基于屏幕空间体素的动态GI

haiaimi

背景介绍

问题

- 无主光间接光

目标

- 有主光间接光

背景介绍

问题

- 无天光遮蔽

目标

- 有天光遮蔽

背景介绍

问题

- 无局部光间接光

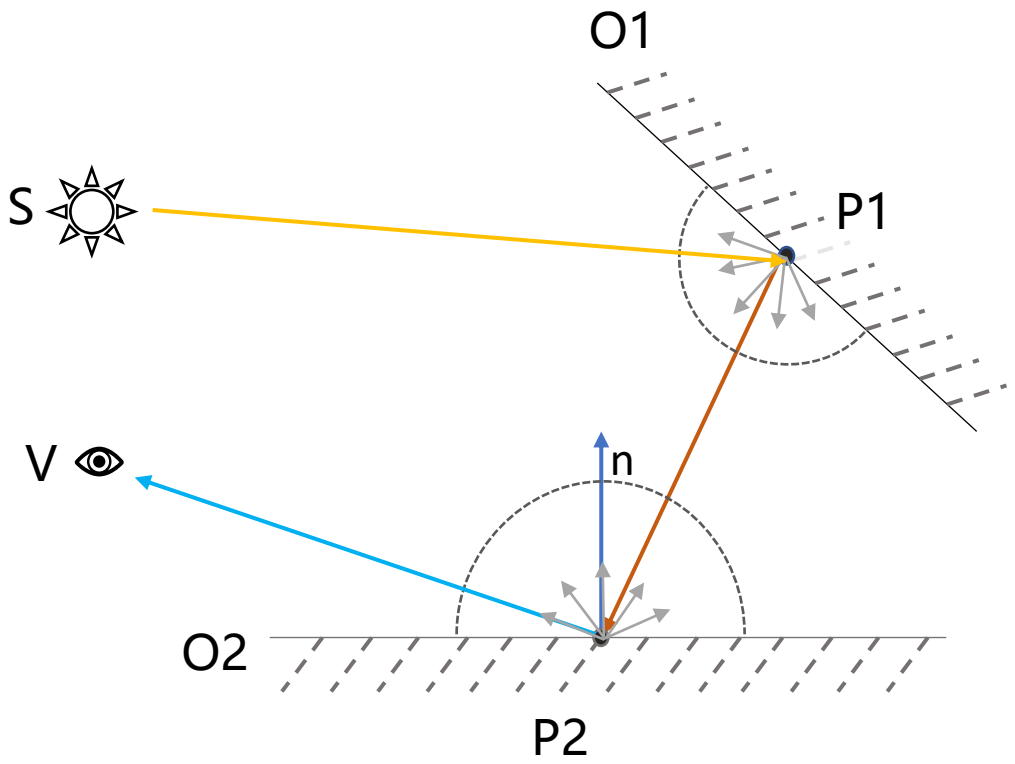
目标

- 有局部光间接光

什么是间接光?

- 例如：S->P1->P2->V
- 如何计算?
  - 射线相交检测
- 计算数量级?
  - 屏幕像素数量: N
  - 一次反弹半球采样数：M
  - 二次反弹半球采样数：K
  - .....

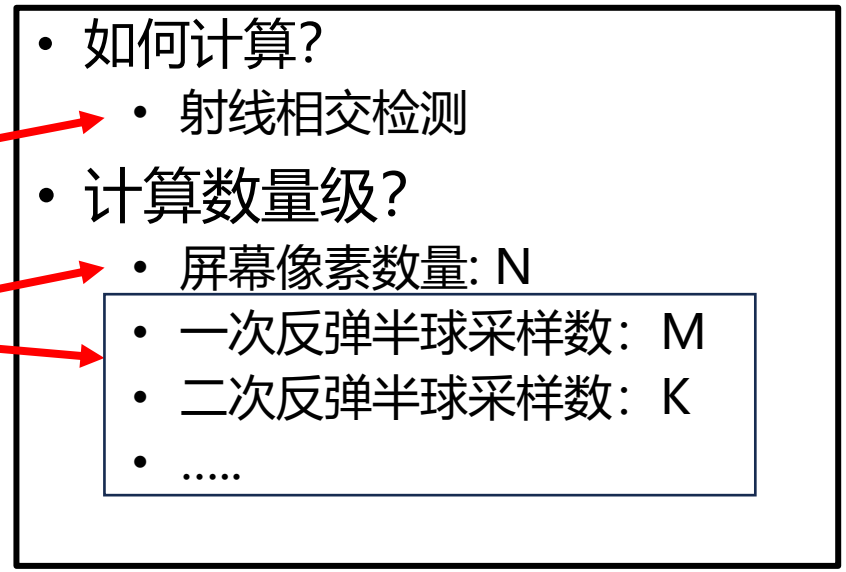
  O(N*M*K*...)　指数级增长!



实时GI都是在减少计算复杂度

实时GI组成部分

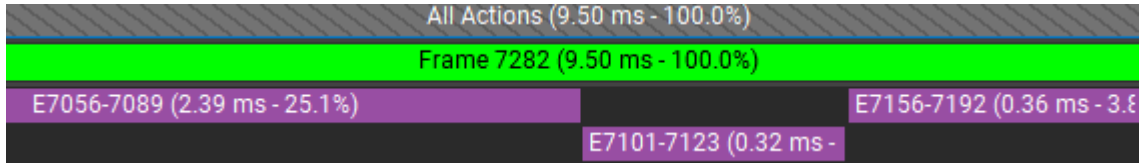
- GI Scene：场景的离散化表达方式
- Indirect Light：间接光收集
- Shading：最终着色



方案选择-目前业内常见方案

- 需求
  - 动态光照/场景下的稳定间接光
  - 无烘焙，低消耗
  - **性能优先，支持移动平台**

- 业内RTGI方案
  - SSGI：消耗高、不稳定
  - VXGI：消耗高
  - DDGI：需硬件支持
  - ……



RTX 2060 UE4.26 SSGI 消耗3ms (1449x921)

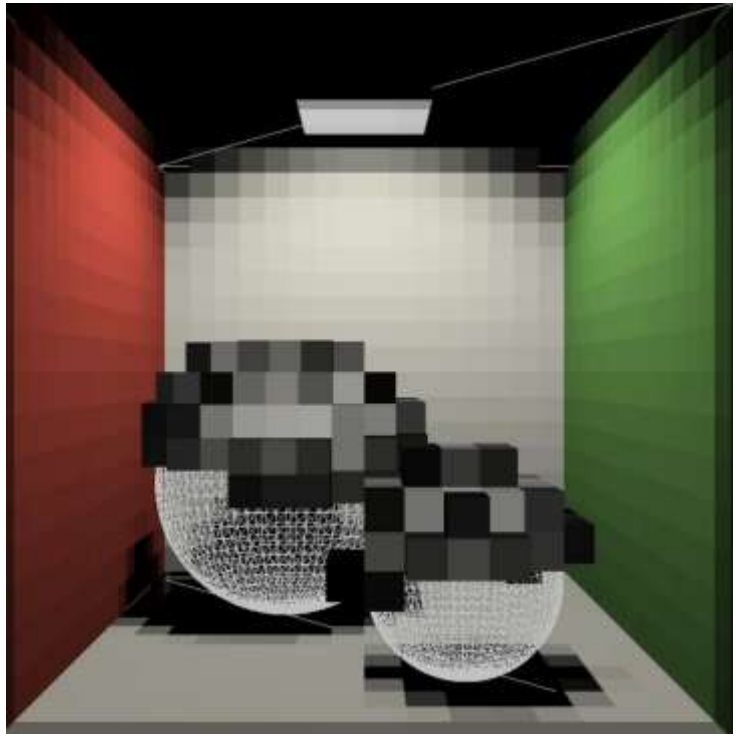
Part1　GI Scene

目的

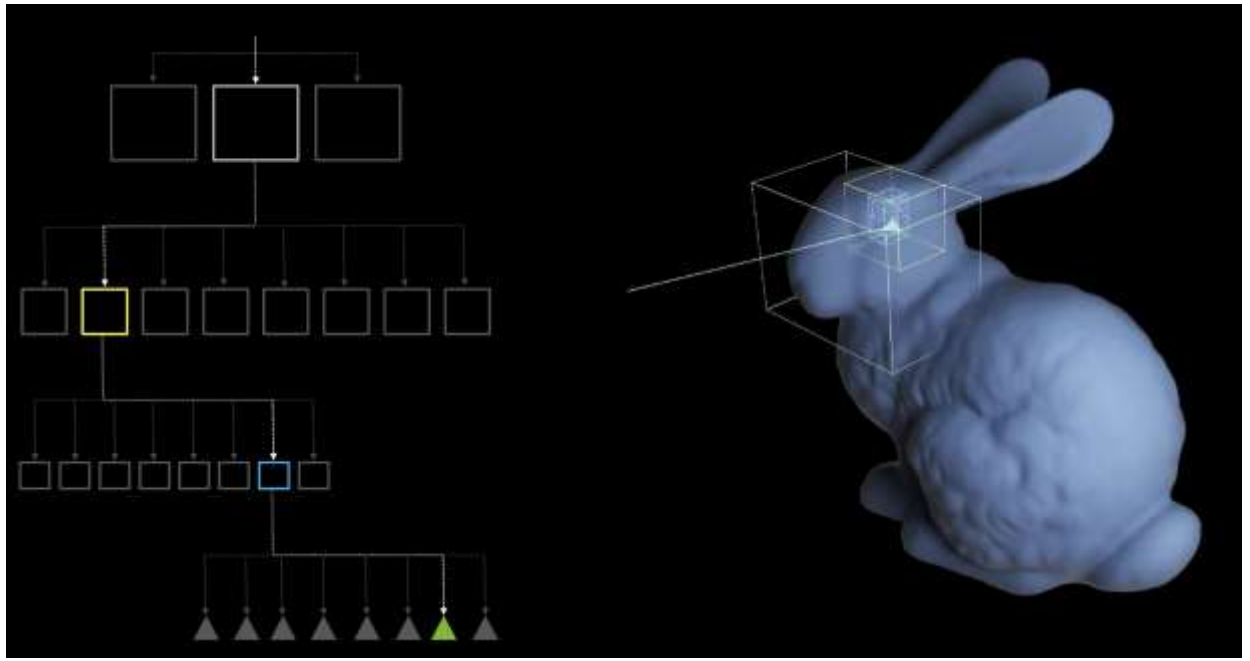
- 简化射线与场景相交检测

- 如何计算?
  - 射线相交检测
- 计算数量级?
  - 屏幕像素数量: N
  - 一次反弹半球采样数：M
  - 二次反弹半球采样数：K
  - .....

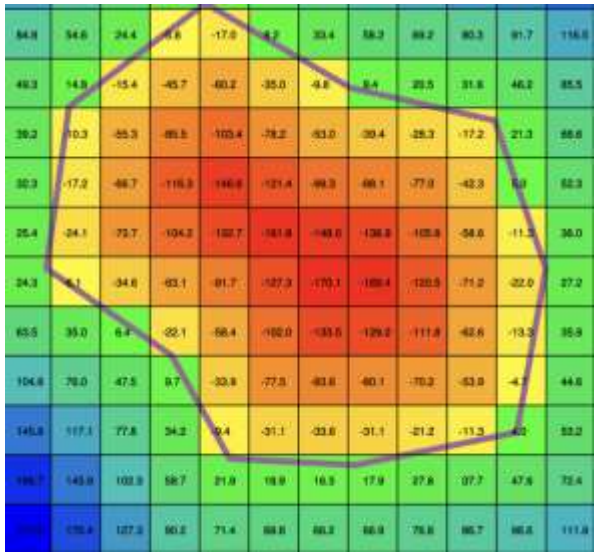
常见GI Scene



Voxel



Hardware Acceletarion Structure(DXR BVH)



SDF

Scene Depth

方案选择

- GI Scene
  - Voxel：Screen Space Scene Color
- Indirect Light Cache
  - Probe: Gather Light From Voxel
- Shading
  - Sample Probe 3D Volume

常见GI Scene

方案	SDF	Voxel	HAS	Scene Depth
优点	结构简单，Trace速度快	结构简单，易生成	Trace精准	获取简单，Trace精准
缺点	生成比较复杂	Trace略复杂	需要硬件支持	不稳定
				

为什么不借鉴Candela SDF GI方案?

iphone 11 pro max上效率（包含一次sdf更新时，一共约8.6ms，多次sdf更新累加sdf更新部分）：

Sdf 更新： 4.45ms（10 倍）

0、color to history 381us

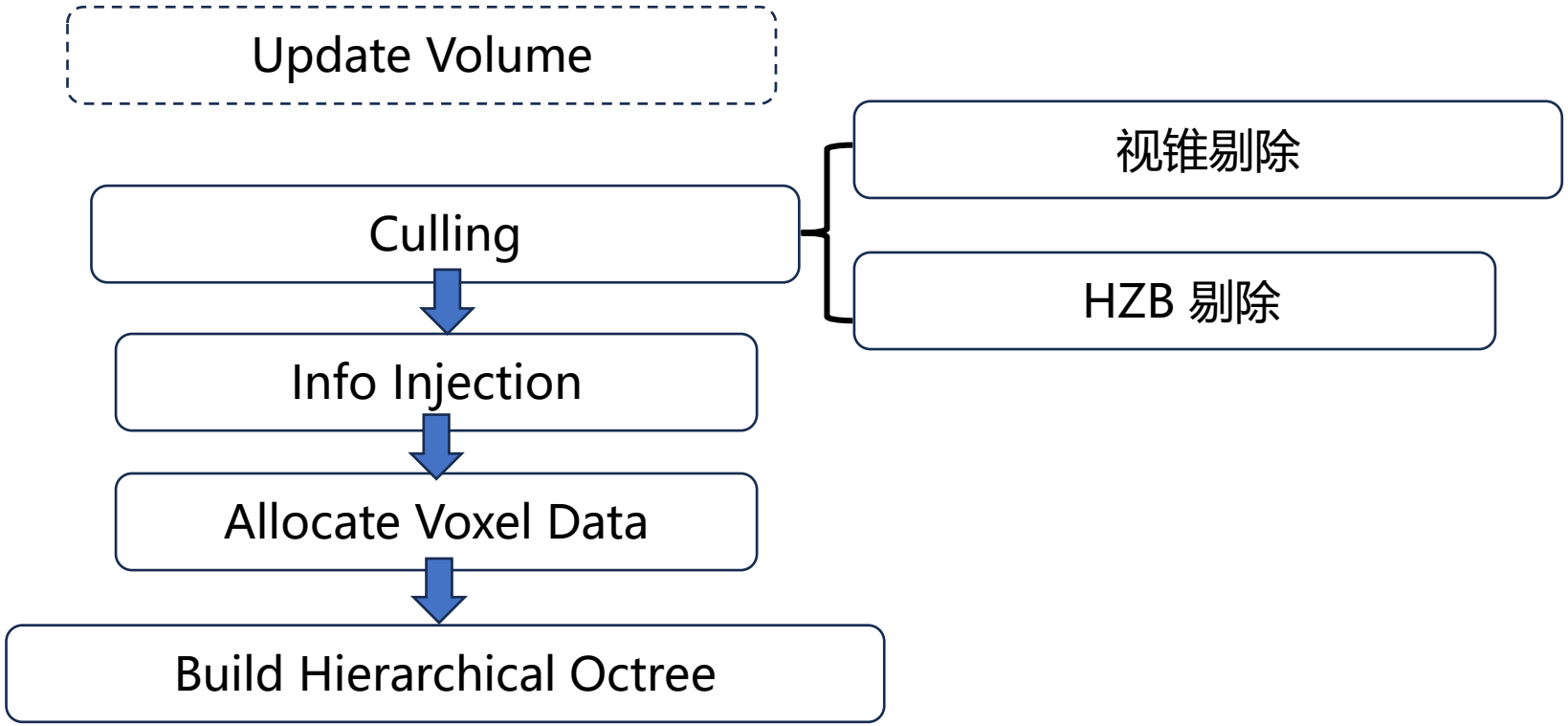
1、mark and inject color 964.8us

2、edge fix 401us

3、propagate 2.71ms

https://km.netease.com/article/397712

Voxel-总体流程


Update Volume
Culling
视锥剔除
HZB 剔除
Info Injection
Allocate Voxel Data
Build Hierarchical Octree

Voxel

- 如何生成Voxel?

GPU Voxelization

- 优点
  - 精确，效果好
- 缺点
  - 消耗高

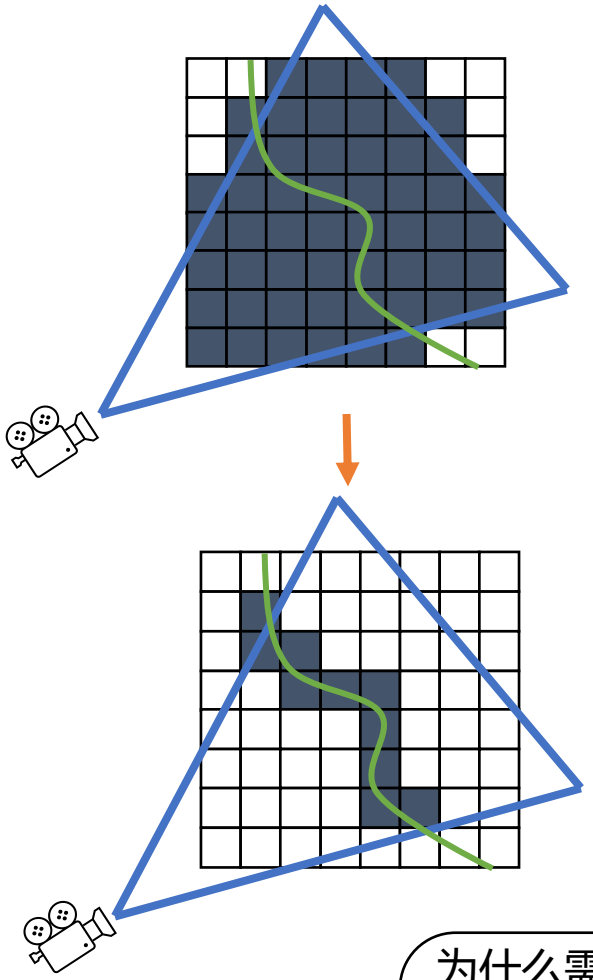
Screen Space Voxelization

- 优点
  - 快速高效
- 缺点
  - 缺少屏幕外信息，不稳定

Voxel-Culling

- 目的
  - 收集当前帧需要更新的Voxel
  - 只更新有效区域Voxel，提高效率
  - 分块更新

Hi-Z Buffer



Brick (4x4x4 Voxels)

- 粗粒度视锥剔除
  - 4x4x4 bricks

- 细粒度HZB剔除
  - Per brick

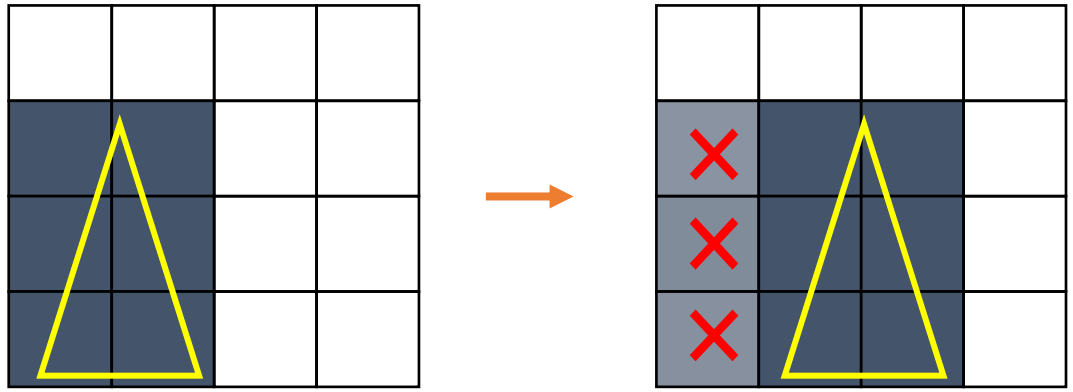
为什么需要Brick?
Voxel大小：0.25m
范围：100mx100mx40m
Voxel映射表： 400x400x160=25,600,000
Brick映射表：100x100x40 = 400000

Voxel-Culling

- 剔除粒度为Brick，Brick内的Voxel如何更新有效性?

