.04	0.00	0.00
glBindRenderbuffer()	1	<0.01	<0.01	0.00	0.00

Almost **100 times faster** than OpenGL built-in mipmap generation method at same configurations!

View Frustum Culling

- To decrease cone tracing and draw call overheads, **cull objects which not visible to viewer**



Hierarchical AABBs (Model-Meshes)

View Frustum Culling

- Performance comparison

Summary

Draws: 114	Sub Draws: 0	Clears: 1	Swap Buffers: 1	Non-API: 1	Total: 3621
	Dispatches: 0	Blits: 0	Misc. Data Update: 0	Other: 3504	

Details

Filter: Enter a filter

API Call	Count	Avg CPU ms	Σ CPU ms	Avg GPU ms	Σ GPU ms
glUniform1i()	1,034	<0.01	0.11	0.00	0.00
glActiveTexture()	626	<0.01	0.03	0.00	0.00
glBindTexture()	626	<0.01	0.14	0.00	0.00
glUniform1f()	602	<0.01	0.05	0.00	0.00
glUniform3fv()	229	<0.01	0.03	0.00	0.00
glBindVertexArray()	228	<0.01	0.12	0.00	0.00
glDrawElements()	114	<0.01	0.57	0.03	3.40
glEnable()	13	<0.01	<0.01	0.00	0.00
glBindRenderbuffer()	1	<0.01	<0.01	0.00	0.00
glBindFramebuffer()	1	<0.01	<0.01	0.00	0.00
glClearColor()	1	<0.01	<0.01	0.00	0.00
glClear()	1	0.05	0.05	<0.01	<0.01
glViewport()	1	<0.01	<0.01	0.00	0.00
glUseProgram()	1	<0.01	<0.01	0.00	0.00
glDisable()	1	<0.01	<0.01	0.00	0.00
SwapBuffers()	1	0.54	0.54	0.00	0.00

Disable View Frustum Culling (**Draw Calls : 114**)

Summary

Draws: 61	Sub Draws: 0	Clears: 1	Swap Buffers: 1	Non-API: 1	Total: 1987
	Dispatches: 0	Blits: 0	Misc. Data Update: 0	Other: 1923	

Details

Filter: Enter a filter

API Call	Count	Avg CPU ms	Σ CPU ms	Avg GPU ms	Σ GPU ms
glUniform1i()	557	<0.01	0.06	0.00	0.00
glActiveTexture()	360	<0.01	0.02	0.00	0.00
glBindTexture()	360	<0.01	0.09	0.00	0.00
glUniform1f()	319	<0.01	0.03	0.00	0.00
glUniform3fv()	123	<0.01	0.02	0.00	0.00
glBindVertexArray()	122	<0.01	0.07	0.00	0.00
glDrawElements()	61	<0.01	0.29	0.05	3.22
glEnable()	5	<0.01	<0.01	0.00	0.00
glBindRenderbuffer()	1	<0.01	<0.01	0.00	0.00
glBindFramebuffer()	1	<0.01	<0.01	0.00	0.00
glClearColor()	1	<0.01	<0.01	0.00	0.00
glClear()	1	0.05	0.05	<0.01	<0.01
glViewport()	1	<0.01	<0.01	0.00	0.00
glUseProgram()	1	<0.01	<0.01	0.00	0.00
glDisable()	1	<0.01	<0.01	0.00	0.00
SwapBuffers()	1	0.45	0.45	0.00	0.00

Enable View Frustum Culling (**Draw Calls : 61**)

Cone Tracing Pass

- Evaluate **4 terms** in Cone Tracing Pass!
  - Direct Diffuse Term (L_{DD})
  - Indirect Diffuse Term (L_{ID})
  - Direct Specular Term (L_{DS})
  - Indirect Specular Term (L_{IS})
  - $Final\ Result = L_{DD} + L_{ID} + L_{DS} + L_{IS}$

