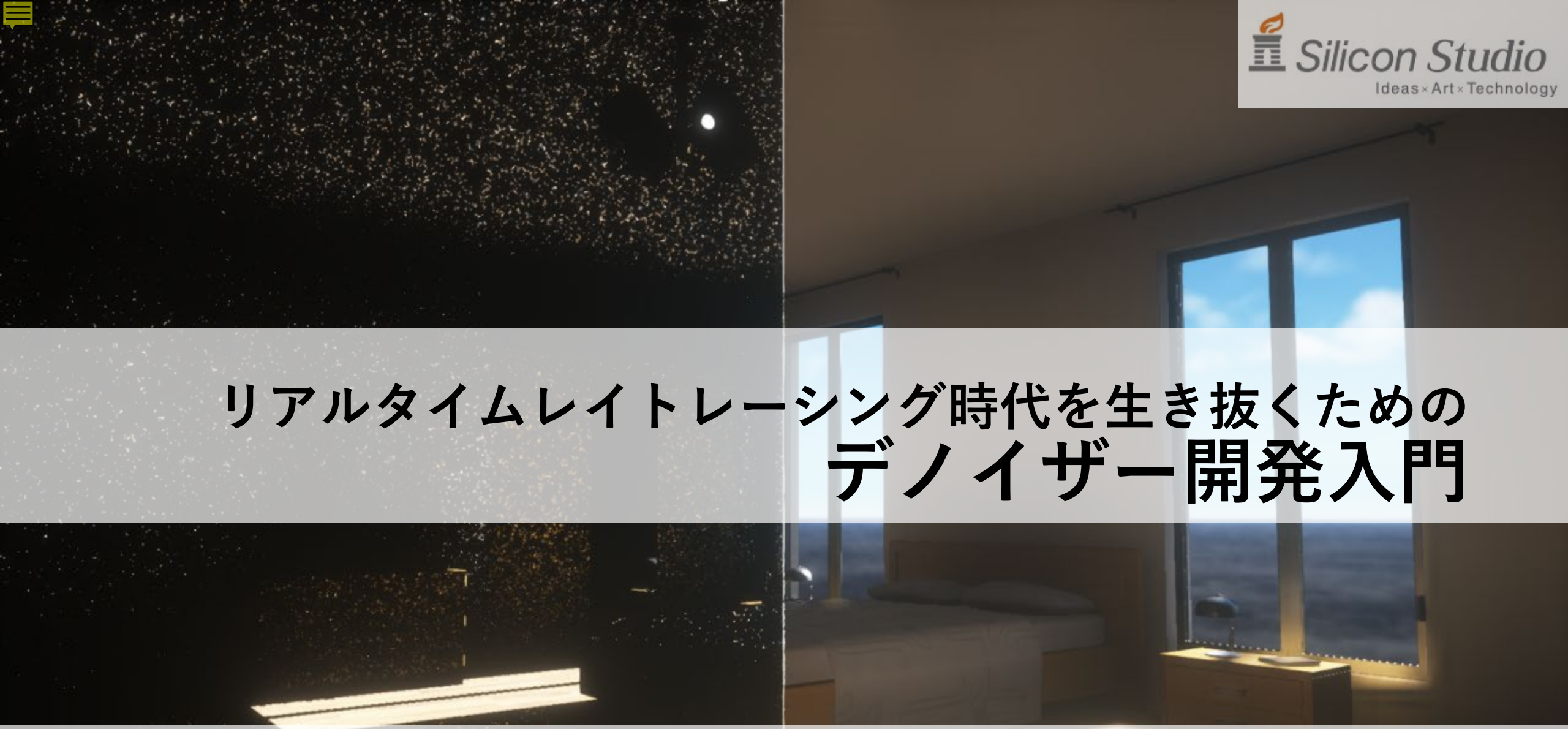


Silicon Studio
Ideas × Art × Technology

# リアルタイムレイトレーシング時代を生き抜くための
# デノイザー開発入門

シリコンスタジオ株式会社　研究開発室　川口 龍樹

2021.08.24 (CEDEC2021)

# はじめに

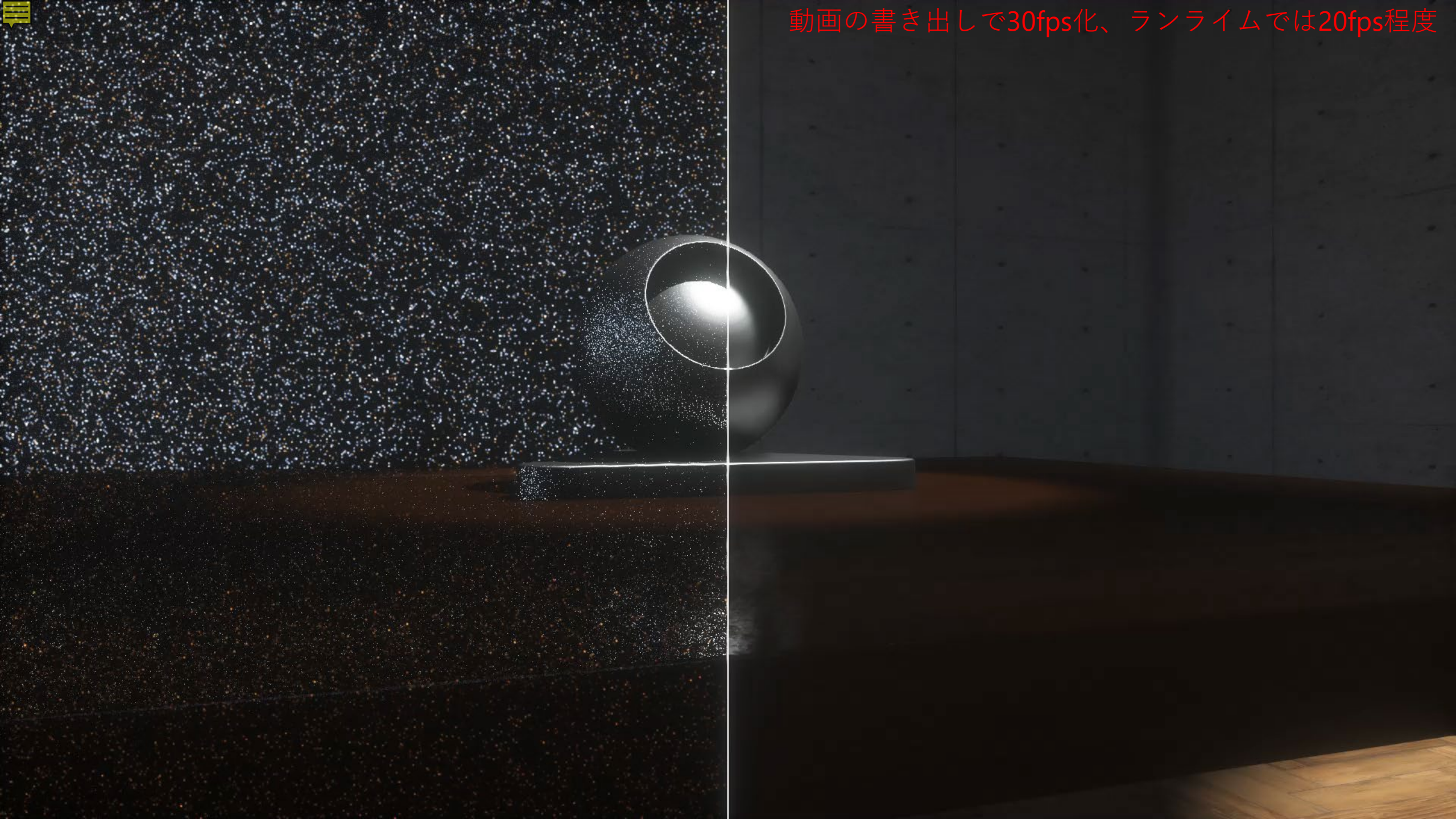
Silicon Studio
Ideas × Art × Technology

**川口　龍樹**　Tatsuki KAWAGUCHI

テクノロジー事業本部　セールス＆BC統括部
研究開発室

2017年シリコンスタジオ株式会社に新卒入社
自社リアルタイムレンダリングにおける技術の研究開発に従事
近年はリアルタイムレイトレーシング表現の研究開発に取り組む

©Silicon Studio Corp., all rights reserved.


動画の書き出しで30fps化、ランライムでは20fps程度

# はじめに

Silicon Studio
Ideas × Art × Technology

レイトレーシング対応のGPUの登場（2018年）
→ リアルタイムレンダリングの表現はより一層のクオリティアップ

高度なレイトレーシング手法は確率的な特性を持つ
→ ノイズが必ず発生する
→ デノイザーの出番

GeForce RTX™ 2080
Founders Edition
2018年9月20日

## 本講演の内容

自社製のリアルタイムレンダリングエンジンにおける、
リアルタイムレイトレーシングのデノイザー開発についての取り組みを紹介

- アルゴリズムの紹介
- 各要素技術の実装方法
- どのような結果が得られたのか
- 細かい問題に対する対処方法

＊本講演内の画像は縮小や配信における画質の劣化により
ノイズの情報が失われている可能性があります

©Silicon Studio Corp., all rights reserved.

# 0 はじめに

Silicon Studio
Ideas × Art × Technology

1 レイトレーシングとノイズ

2 デノイザー概要

3 各種デノイザー紹介

4 アルゴリズム解説と実装

5 デノイザー設計

6 デノイザーの未来とセッションまとめ

©Silicon Studio Corp., all rights reserved.

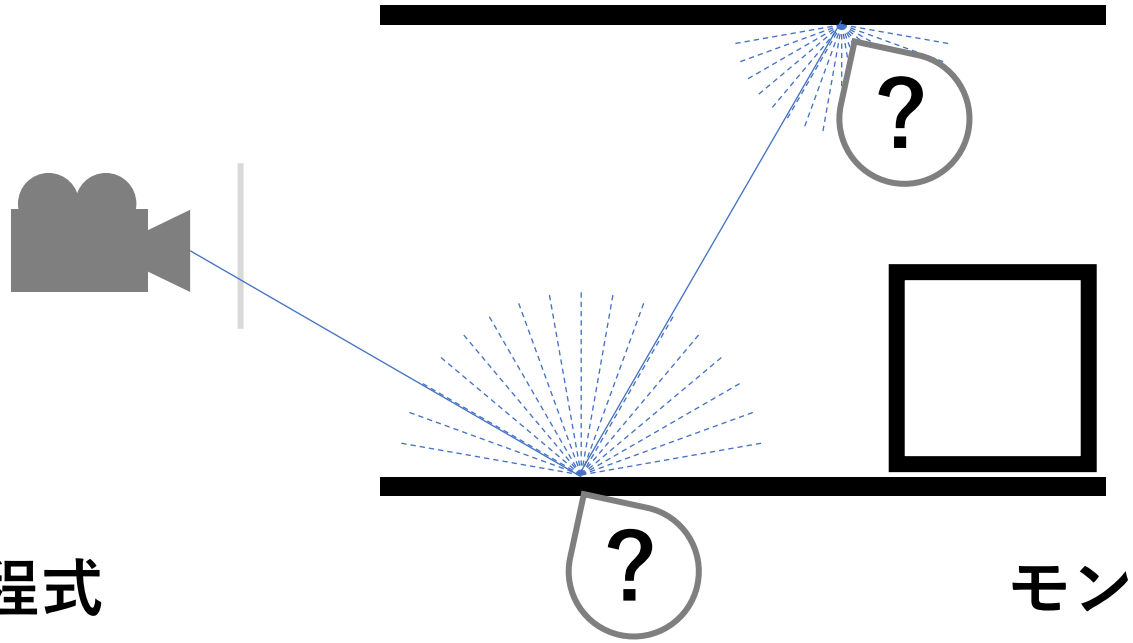
# 1 レイトレーシングとノイズ

©Silicon Studio Corp., all rights reserved.

# レイトレーシングとノイズ

Silicon Studio
Ideas × Art × Technology

**レイトレーシング（e.g. パストレーシング）**
光線（レイ）を追跡（トレース）して3DCGの絵を描画する
レイの衝突したある点からさらに周囲へレイを飛ばして色を決める（半球積分）
レンダリング方程式で表され解くときは確率的なモンテカルロ積分を使う




**レンダリング方程式**

$$I(\boldsymbol{x}) = \int_{\Omega} f(\boldsymbol{x}, \boldsymbol{\omega_i}) L(\boldsymbol{x}, \boldsymbol{\omega_i}) |\boldsymbol{n} \cdot \boldsymbol{\omega_i}| d\boldsymbol{\omega_i}$$

**確率的に解く**

**モンテカルロ積分**

$$I(\boldsymbol{x}) \sim \frac{1}{N} \sum_{i=0}^{N} \frac{f(x_i)}{p(x_i)}$$

©Silicon Studio Corp., all rights reserved.

# レイトレーシングとノイズ

Silicon Studio
Ideas × Art × Technology

レイトレーシングの計算（ある点の明るさを求める）

## レンダリング方程式

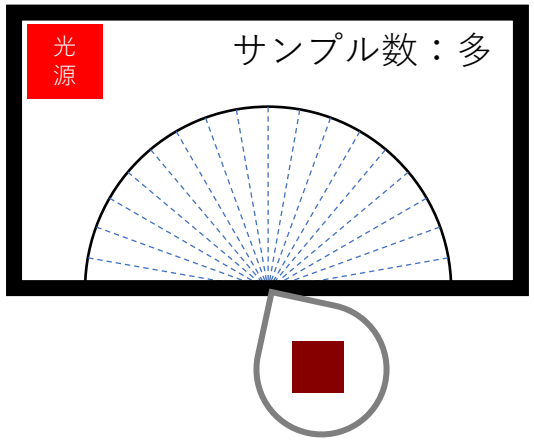
$$I(\boldsymbol{x}) = \int_{\Omega} f(\boldsymbol{x}, \boldsymbol{\omega_i}) L(\boldsymbol{x}, \boldsymbol{\omega_i}) |\boldsymbol{n} \cdot \boldsymbol{\omega_i}| d\boldsymbol{\omega_i}$$

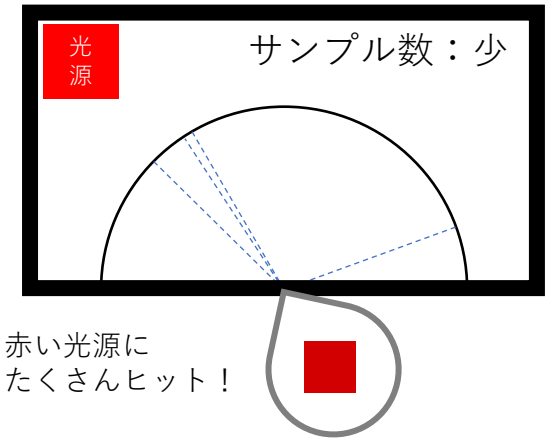
**確率的に解く**

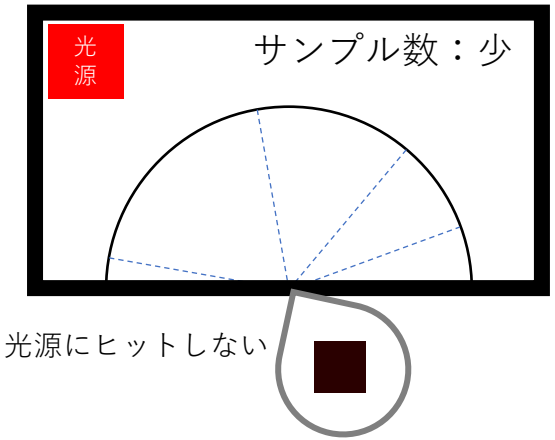
## モンテカルロ積分

$$I(\boldsymbol{x}) \sim \frac{1}{N} \sum_{i=0}^{N} \frac{f(x_i)}{p(x_i)}$$

飛ばすレイが少ないと同じ本数でも結果にばらつきが出る



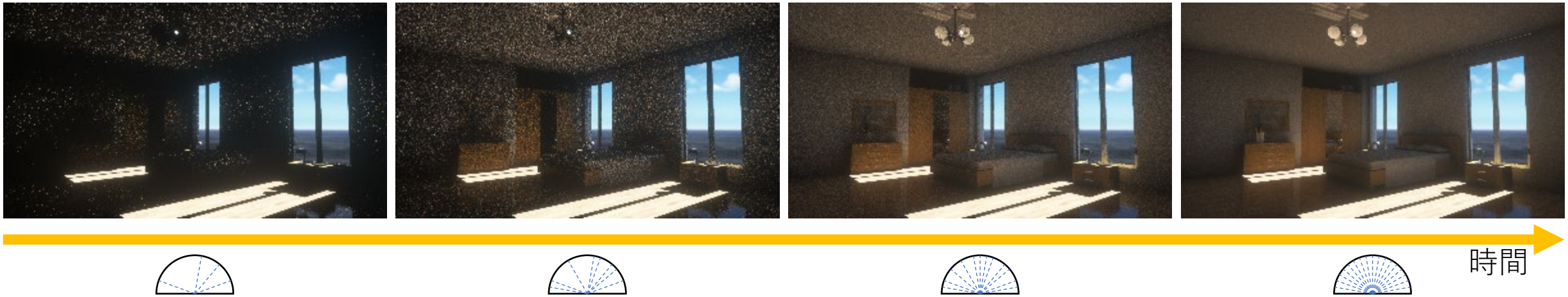

確率的に解くためのサンプル数が少ないと真値との誤差が大きくなる
視覚的なノイズとして現れる！

©Silicon Studio Corp., all rights reserved.

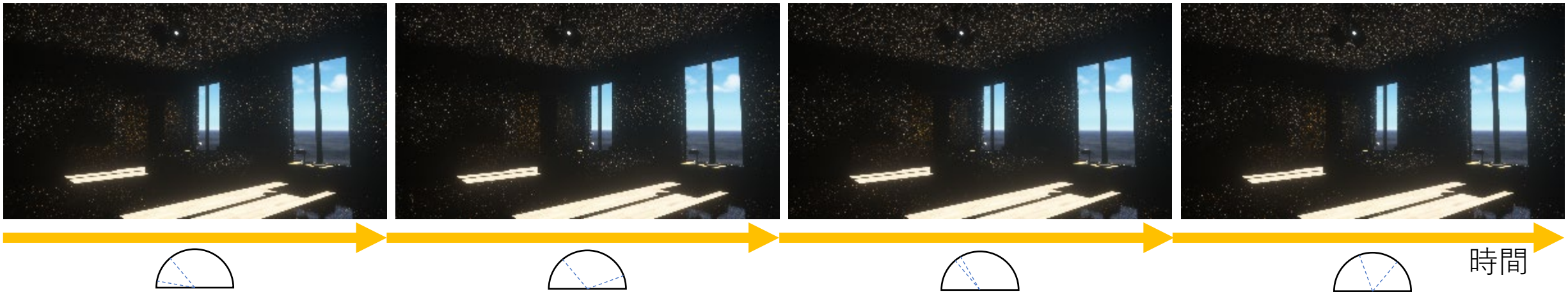
# 1 レイトレーシングとノイズ

Silicon Studio
Ideas × Art × Technology

**オフラインレンダリング**　時間をかけてサンプル数を増やす




**リアルタイムレンダリング**　1フレームに使える時間が限られているのでサンプル数を増やせない



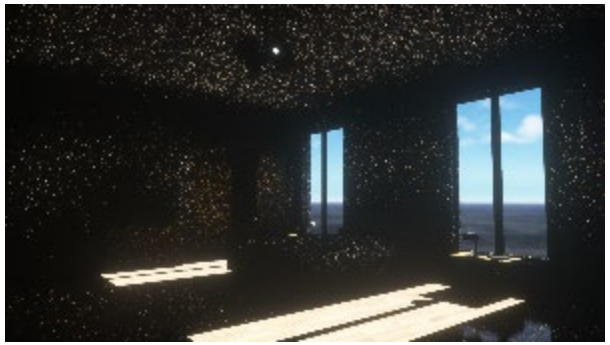

**デノイザーが必要！**

©Silicon Studio Corp., all rights reserved.

# 2 デノイザー概要

©Silicon Studio Corp., all rights reserved.