Last Hit Counter

- 距离上次成功注入的帧数
  - 超出一定的阈值就标记为失效



物体Brick内移动

Voxel-Info Injection

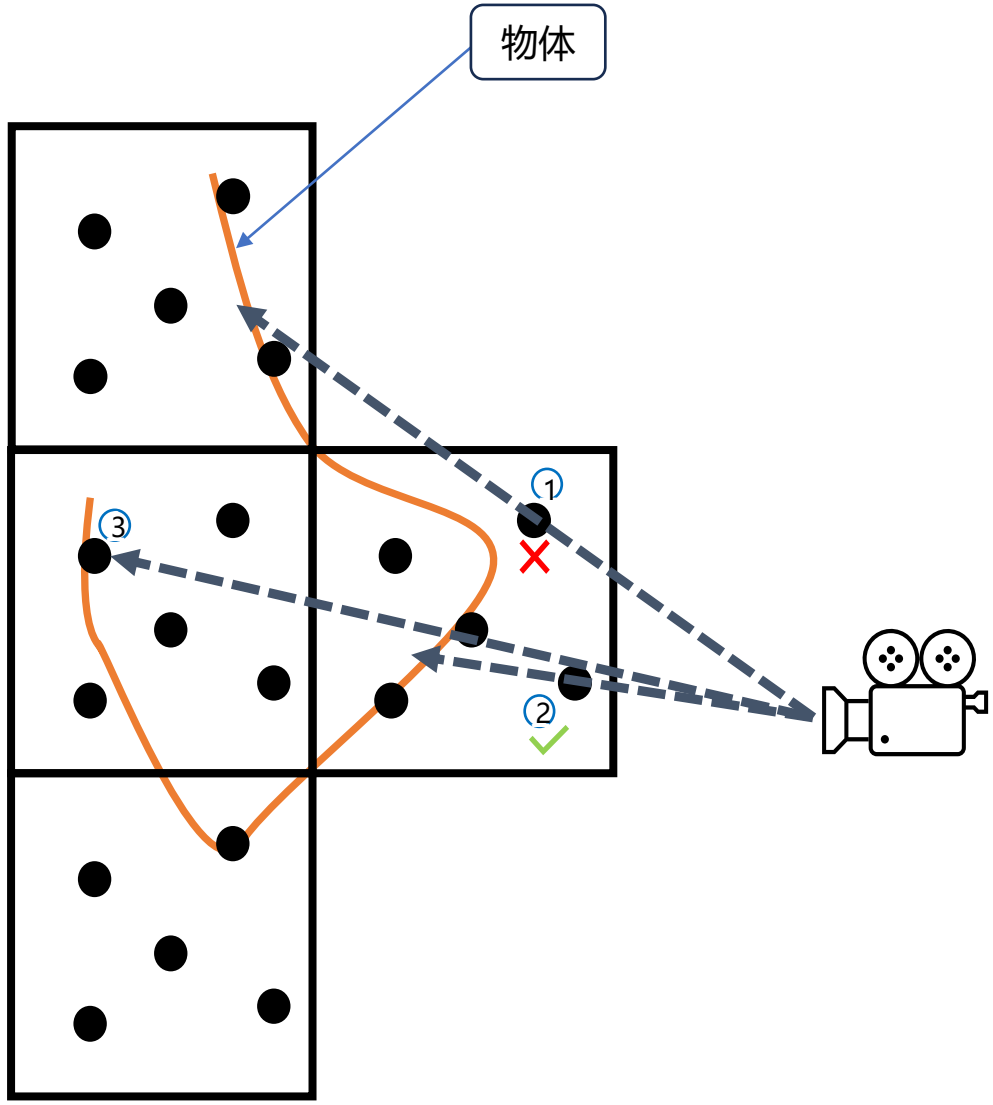
- 如何从屏幕空间注入?
  - Voxel->SceneColor(避免数据同时写入冲突)
  - 选取体素内随机点（世界坐标)
  - 投影到Screen Space，获取对应场景深度
  - 比较深度位置是否在voxel内

- 写入数据
  - Color: R11G11B10
  - Normal: R8G8
  - Last Hit Counter: R16

① Invalid：累加LastHitCounter

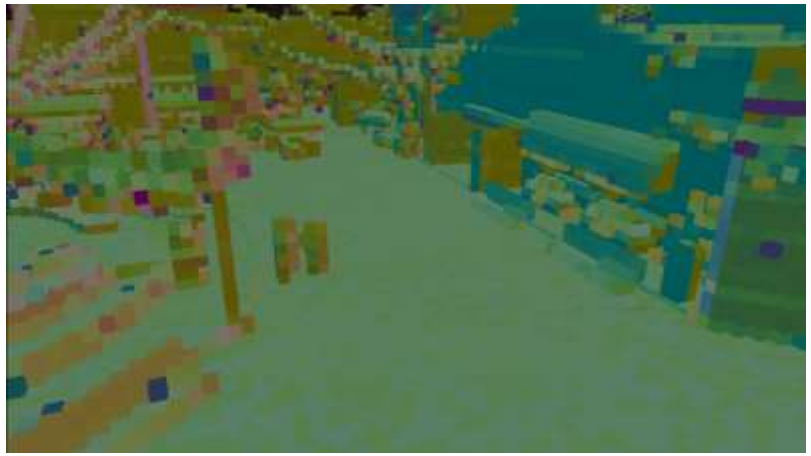
② Valid： 写入Color/ Normal, 重置LastHitCounter

③ Skip：Do noting



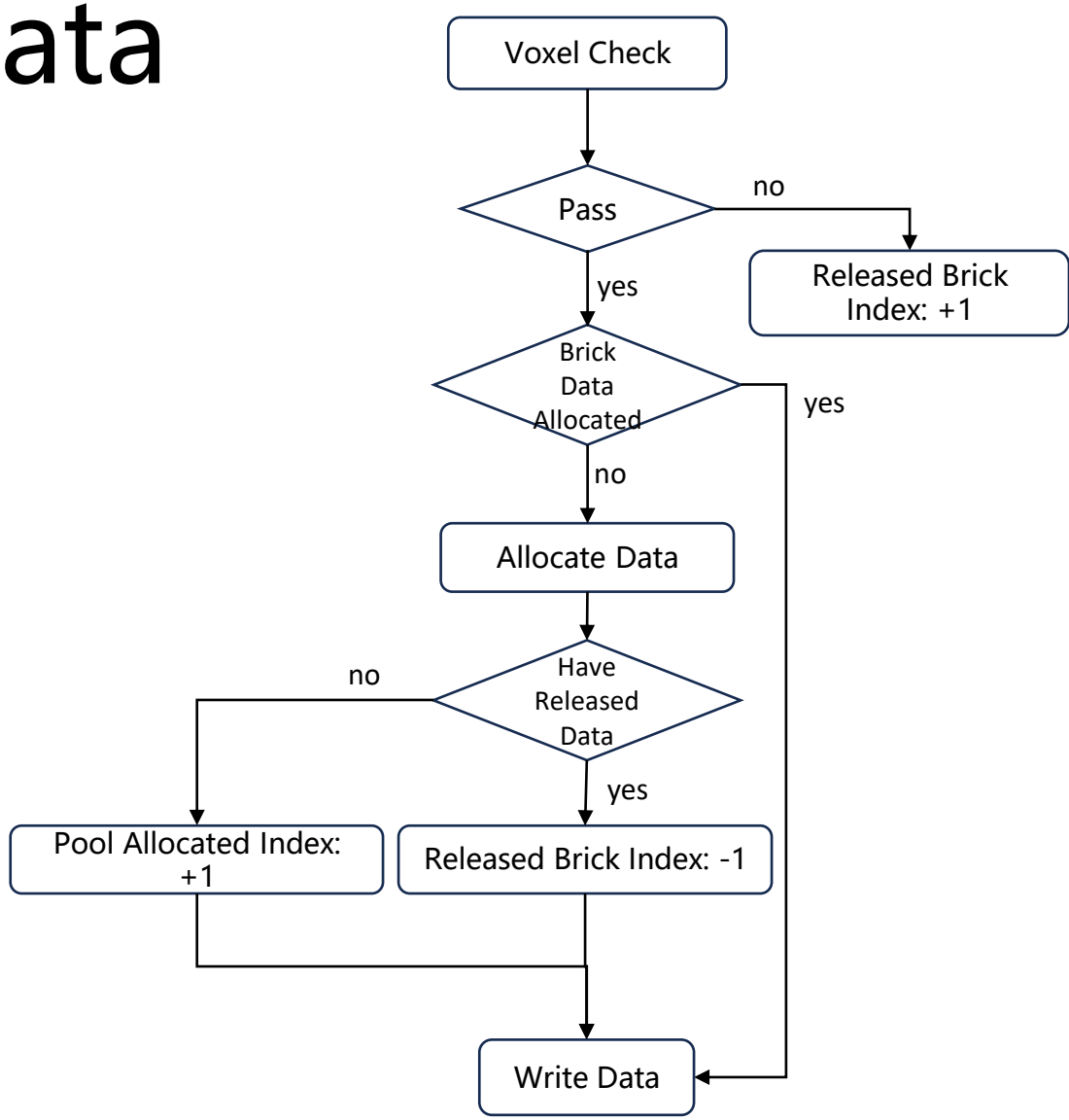
Voxel-Info Injection

- 输入：Depth + SceneColor
- 输出：Voxel(color + normal)

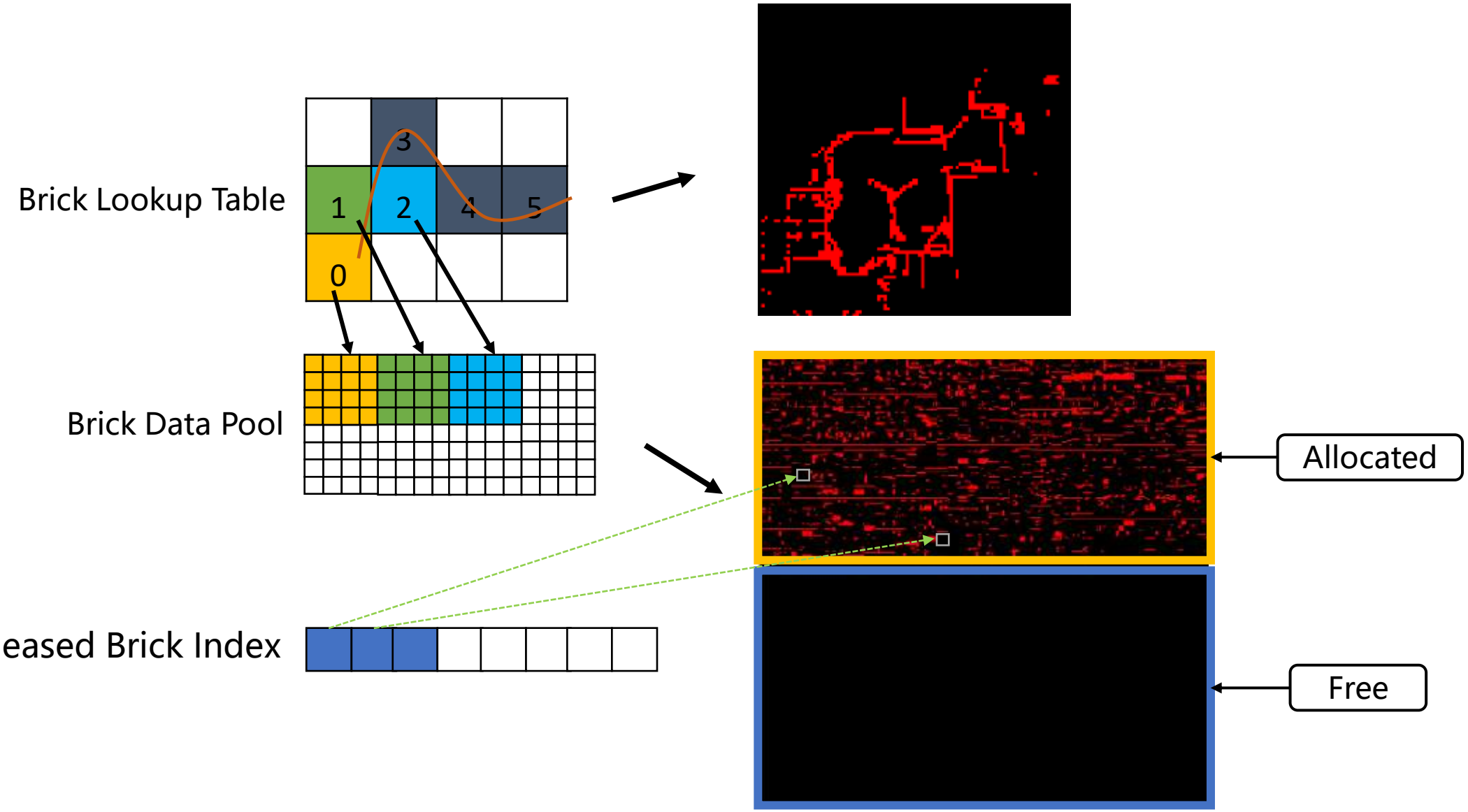


Voxel-Allocate Voxel Data

- 存储单位：Brick
- 数据结构
  - Brick Data Pool: int32 Texture3D
    - 存储Brick数据的Pool
  - Brick Lookup Table:int32 Texture3D
    - 世界空间Brick在Pool存储位置的索引
  - Pool Allocated Index: int32
    - Pool中已被分配数量
  - Released Brick Index: int32 Buffer
    - Pool中被分配又被释放的Brick索引

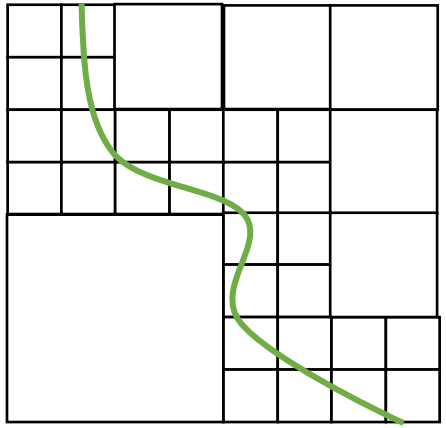


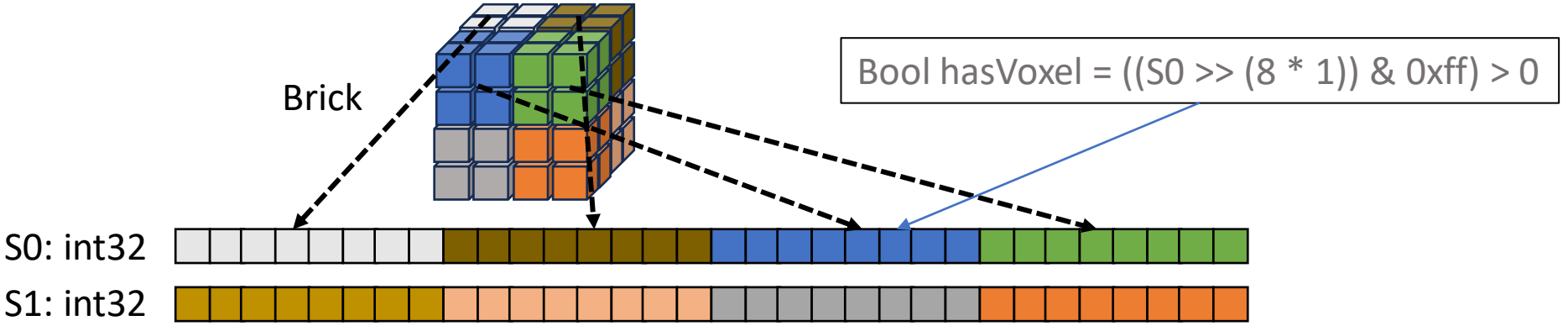
Voxel Data


Brick Lookup Table
3
1
2
4
5
0
Brick Data Pool
Released Brick Index
Allocated
Free

Voxel-Build Octree

- 为什么需要八叉树?
  - 加速Voxel Trace
  - 生成Probe位置
- 实现
  - 5 Level
  - 2x2x2 Per Level
  - 1brick 64bit