Cone Tracing Pass

- Direct Diffuse Term (Local Illumination)

$$L_{DD} = Visibility \cdot L_i \frac{\alpha}{\pi} dot(\hat{n}, \hat{l})$$

$Diffuse\ BRDF : Lambertian\ Reflectance$

$\alpha : Albedo\ of\ material$

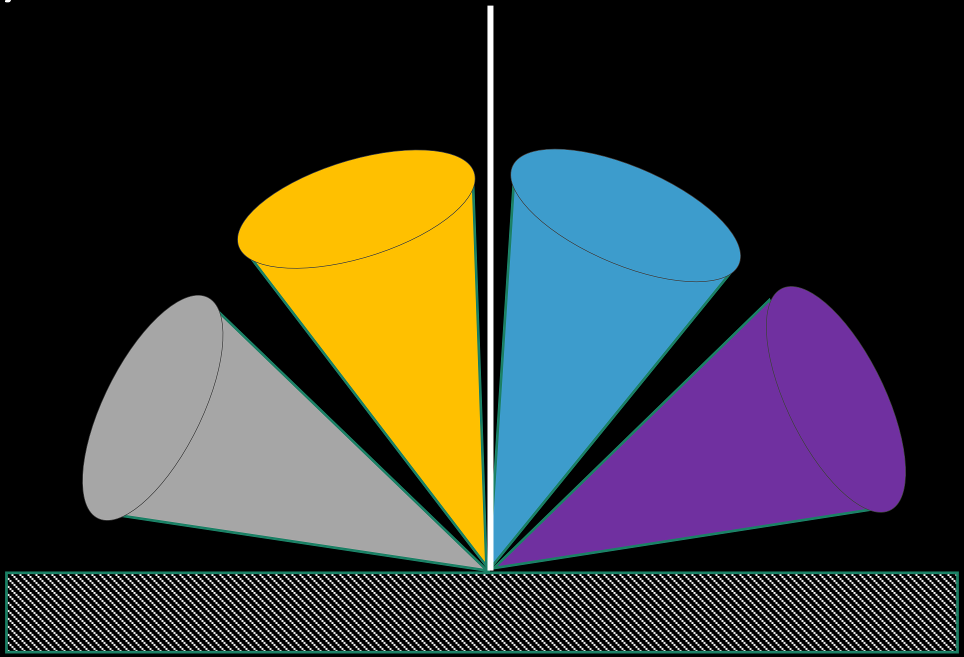
$L_i : Light\ Intensity$

$n : Fragment\ normal$

$l : Light\ Direction$

Cone Tracing Pass

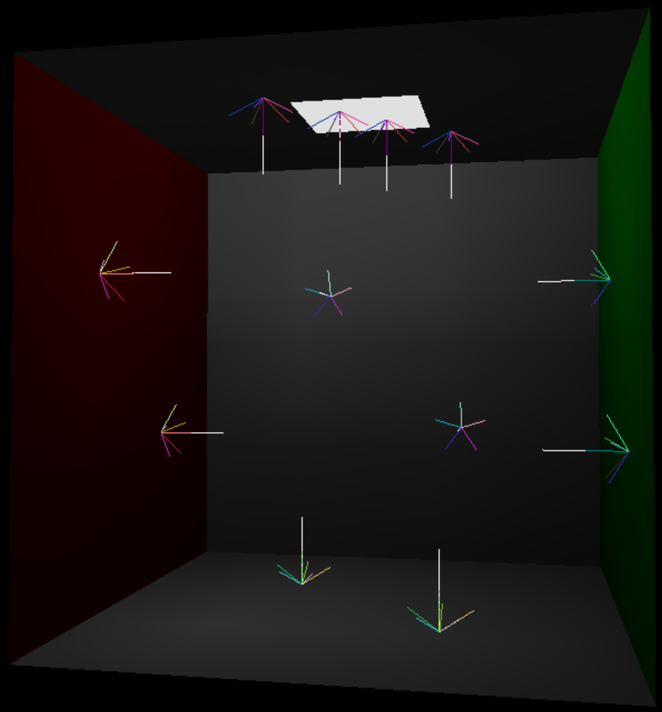
- Indirect Diffuse Term (Global Illumination)