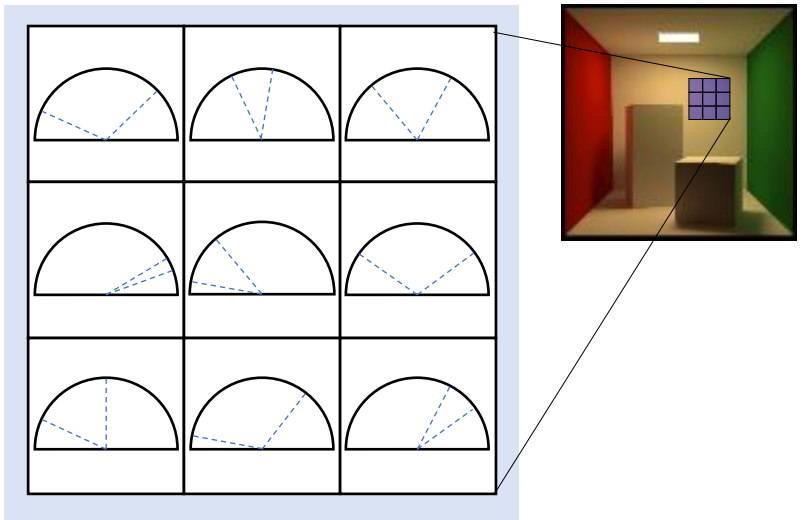
# デノイザー概要



**Denoiser**

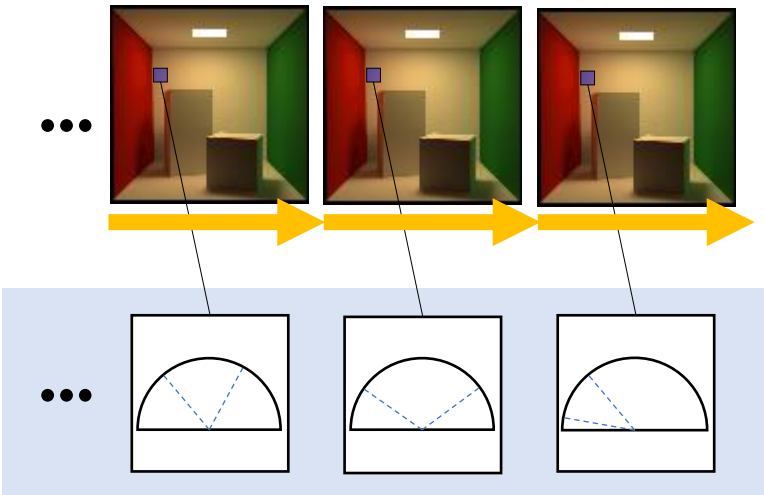
足りないサンプル数を
補ってノイズを減らす

## Spatial Accumulation



注目している点の周辺から
サンプルを集めて不足を補う
≃ 平均値フィルタ

## Temporal Accumulation



注目している点の過去のサンプル
を集めて不足を補う
≃ Temporal super sampling

©Silicon Studio Corp., all rights reserved.

# デノイザー概要

Silicon Studio
Ideas × Art × Technology

## 2.1. デノイザーの対象

**手法**

| パストレーシング | フォトンマッピング | 古典的レイトレーシング |
|---|---|---|
| 収束に非常に時間がかかるのでデノイザーが必要 | 一度に収集するフォトンには限りがあるのでデノイザーが必要 | 確率的な処理が無ければデノイズは不要 |

**信号**

| シャドウ | 直接光 / 間接光 / 鏡面反射 | 直接光 / 間接光 / 拡散反射 |
|---|---|---|
| 色の情報は基本必要ない動的な光源に対応するためには時間的な処理は抑える必要がある | 材質によって性質が大きく変わり視点依存があるので柔軟な設計とパラメータ調整が必要 | サンプル数が非常に不足するがディティールはあまりいらないので大きなフィルタで処理する |

**今回はパストレーシングにおける間接鏡面反射光を主に取り上げる**

©Silicon Studio Corp., all rights reserved.

# デノイザー概要

# 2.2. デノイザーの構成

Silicon Studio
Ideas × Art × Technology

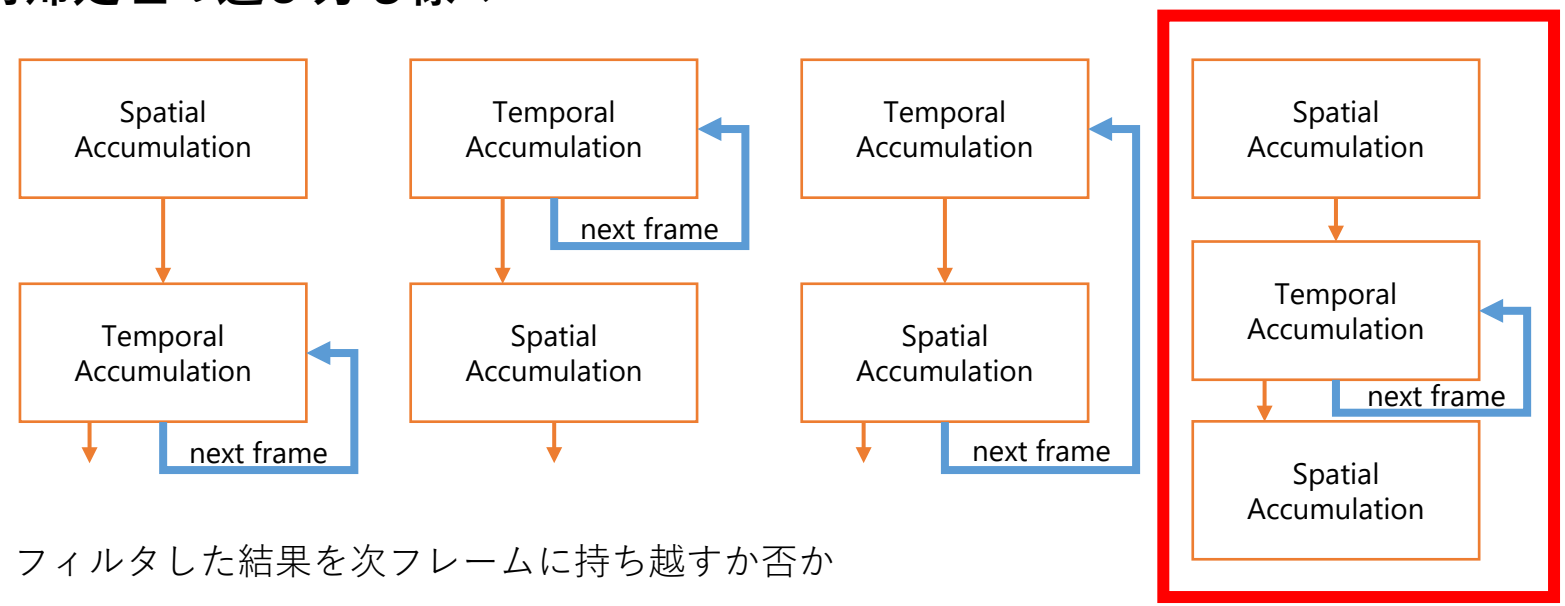
## 基本的な流れ

- Ray Tracing
- Pre- Spatial Accumulation
- Temporal Accumulation
- Post- Spatial Accumulation

## Optional : 必要に応じて前処理や利用するデータの管理

Disocclusion | G-Buffer | Mip Generation | ...

## 再帰処理の選び方も様々




フィルタした結果を次フレームに持ち越すか否か

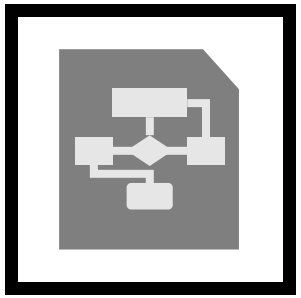
©Silicon Studio Corp., all rights reserved.

# 3 各種デノイザー紹介

©Silicon Studio Corp., all rights reserved.

## 各種デノイザー紹介

# 既存のリアルタイムデノイザー

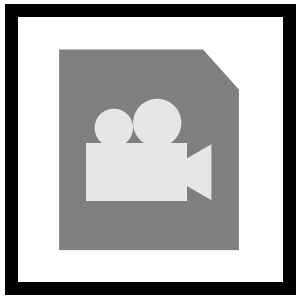


### アルゴリズム

- SVGF
- A-SVGF
- SLGF
- Recurrent Blur

### ライブラリ

- NVIDIA RT Denoiser
- AMD FFX Denoiser



### リアルタイムレンダラー

- bgfx
- NVIDIA Falcor
- The-Forge

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 3.1 SVGF Spatiotemporal Variance-Guided Filtering

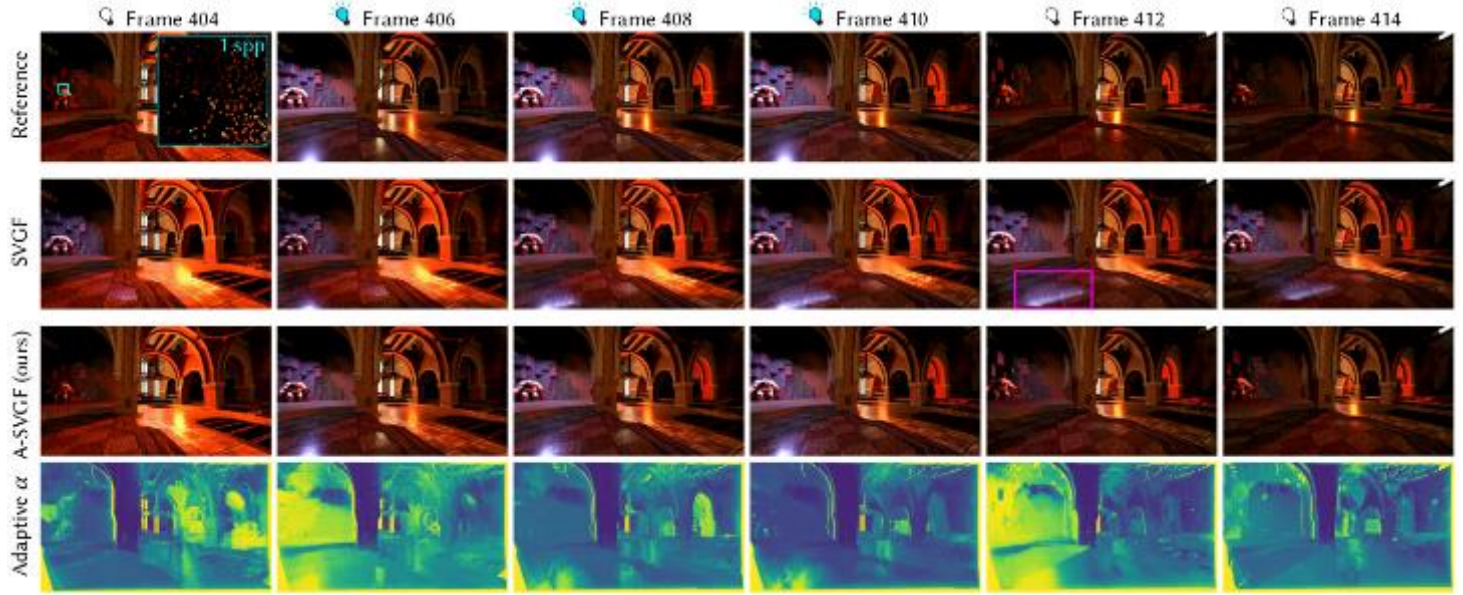
https://research.nvidia.com/publication/2017-07_Spatiotemporal-Variance-Guided-Filtering%3A

Spatiotemporal（時空間）な Variance （分散）をデノイズに利用する
分散はノイズの量を表す指標になるのでフィルタの大きさを調整するのに利用する
過剰ブラーを抑制して高品質な結果を作る

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 3.2 A-SVGF Adaptive SVGF




https://cg.ivd.kit.edu/atf.php

SVGF を光源環境の変化に対応させた手法
輝度の時間勾配をみて大きく変化がある場合は Temporal Accumulation を弱める
時間勾配の正確な追跡に Gradient pixel の実装がレイトレーシング側に必要

Quake II RTX というゲームに採用されている
CEDEC+KYUSHU 2019 で解説したセッションあり http://cedec-kyushu.jp/2019/session/04.html

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 3.3 SLGF Spatiotemporal Luminance-Guided Filter

https://cedec.cesa.or.jp/2019/session/detail/s5ce39bf17ac37

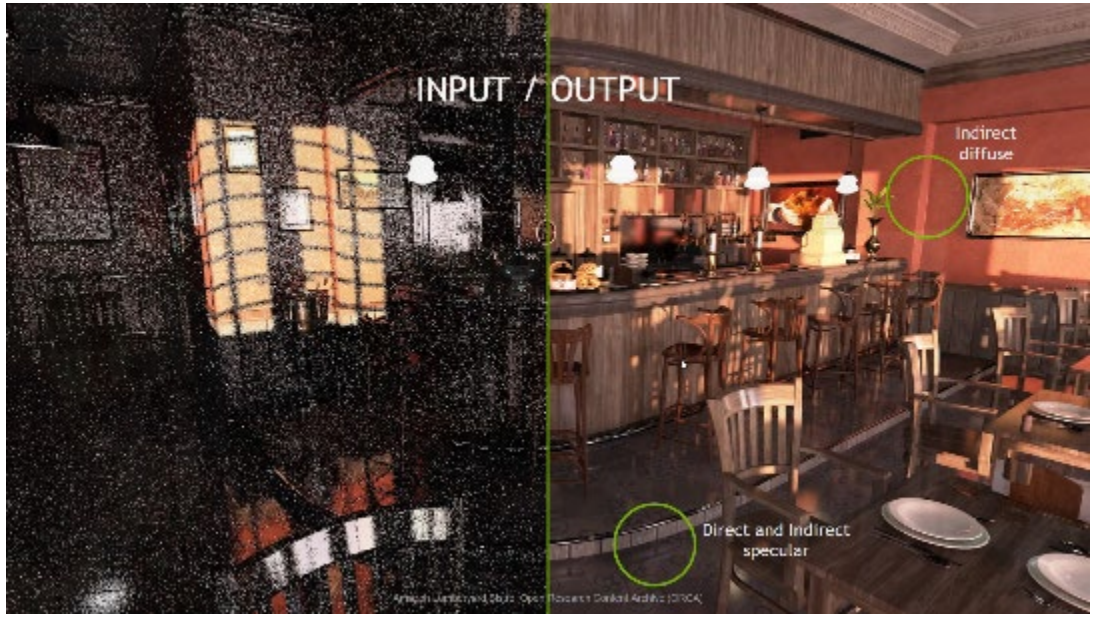
ルミナスエンジンのリアルタイムレイトレーシングデモで開発された手法（CEDEC 2019）

A-SVGF のGradient pixelのような実装や分散パラメータなど複雑な設計を必要としない
レイトレーシングの結果から輝度推定を行いピクセルの取りうる輝度で Temporal Accumulation を調節

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 3.4 Recurrent Blur




https://developer.nvidia.com/gtc/2020/video/s22699-vid
Fast Denoising with Self Stabilizing Recurrent Blurs という GTC 2020 で発表されたアルゴリズム

少ないサンプル数のフィルタを時間的に再帰することで効率的にサンプル数を増やす
適応的なブラー半径や Mipmap を使った階層的な信号復元手法を取り入れて効率的なデノイズを達成

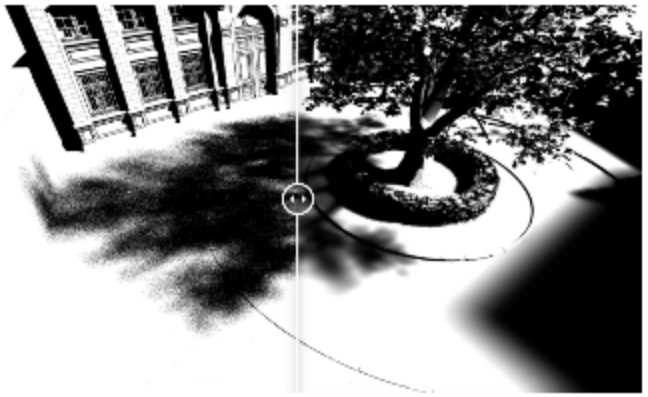
Ray Tracing Gems II にも解説あり

©Silicon Studio Corp., all rights reserved.

# 各種デノイザー紹介

Silicon Studio
Ideas × Art × Technology

## 3.5 NVIDIA RT Denoisers



https://developer.nvidia.com/nvidia-rt-denoiser

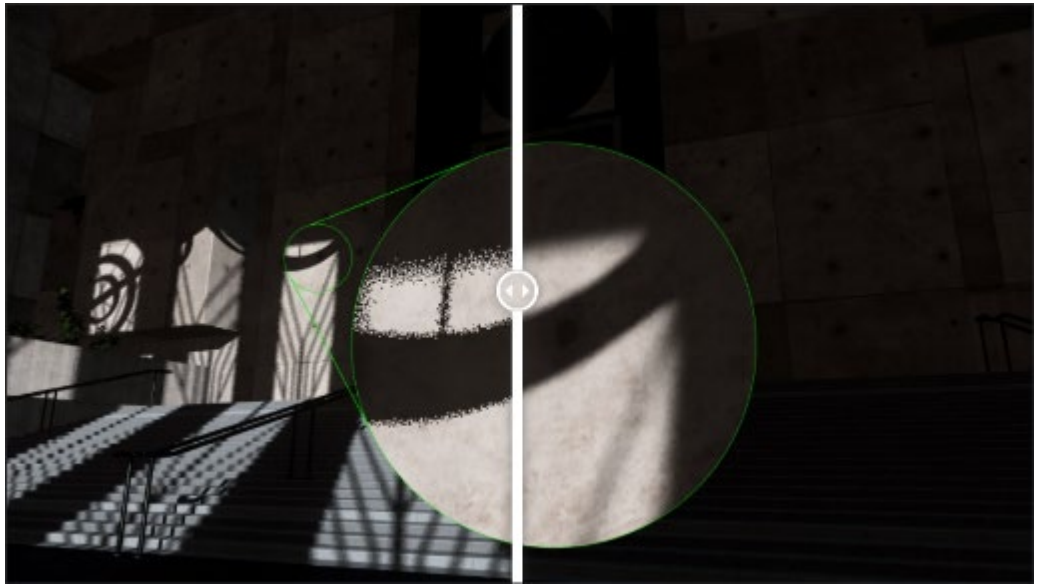
NVIDIA が提供するリアルタイムレイトレーシングをターゲットとしたデノイザーライブラリ
現在は申請するとソースコードへのアクセスを得られる
ReBLUR、SIGMA、ReLAXと用途に応じた三つのデノイザーが提供

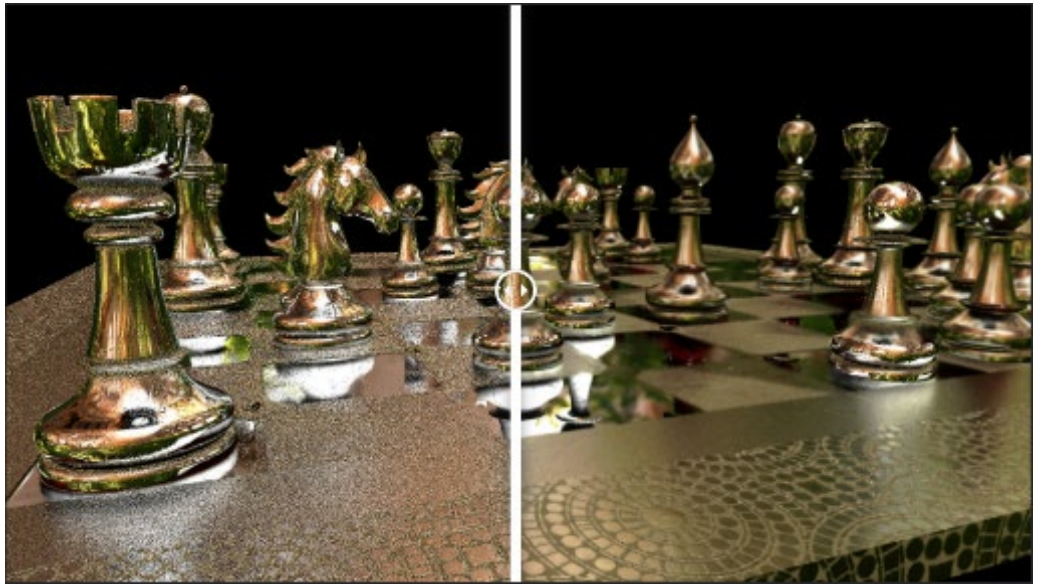
いずれも Spatiotemporal なデノイザーですがフィルタの設計やターゲットが異なります
**ReBLUR**：Recurrent Blur をもとにしたデノイザー
**SIGMA**：高速な影のデノイザーで物理ベースのフィルタで半影表現を強化する
**ReLAX** ：レイトレの Direct Illumination 向けに改良された SVGF

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 3.6 AMD FidelityFX Denoiser





https://gpuopen.com/fidelityfx-denoiser/

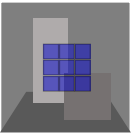
AMD が提供するオープンソースのリアルタイムレイトレーシング向けデノイザー
影用の Shadow Denoiser と反射光用の Reflection Denoiser がある
共に Spatiotemporal なデノイザー
Ray Classifier でタイルごとに飛ばすレイを動的に調整する

©Silicon Studio Corp., all rights reserved.

# 4 アルゴリズム解説と実装

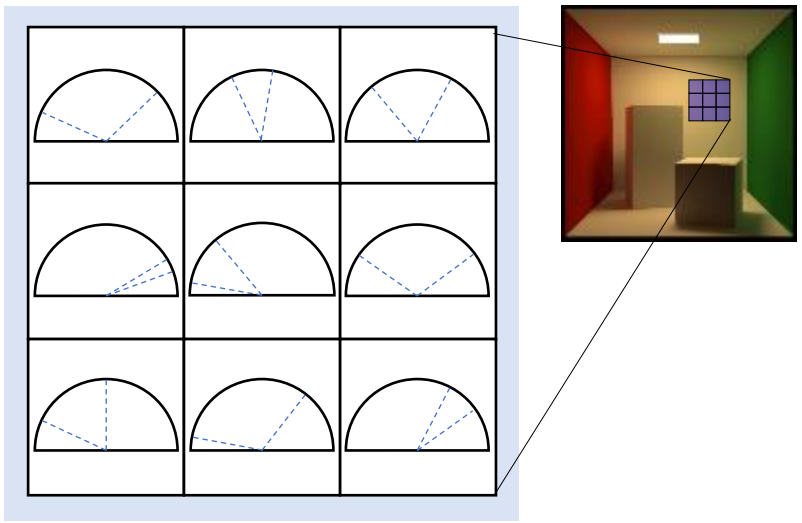
©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology



# 4.1. Spatial Accumulation 詳解

空間的に近傍の情報を利用してサンプル不足を補う＝平均フィルタ
近傍ピクセル間の相関が高ければ機能する

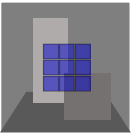


リアルタイムレイトレーシングは極端にサンプルが少ないので大きなフィルタサイズが必要
→高速で効率的な広範囲のフィルタリング処理が求められる

**空間的**＝「2次元スクリーンスペース＋3次元空間のジオメトリ情報」の処理であることが多い

©Silicon Studio Corp., all rights reserved.