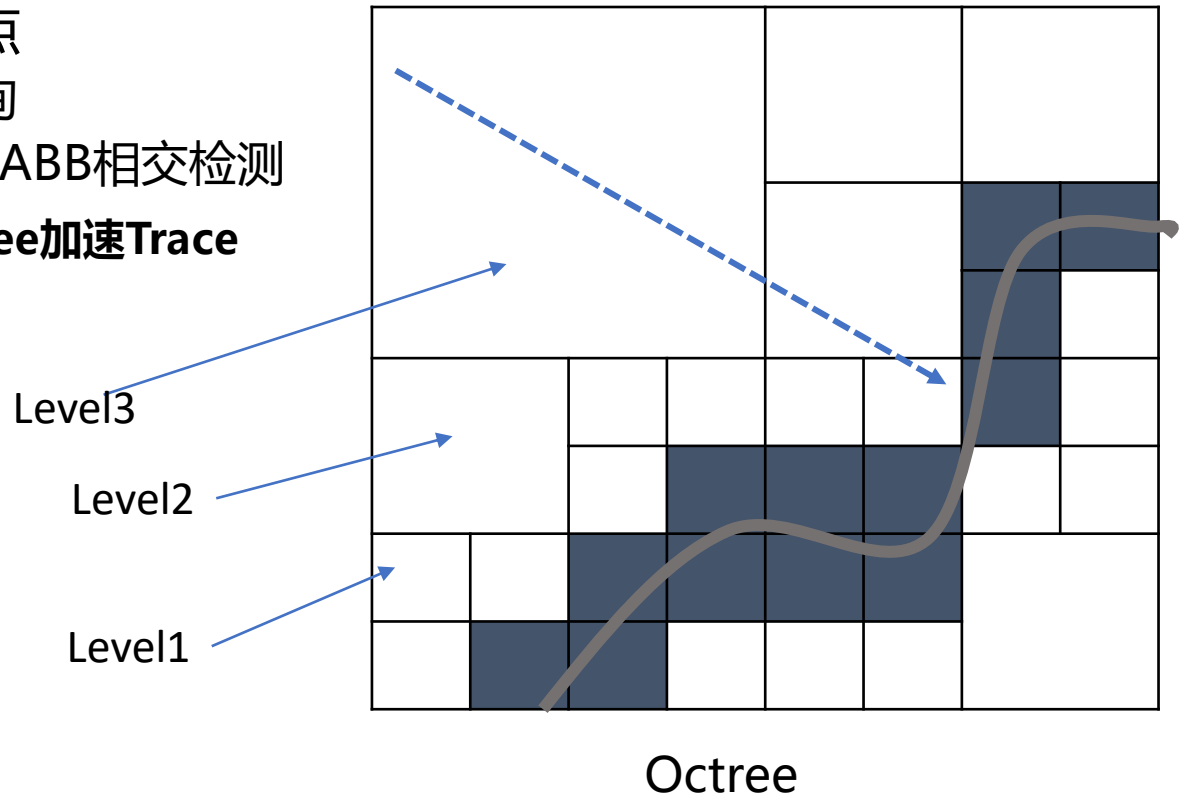




Voxel Trace

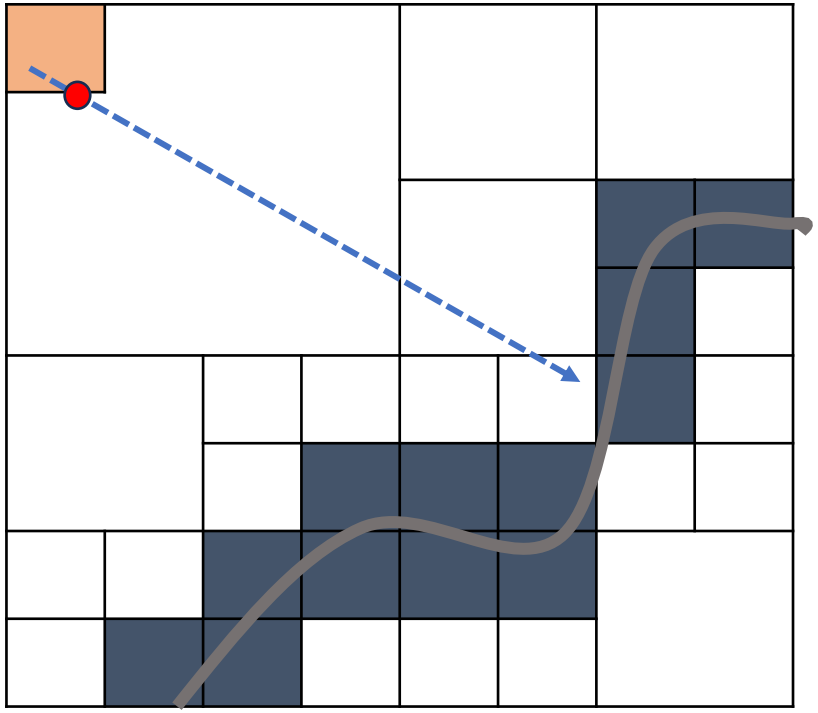
- Trace消耗点
  - 数据查询
  - Ray->AABB相交检测

利用Voxel Octree加速Trace



Trace Level: Trace步进大小

Voxel Trace

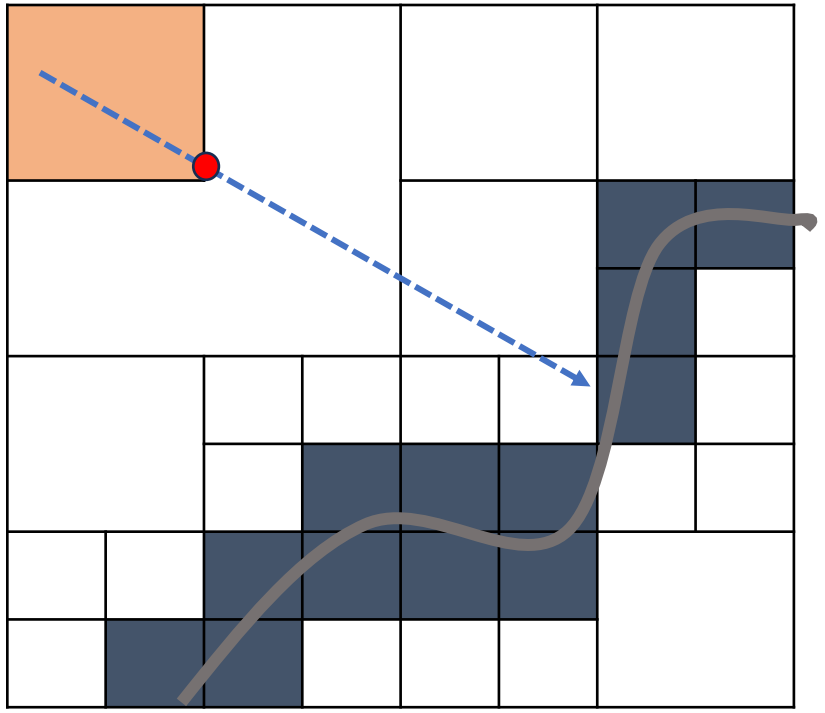


Step 1

(Level = 1) 节点为空，Trace一步，Level+1

Level1：最小层级

Voxel Trace

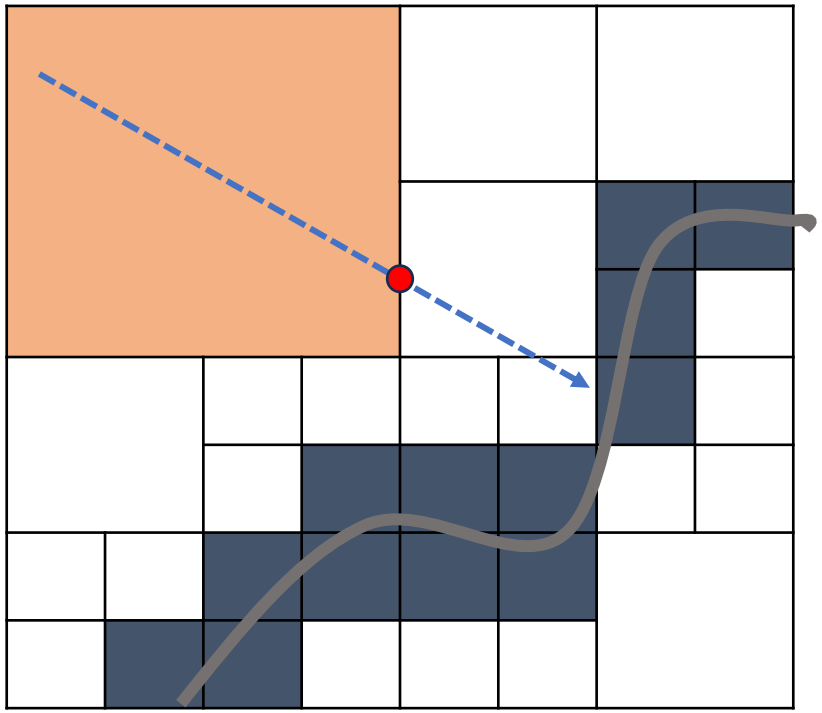


Step 2

(Level = 2) 节点为空，Trace一步，Level+1

Level1：最小层级

Voxel Trace

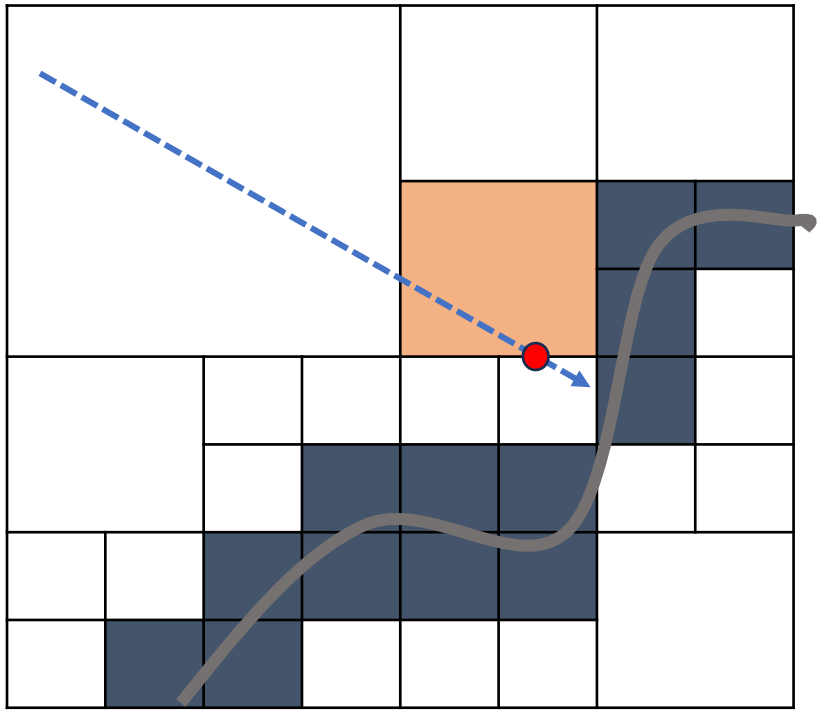


Step 3

(Level = 3) 节点为空，Trace一步，Level+1

Level1：最小层级

Voxel Trace

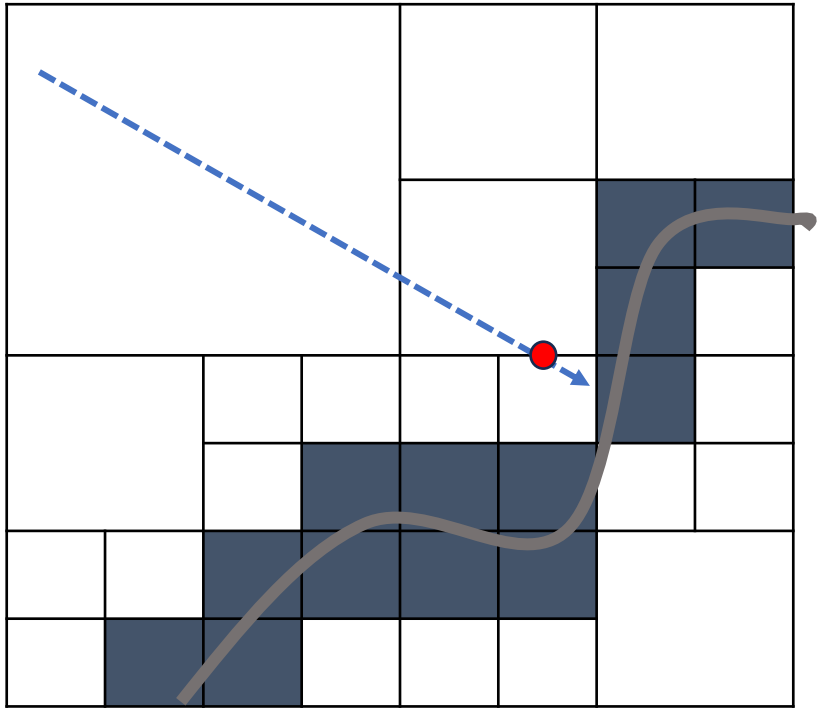


Step 4

(Level = 3) 节点不为空，Level-1，节点为空，Trace一步，Level + 1

Level1：最小层级

Voxel Trace

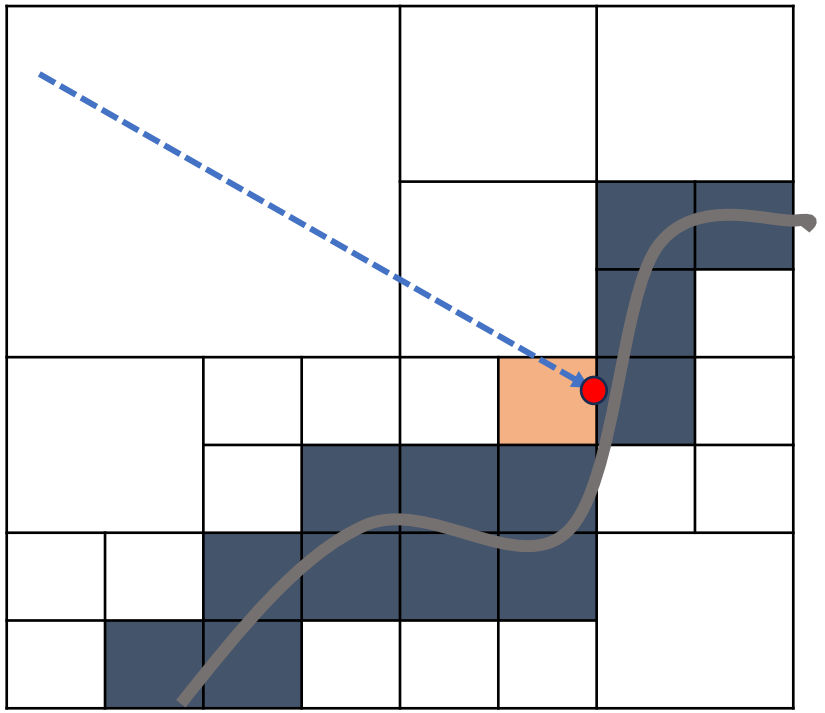


Step 5

(Level = 3) 节点不为空，Level-1，节点不为空，Level – 1

Level1： 最小层级

Voxel Trace

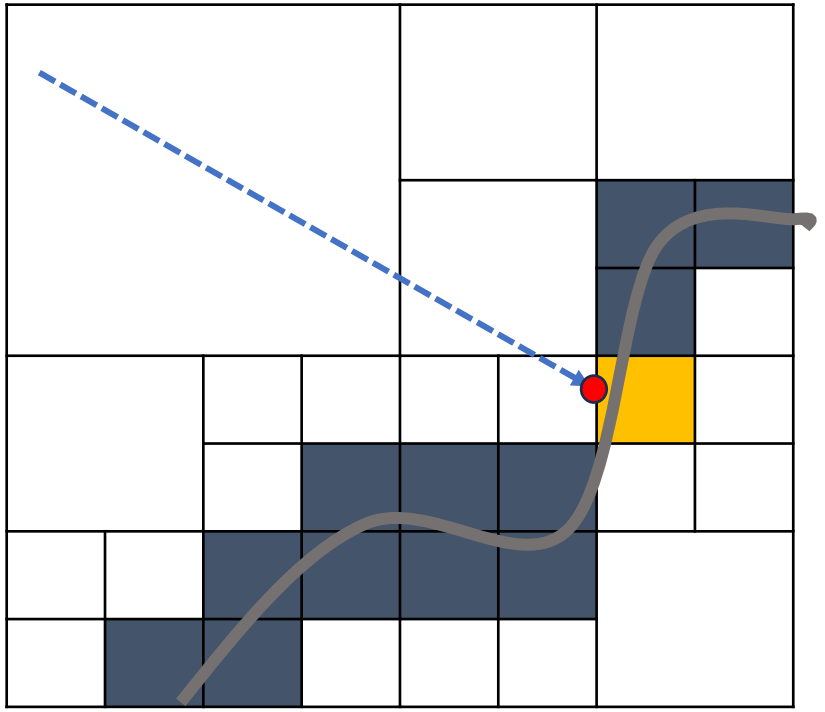


Step 6

(Level = 1) 节点为空，Trace一步， Level+1

Level1： 最小层级

Voxel Trace



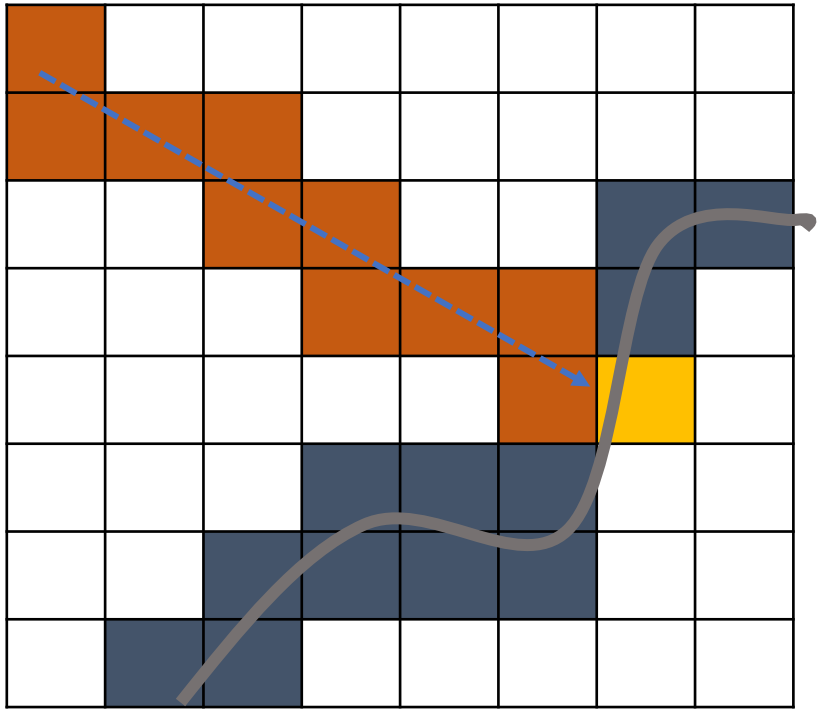
Step 7

(Level = 2) 节点不为空，Level-1，节点不为空，且当前Level=1，Trace结束

Level1： 最小层级

Voxel Trace

- 直接Trace
  - 10次相交检测
  - 11次数据查询
- Octree Trace
  - 5次相交检测
  - 10次数据查询



直接Trace

Part2 Indirect Light

目的

- 减少光照反弹采样数M*K*...