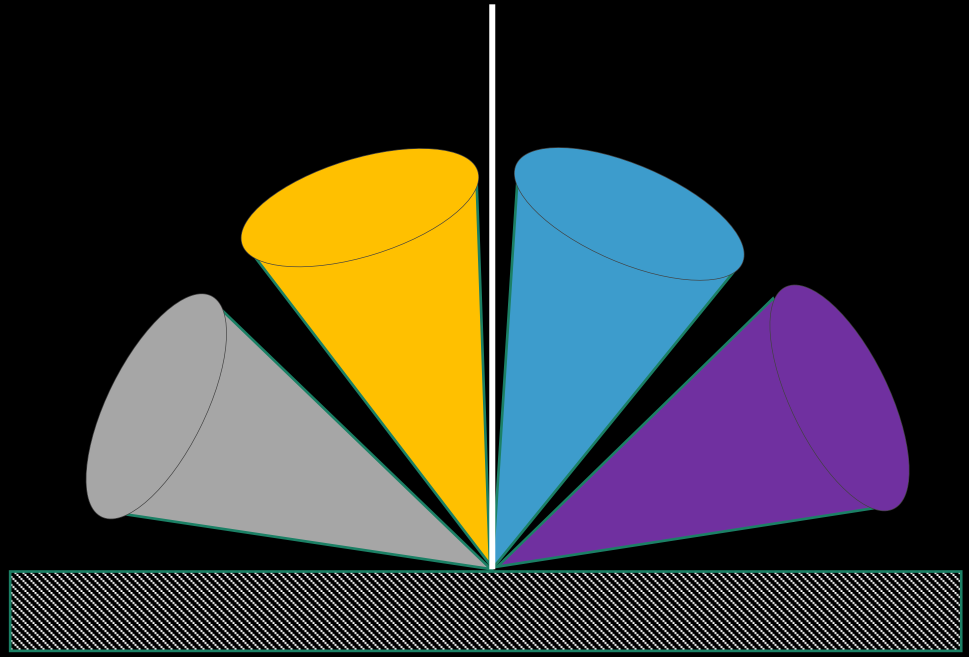


$$L_{ID} = \frac{\alpha}{\pi}\sum_{i=1}^{6} w_i \cdot ConeTrace(\hat{n}, \hat{c_i}, 60°)dot(\hat{n}, \hat{c_i})\ (\ c_i\!: Cone\ Direction, w_i\!: Cone\ Weights\ )$$

Cone Tracing Pass

- Visualize Diffuse Cone directions

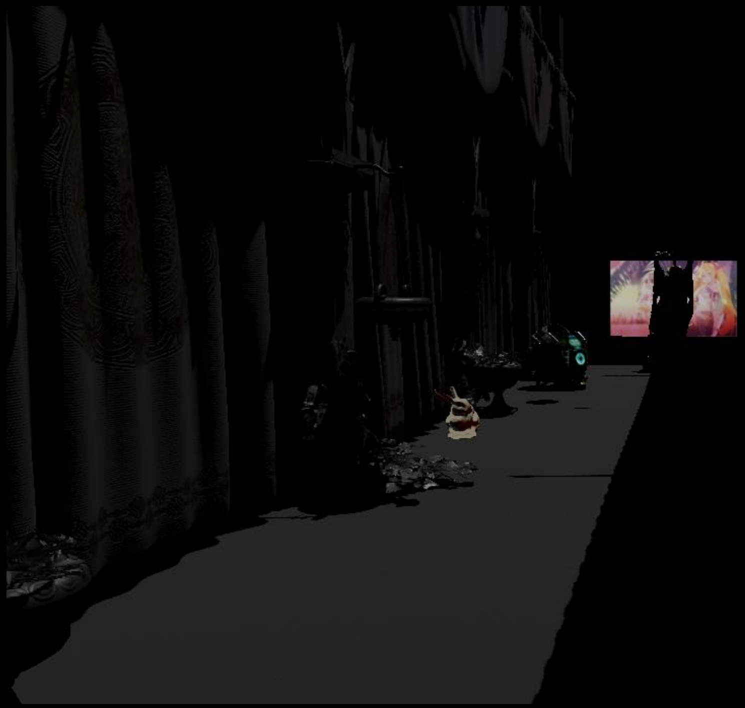




$$L_{ID} = \frac{\alpha}{\pi}\sum_{i=1}^{6} w_i \cdot ConeTrace(\hat{n}, \hat{c_i}, 60°)dot(\hat{n}, \hat{c_i})\ (\ c_i\colon Cone\ Direction, w_i\colon Cone\ Weights\ )$$

Cone Tracing Pass

- Direct Specular (Local Illumination)



$$\mathrm{L_{DS}} = Visiblity \cdot L_i \frac{D \cdot G \cdot F}{4 \cdot dot(\hat{l}, \hat{h}) dot(\hat{v}, \hat{h})} dot(\hat{n}, \hat{l})$$