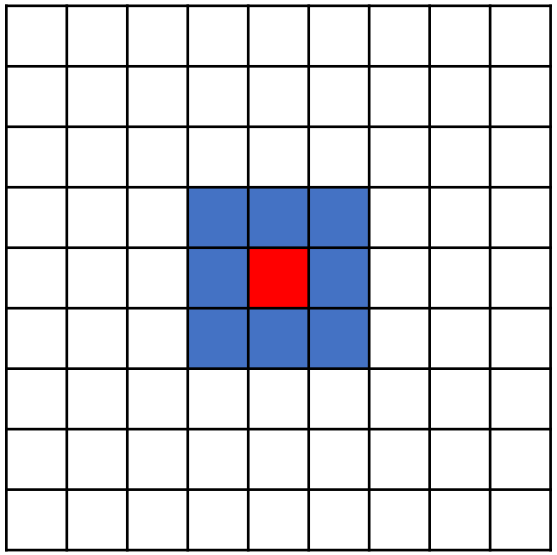
Silicon Studio
Ideas × Art × Technology



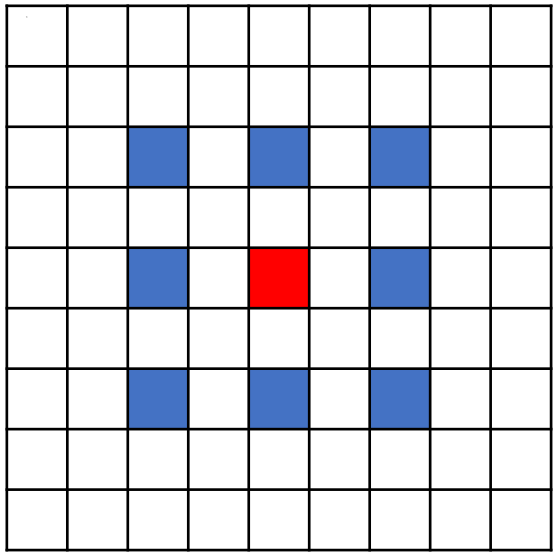
# 4.1.1. À-Trous Filter

再帰的な処理で広範囲をフィルタリングする手法

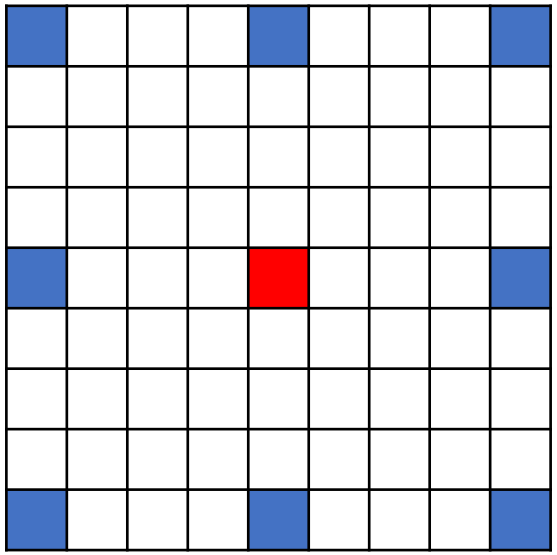
1パス目（影響範囲3x3）



2パス目（影響範囲7x7）



3パス目（影響範囲15x15）



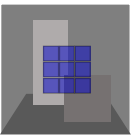
信号解析の分野で高速な Wavelet 変換の手法として提案

**A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform [Holschneider et al. 1989]**

リアルタイムレイトレーシングのデノイズフィルタに応用

**Edge-avoiding À-Trous wavelet transform for fast global illumination filtering [Dammertz et al. 2010]**

©Silicon Studio Corp., all rights reserved.



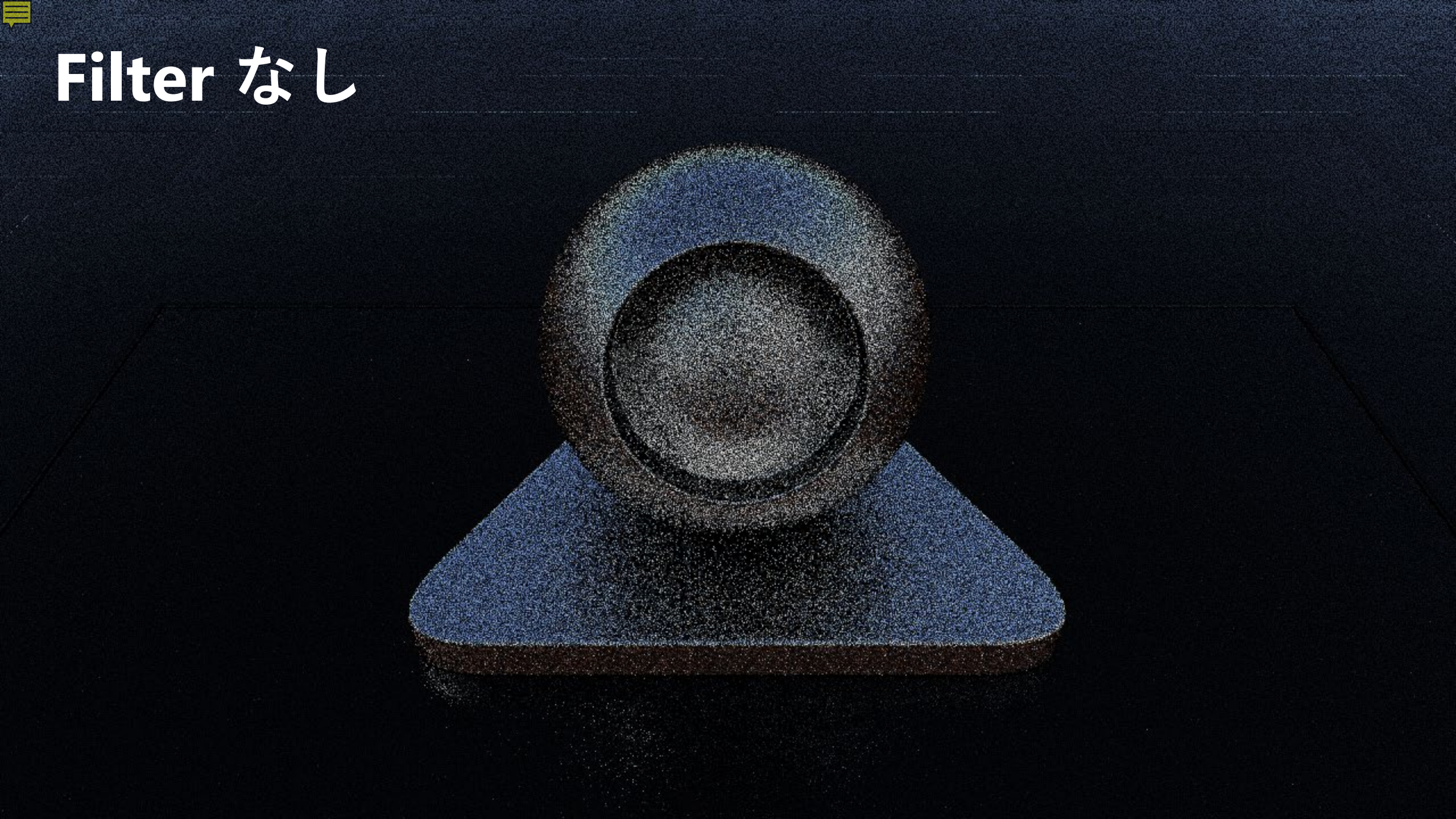
# 4.1.1. À-Trous Filter

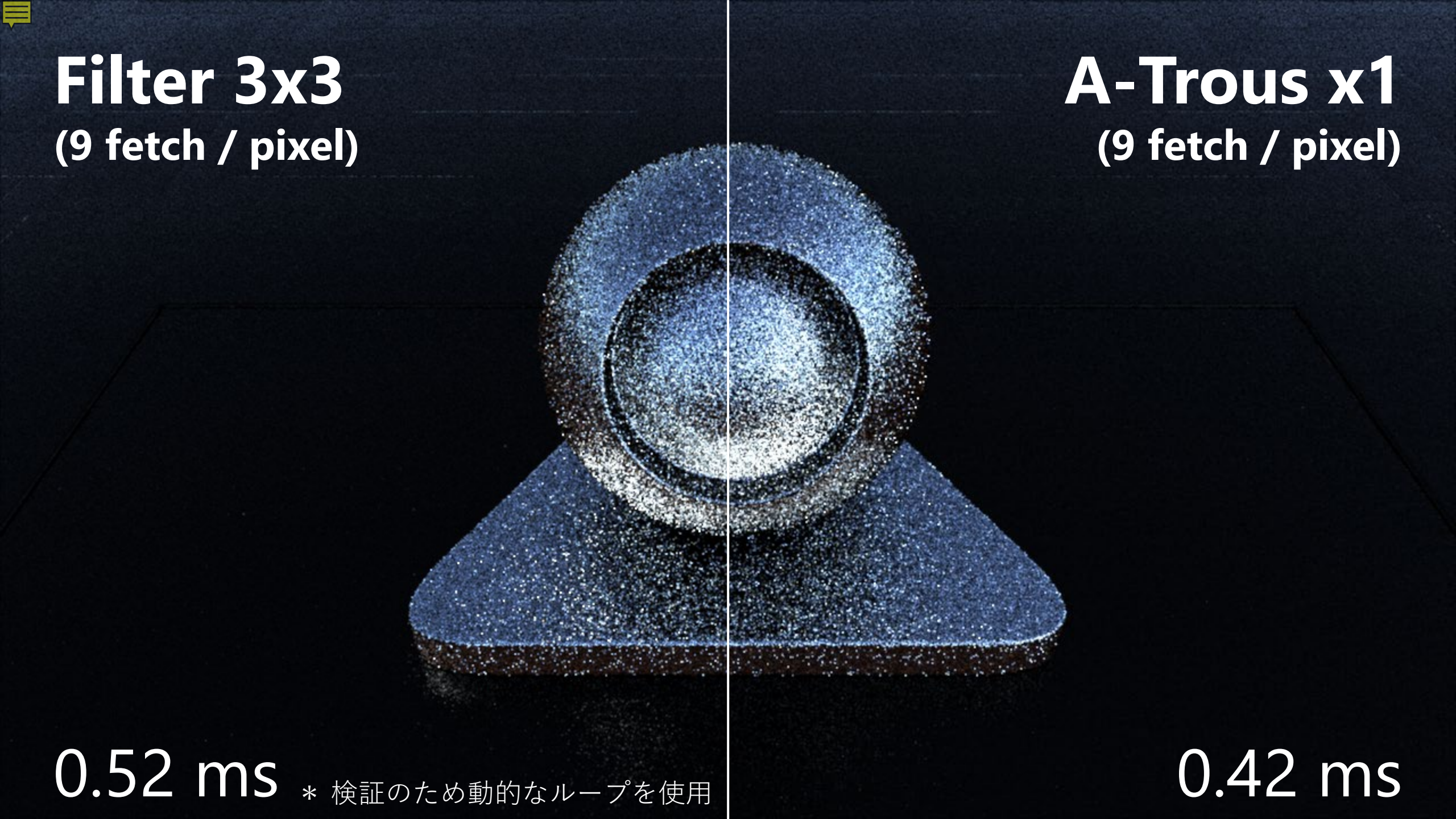
```
float4 ATrous(...)
{
    int N = 8;
    int2 offsets[] = {
        int2(-1,  0), int2( 0, -1), int2( 1,  0), int2( 0,  1),
        int2(-1, -1), int2( 1, -1), int2(-1,  1), int2( 1,  1)
    };

    float  weight =  1.0;
    float4 color = LoadColor(texidx);
    for(int i=0; i<N; ++i)
    {
        float2 offset = offsets[i] * (1 << recursionCount);
        float w = ComputeWeight(...);
        color += w * LoadColor(texidx + offset);
        weight += w;
    } // End for
    return color / (weight);
}
```

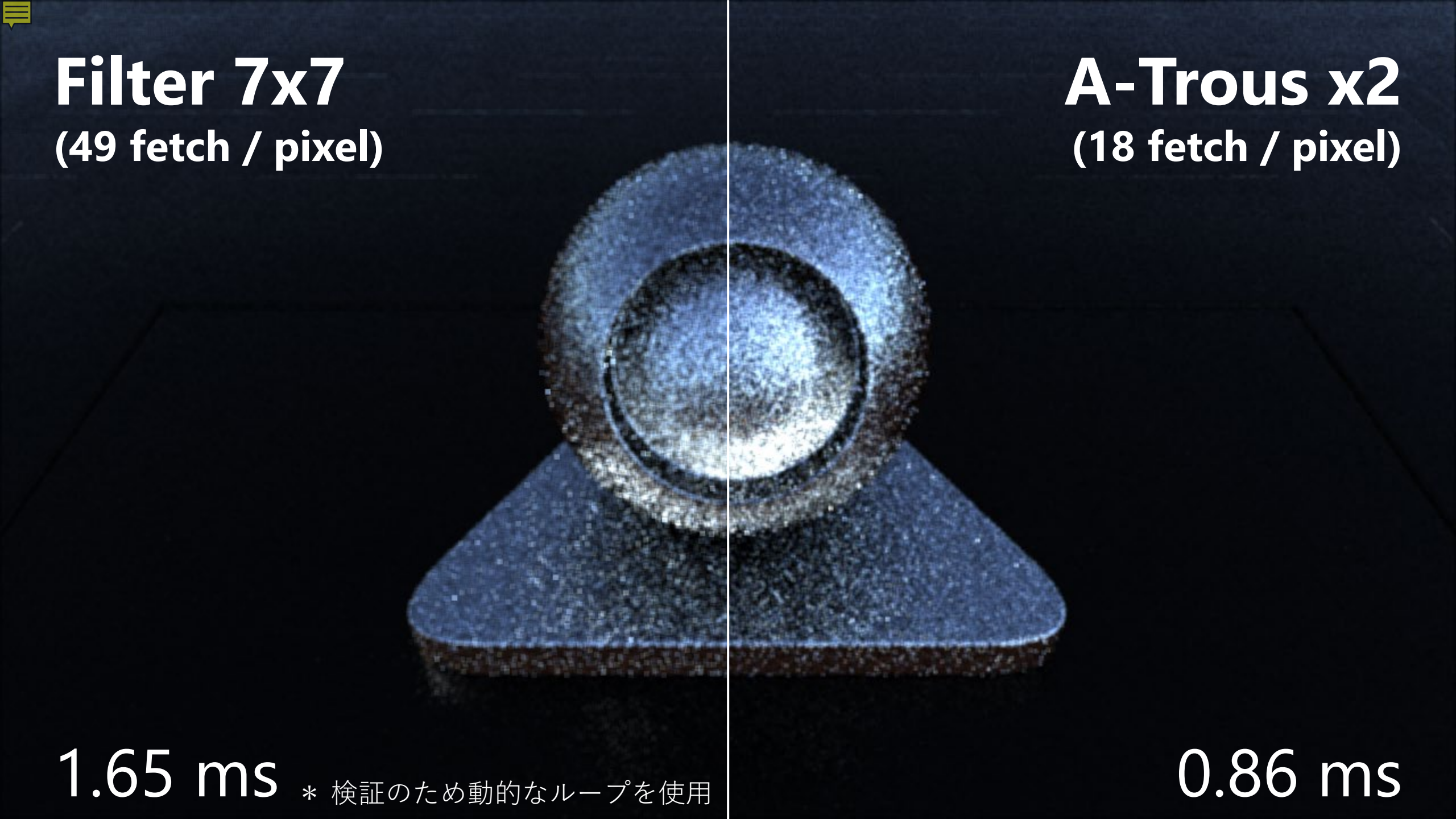
1パス目：recursionCount=0 → オフセット 1 pixel
2パス目：recursionCount=1 → オフセット 2 pixel
3パス目：recursionCount=2 → オフセット 4 pixel

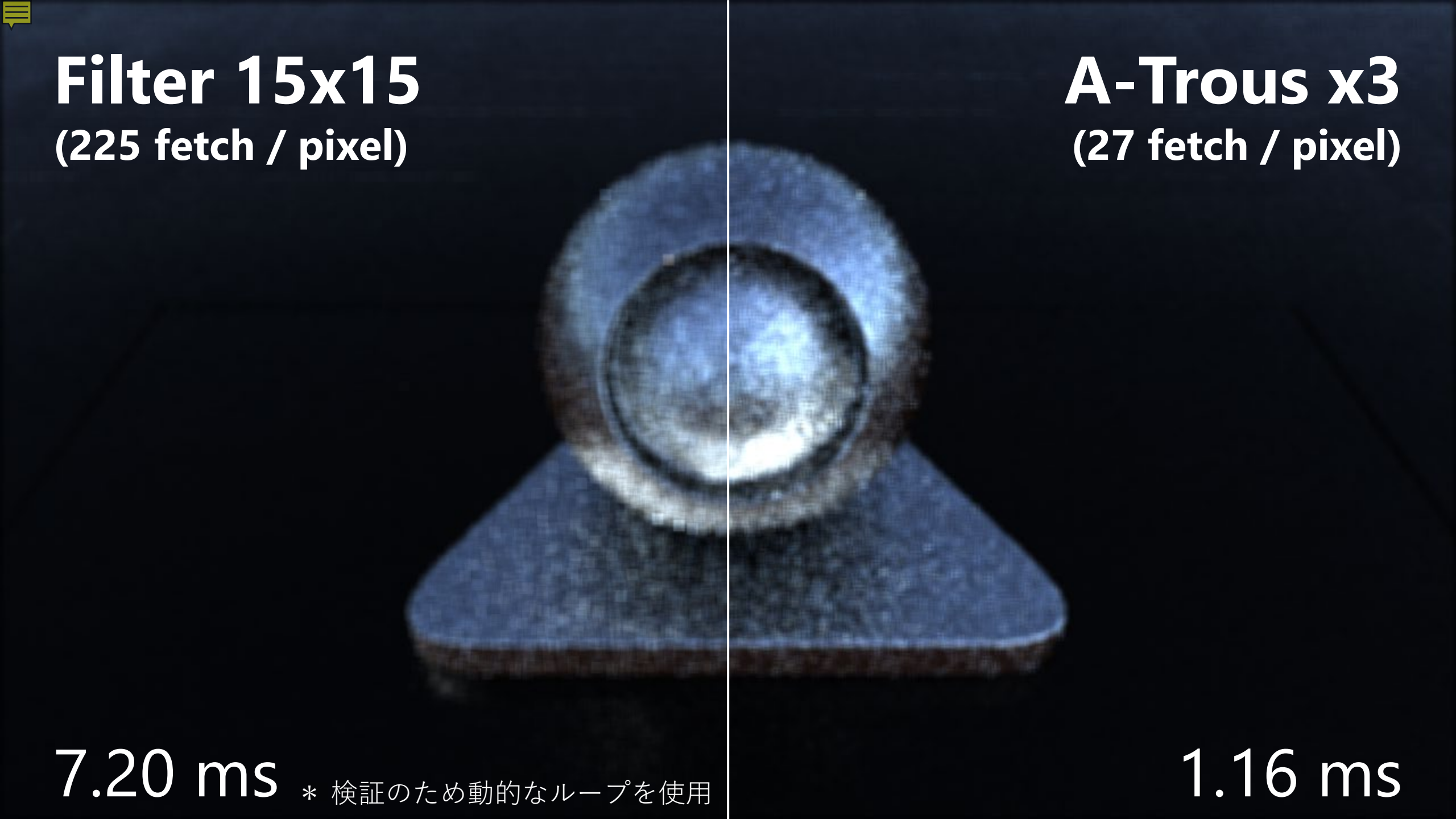
©Silicon Studio Corp., all rights reserved.