- 如何计算?
  - 射线相交检测
- 计算数量级?
  - 屏幕像素数量: N
  - 一次反弹半球采样数：M
  - 二次反弹半球采样数：K
  - .....

如何获取间接光?

- 如何计算?
  - Voxel Trace

- 在哪计算?

Screen Space Trace

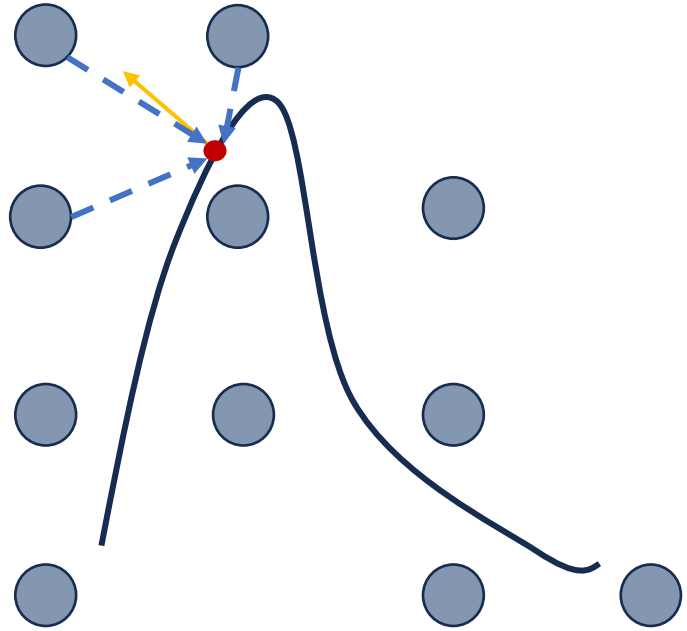
- 优点
  - 效果好
- 缺点
  - 消耗大
  - 需要降噪

逐像素计算量太大:
720p: 100w像素
1080p: 200w像素

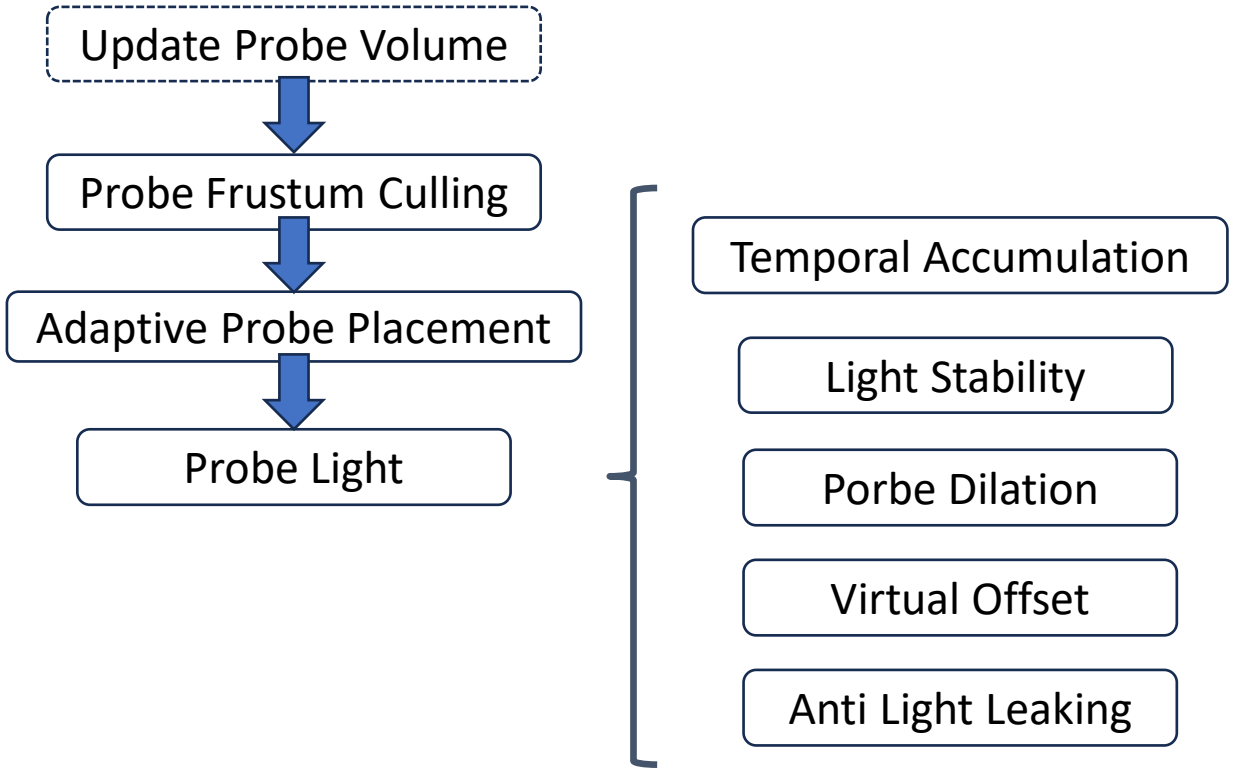
World Space Probe

- 优点
  - 可Cache
  - Trace量小（易调）
  - 无需降噪
- 缺点
  - 效果更粗糙

更新数量可调:
视野Probe：2000-4000



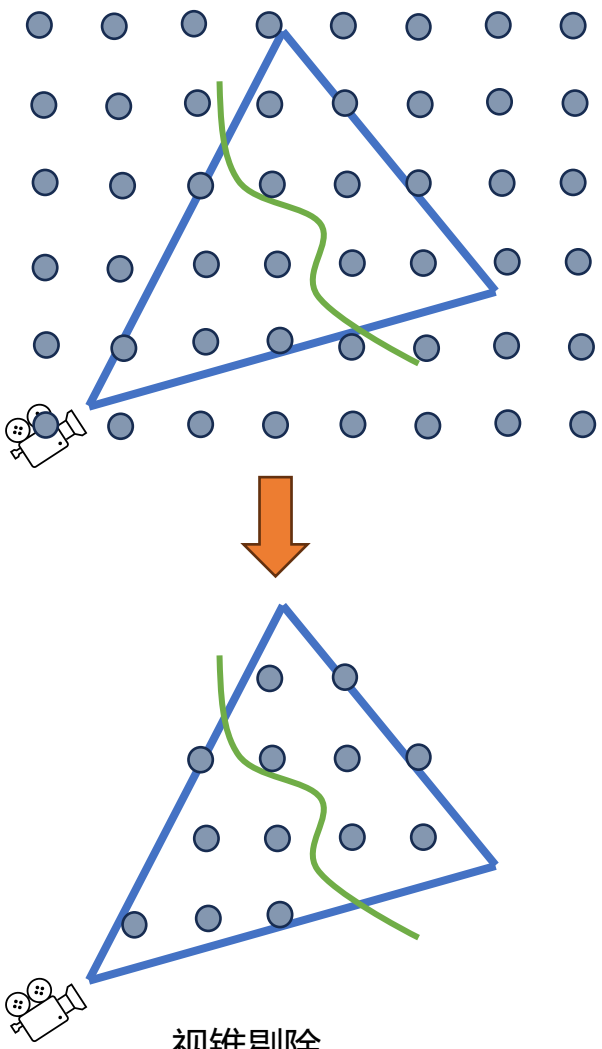
Indirect Light-总体流程



Indirect Light-Probe Frustum Culling

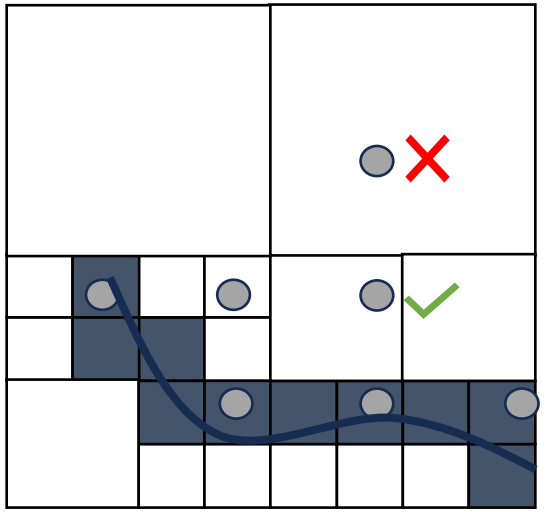
- ProbeVolume视锥剔除
  - 目的
    - 视锥剔除获得当前帧视野所有Probe
  - 方法
    - Frustum AABB Test
    - CPU粗粒度剔除

- Probe Volume Clipmap
  - 3级Probe Volume，每帧处理一级

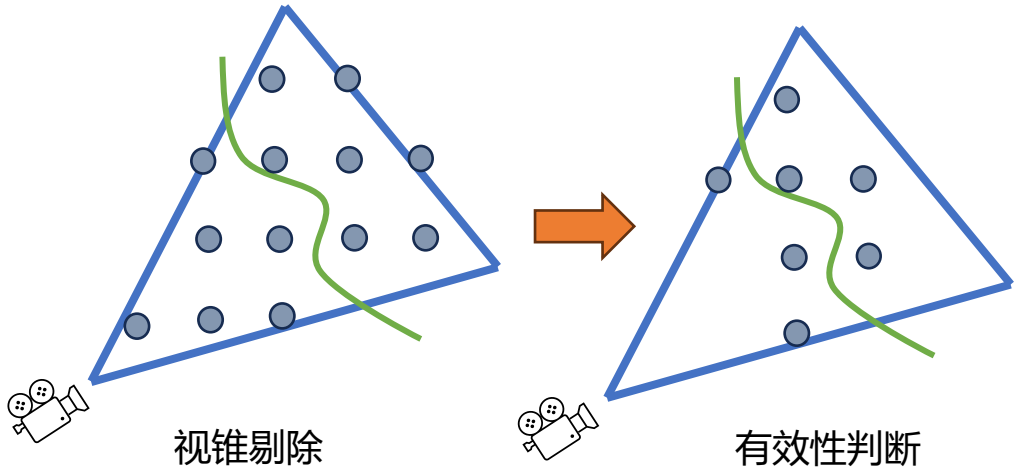


Indirect Light-Adaptive Placement

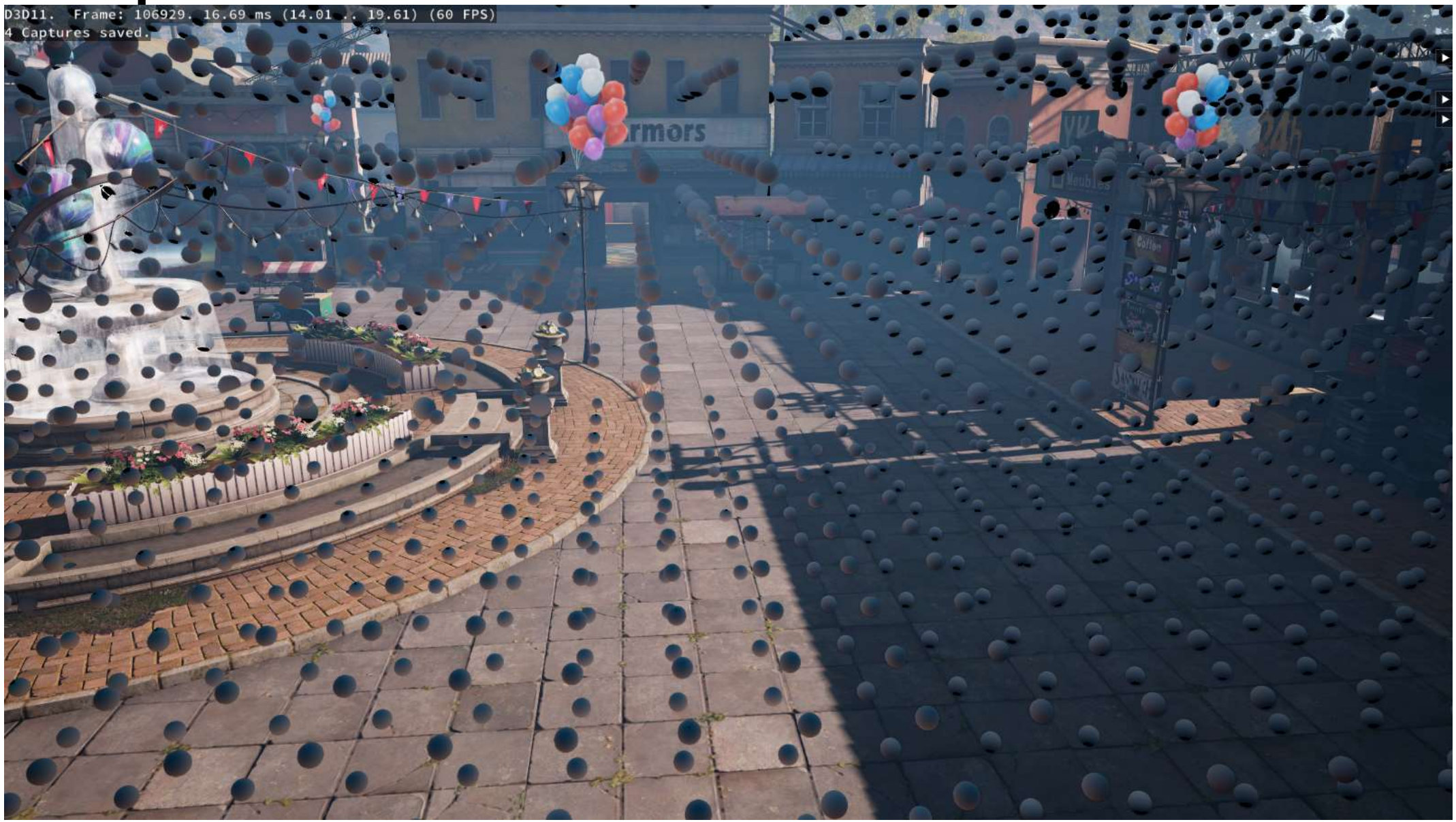
- 自适应摆放
  - 目的
    - 只在场景周围生成Probe，减少Probe更新量
  - 判断方法
    - 在Probe有效范围内查找是否存在Voxel



不同级Probe利用Voxel Octree加速查找



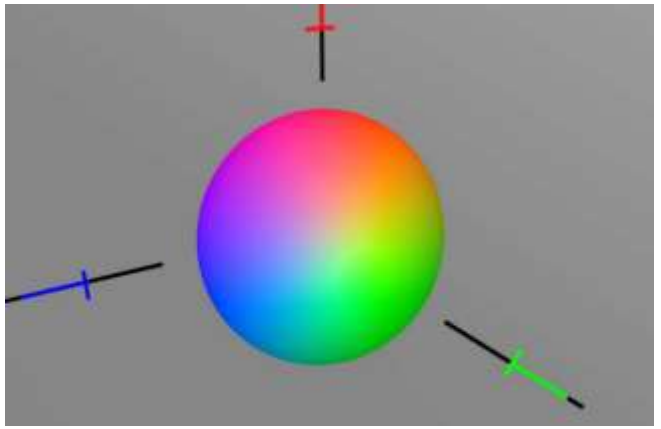
Adaptive Placement



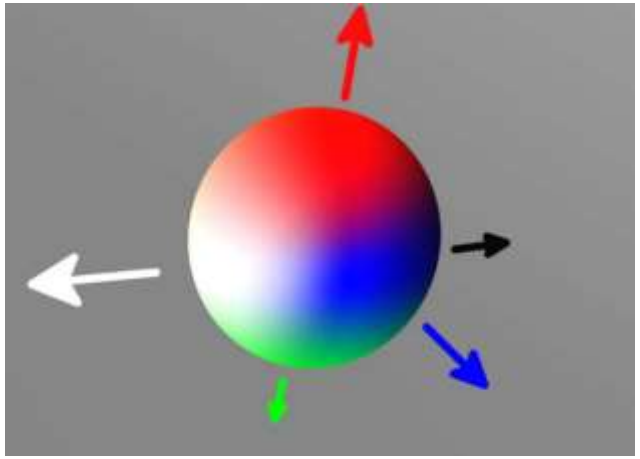
Probe数据格式选择

- Probe数据
  - SH、Ambient Cube、Octahedral map 都可以
  - Why SH(Spherical Harmonic)?
    - Diffuse GI 球谐表示足够，2阶球谐表示
    - 内存需求小，3 x half4，Ambient Cube(6 x half3), Octahedral map(4x4 half3)
    - 球面函数，插值效果平滑
    - Radiance->Irradiance转换方便，Oct map需要单独的Pass转换

8x8 Octahedral map



SH

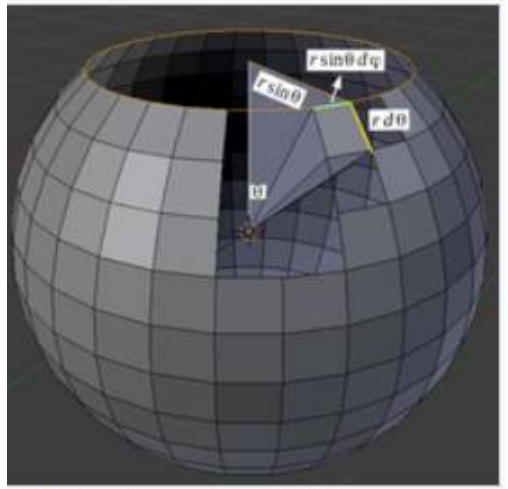


Ambient Cube

Spherical Harmoinc Light

- 投影
  - $L(\theta,\varphi)$: 球面入射光，需要计算得地方
  - L_{lm}: 最终球谐系数，结果
  - $Y_{lm}(\theta,\varphi)$: 球谐基函数
  - 球面光照信息投影到球谐基上:

$$L_{lm} = \int_{\theta=0}^{\pi}\int_{\varphi=0}^{2\pi} L(\theta,\varphi)Y_{lm}(\theta,\varphi)\underbrace{sin\theta d\theta d\varphi}_{\text{微分立体角}}$$