Specular BRDF : GGX Microfacet BRDF

D: *Normal Distribution Term*

G : *Geometry Term*

F : *Fresnel Term*

l : *Light Direction*

v : *View Direction*

h: *Halfway Direction*

Cone Tracing Pass

- Indirect Specular (Global Illumination)

$$L_{IS} = \frac{1}{N}\sum_{k=1}^{N}\frac{L_i(\widehat{l_k})GGX(\widehat{l_k},\hat{v})dot(\hat{n},\widehat{l_k})}{p(\widehat{l_k},\hat{v})}$$

Importance Sampled GGX Microfacet BRDF

p : $GGX\ Normal\ Distribution$

n : $Normal\ Direction$

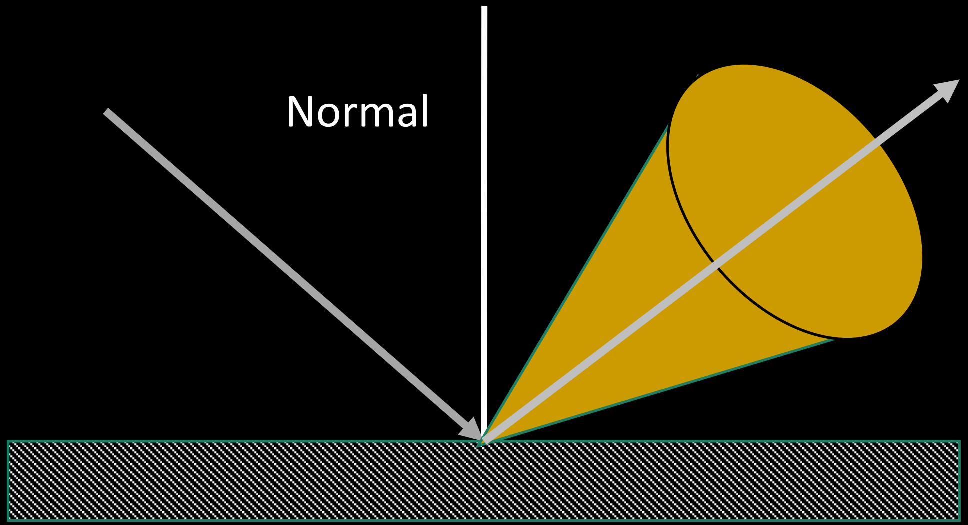
l_k : $Sampled\ Light\ Direction$

v : $View\ Direction$

N: $Sample\ Size(Default : 2{\sim}4)$

Cone Tracing Pass

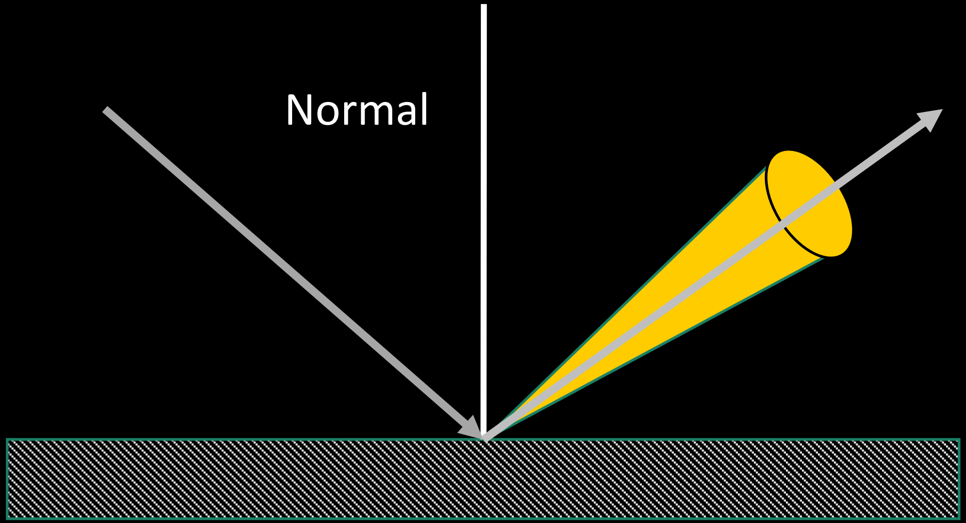
- Rough Specular (High roughness)



$$Roughness \propto Cone\ Diameter$$

Cone Tracing Pass

- Fine Specular (Low Roughness)



$$Roughness \propto Cone\ Diameter$$

Cone Tracing Pass

- Final Result (Combine every evaluated terms)

$$L = L_{DD} + L_{ID} + L_{DS} + L_{IS}$$

Performance

Configuration / Performance	Crytek Sponza (GI OFF)	Crytek Sponza (GI ON)	Demo Sponza (GI OFF)	Demo Sponza (GI ON)
Lowest FPS	480.6 FPS	144.8 FPS	450.4 FPS	107.3 FPS
Highest FPS	503.4 FPS	220.3 FPS	480.3 FPS	163.7 FPS
$\boldsymbol{\Delta t_{Avg}}$	2.03 ms	5.48 ms	2.22 ms	7.38 ms