# Filter なし

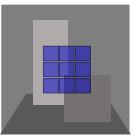



Filter 3x3
(9 fetch / pixel)
A-Trous x1
(9 fetch / pixel)
0.52 ms
＊ 検証のため動的なループを使用
0.42 ms


Filter 7x7
(49 fetch / pixel)
A-Trous x2
(18 fetch / pixel)
1.65 ms
＊ 検証のため動的なループを使用
0.86 ms

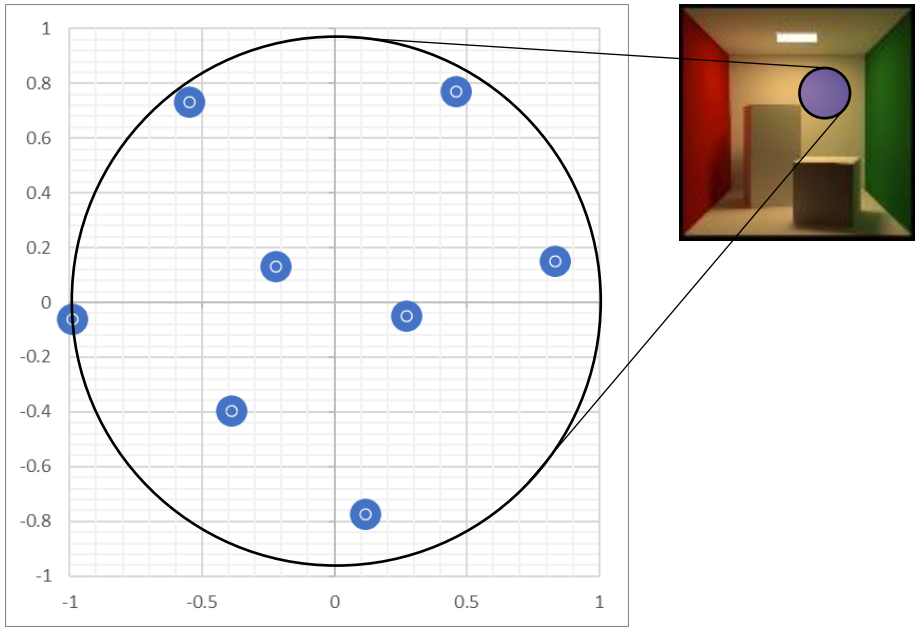

Filter 15x15
(225 fetch / pixel)
A-Trous x3
(27 fetch / pixel)
7.20 ms
＊ 検証のため動的なループを使用
1.16 ms

Silicon Studio
Ideas × Art × Technology

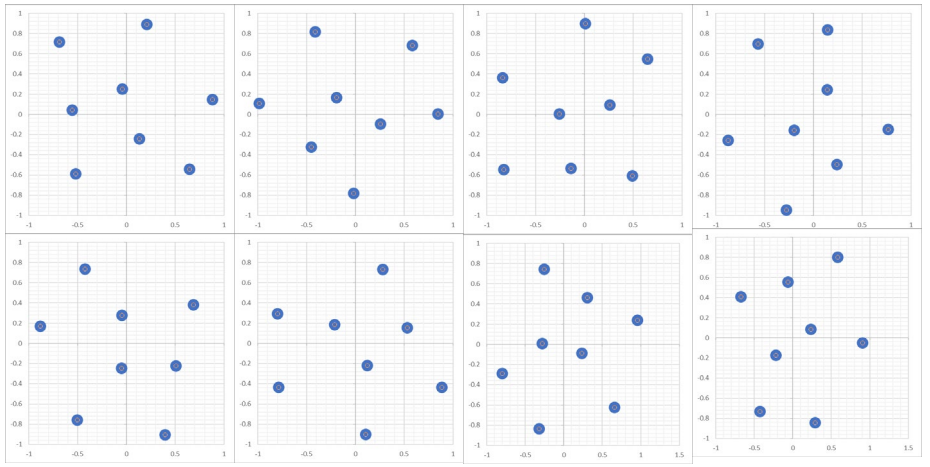


# 4.1.2. Poisson Disk Blur

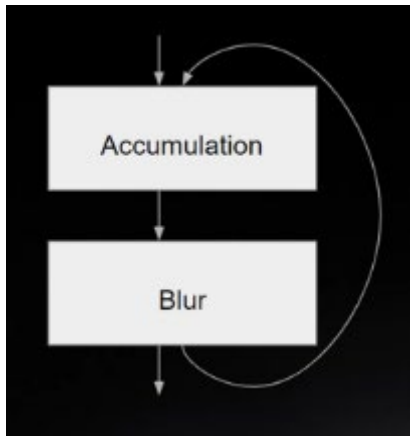
疎な効率のいいサンプリングパターンを利用する方法




Poisson Disk (8 samples)



サンプル点を回転して効率化




Recurrent Blur では
Blur結果を再帰する

半径のパラメータで影響範囲の調整が容易
サンプルパターンをランダマイズ（回転）することで時間的に効率のいいサンプル収集が可能
単一でのサンプル数が足りないケースでは時間的な再帰に含める

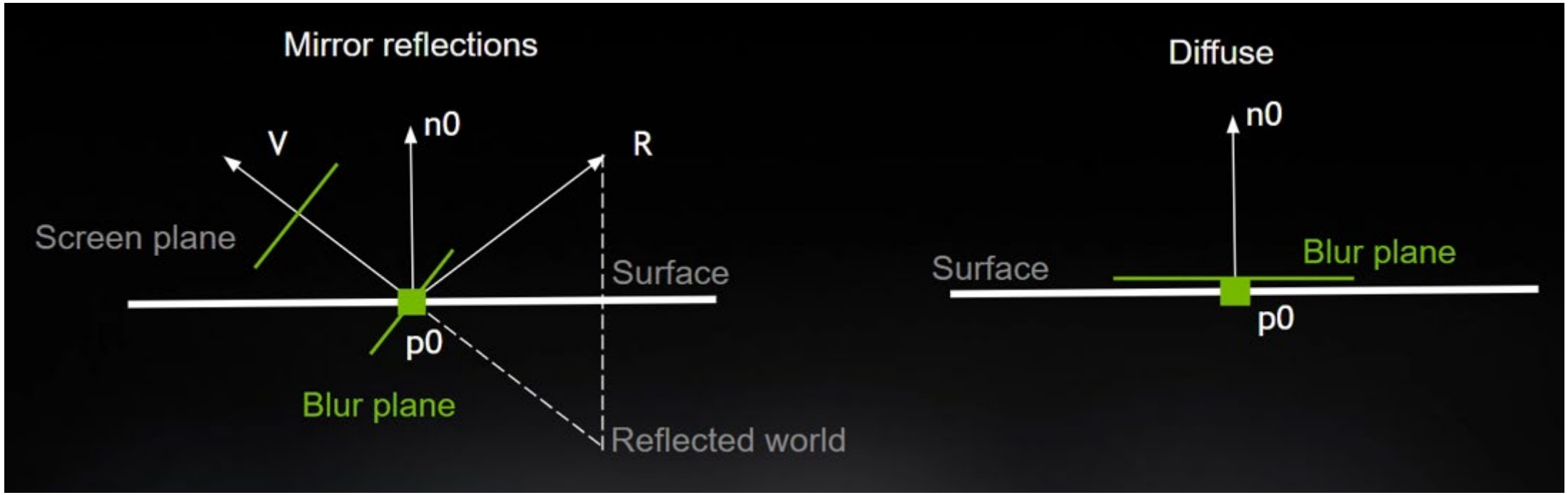
©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 4.1.2. Poisson Disk Blur

デノイズに合わせたディスクの改良も提案されている

**Fast Denoising with Self Stabilizing Recurrent Blurs [Zhadan 2020] (GTC2020)**



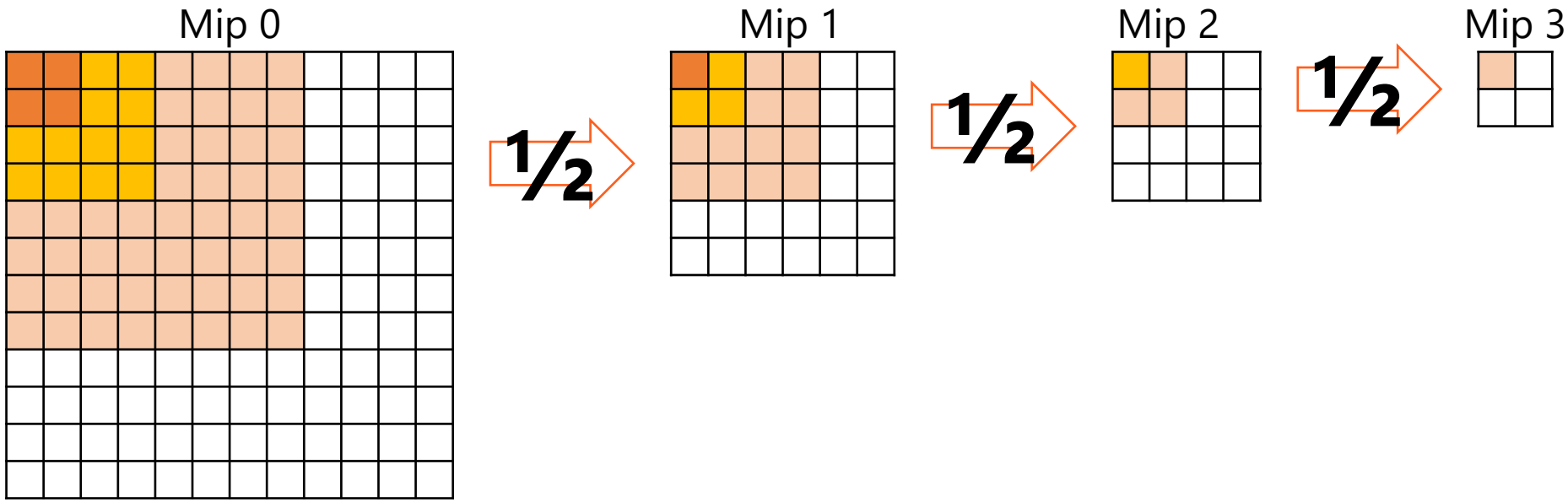

ディスクをワールド空間上のジオメトリに合わせて配置
スクリーンに投影したディスクでフィルタリング

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 4.1.3. Mipmap & Down Sampling

解像度を落とす＝フィルタリング




解像度半分（Mip 1 Level）ごとに2の累乗でサンプル数を補える
（Mip0:1x1, Mip1:2x2, Mip2:4x4, ..., Mip4:16x16）
アップスケーリング処理が必要（Bilateral Upsampling）

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

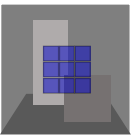
# 4.1.4. Edge Stopping Function

スクリーンスペースの Spatial Accumulation は
近傍ピクセルが3次元空間的にも近傍でなければならない

Edge Stopping Function はエッジ保存フィルタにおける重み付け関数
3次元空間的に離れている・異なる物体（＝エッジ）を跨いだフィルタリングを抑制する

©Silicon Studio Corp., all rights reserved.

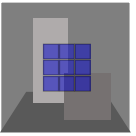
Silicon Studio
Ideas × Art × Technology



# 4.1.4. Edge Stopping Function

| | | |
|---|---|---|
| 法線 | $w_n = \max\left(0.0, \mathbf{n}(\mathbf{p}) \cdot \mathbf{n}(\mathbf{q})\right)^{\sigma_n}$ | 単に法線の差の絶対値を使うこともある |
| 深度 | $w_z = \exp\left(-\frac{\lvert z(\mathbf{p}) - z(\mathbf{q})\rvert}{\sigma_z + \varepsilon}\right)$ | 深度勾配で正規化する手法も提案されている |
| 空間位置 | $w_s = \exp\left(-\frac{(\mathbf{p}-\mathbf{q})\cdot(\mathbf{p}-\mathbf{q})}{\sigma_s + \varepsilon}\right)$ | スクリーン位置だとガウシアンになる<br>3次元位置だと深度項の代替としても機能する |
| 輝度 | $w_l = \exp\left(-\frac{\lvert l(\mathbf{p}) - l(\mathbf{q})\rvert}{\sigma_l \sqrt{g_{3\times3}\left(V\left(l(\mathbf{p})\right)\right)} + \varepsilon}\right)$ | 分母の$V\left(l(\mathbf{p})\right)$が輝度分散を表す<br>分散によってフィルタサイズを調整できる<br>**Spatiotemporal Variance-Guided Filtering [Schied 2017]** |
| 反射形状 | $w_{spec} = \left(\frac{2\sqrt{\bar{\lambda}_i \bar{\lambda}_j}}{\bar{\lambda}_i + \bar{\lambda}_j}\right)^{\beta} G\left(\xi_i, \xi_j, \frac{\beta \bar{\lambda}_i \bar{\lambda}_j}{\bar{\lambda}_i + \bar{\lambda}_j}\right)$ | 反射形状の球面ガウス関数表現で定義される<br>**Specular Lobe-Aware Filtering and Upsampling for Interactive Indirect Illumination [Tokuyoshi 2016]** |

©Silicon Studio Corp., all rights reserved.

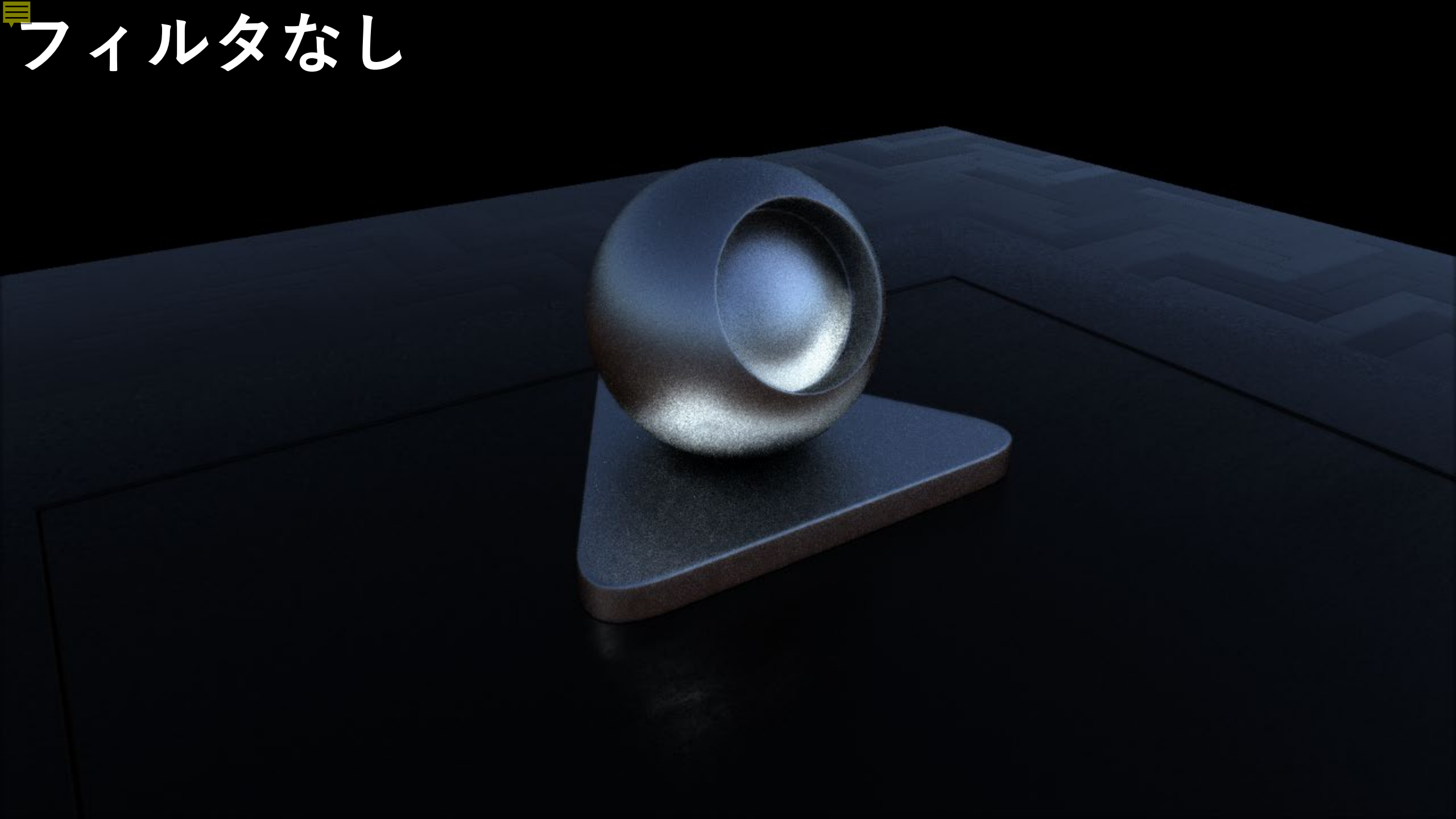


# 4.1.4. Edge Stopping Function

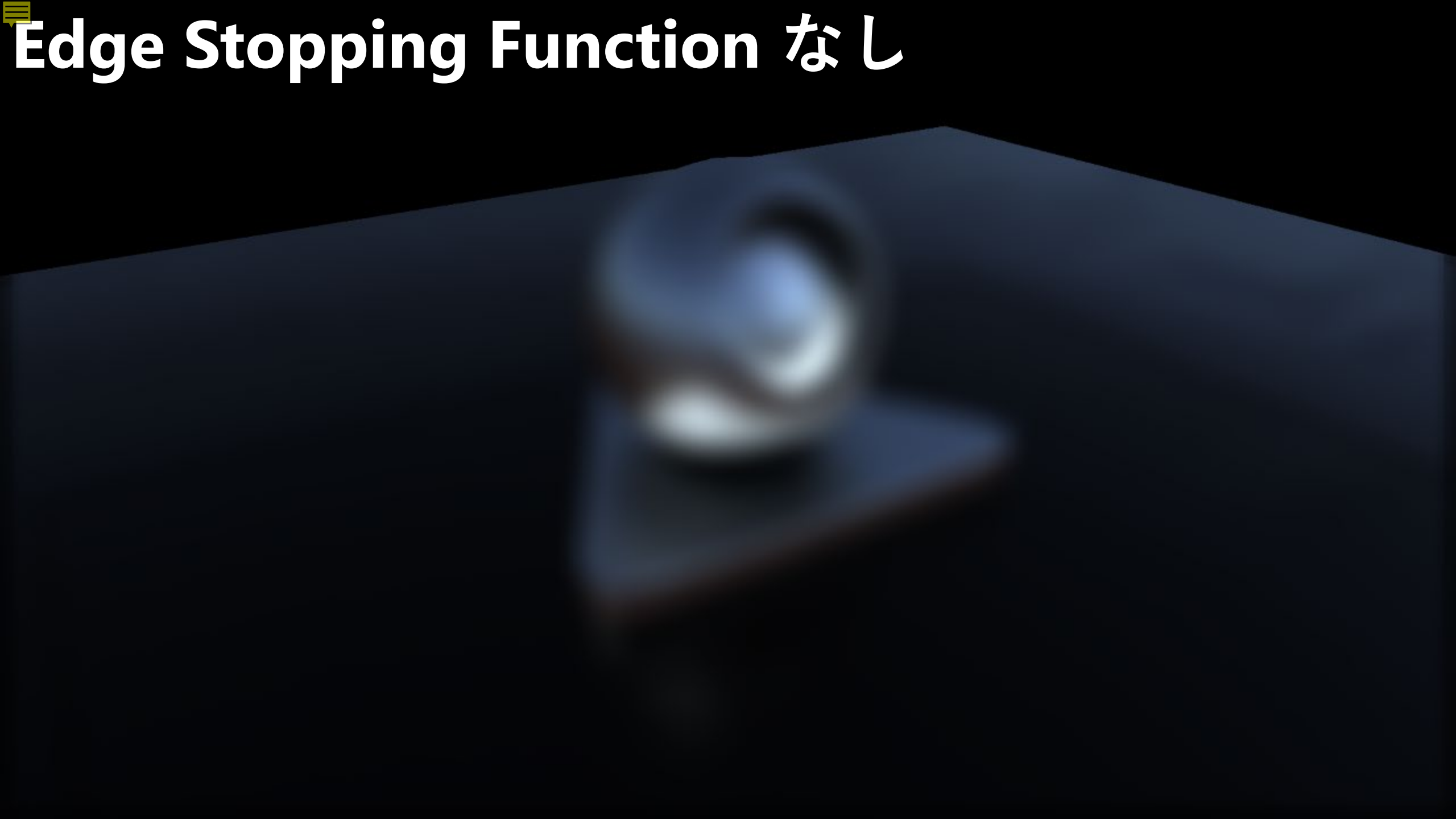
```
for(...)
{
    float w = 0.0;
    w += -abs(Depth(p) - Depth(q))/(float)(sigma_z + 0.001);  //深度
    w += -abs(dot((p - q), (p - q))/(float)(sigma_p + 0.001); //3D位置
    w += -(float)(ID(p) != ID(q)) * sigma_id; //ID

    float weight = exp(w);
    weight       *= pow(max(0.0, dot(Normal(p), Normal(q))), sigma_n); // 法線

    color        += (weight * Color(q));
    totalWeight  += weight;
}
color /= totalWight;
```

©Silicon Studio Corp., all rights reserved.