立体角 $d\Omega = \sin\theta\, d\theta\, d\varphi$

wiki

  - Monte Carlo积分:

$$\int_{\theta=0}^{\pi}\int_{\varphi=0}^{2\pi} sin\theta d\theta d\varphi = 4\pi \quad pdf = \frac{1}{4\pi}$$

$$L_{lm} = \frac{1}{N}\sum_{i=1}^{N}\frac{L(\theta_i,\varphi_i)}{pdf}Y_{lm}(\theta_i,\varphi_i) = \frac{4\pi}{N}\sum_{i=1}^{N}L(\theta_i,\varphi_i)Y_{lm}(\theta_i,\varphi_i)$$

  - $Y_{lm}(\theta,\varphi)$: 2阶球谐，已知 $Y_{00} = \sqrt{\frac{1}{4\pi}}, Y_{1m} = \sqrt{\frac{3}{4\pi}}(x,y,z)$
  - 只需求出$L(\theta_i,\varphi_i)$即可

Spherical Harmoinc Light

- 重建

$$L(\theta,\varphi)=\sum_{l,m}L_{lm}Y_{lm}(\theta,\varphi)$$

  - 球谐系数和指定方向球谐级函数点积即可重建当前方向得Radiance
  - 渲染中主要考虑的是Irradiance，需要从Radiance球谐获得Irradiance
    - 由于是DiffuseGI，BRDF与入射光方向无关

$$E(n)=\int_{\Omega}L_i(\omega_i)\mathrm{n}\cdot\ \omega_i d\omega_i$$

  - 球谐cosin lobe具有解析解：

$$E(n)=\sum_{l,m}\sqrt{\frac{4\pi}{2l+1}}A_l L_l^m Y_l^m(n)$$

$$A_0=\sqrt{\frac{\pi}{4}}\qquad A_1=\sqrt{\frac{\pi}{3}}$$

$$A_l=\int_0^{2\pi}\int_0^{\pi}\lfloor\cos(\theta_i)\rfloor Y_l^0(\theta_i)\sin(\theta_i)\,\mathrm{d}\theta_i\,\mathrm{d}\phi_i$$

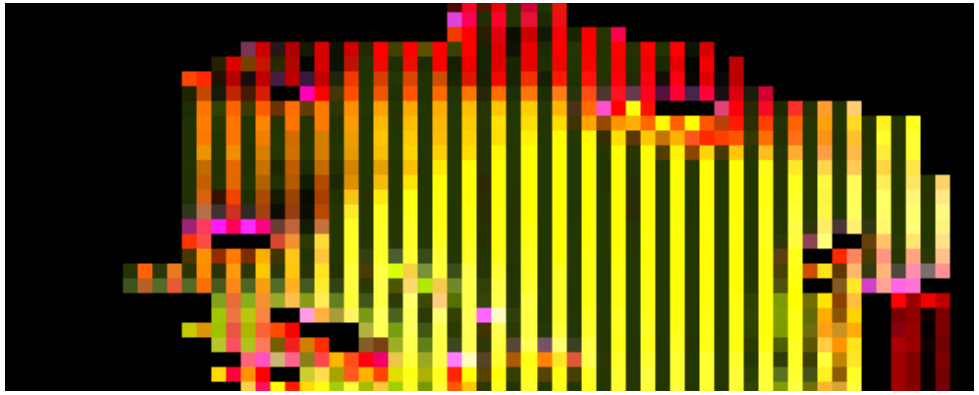
$$=2\pi\int_0^{\frac{\pi}{2}}\cos(\theta_i)Y_l^0(\theta_i)\sin(\theta_i)\,\mathrm{d}\theta_i\qquad(4)$$

细节：https://patapom.com/blog/SHPortal/

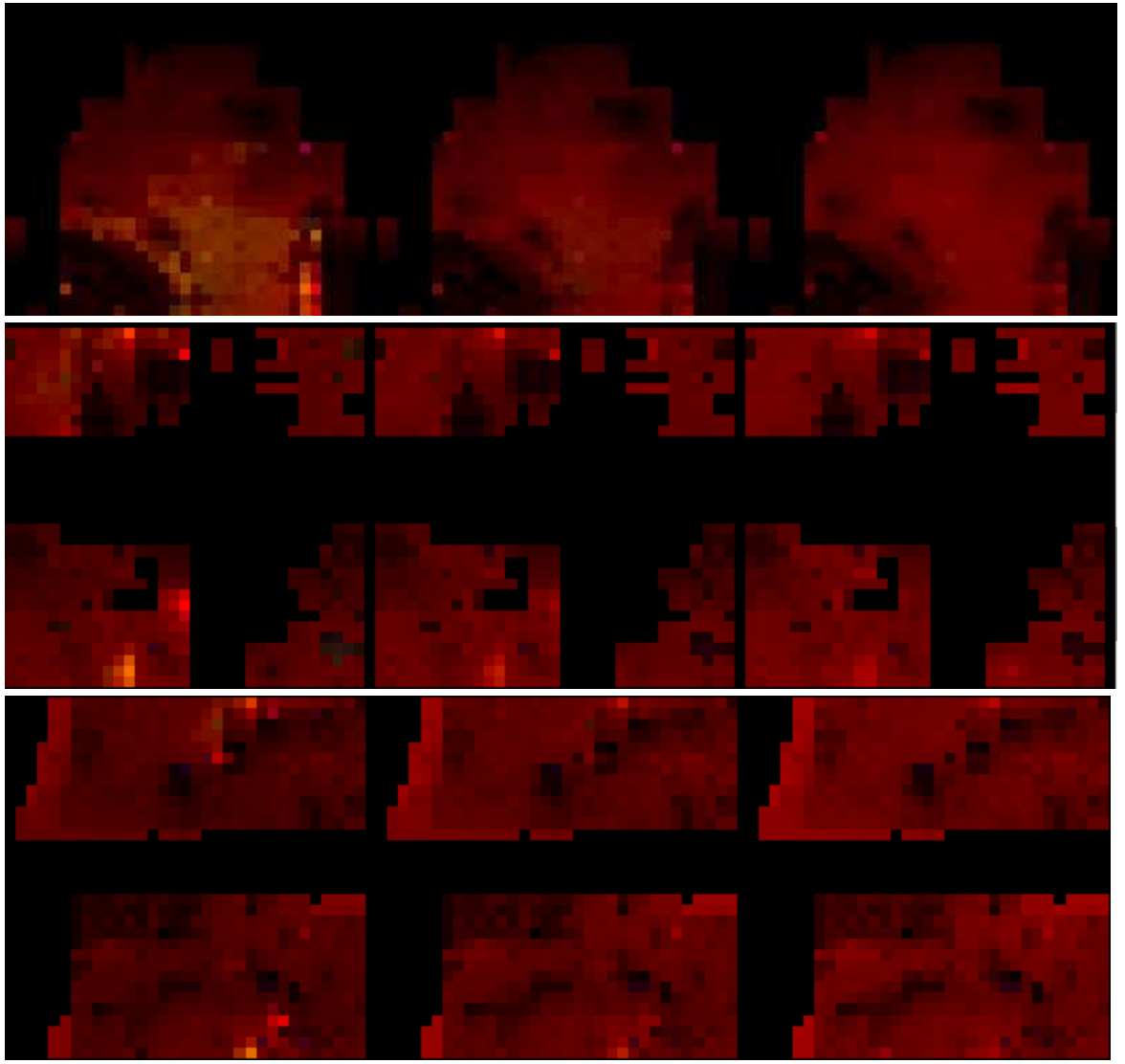
  - 结果十分简单，对0阶，1阶基分别乘π，$\frac{2\pi}{3}$

Indirect Light-Probe Light

- 如何Trace?
  - 球面均匀随机方向采样
  - Halton序列生成采样方向
  - 1024 rays, 32 rays per frame
- Trace存储的数据
  - Indirect/Sky Light (SH2)
  - Distance (SH2)
  - Normal_HitDir_Dot (SH2)



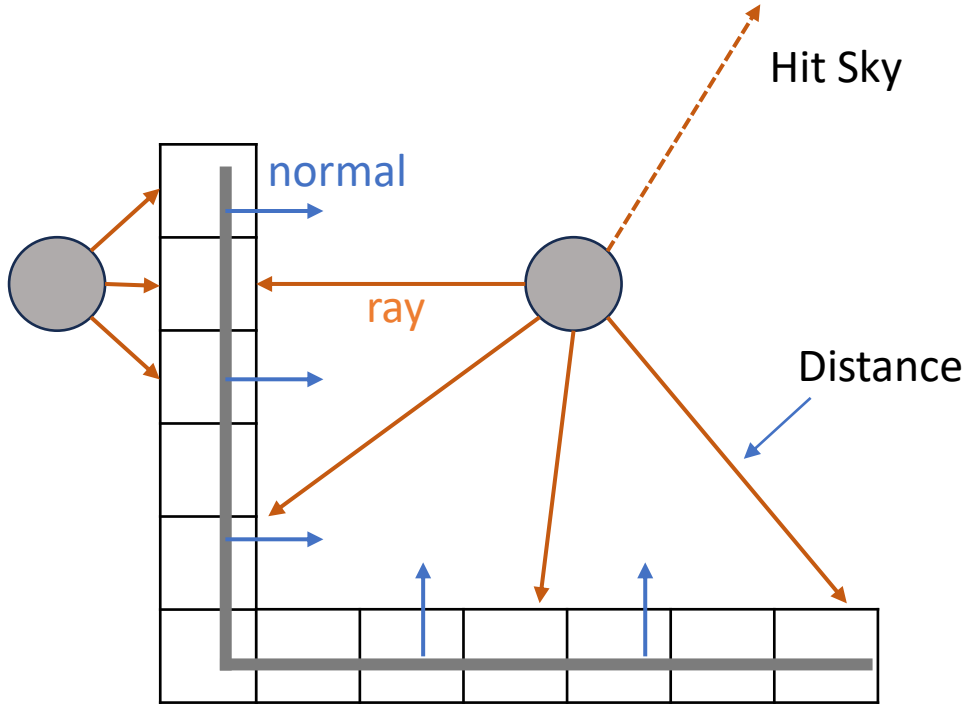
Distance+Normal_HitDir_Dot



Indirect/sky Light

Indirect Light-Probe Light

- Indirect/Sky Light
  - Hit: Voxel Color
  - No Hit: Sky Light (Sky Visibility)
- Distance
- Normal_HitDir_Dot
  - dot_value = dot(-ray_dir, normal_dir)
  - 用于区分Probe位于Mesh正反面



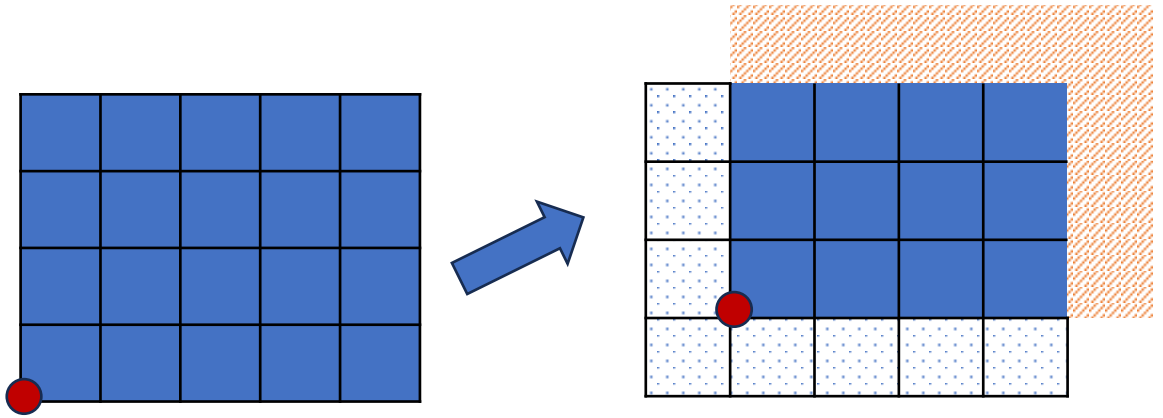
Sky Visibility很重要!

No Sky Visibility

Sky Visibility

Indirect Light-Volume Shift

- Volume移动只更新变化部分
- 只改变VolumeTexture起始坐标
- 硬件采样时需要拷贝边界



Indirect Light

- 一帧Trace的数量有限，如何保证数据准确性？

Indirect Light-Temporal Accumulation

- 一帧的计算量有限，需要历史帧累积
- Temporal 累积基本计算方式

$$S = (1 - alpha) * S_{history} + alpha * S_{cur}$$

缺点：固定alpha更新不够及时

- 改进：增加历史累计帧数m

alpha = 1/(m+1) = 1　　alpha = 1/(m+1) = 0.5　　alpha = 1/(m+1) = 0.333　　....