Δt_{Avg} : $Average\ of\ times\ to\ process\ a\ frame$

- Configurations
  - CPU : AMD Ryzen 2700x
  - GPU : NVIDIA GeForce RTX 2080
  - Display Resolution : 1280 x 720
  - Voxelized Scene : RGBA8(512^3)

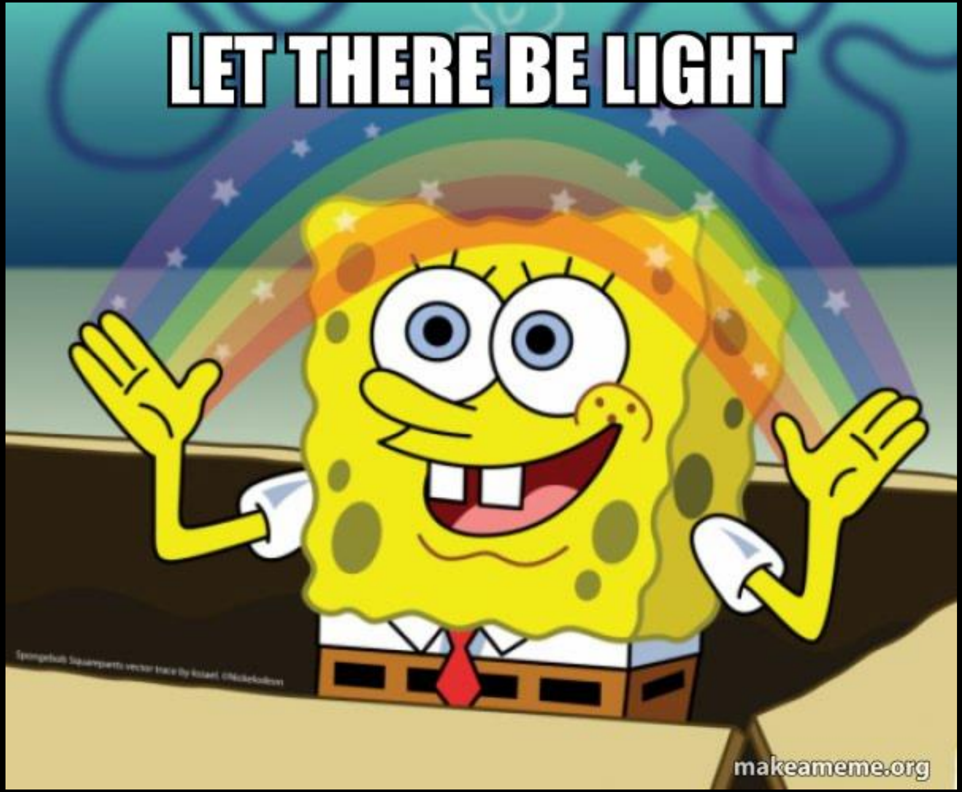
- Observations
  - **Reasonable performance** for real time applications!
  - Performance is related to **number of objects** in the scene.

Results (Demo Video)

Future Works

- **Extend Indirect Light Bounces**
  - Using LPV(Light Propagation Volume) for 1st bounce to extend 2nd bouncing at VCT.
  - Or through compute shader to simulate (N-1) times light bouncing.
- **Improve Voxelization Method**
  - Clip-map based Voxelization to reducing memory footprint.
  - Split Geometry data(Normal, Albedo, Opacity, ..) to handle more complex scenes.
    (ex. Light has physical quantity units)
- **Implement Post-Process Effects**
  - To get more beautiful final result, we need to consider about post-process effects like DOF, Bloom, Exposure, Bokeh, etc…
- **Find more flexible and physically plausible BSDFs (Not a BRDF)**

Thanks for your attention



Additional Figures

Additional Figures

Additional Figures

Additional Figures

Additional Figures