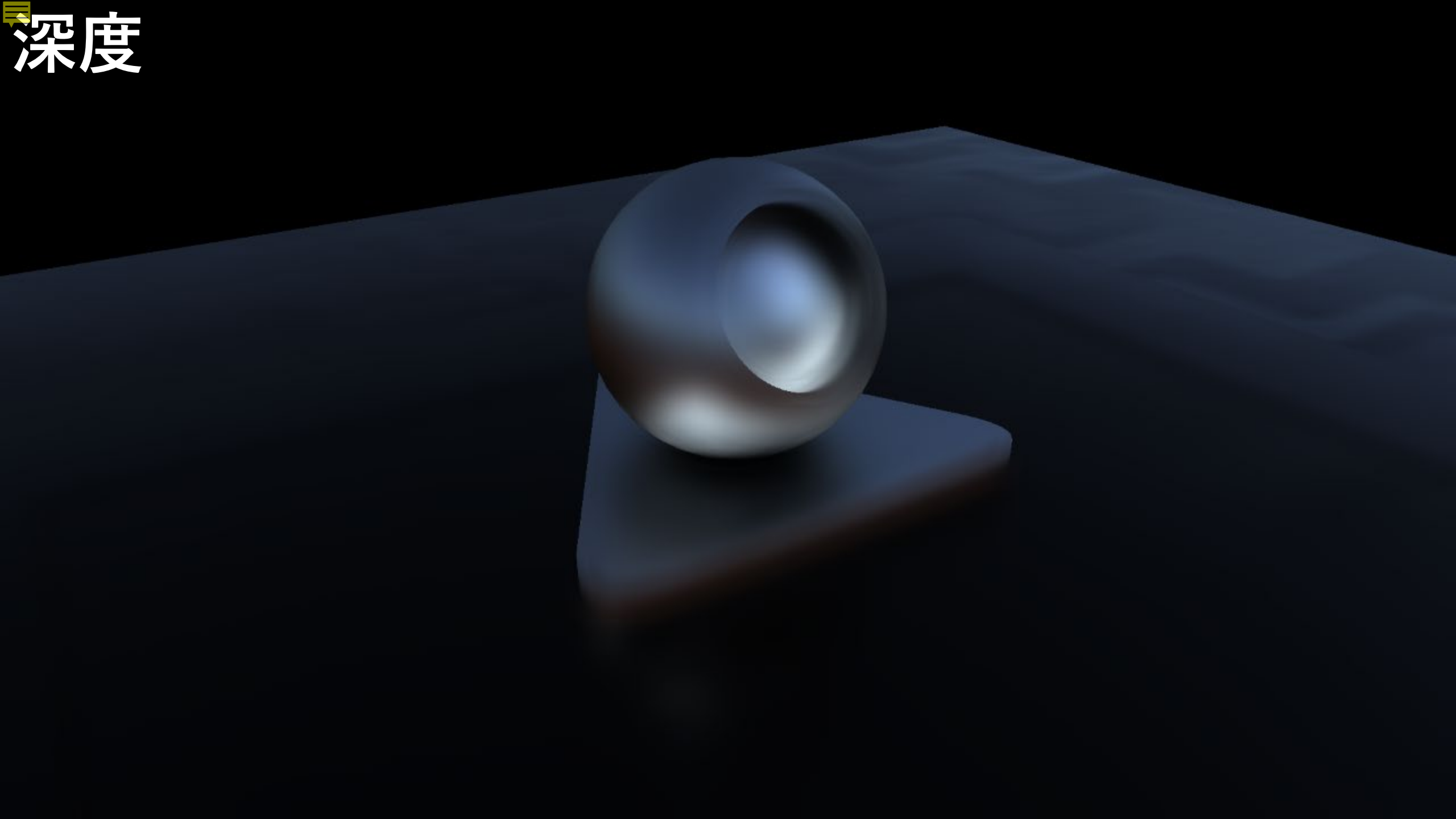
# フィルタなし



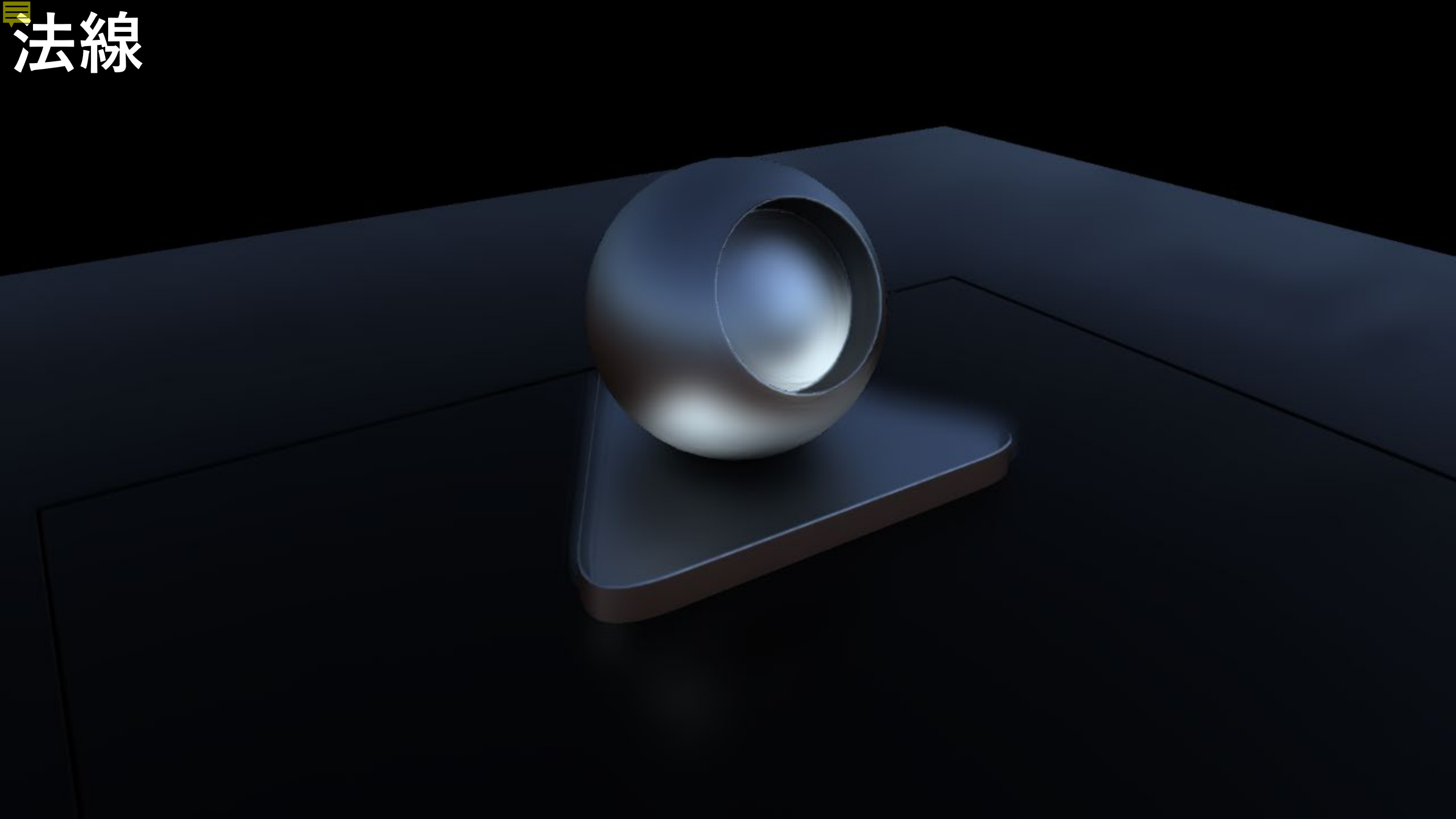
# Edge Stopping Function なし



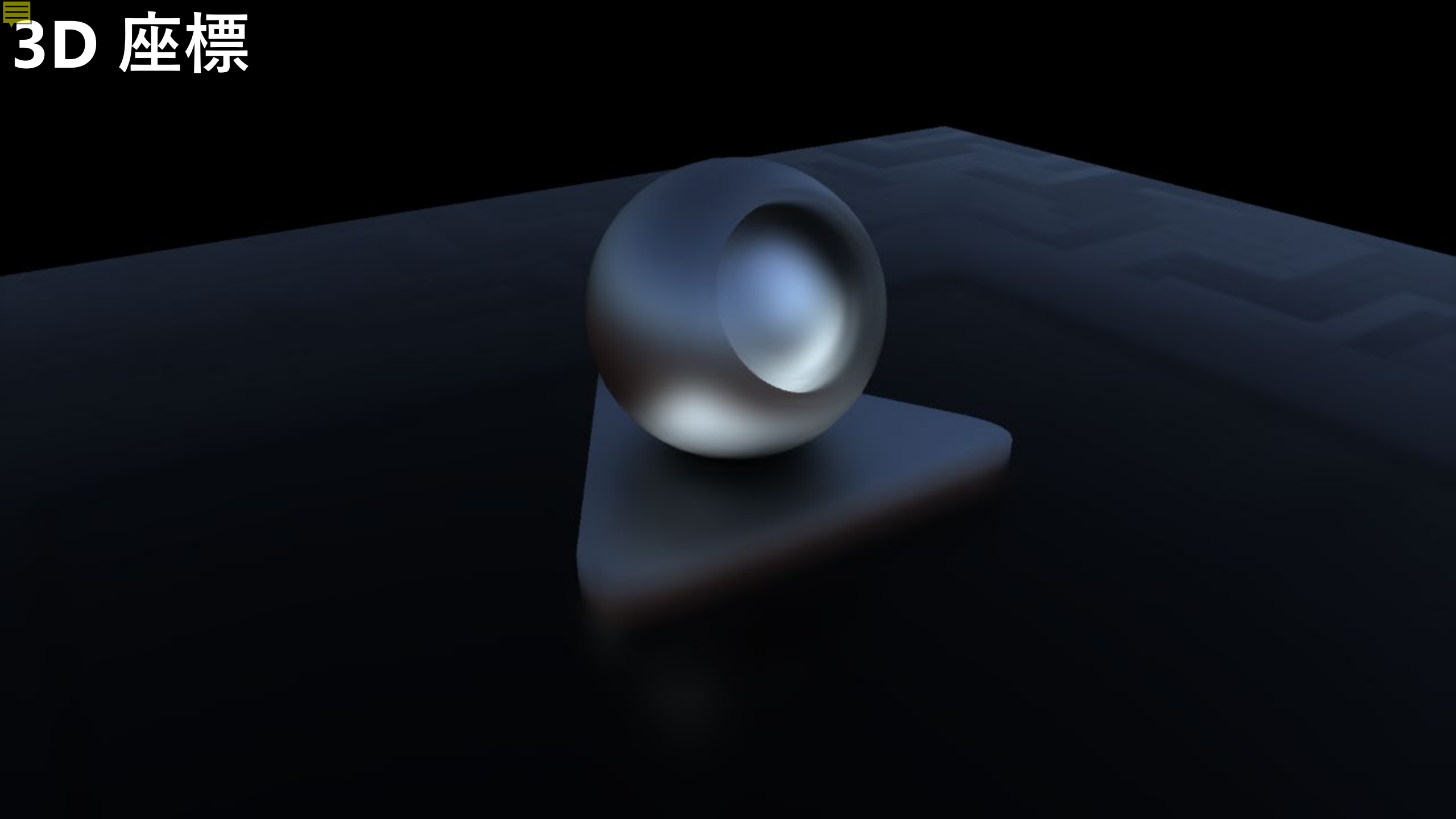
# 深度



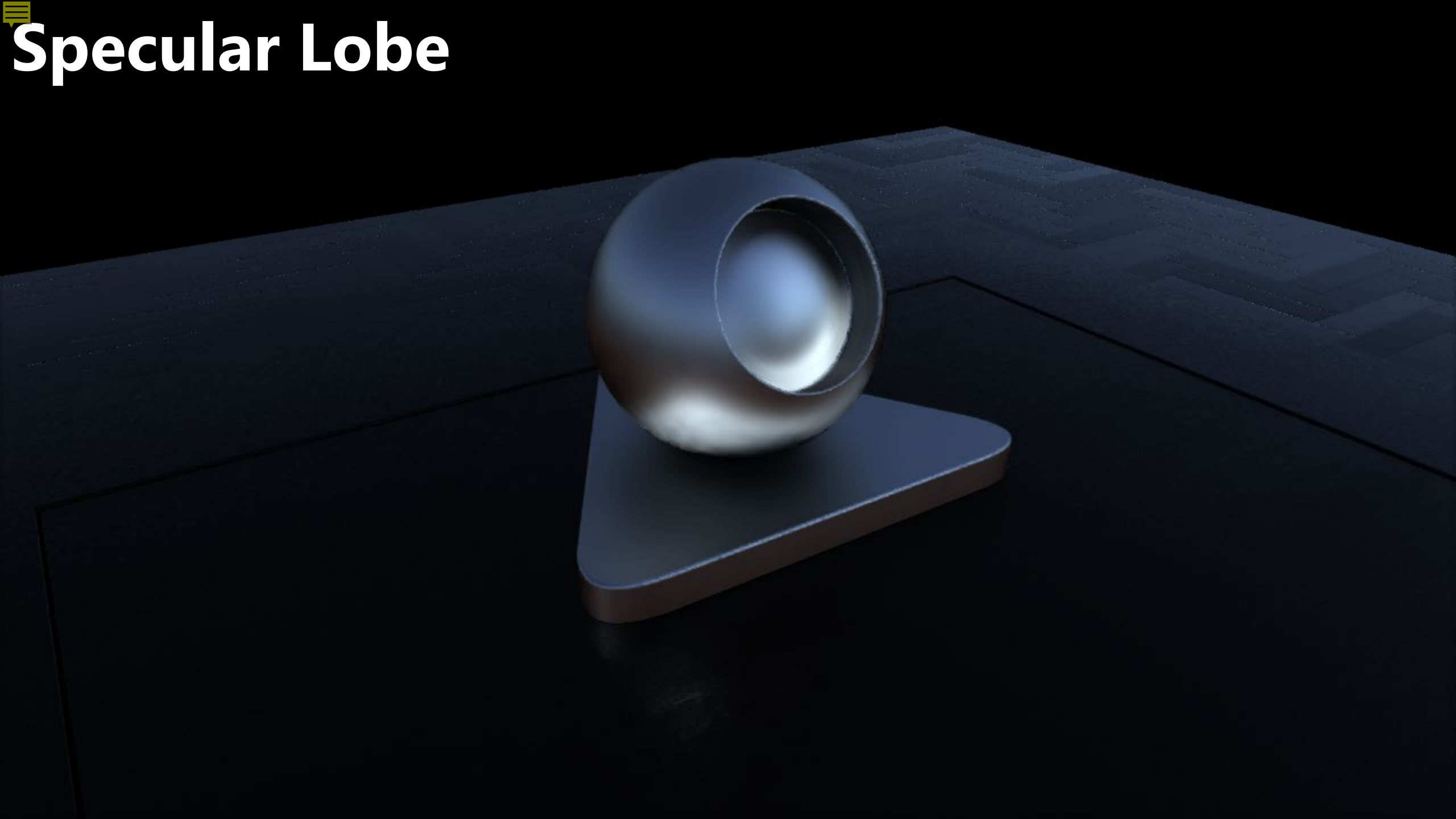
# 法線



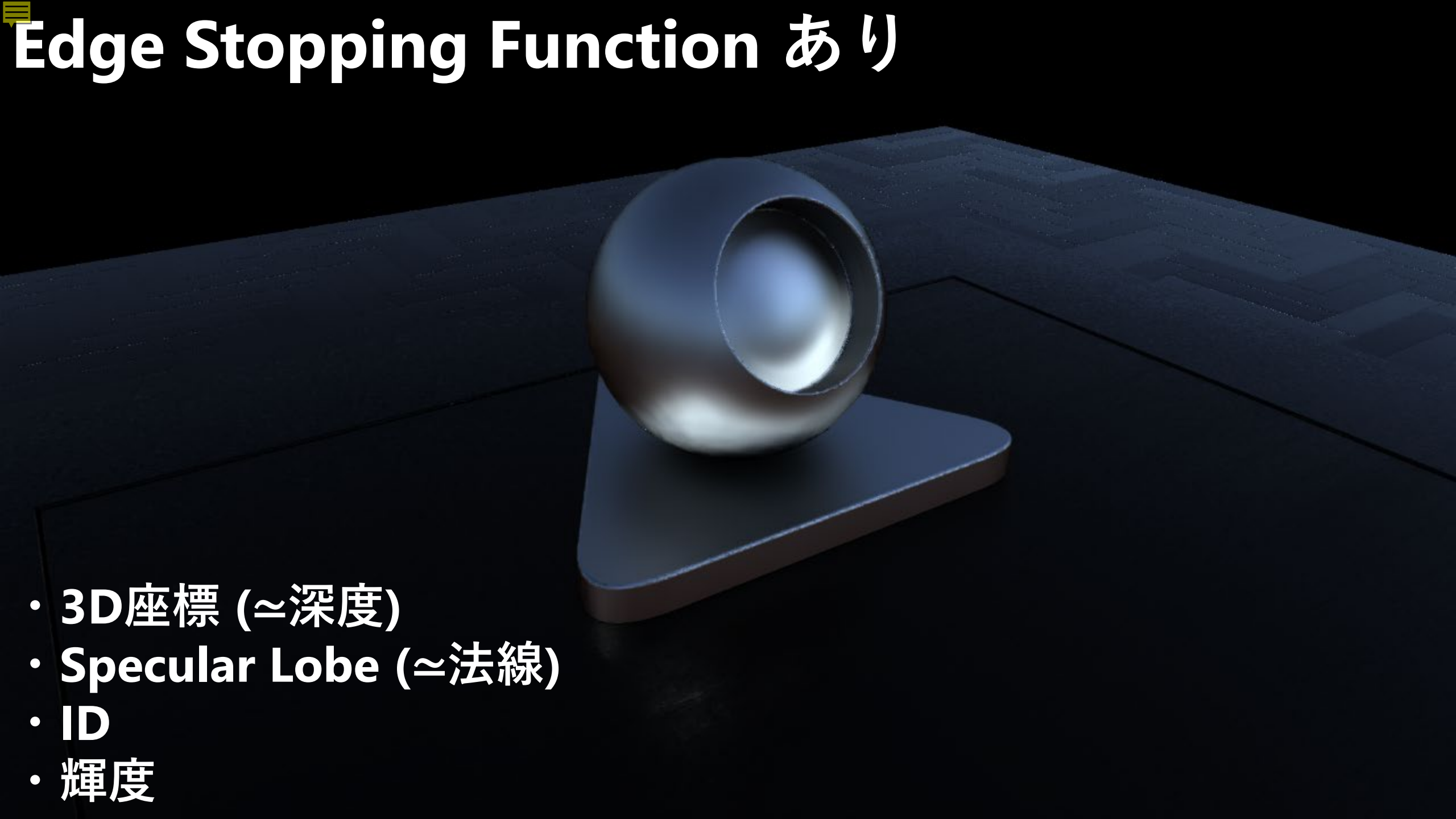
# 3D 座標



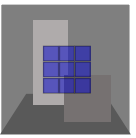
# Specular Lobe



# Edge Stopping Function あり



- 3D座標 (≃深度)
- Specular Lobe (≃法線)
- ID
- 輝度



# 4.1.5. 分散の利用

分散（Variance）はノイズの量を表す指標になる
- 分散(**小**)→ノイズ(**少**)→フィルタ半径(**狭**)
- 分散(**大**)→ノイズ(**多**)→フィルタ半径(**広**)

$$V[X] = E[(X - E[X])^2] = E[X^2] - E[X]^2$$

求めたい**分散の値**

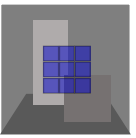
**輝度の平均値**
Spatial / Temporal Accumulation の結果がそのまま平均値として使える

**輝度の二乗の平均値（二次モーメント）**
輝度の二乗値を描き出しておいて同じように Accumulation する

**Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination [Schied et al. 2017]**

©Silicon Studio Corp., all rights reserved.

# アルゴリズム解説と実装



# 4.1.5. 分散の利用

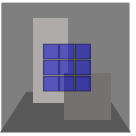
Spatial Accumulation → **Spatial Variance**　≃空間的なノイズの量
画像のエッジなどでも分散は高くなるのでノイズだけを表す指標ではない

Temporal Accumulation → **Temporal Variance**　≃時間的なノイズの量
（光源環境が変わらないなど）安定した環境下ならノイズの指標になる

基本的に Temporal Variance を使う
Disocclusion（後述）発生直後は Spatial Variance を使う

©Silicon Studio Corp., all rights reserved.