第0帧, m = 0　　第1帧, m = 1　　第2帧, m = 2

* 第一帧值为x，第N帧为x/N

Indirect Light-Temporal Accumulation

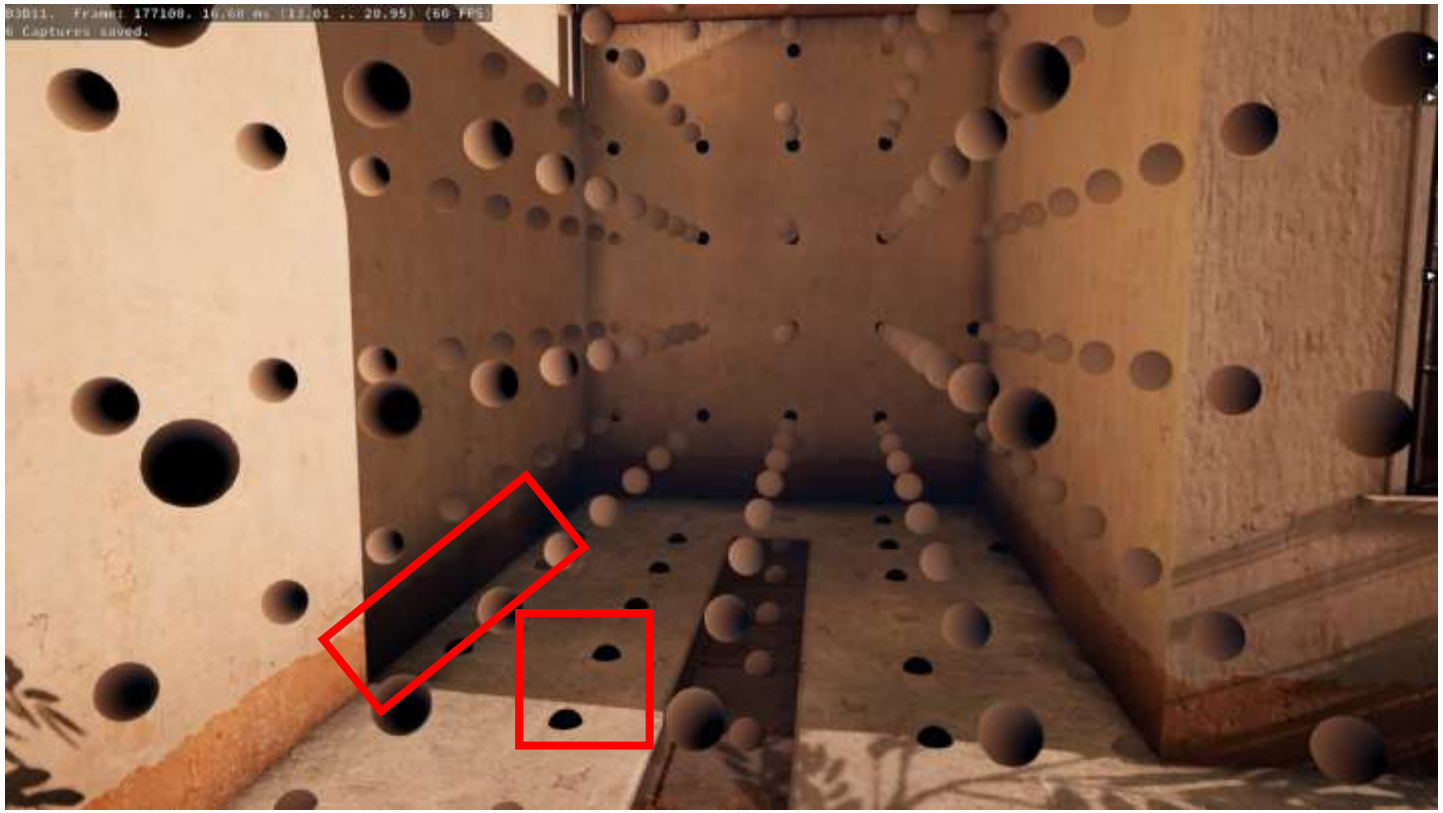
- 结果对比

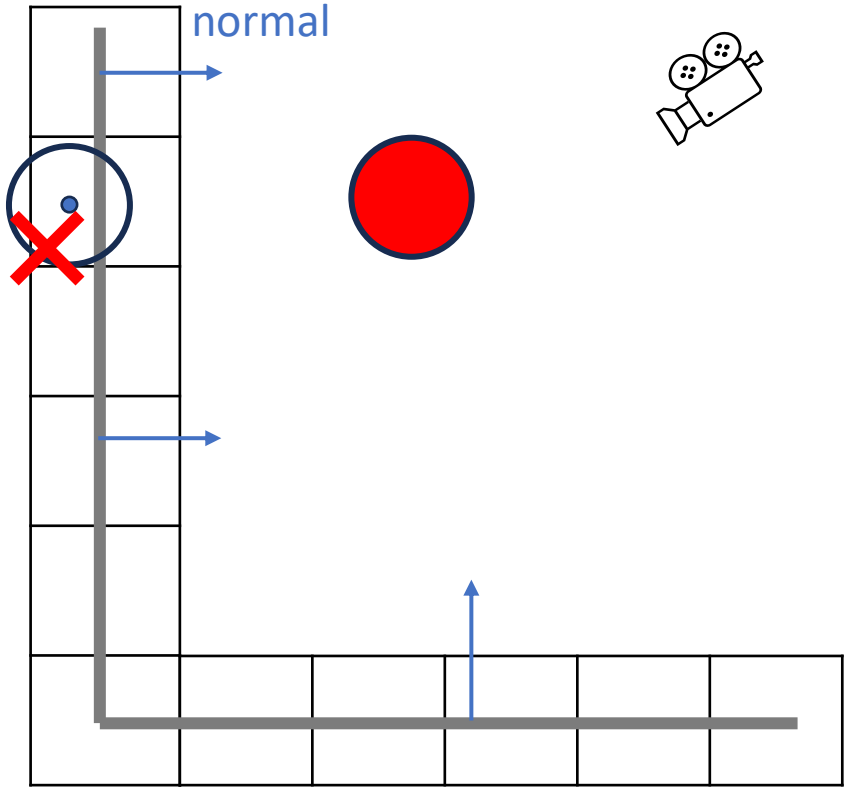
固定alpha

动态alpha

Indirect Light

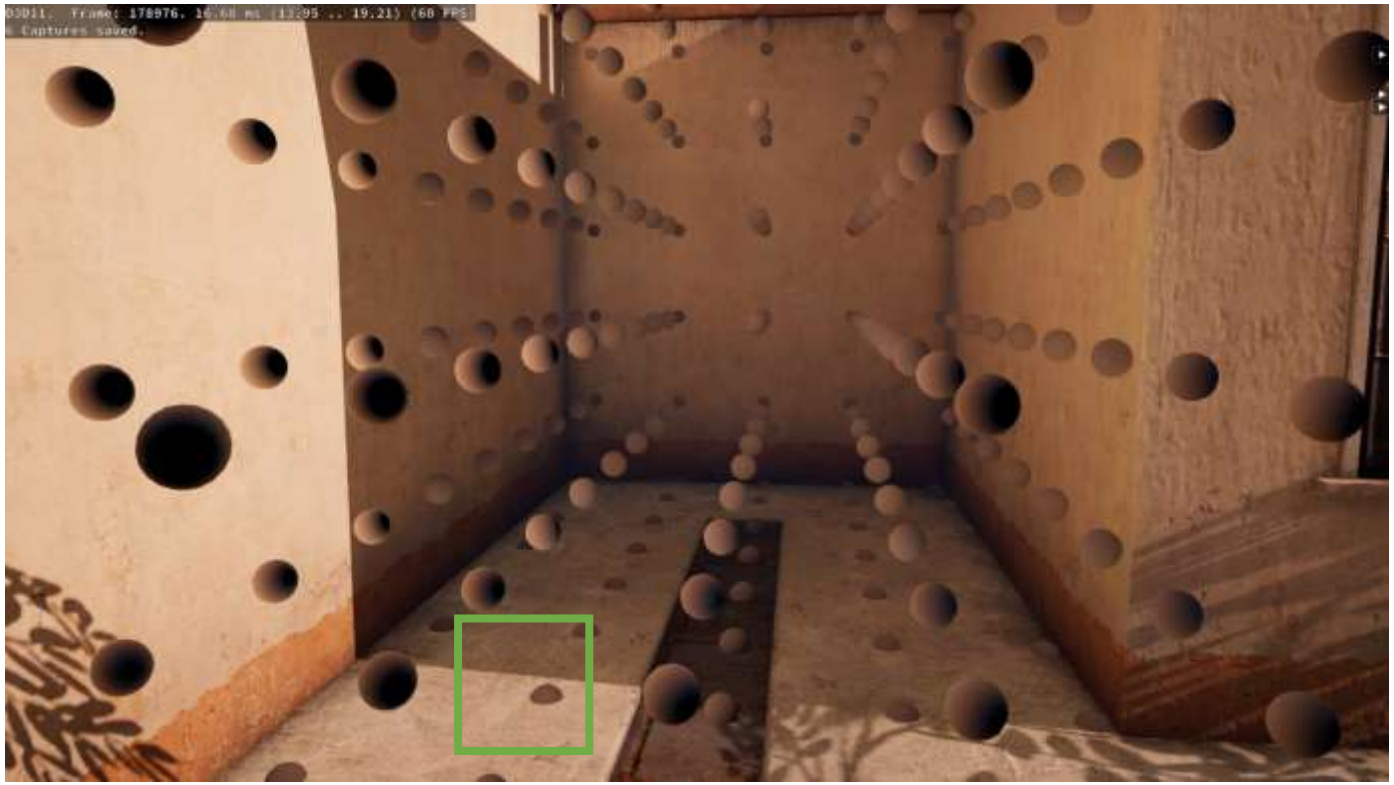
- 问题
  - 画面有奇怪的黑色
- 原因
  - Probe在体素里面

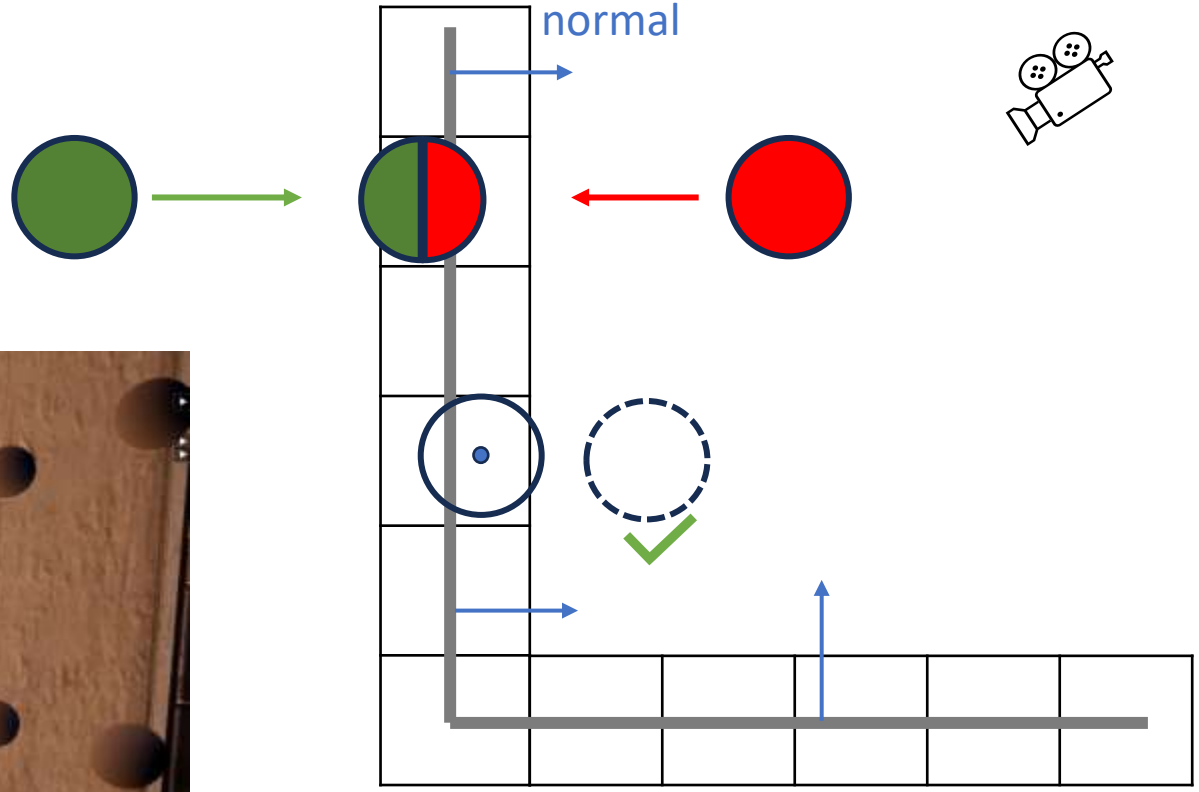




Indirect Light-Probe Dilation

- 解决方法
  - Case 1：SceneDepth之后
    - Dilation: 混合邻接Probe
  - Case 2：SceneDepth之前
    - Virtual Offset：法线方向偏移

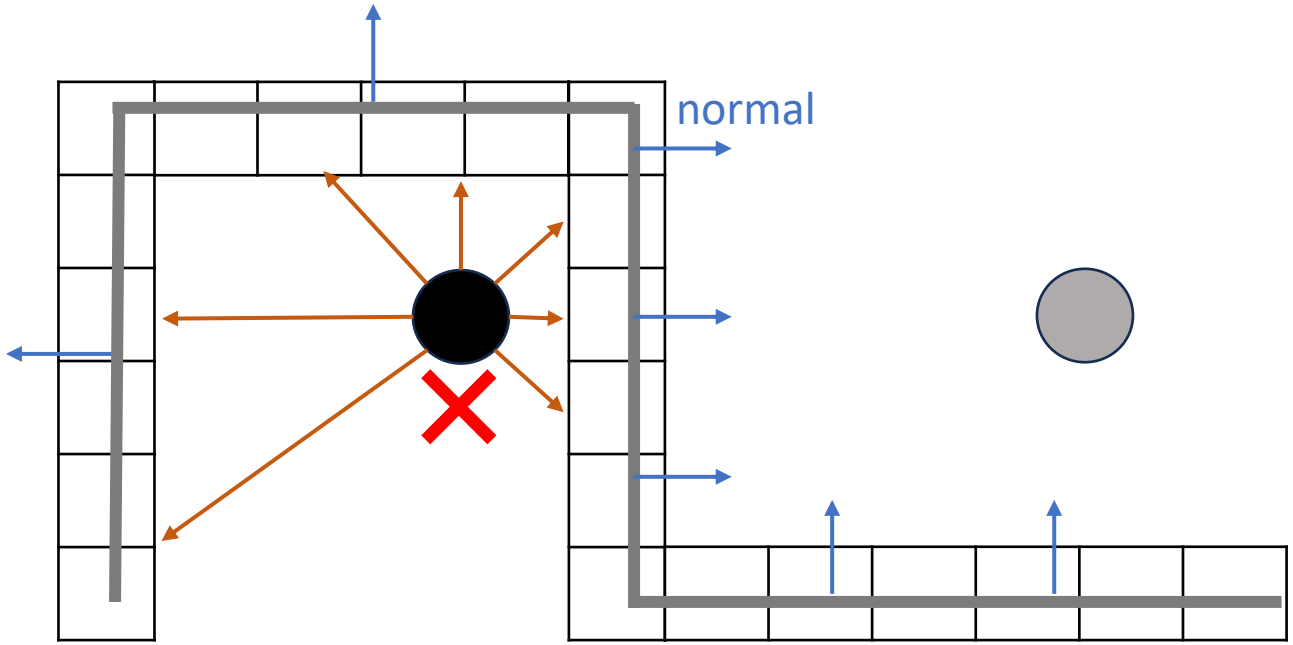




Indirect Light

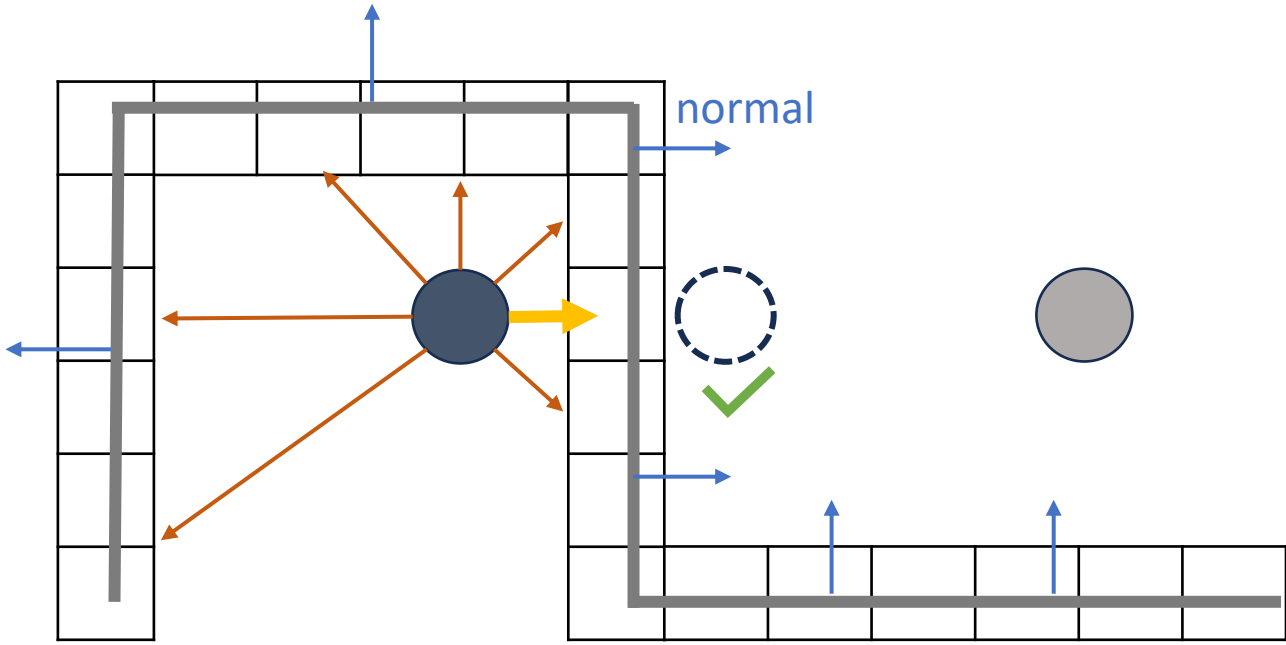
- Dilation后依然有问题
  - Probe In Geometroy(BackFace)

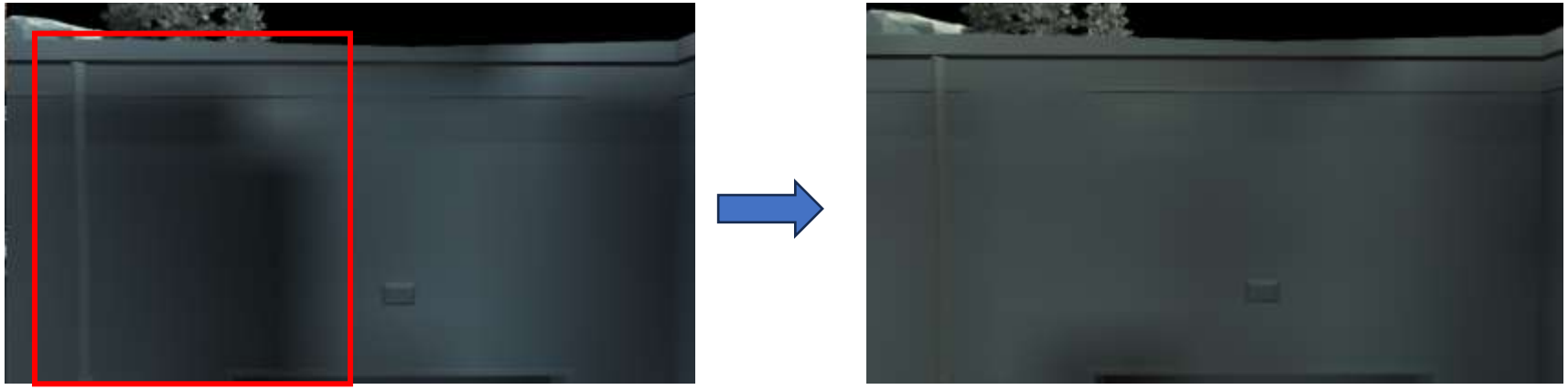
Probe Dilation



Indirect Light-Virtual Offset

- 解决方法
  - Probe偏移
    - 如何判断是否需要偏移?
      - Min Normal_HitDir_Dot < 0
    - 偏移方向?
      - Min Distance Direction



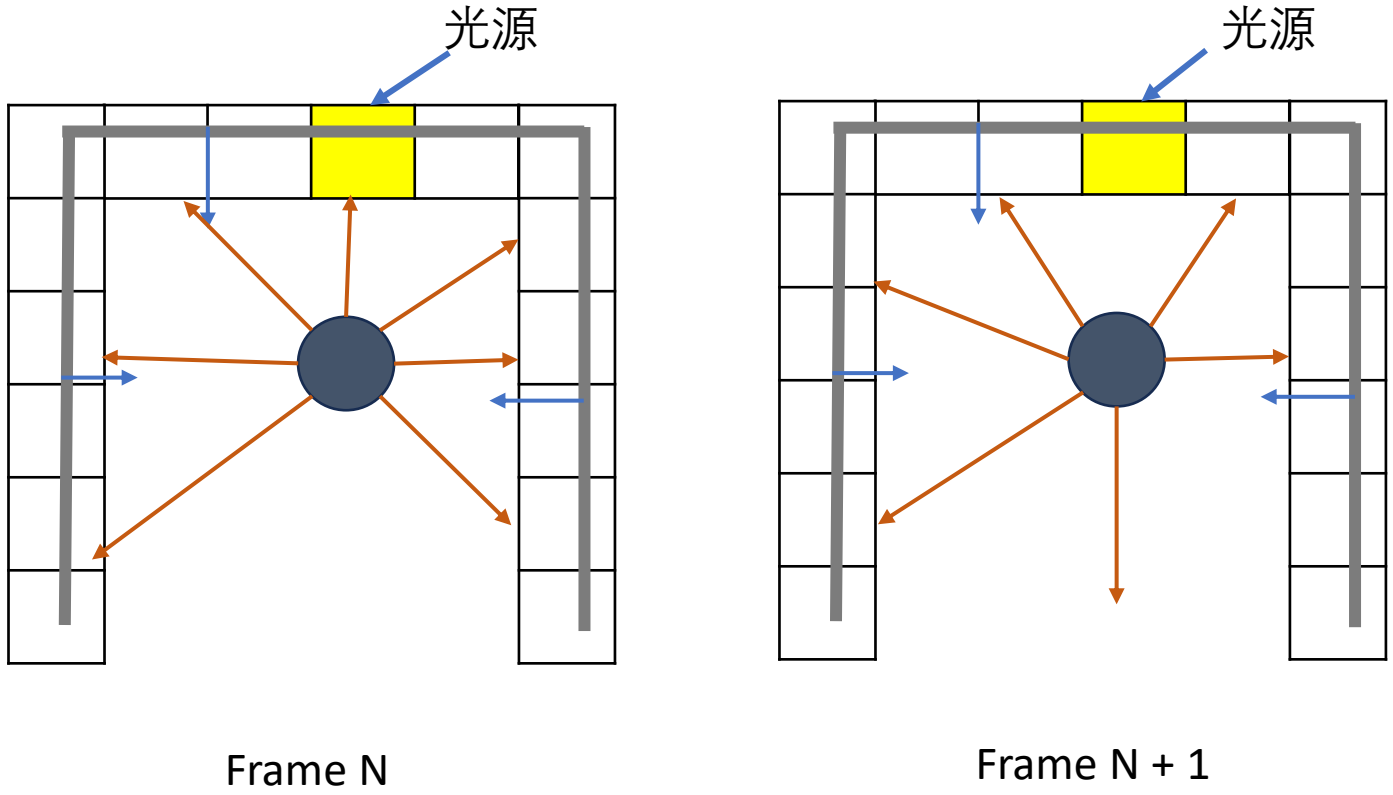


Indirect Light

- 出现的问题
  - 暗光画面闪烁

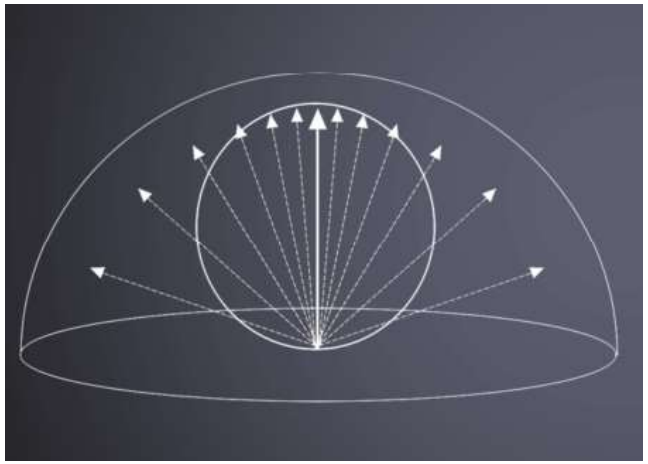
Indirect Light-Light Stability

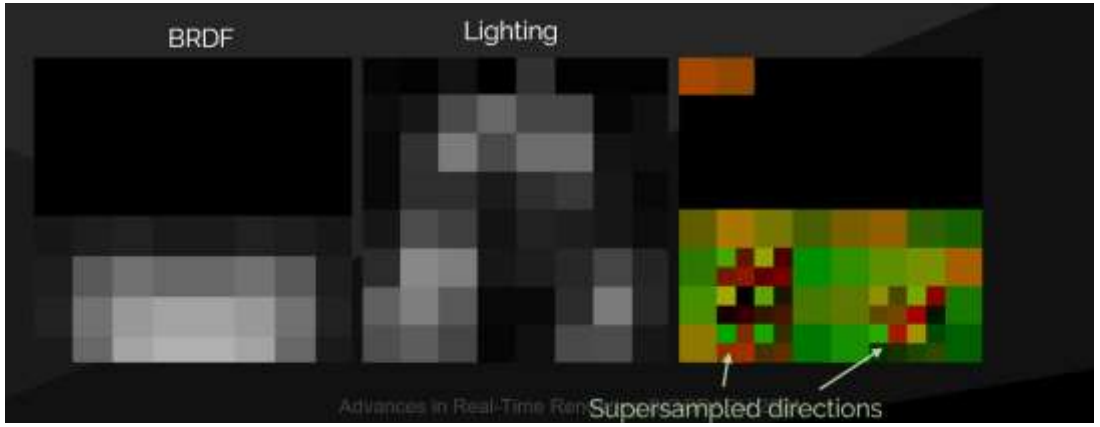
- 原因：
  - 光源少，Trace命中率低
  - 前后帧Trace结果方差太大
  - Temporal不能完全解决

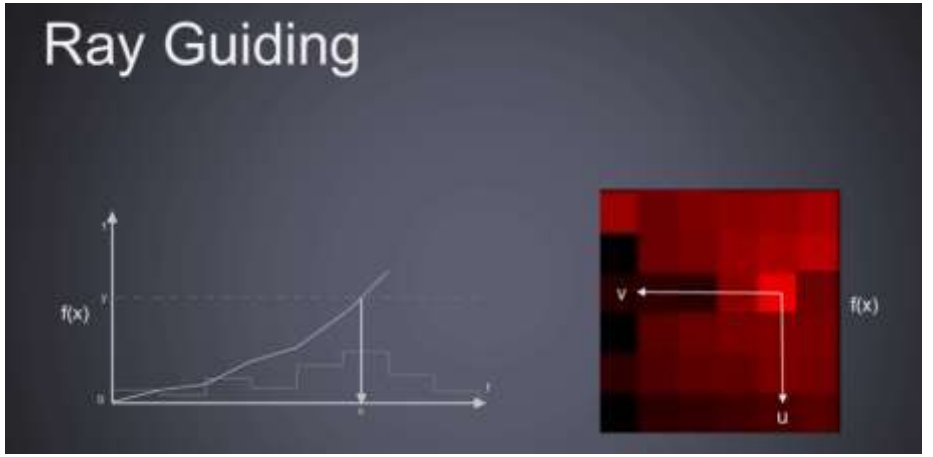


Indirect Light-Light Stability

- 如何解决
  - 重要性采样
    - BRDF Importance Sampling
    - Lumen: Structured Importance Sampling
    - Surfel GI: Ray Guiding







本质就是根据Trace历史结果的亮度作为pdf调整当前Trace方向分布

Restir(Resvoir Spatial-Temporal Importance Resample)

- 重要性采样，Monte Carlo积分 $I = \sum_1^N \frac{f(x)}{p(x)}$
  - p(x)需要方便计算或采样（我们无法获得亮度分布）

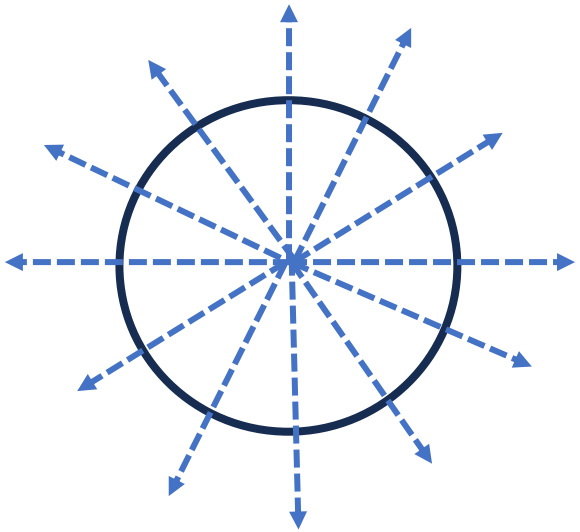
- Resampled Importance Sampling(RIS) 重要性重采样（无需知道精确目标分布）
  - 步骤：
    - 根据已知分布$\hat{p}(x)$生成指定数量样本
    - 根据proposal pdf $p(x)$为每个样本计算权重$\frac{p(x)}{\hat{p}(x)}$
    - 根据权重$\frac{p(x)}{\hat{p}(x)}$重新生成指定数量样本
  - 缺点：
    - 需要提前生成一定数量样本，并重新生成样本，O(N)复杂度
- Weighted Reservoir Sampling(WRS)
  - 利用蓄水池算法解决RIS的问题，O(1)复杂度
- Temporal Reuse
  - 历史数据重用

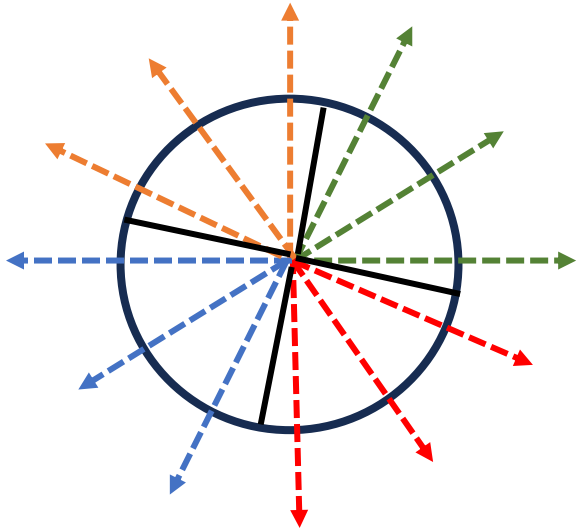
$$\hat{p}(x) = \frac{1}{4\pi}$$

$$p(x) = \text{Sample Luminance}$$

Indirect Light-Light Stability

- 一个蓄水池无法描述一个球面的分布
  - 分层采样
    - 对球面均匀划块
    - 每块维护一个蓄水池
- 根据历史数据的最大亮度限制当前帧亮度





Indirect Light-Light Stability

- 优化结果

Indirect Light-Light Stability

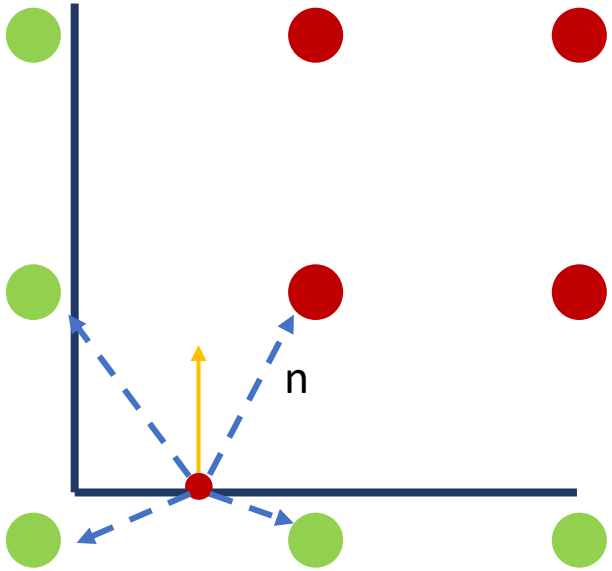
- 移动时新增Probe默认值为0，有明显闪烁

Indirect Light-Light Stability

- 解决方法
  - 初始化利用高一级Probe Volume

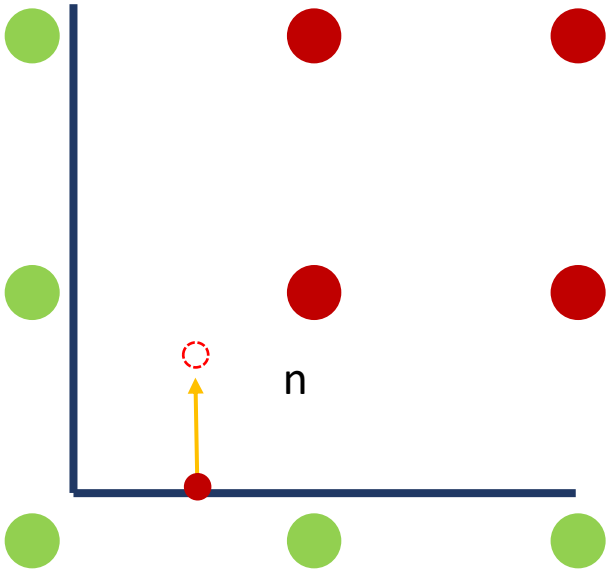
Indirect Light

- 漏光
  - 产生原因
    - Probe采样



Light Leaking-解决方法

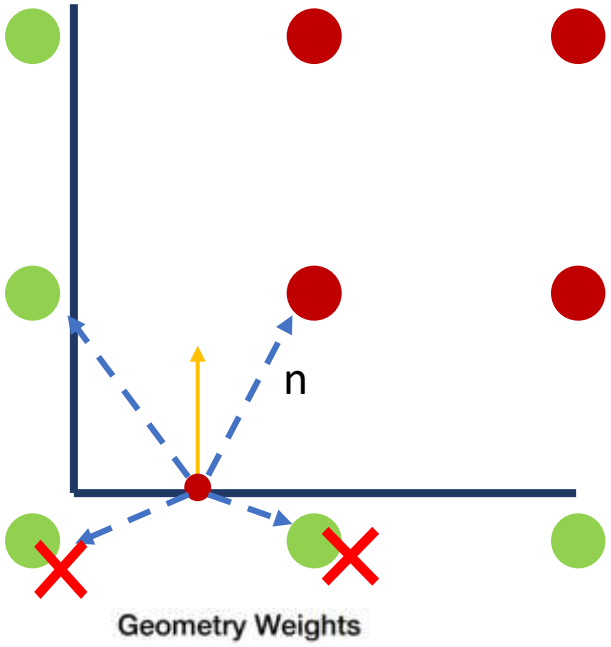
- Normal Bias

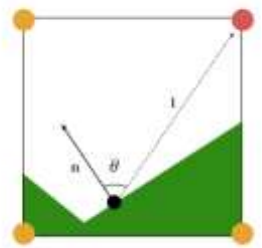


效果图： Normal Bias

Light Leaking-解决方法

- Normal Weight

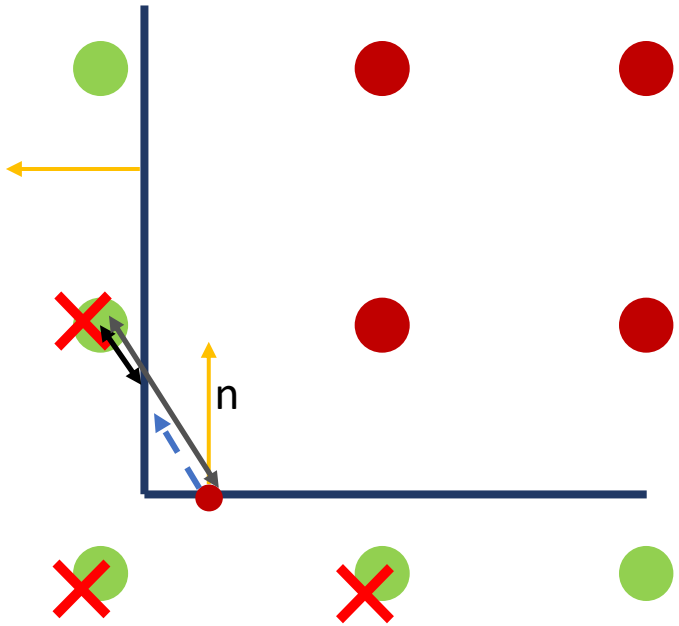




效果图： Normal Bias+Normal Weight

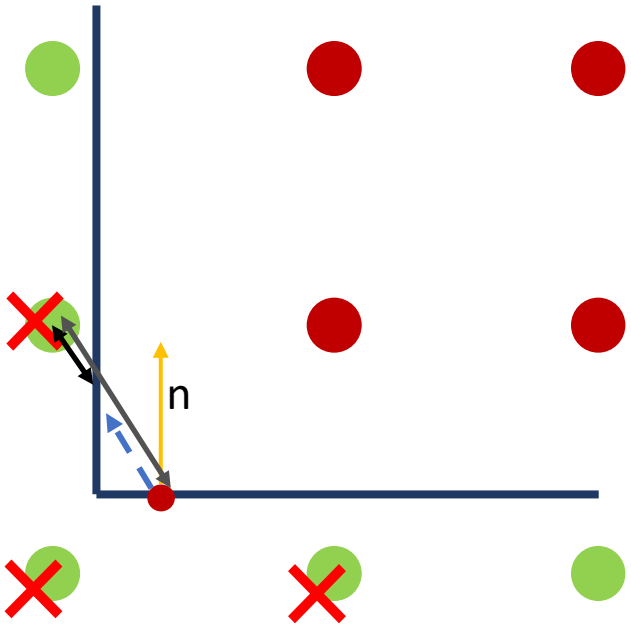
Light Leaking

- Distance Weight
  - 法线权重无法解决墙角处漏光
  - 比较到采样点的距离与Trace距离
  - 可以再缓解一部分漏光
    更好的解决需要用DDGI方案



Light Leaking-解决方法

- Distance Weight
  - 比较Probe到着色点的距离与 Probe Trace距离
  - 可以再缓解一部分漏光
    更好的解决可以用DDGI方案



效果图： Normal Bias+Normal Weight+Distance Weight

Part3 Shading

目的

- 减少屏幕像素采样数N

- 如何计算?
  - 射线相交检测
- 计算数量级?
  - 屏幕像素数量: N
  - 一次反弹半球采样数：M
  - 二次反弹半球采样数：K
  - .....

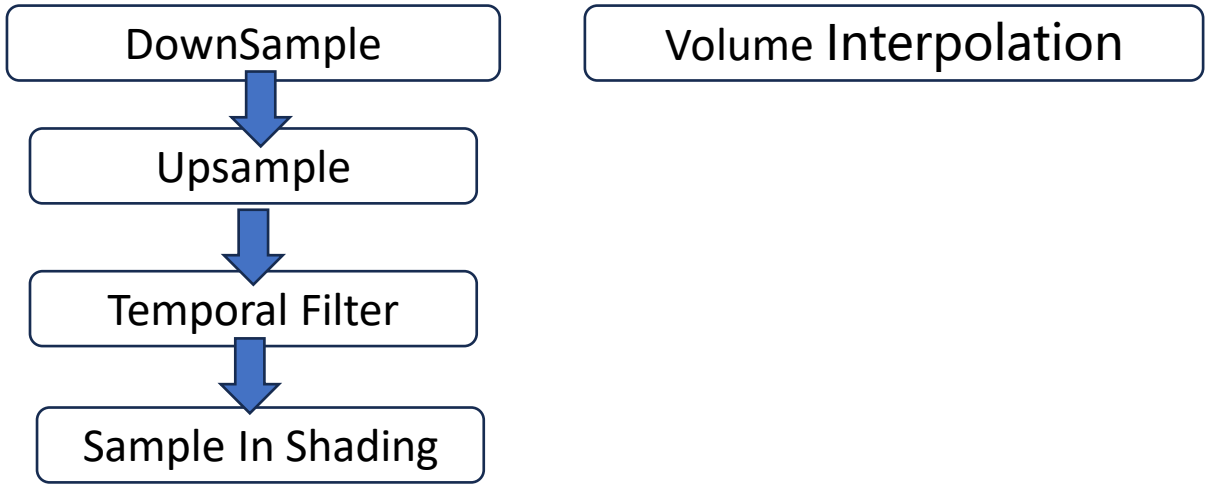
Light Gather

- Shading消耗来源
  - SH Texture3D的采样
  - 防漏光计算自定义权重

- 采样方法
  - Per Pixel Sample
    - 优点
      - 操作简单直接
      - 支持Forward/Deferred管线
    - 缺点
      - 逐像素消耗太高
  - DownSample
    - 优点
      - 性能友好
    - 缺点
      - 处理更复杂，需要Upsample
      - 对Forward支持不友好