# アルゴリズム解説と実装



# 4.1.5. 分散の利用

Edge Stopping Function の輝度項で分散を使う

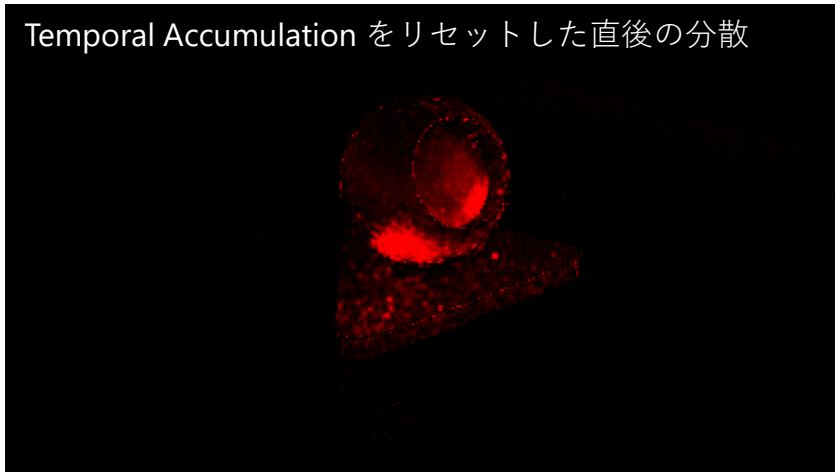
**輝度**

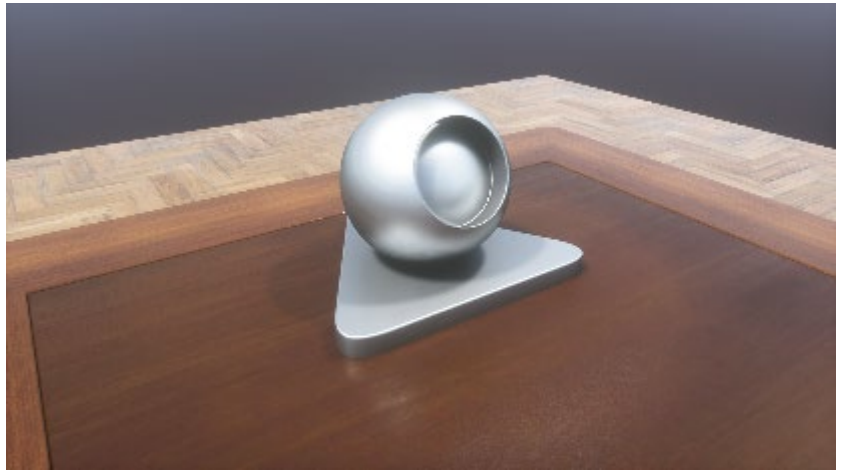
$$w_l = \exp\left(-\frac{|l(\mathbf{p}) - l(\mathbf{q})|}{\sigma_l \sqrt{g_{3\times3}\left(V(l(\mathbf{p}))\right)} + \varepsilon}\right)$$

分母の$V(l(\mathbf{p}))$が輝度分散を表す
分散によってフィルタサイズを調整できる

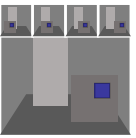
分散の低い箇所でフィルタを抑制するように働く
（絵としては効果がわかりにくい）
再帰回数やフィルタサイズに作用させる方法も考えられる

$l(\mathbf{p})$：ピクセル$\mathbf{p}$の輝度値
$V(l(\mathbf{p}))$：輝度分散
$g_{3\times3}(\cdot)$：ガウシアンカーネル



©Silicon Studio Corp., all rights reserved.

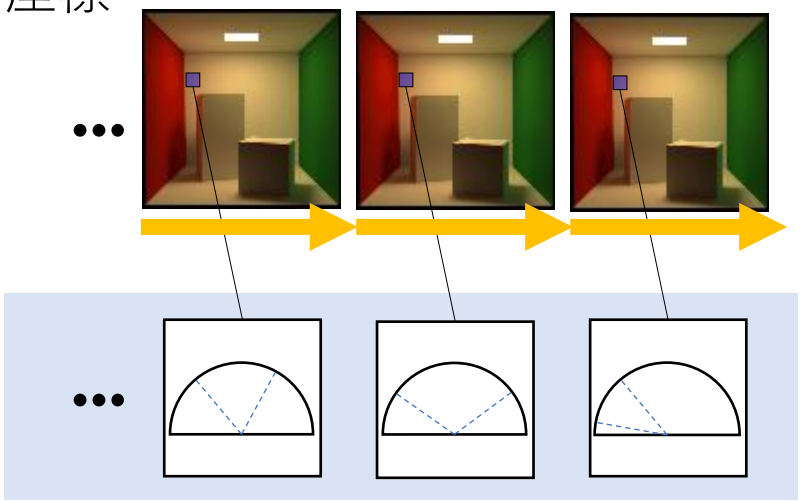


# 4.2. Temporal Accumulation 詳解

同一点の過去の結果を利用してサンプル数を補う

Temporal Anti-Aliasing と実質同じだが必要な過去の情報が多い

＊同一点≠スクリーン上の同じ座標



動くものやカメラに応じて3次元空間内の同一点を追跡する

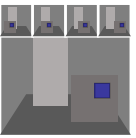
→ **Reprojection, Backprojection**

過去フレームではスクリーンに映っていなかった箇所も存在する（Reprojection が失敗する）

→ **Disocclusion**

©Silicon Studio Corp., all rights reserved.

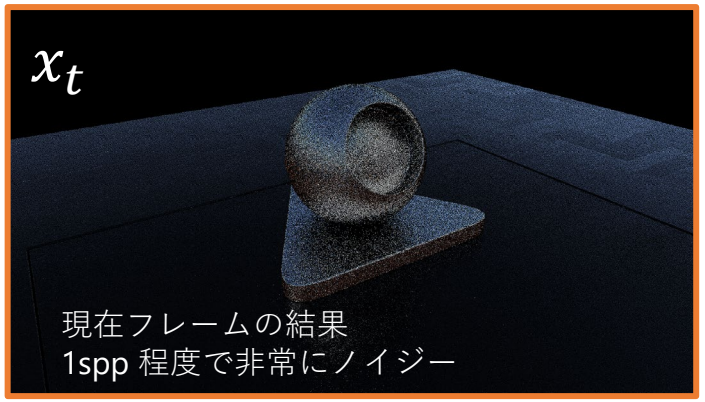
Silicon Studio
Ideas × Art × Technology

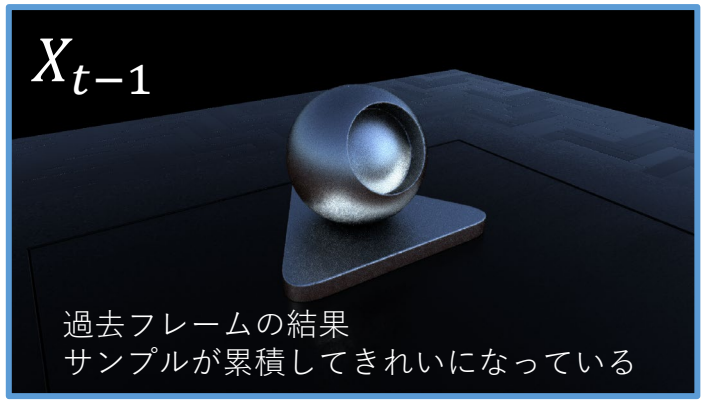


# 4.2.1. 指数移動平均フィルタ（EMAF）

ヒストリバッファに過去のすべてのサンプルの累積が記録される

指数移動平均（lerp）：$X_t = x_t \cdot \alpha + X_{t-1} \cdot (1 - \alpha)$




$\alpha$ **大**：現在のサンプルが支配的、補えるサンプル数が少なくノイズの多い結果
$\alpha$ **小**：過去のサンプルが支配的、動的なシーンの変化が残像となって表れやすい

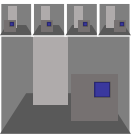
ピクセル毎の Reprojection の成功回数（History Length）を記録しておき $\alpha$ の決定に利用する
連続して成功する（安定している） → 過去フレームを信頼して利用できる→ $\alpha$ を小さくとれる

$$\alpha = \frac{1}{\text{History Length}}$$

History Length は最大24～30フレーム程度

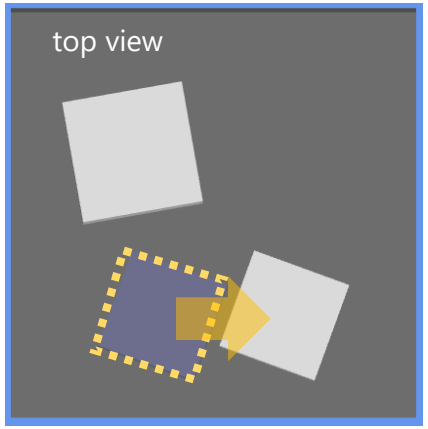
©Silicon Studio Corp., all rights reserved.

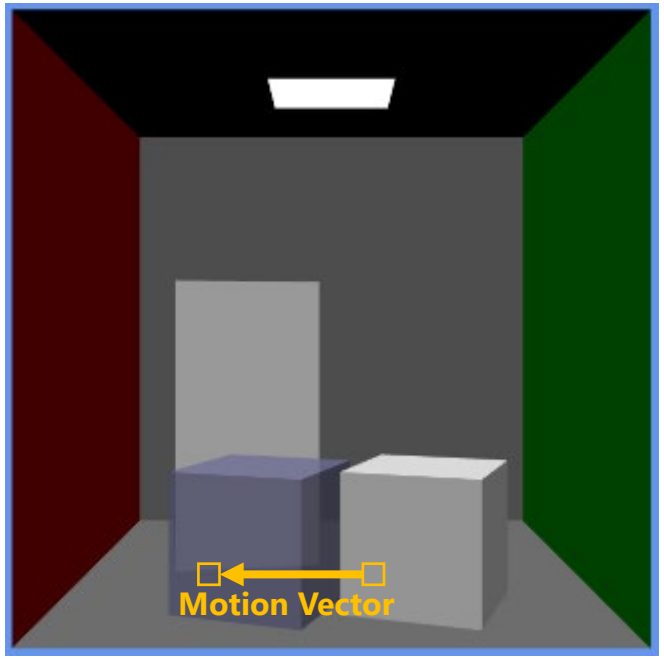
Silicon Studio
Ideas × Art × Technology



# 4.2.2. Reprojection

## **Motion Reprojection**：Motion Vector で同一点を追跡する



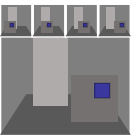




```
float2 MotionReprojection2D(...) // Motion Vector がスクリーン上の2Dのケース
{
    float2 UV_prev = UV_curr + motionVector.xy;
    return UV_prev;
}
float2 MotionReprojection3D(…) // Motion Vector が View-space の3Dのケース
{
    float3 vPos        = CalcViewSpacePos(…);
    float3 vPosPrev    = vPos +  motionVector.xyz;
    float4 ClipPosPrev = mul(float4(vPosPrev, 1.0), previousProjectionMatrix);
    float2 UV_prev     = ClipPosprev.xy / ClipPosprev.w * float2(0.5, -0.5) + 0.5;
    return UV_prev;
}
```

```
float2 UV_prev = MotionReprojection2D(...);
float4 CurrentColor = LoadColor(UV_curr);
float4 PreviousColor = LoadColor(UV_prev);
float4 AccumlatedColor = lerp(PreviousColor, CurrentColor, alpha);
```

動く物体の Motion Vector を使って前フレームの画面でどこに映っていたかを追跡する
再投影したスクリーン座標でヒストリバッファからサンプルを得て EMAF でブレンドする
スムースな面の鏡面反射は視線依存の特性が強く Motion Vector が役立たない

©Silicon Studio Corp., all rights reserved.



# 4.2.2. Reprojection

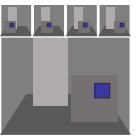
**Reprojection なし**

ハイライトがオブジェクトの動きについてこない

**Reprojection あり**

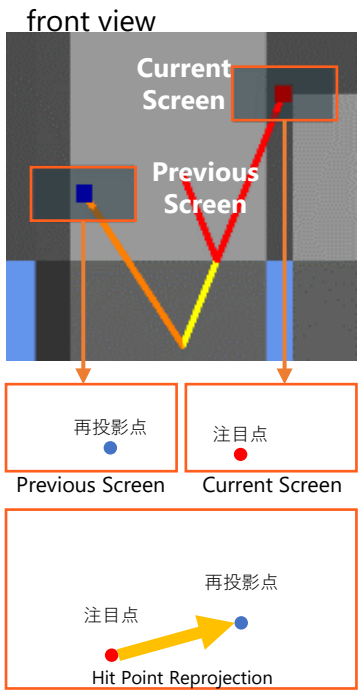
ハイライトがオブジェクトの動きに追従する

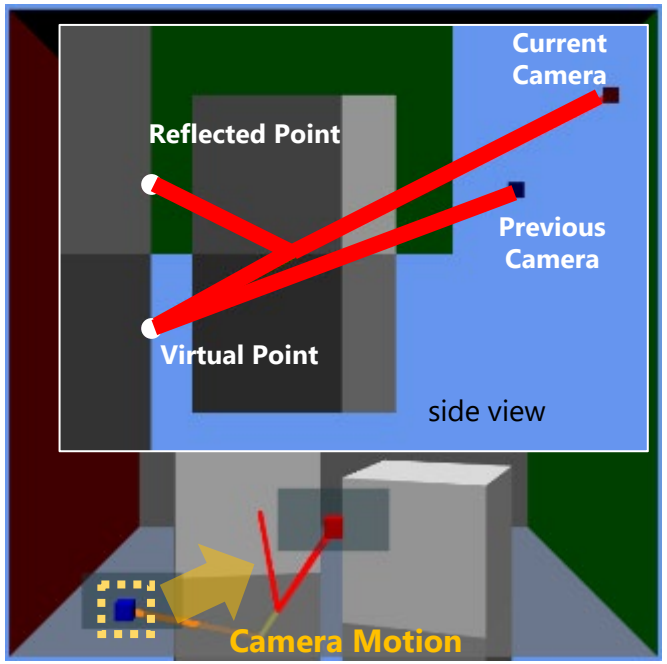
©Silicon Studio Corp., all rights reserved.



# 4.2.2. Reprojection

## Hit Point Reprojection：鏡面反射の仮想的なヒット点を追跡する



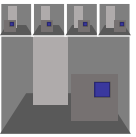




```
float2 HitPointReprojection(...)
{
    float3 vPos = CalcViewSpacePos(...);
    float primary_ray_length = length(vPos);
    vPos = normalize(vPos); // view vector

    float ray_length = primary_ray_length + reflected_ray_length;
    vPos *= ray_length; // extend ray to virtual reflected point

    float3 wPos = mul(float4(vPos, 1.0), inverseViewMatrix).xyz;
    float4 ClipPosPrev = mul(float4(wPos, 1.0), previousViewProjectionMatrix);
    float2 UV_prev = ClipPosPrev.xy / ClipPosPrev.w * float2(0.5, -0.5) + 0.5;
    return UV_prev;
}
```

```
float2 UV_prev_motion = MotionReprojection2D(...);
float2 UV_prev_hitpt  = HitPointReprojection(...);
float2 UV_prev = lerp(UV_prev_motion, UV_prev_hitpt, roughness > threshold);
```

仮想的な反射空間における点を過去のカメラのスクリーンに投影する
反射レイの長さを記録しておく必要がある（鏡面反射バッファのαに格納しておく）
ラフネスで Motion Reprojection と Hit Point Reprojection を選択する

©Silicon Studio Corp., all rights reserved.



# 4.2.2. Reprojection

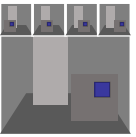
**Motion Reprojection**

外の映り込みが床に追従してしまう

**Hit Point Reprojection**

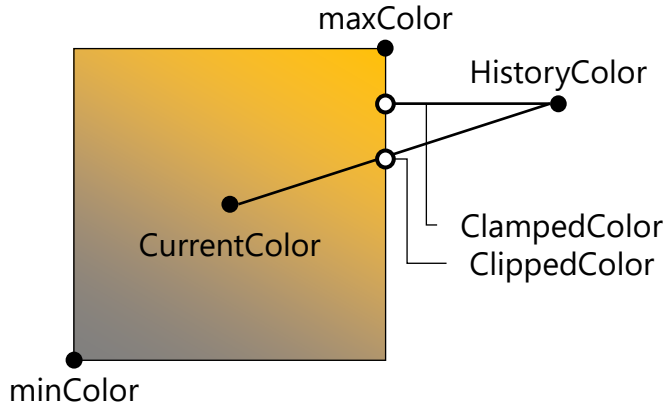
残像はあるが反射の位置はあっている

©Silicon Studio Corp., all rights reserved.



# 4.2.3. Clip / Clamp

Reprojection の精度には限界があり残像の発生は避けられない
注目ピクセルの周辺カラーの min/max で過去カラーの値を制限



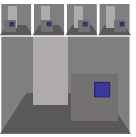

Clamp：min/max で色を Hisoty の値を制限
Clip：min/max で作った AABB とCurrent-History 線分の交差位置の色を取る

```
float4 mean  = filteredColor;  // 1次モーメントのフィルタカラー
float4 mean2 = filteredColor2; // 2次モーメント（輝度の二乗値）のフィルタカラー
float4 variance = mean2 - mean * mean;
float4 radius = StdDevScalar * sqrt(variance); // 標準偏差の半径スケール
float4 minColor = mean - radius;
float4 maxColor = mean + radius;
```

```
// Clip (AABB-Ray Intersection)
float3 dir = currentColor - historyColor;
float3 p0 = (minColor - historyColor) / dir;
float3 p1 = (maxColor - historyColor) / dir;
float t = max(max(min(p0.x, p1.x), min(p0.y, p1.y)), min(p0.z, p1.z));
historyColor = lerp(historyColor, currentColor, t);

// Clamp
historyColor = clamp(historyColor, minColor, maxColor);
```

Temporal AA でも使われる残像の低減手法　**High Quality Temporal Supersampling [Karis 2014]**
分散から min/max を求める Variance Clipping は Pre- Spatial Accumulation で効率的に求まる
**An Excursion in Temporal Supersampling [Salvi 2016]**

©Silicon Studio Corp., all rights reserved.



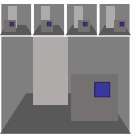
# 4.2.3. Clip / Clamp

Hit Point Reprojection のみ

Hit Point Reprojection + Variance Clipping

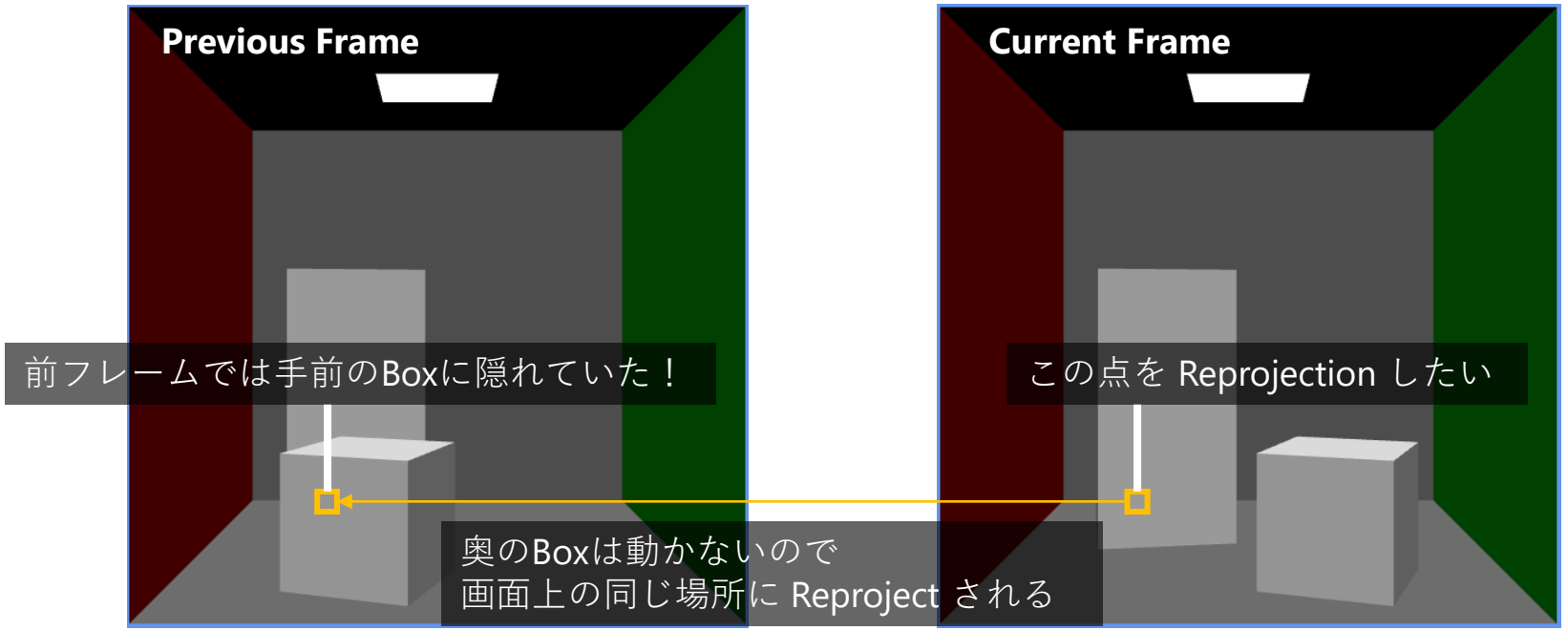
Hit Point Reprojection と Variance Clipping を組み合わせて
残像感の無い綺麗な Accumulation が達成された

©Silicon Studio Corp., all rights reserved.