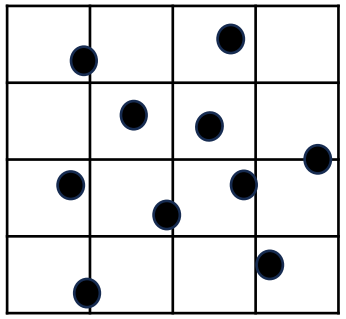
Light Gather

- 流程

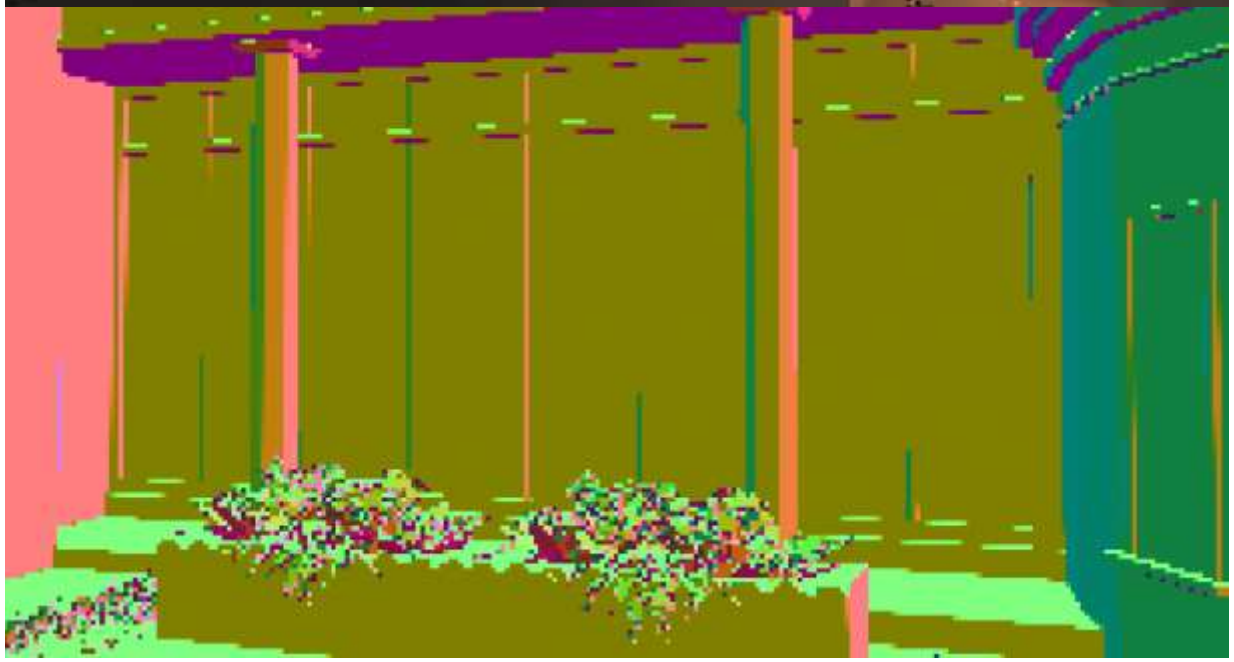


DownSample

- Probe Volume 采样
  - 自定义权重采样（处理漏光）
- 为什么可以DownSample?
  - Probe本身精度较低，降采样没有明显效果损失
- 1Sample per 4x4 pixels
  - Temporal Jitter
    - 保证场景信息完整

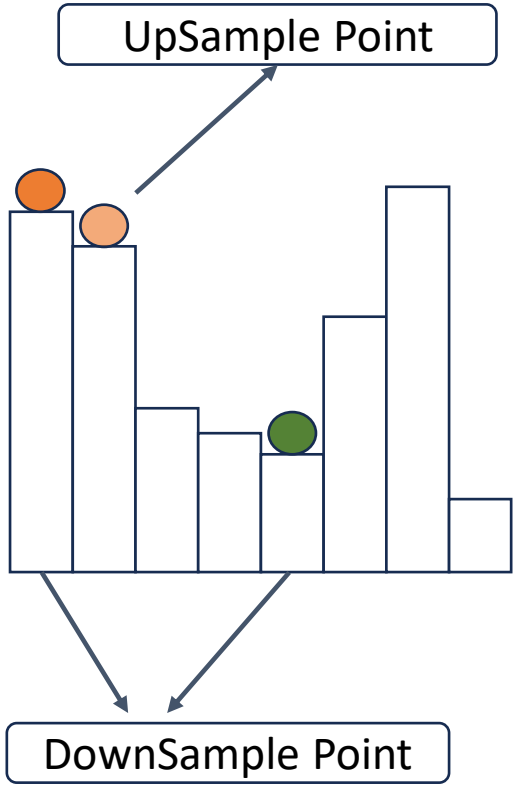


- 输出
  - Indirect Light
  - Normal+Depth



UpSample

- UpSample to ½ Render Resolution
- 双边滤波，深度、法线权重



Temporal Filter

- 对Upsample后的结果Temporal Filter

初始降采样

最终结果

Shading

- Shading时直接从最终间接光贴图采样

GI

Shading

- 问题
  - 不同级别的Probe采样接缝明显

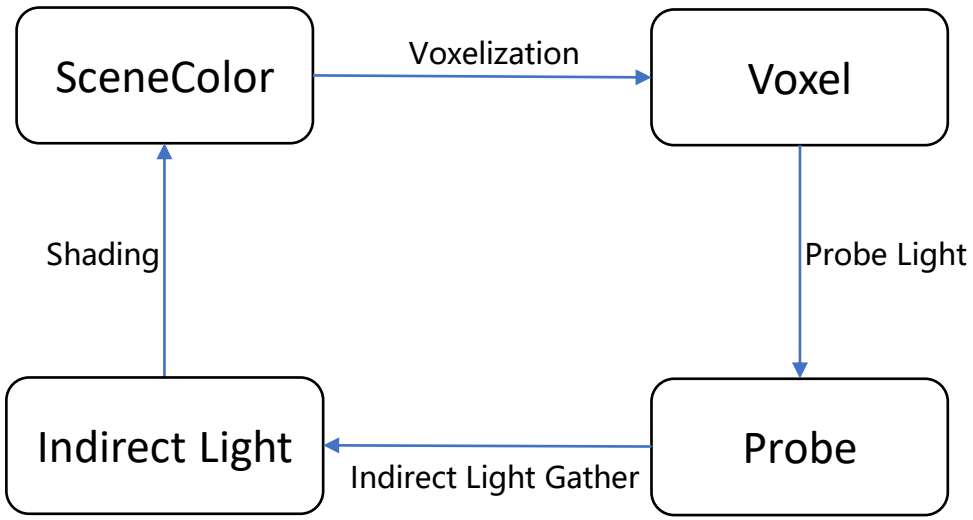
Probe Volume Interpolation

- 解决方法
  - Noise + Light Gather Temporal Filter

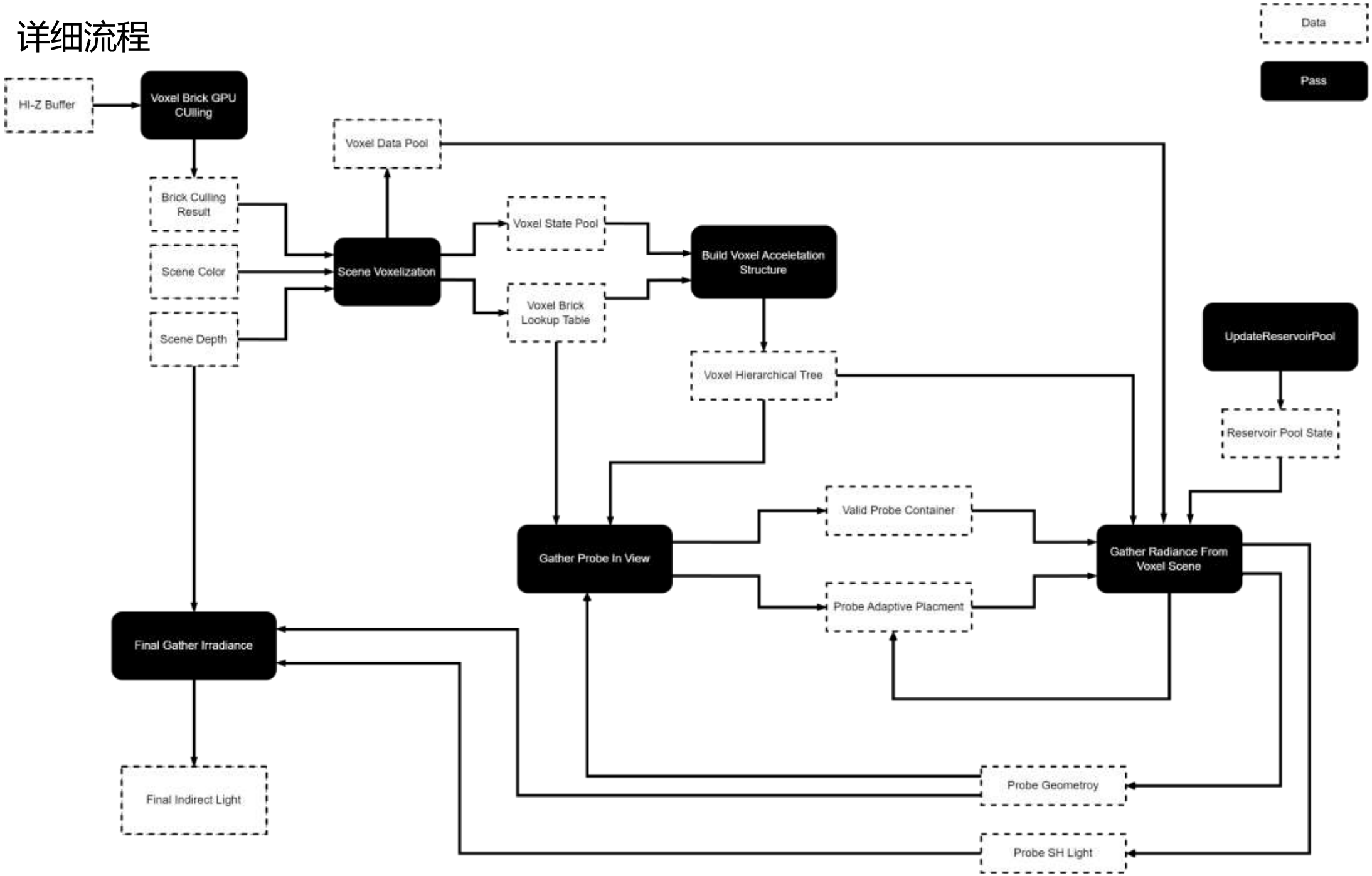
Part4 总结

总结

- 大致流程
  - Voxelization
  - Probe Light
  - Indirect Light Gather
  - Shading



详细流程


Data
Pass
HI-Z Buffer
Voxel Brick GPU CUlling
Voxel Data Pool
Brick Culling Result
Scene Color
Scene Depth
Scene Voxelization
Voxel State Pool
Voxel Brick Lookup Table
Build Voxel Acceletation Structure
Voxel Hierarchical Tree
UpdateReservoirPool
Reservoir Pool State
Valid Probe Container
Gather Probe In View
Probe Adaptive Placment
Gather Radiance From Voxel Scene
Final Gather Irradiance
Final Indirect Light
Probe Geometroy
Probe SH Light

Memory

120m x 120m x 40m范围

Resource	Type	Size	Format	Video Memory(MB)
t_brick_lookup_table	Texture3D	100x100x32	r32	1.22
t_scene_voxel_iso	Texture3D	1024x512x4	rgba16f	16
t_hierarchical_voxel_as_lod0	Texture3D	200x100x32	r32	2.44
t_hierarchical_voxel_as_lod1	Texture3D	50x25x8	r32	0.04
t_aniso_voxel_lookup_table	Texture3D	120x120x48	r32	3
t_scene_voxel_aniso	Texture3D	768x256	rgba16f	1.5
t_probe_placement_state	Texture3D	60x60x20	r32	0.27
t_probe_volume_sh2	Texture3D	183x61x21	rgba16f	1.8
b_allocated_free_far_brick	Buffer	65536	rgba16f	0.25
b_allocated_free_near_voxel	Buffer	65536	rgba16f	0.125
b_probe_coord_container	Buffer	32768	rgba16f	0.125

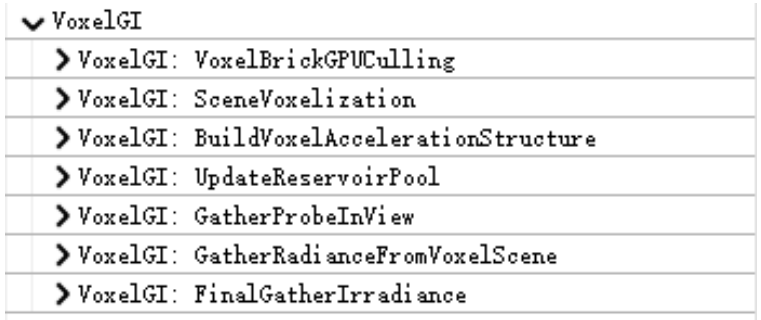
GI数据: 30mb左右

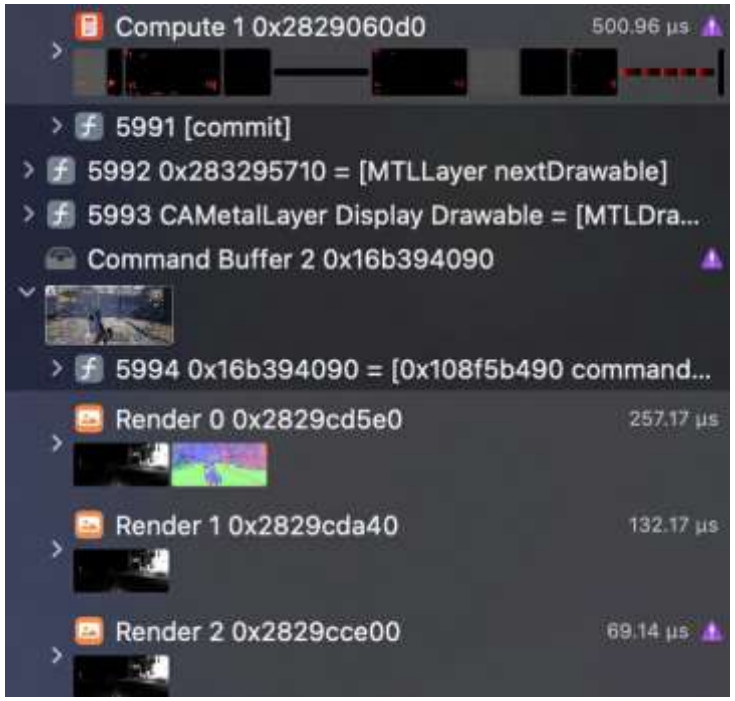
Final Gather: 10mb (与屏幕分辨率有关)

Performance

- 120mx120mx40m

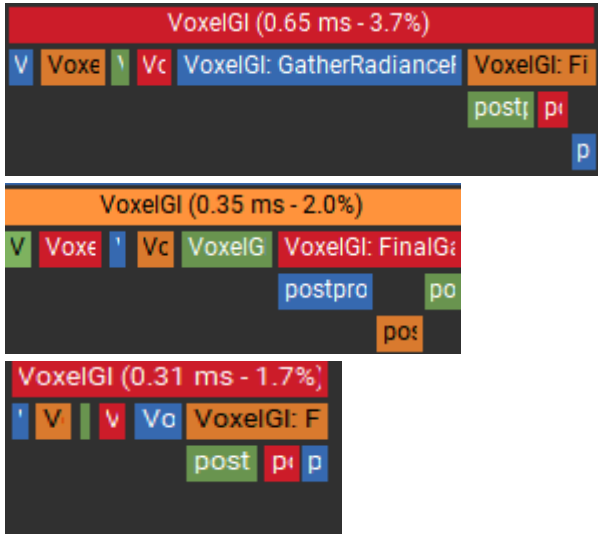
	Time(ms)	
Pass Name	RTX 2060	iPhone14Pro
VoxelBrickCulling	0.02	0.05
Voxelization	0.06	0.06
BuildOctreeStrcture	0.01	0.05
UpdateReservoirPool	0.01	0.02
Probe Placement	0.03	0.04
GatherLightFromeVoxel	0.10	0.28
FinalGather	0.13	0.45
Sum	0.36	1.0





客户端分级

- 目前分有低、中、高，主要区别就是GI范围和更新频率
- PC下不同级的性能消耗
  - 高：0.65ms
  - 中：0.35ms
  - 低：0.31ms
- 手机平台目前对应PC的低级别




D3D11. Frame: 11454. 16.67 ms (13.06 .. 21.08) (60 FPS)
0 Captures saved.
GI ON
VoxelGI
enable voxel gi


D3D11. Frame: 14302. 16.58 ms (11.88 .. 21.26) (60. FPS)
0 Captures saved.
GI OFF
VoxelGI
enable voxel gi


D3D11. Frame: 49844. 16.67 ms (11.88 .. 20.54) (60 FPS)
0 Captures saved.
Coffee
SeaFood
FooD
And
beer
SANDWICH
CAFE & CATERING
GI ON
VoxelGI
enable voxel gi


D3D11. Frame: 51064. 16.68 ms (13.73 .. 19.54) (60 FPS)
0 Captures saved.
Coffee
SeaFood
FooD
And
beer
SANDWICH
CAFE & CATERING
GI OFF
VoxelGI
enable voxel gi


66.7 FPS, 14.99 ms/frame
Hit F11 for frame capture
GI ON
VoxelGI


62.8 FPS, 15.93 ms/frame
Hit F11 for frame capture
GI OFF
VoxelGI


D3D11. Frame: 77333. 16.67 ms (11.32 .. 21.80) (60 FPS)
0 Captures saved.
SMART SHOP
GI ON
VoxelGI
enable voxel gi


D3D11. Frame: 78763. 16.68 ms (13.19 .. 20.36) (60 FPS)
0 Captures saved.
SMART SHOP
GI OFF
VoxelGI
enable voxel gi


D3D11. Frame: 104886. 16.64 ms (13.49 .. 19.33) (60. FPS)
0 Captures saved.
VoxelGi
enable voxel gi


D3D11. Frame: 29094. 16.65 ms (11.09 .. 20.31) (60. FPS)
0 Captures saved.
SSAO
VoxelGI
enable voxel gi

Future

- 支持屏幕外的Voxel
- 提高光照收敛速度，减少lag

References

- [Robin Green] Spherical Harmonic Lighting: The Gritty Details, 2003
- [Gilabert, Stefanov] Deferred Radiance Transfer Volumes – Global Illumination in Far Cry 3, GDC 2012
- [Yuriy O'Donnell] Precomputed Global Illumination in Frostbite, GDC 2018
- [Peter-Pike Sloan] Stupid Spherical Harmonics (SH) Tricks, GDC 2008
- [Ari Silvennoinen, Ville Timonen] Multi-Scale Global Illumination in Quantum Break, siggraph 2015
- [Daniel Wright] Radiance Caching for real-time Global Illumination
- [EA] Global Illumination Based on Surfels
- [Unity] Probe-based Lighting in Unity's 'Enemies'
- [Nvidia] Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting

谢谢