# 4.2.4. Disocclusion –概要

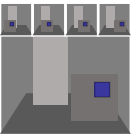
オブジェクトの移動などで画面に現れる Reprojection が失敗する点




Disocclusion 検出の専用パスで領域をマスクとして書き出しておくと利用しやすい

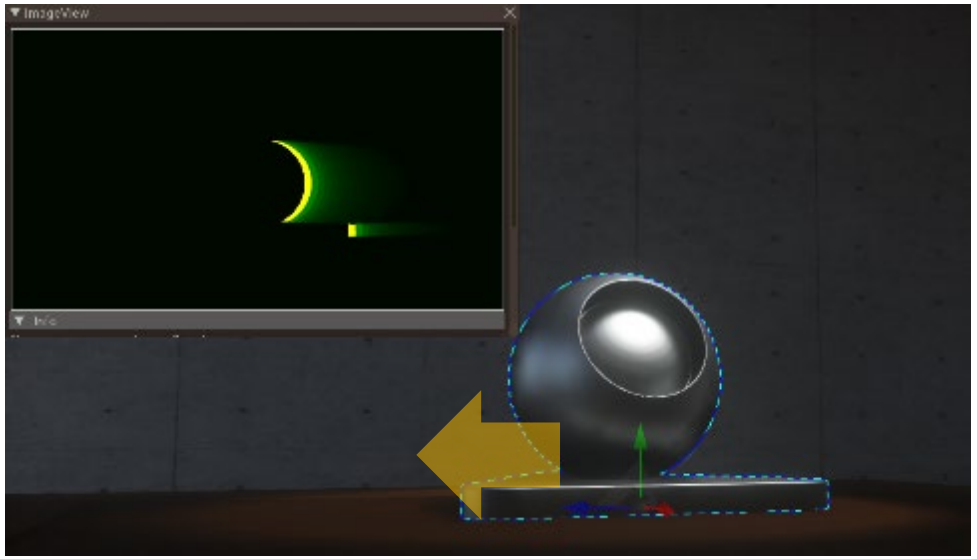
- 同時に History Length も記録できる
- 後処理でDisocclusion領域のノイズ（ジッタリングなど）を除去できる
- 後処理で軽いブラーを適用して境界をなじませることができる

©Silicon Studio Corp., all rights reserved.



# 4.2.4. Disocclusion –検出

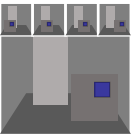
Disocclusion 検出には過去フレームのジオメトリ情報を利用する



```
bool bSuccess = all( UVprev == SATURATE(UVprev)); // in screen
bSuccess = bSuccess && (abs(historyDepth - centerDepth)  < depthThreshold);
bSuccess = bSuccess && (-dot(historyNormal, centerNormal) * 0.5 + 0.5) < normalThreshold;
bSuccess = bSuccess && (centerID == historyID);

float historyLength = rcp(DisocclusionOutput[pixelIDX].y);
historyLength = bSuccess ? min(historyLength + 1, MAX_FRAMES_COUNT) : 1.0;
DisocclusionOutput[pixelIDX]   = float2(!bSuccess, rcp(historyLength));
```

Reprojection 前後が異なるオブジェクトであったら Disocclusion が発生したと考える
異なるオブジェクトかどうか現在と過去のジオメトリ情報を比較して判定する
・オブジェクトID、法線方向、深度、etc...
過去フレーム GBuffer や深度バッファを保持しておく必要がある

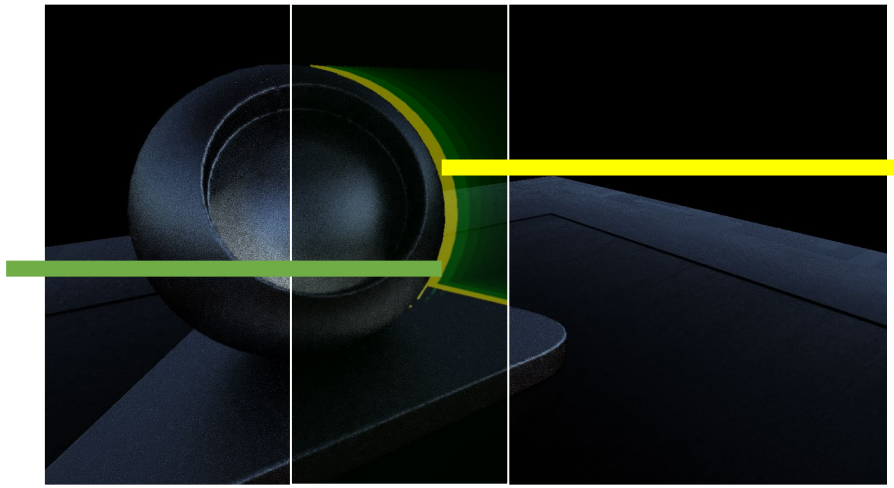
©Silicon Studio Corp., all rights reserved.



# 4.2.4. Disocclusion –補完

Disocclusion 領域はTemporal Accumulation が使えない

History Length が短い（緑）
現在フレームの割合が多め



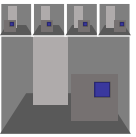
Disocclusion Mask（黄色）
現在フレームの結果だけで補完する

強い Spatial Accumulation で補完する必要がある（Disocclusion Fix）
・Temporal Accumulation 前に Pre- Spatial Accumulation を行う
・Temporal Accumulation 後に Disocclusion 領域を強くフィルタする

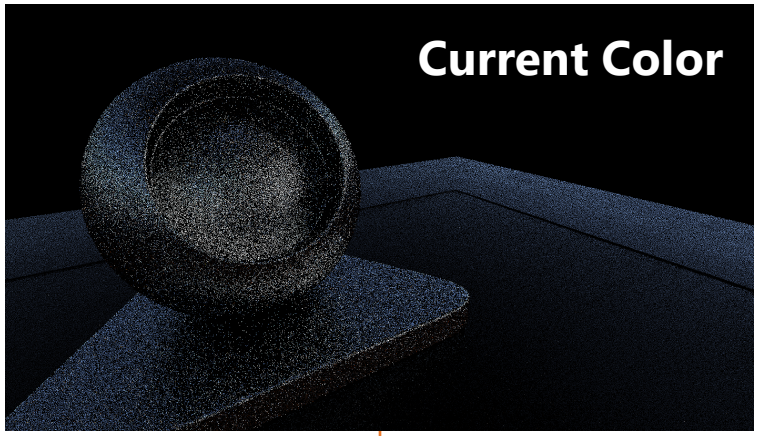
今回は Pre- Spatial Accumulation を採用しているので前者

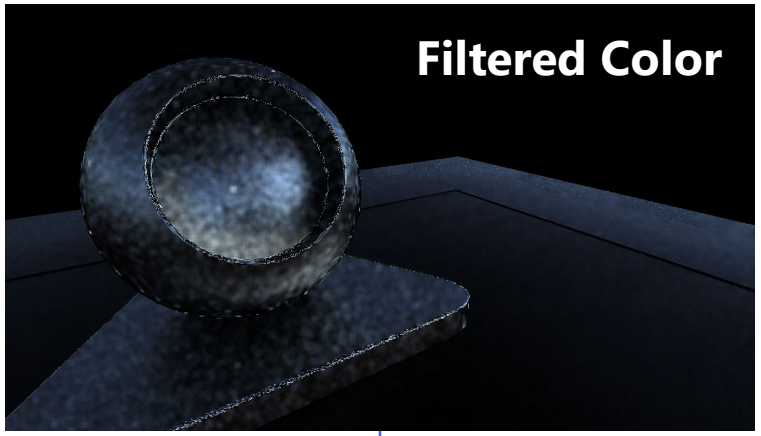
©Silicon Studio Corp., all rights reserved.

# アルゴリズム解説と実装



## 4.2.4. Disocclusion –補完



with Reprojection

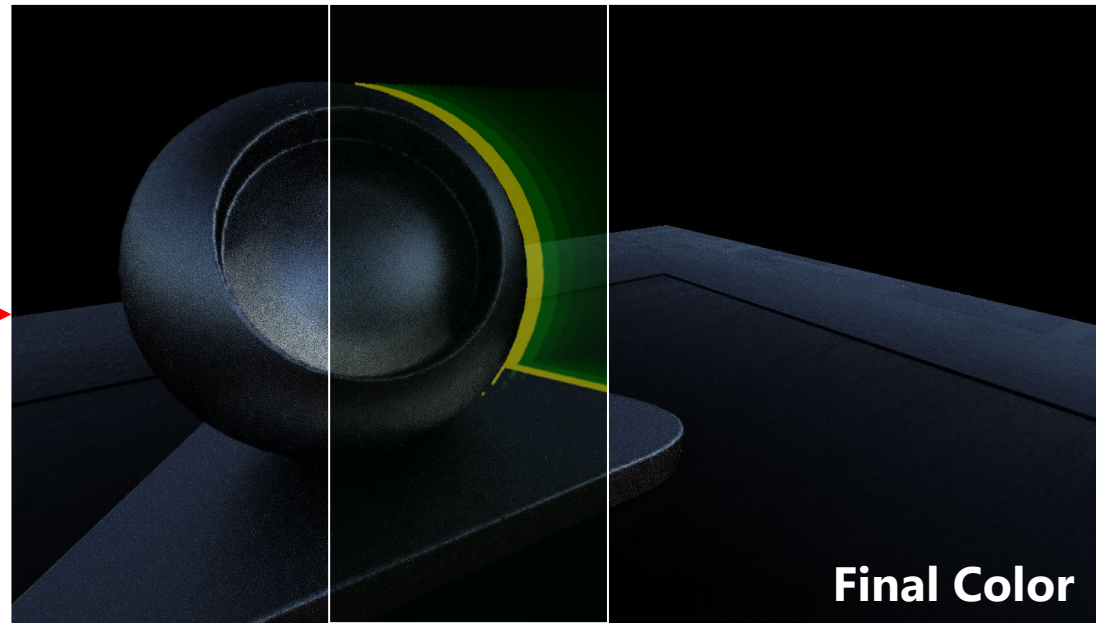
$$X_t = x_t \cdot \alpha + X_{t-1} \cdot (1 - \alpha)$$

緑← (短) History Length (長) →黒

大　←α→　小

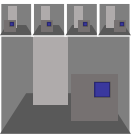
$$X'_t = X_t \cdot (1 - \beta) + g(x_t) \cdot \beta$$

黄：Disocclusion Mask→β=1




©Silicon Studio Corp., all rights reserved.

# アルゴリズム解説と実装

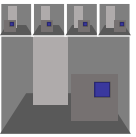


## 4.2.4. Disocclusion

Disocclusion 補間無し

Disocclusion 補間あり

©Silicon Studio Corp., all rights reserved.



# 4.2.5. 動的な光源

## Reprojection は光源環境の変化を追跡できない

- 現在フレームの推定輝度を求めて Temporal Accumulation の結果を補正する（SLGF）
  現在フレームの推定輝度として Pre- Spatial Accumulation が利用できる

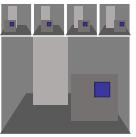
- 現在と過去フレームの輝度変化率を求め変化が大きいときは現在フレームの結果を多く使う
  前フレームの位置を完全に追跡するために乱数シードの再利用などが必要（A-SVGF）

**採用手法（さらなる改善が必要）**
輝度を見て輝度差が大きいときは現在フレームのフィルタ値を多く混ぜる
A-SVGF のような Reprojection の信頼性は保証しない
輝度差の計算にはPre- Spatial Accumulation の結果を使う
閾値の調整が難しく残像やちらつきが生じやすい

```
float3 accumulatedColor = lerp(historyColor, currentColor, alpha);
float luminanceDiff = abs(Luma(filteredColor) - Luma(accumulatedColor)); // 輝度差の計算はフィルタした値を使う
float luminanceCheck = (float)(luminanceDiff > luminanceThreshold);
resultColor = lerp(accumulatedColor, filteredColor, luminanceCheck);
```

©Silicon Studio Corp., all rights reserved.

# アルゴリズム解説と実装



## 4.2.5. 動的な光源

**輝度閾値処理無し**
一度影の明るく（暗く）なると戻りにくい

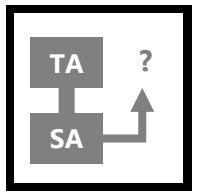
**輝度閾値処理あり**
しっかりと残像なく影の移動が見て取れる

閾値設定が甘いと汚い残像が残る→

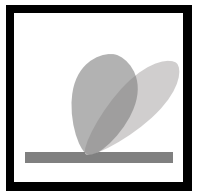
©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

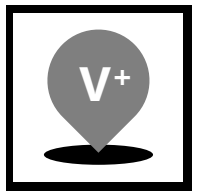
# 4.3.1. 実装の考察点



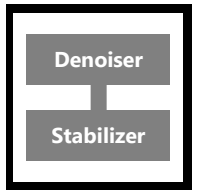

Temporal Accumulation で Spatial Accumulation の**結果を再帰すべきかどうか**は重要な問題
強いフィルタ結果を再帰するとぼやけた結果は残り続けて NG
サンプル数が少ないフィルタを再帰するのは許容範囲（再帰が進んだら半径を小さくする）



Specular Lobe Aware Filter はかなり有効だが**サンプル数が少ないとノイズ**が残りやすい
Temporal Accumulation が進むまでは効果を抑える方がよい




輝度分散は黒い領域に反応できないが、分散に意図的な値を加える**分散ブースト**も有効
ただし分散を使った処理そのものがモーメント保持や品質、調整しやすさから**オミットも視野**




一度の Temporal Accumulation では時間的なジラジラが残りやすい
追加で軽い Temporal Accumulation を行って時間的な安定性を上げる **Temporal Stabilization**
Temporal Anti-Aliasing がエンジンに搭載されていればそのまま利用すればよい

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 4.3.2. 特殊な信号の扱い

## 完全鏡面の扱い

実質的にフルのパストレーシングを扱うことになる

**完全鏡面は間接鏡面反射光**

・ただし拡散反射成分もそのまま含まれる
・さらに GBuffer にジオメトリ情報が存在しない

**信号の分離**

完全鏡面上の拡散反射成分は拡散反射の信号に含める
鏡に色がついていると扱いが複雑
今回は非採用（完全鏡面上の拡散反射光も鏡面反射光のデノイザーで扱う）

**セカンダリ GBuffer**

レイトレーシング時に完全鏡面上のジオメトリ情報を描きだす

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 4.3.2. 特殊な信号の扱い

## 拡散反射・半透明

鏡面反射以外の信号は異なるテクニックが求められる

**間接拡散反射光**

サンプル数が大きく不足する→フィルタ半径を非常に大きくとる、解像度を落とす
BRDF 適用前の入射輝度と入射方向 を SH で管理する
デノイズの過程で一次サーフェスの情報が失われない
レイトレ輝度(RGB)→ (YCoCg) → [Y, 方向] の SH (L1 まで 4成分)＋CoCg
結果をシーンに合成する際に法線で SH を展開し BRDF を適用

**Precomputed Global Illumination in Frostbite (GDC 2018)**

**半透明（Future works）**

信号の性質自体は鏡面反射に近く、鏡面反射に含めて扱うケースも多い
ただし屈折を伴う半透明は Reprojection が非常に難しい

**その他（Future works）**：クリアコート、SSS など特殊材質

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 4.3.3. その他のテクニック

## Firefly Rejection

確率的なパストレーシングでは高輝度の点が生じることがある
ただ Spatial Accumulation すると輝点が広がってしまう

- **Clamp**
  輝度値の範囲がある程度わかっているなら有効
  ただしTemporal Accumulation でエネルギー損失が累積する
  分散を使って外れ値を補正する方法もあるがエネルギー損失は同様に起こる

- **Path Regularization**
  BRDF そのものをブラーして高輝度を避けるという考え方
  レイが反射するたびにラフネスを増やすことで得られる

- **Median Filter, Min/Max Filter (Opening)**
  ゴマ塩ノイズ除去に適した画像処理フィルタを Spatial Accumulation の前処理に入れる
  ただしエネルギーが狂うので Temporal Accumulation に再帰させるのはNG

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 4.3.3. その他のテクニック

## Blue Noise

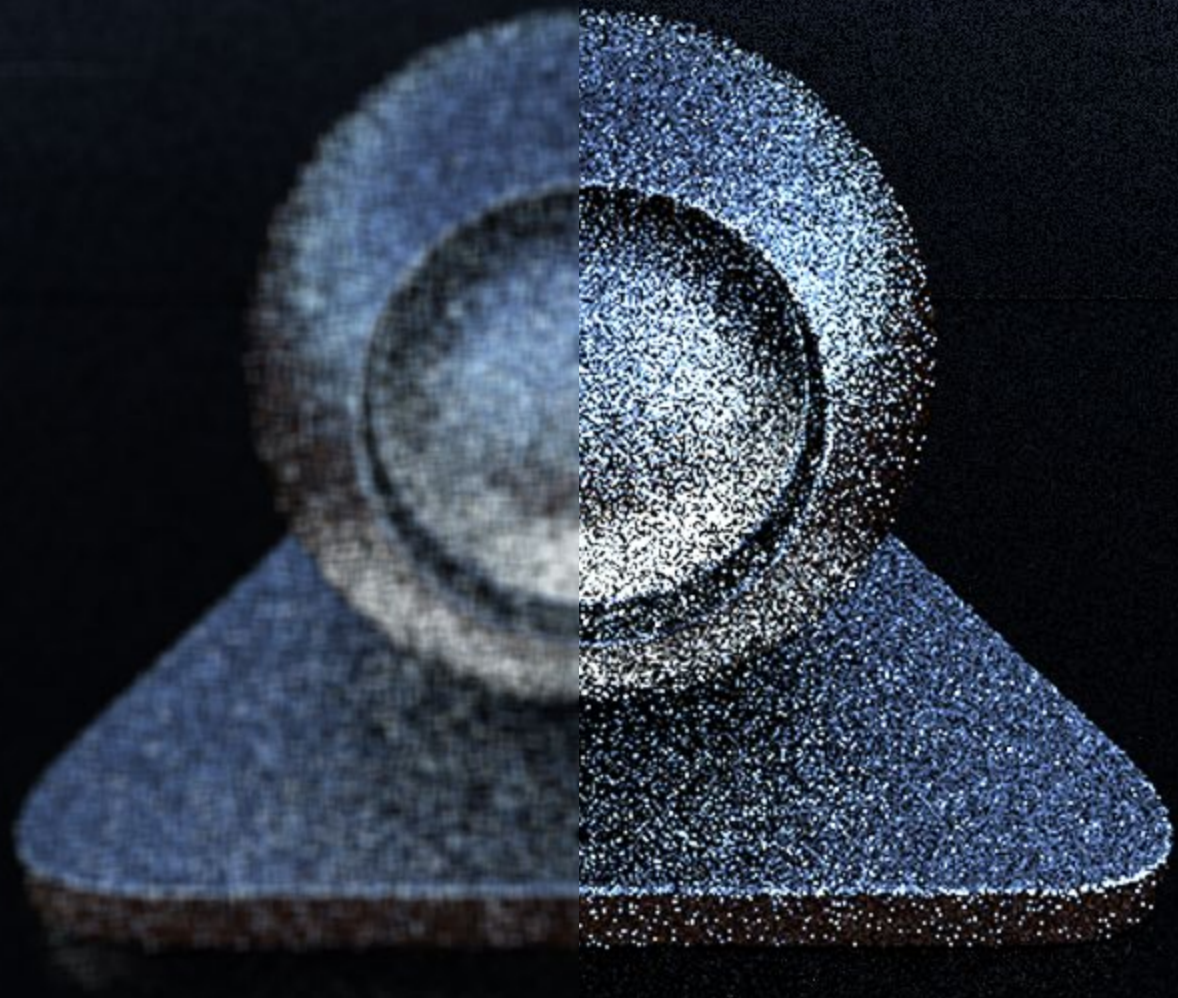
パストレーシングのノイズ自体に良好な特性を持たせることも重要

パストレーシング結果のノイズ分布が画面上で Blue Noise 特性を持つとデノイズしやすい
Blue Noise：低周波成分が少ないノイズ → 狭いフィルタ半径でノイズを消せる

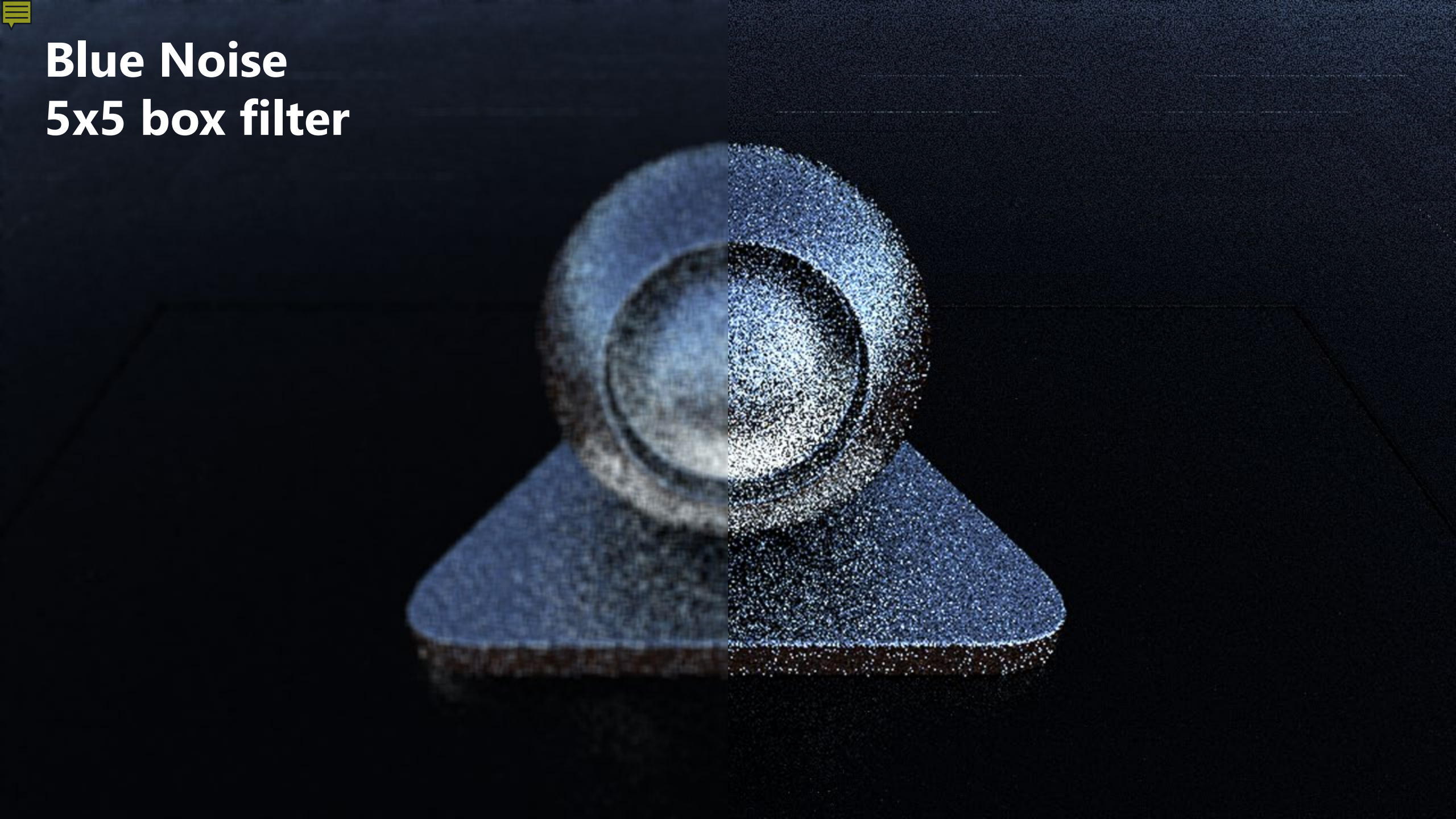
**Distributing Monte Carlo Errors as a Blue Noise in Screen Space by Permuting Pixel Seeds Between Frames [Heitz and Belcour 2019]**

©Silicon Studio Corp., all rights reserved.

# White Noise
# 5x5 box filter



# Blue Noise
# 5x5 box filter



# 5 デノイザー設計

©Silicon Studio Corp., all rights reserved.

# デノイザー設計とパフォーマンス

Silicon Studio
Ideas × Art × Technology

# 5.1. デノイザーの処理フロー

- GBuffer
- RayTracing
- Metadata
- Disocclusion
- Pre- Spatial Accumulation
- Temporal Accumulation
- Post- Spatial Accumulation

ジオメトリ情報を **GBuffer** に描き出す

・深度
・法線
・モーションベクトル
・ラフネス

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 5.1. デノイザーの処理フロー

- GBuffer
- RayTracing
- Metadata
- Disocclusion
- Pre- Spatial Accumulation
- Temporal Accumulation
- Post- Spatial Accumulation

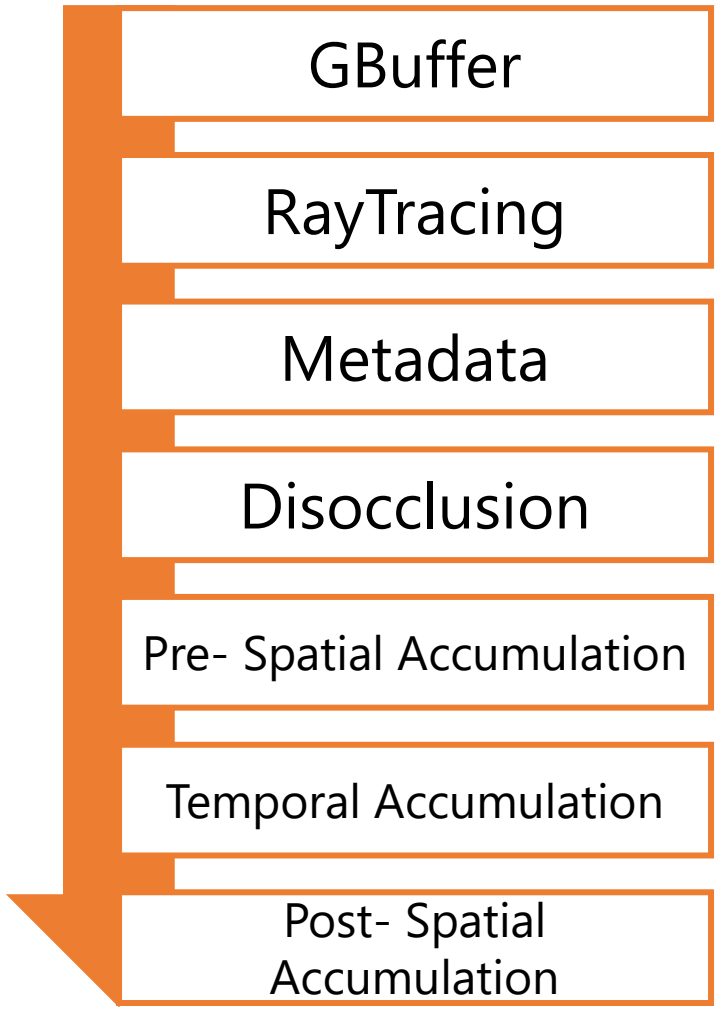
各信号ごとにレイトレーシング
- 影
- **間接鏡面反射**
- 間接拡散反射
- 半透明
- **完全鏡面上の二次ジオメトリ情報**

分散を利用するため輝度の二乗も描き出す
（鏡面反射成分のみ）

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 5.1. デノイザーの処理フロー




デノイズに利用する情報を **Metadata** として整理しておく

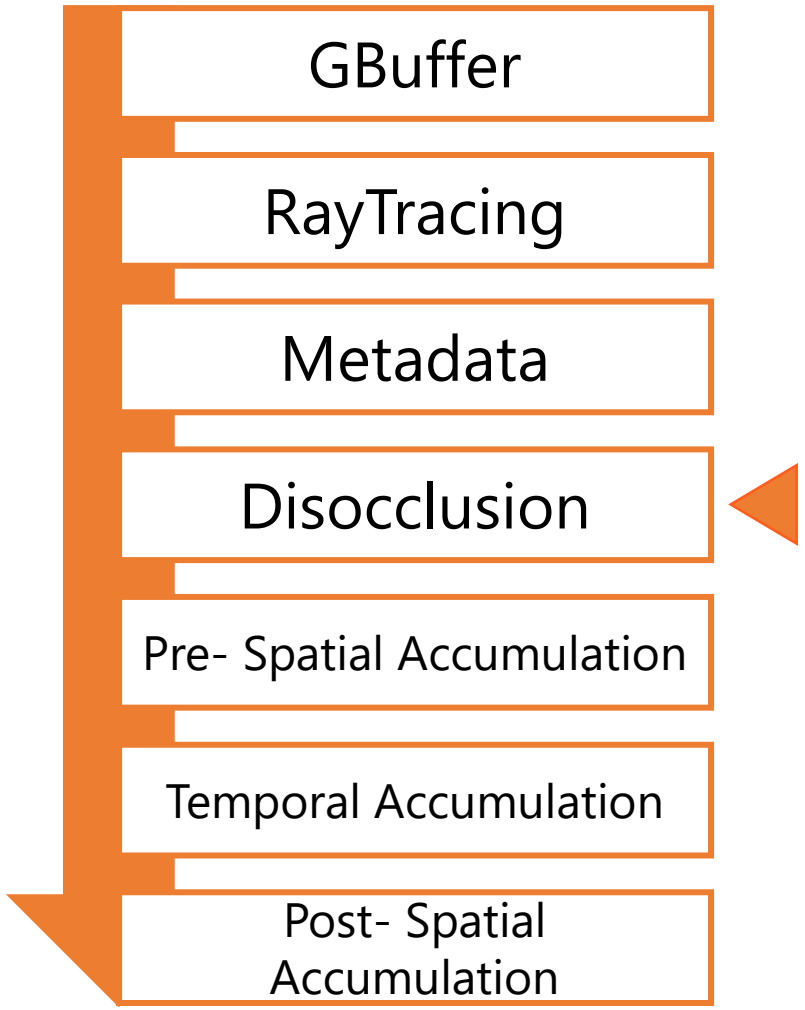
必要に応じて一次ジオメトリ（GBuffer）と完全鏡面上の二次ジオメトリ（RayTracing）の情報を統合する

- 深度（View Z も可）
- 法線
- モーションベクトル
- ラフネス
- ID（レイトレのインスタンスID）

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 5.1. デノイザーの処理フロー




Disocclusion を検出してマスクとして描き出す
同時にReprojection の成功回数も描き出す
（History Length）

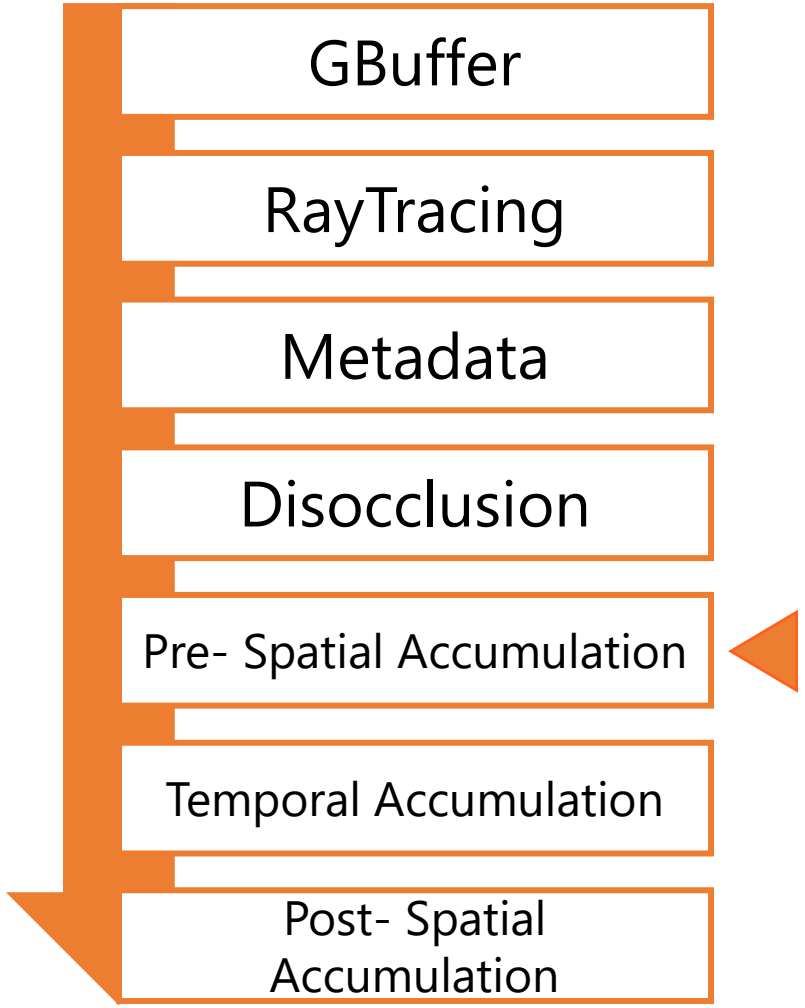
必要に応じて平均値フィルタをかける
（領域の広がる方向にのみ）

©Silicon Studio Corp., all rights reserved.

# デノイザー設計とパフォーマンス

Silicon Studio
Ideas × Art × Technology

## 5.1. デノイザーの処理フロー



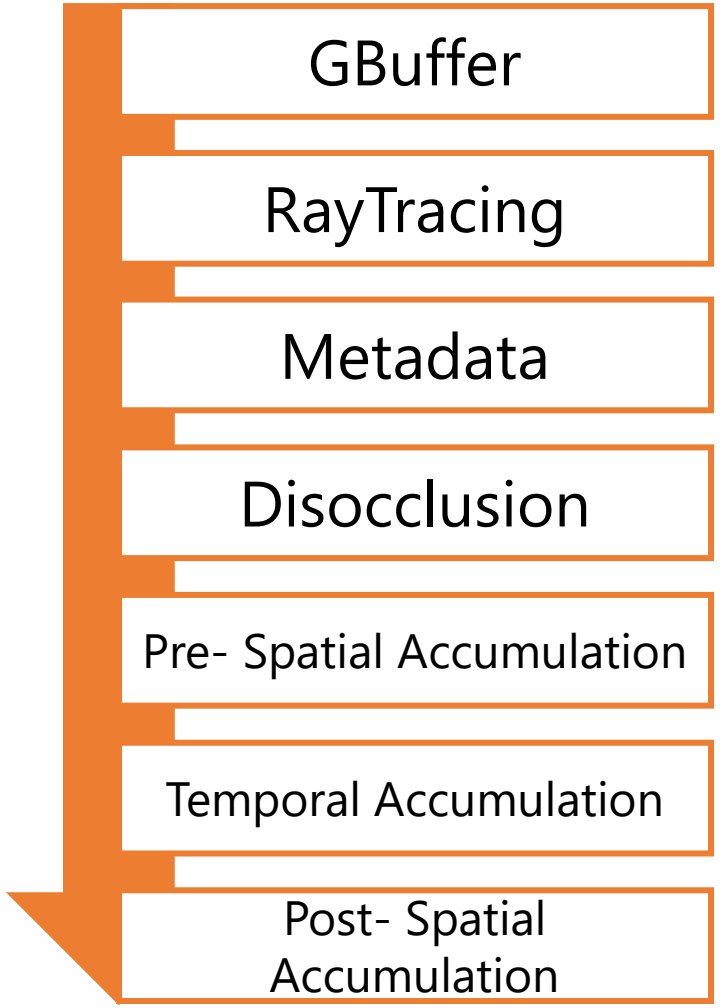

事前 Spatial Accumulation を行う
Temporal Accumulation の処理に利用する
・Disocclusion補完
・Variance Clipping

必要に応じてフィルタ結果の Mipmap を生成
より強いフィルタ結果として利用する
MaxMipLevel*roughness で選ぶとよい

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 5.1. デノイザーの処理フロー




Temporal Accumulationを行う
- Reprojection
- EMAF
- Disocclusion 補完
- Variance Clipping

Temporal Accumulation の結果で History Buffer を更新する
（Post- Spatial Accumulation は再帰させない）
Metadata の History も更新しておく

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 5.1. デノイザーの処理フロー

GBuffer

RayTracing

Metadata

Disocclusion

Pre- Spatial Accumulation

Temporal Accumulation

Post- Spatial Accumulation

事後Spatial Accumulationを行う

必要に応じて鏡面反射は1パス目で Median Filter をかけて Firefly Rejection を行う

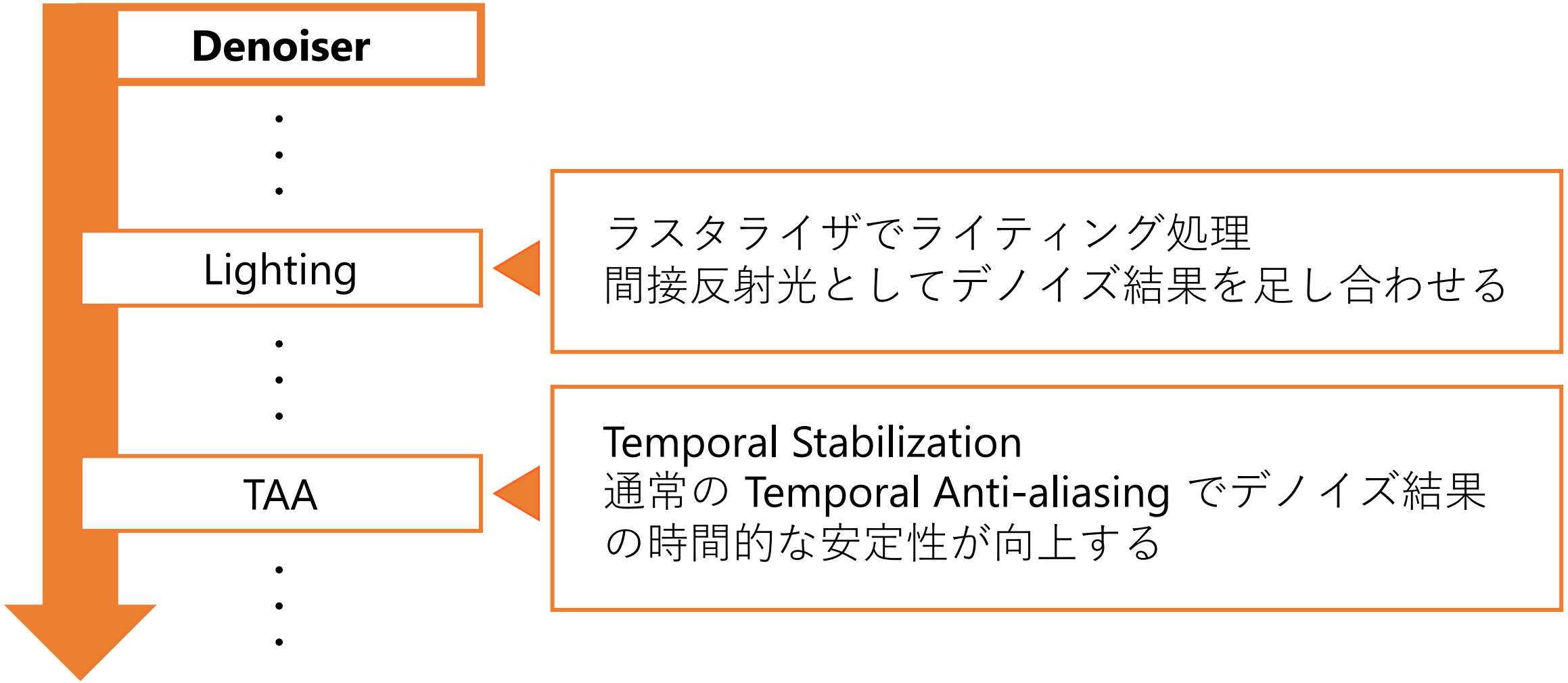
必要に応じてより強いフィルタのため Mipmap を生成する
Mipmap をシーン解像度に合わせるための Upsampling も行う

得られた最終的な絵はライティング時に間接光として合成する

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 5.1. デノイザーの処理フロー

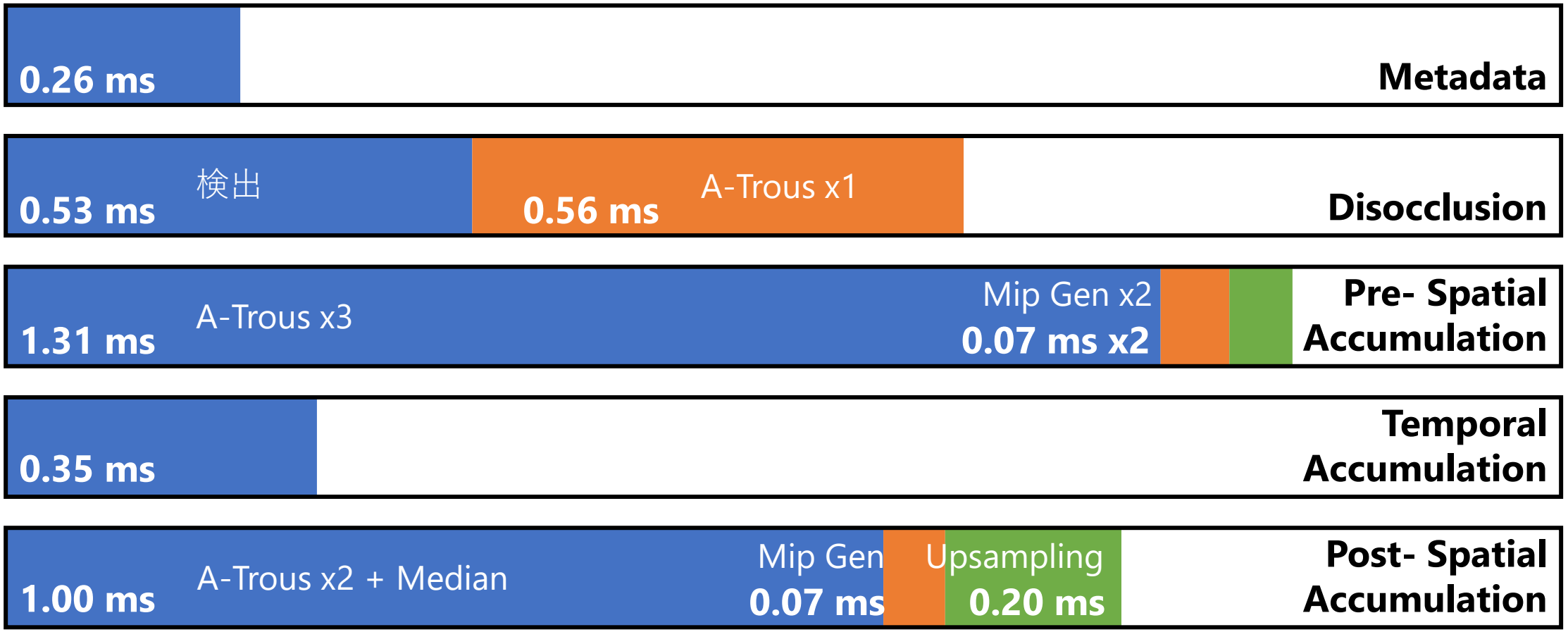



©Silicon Studio Corp., all rights reserved.

# デノイザー設計とパフォーマンス

Silicon Studio
Ideas × Art × Technology

## 5.2. パフォーマンス




＊1920x1080 NVIDIA TITAN RTX で測定

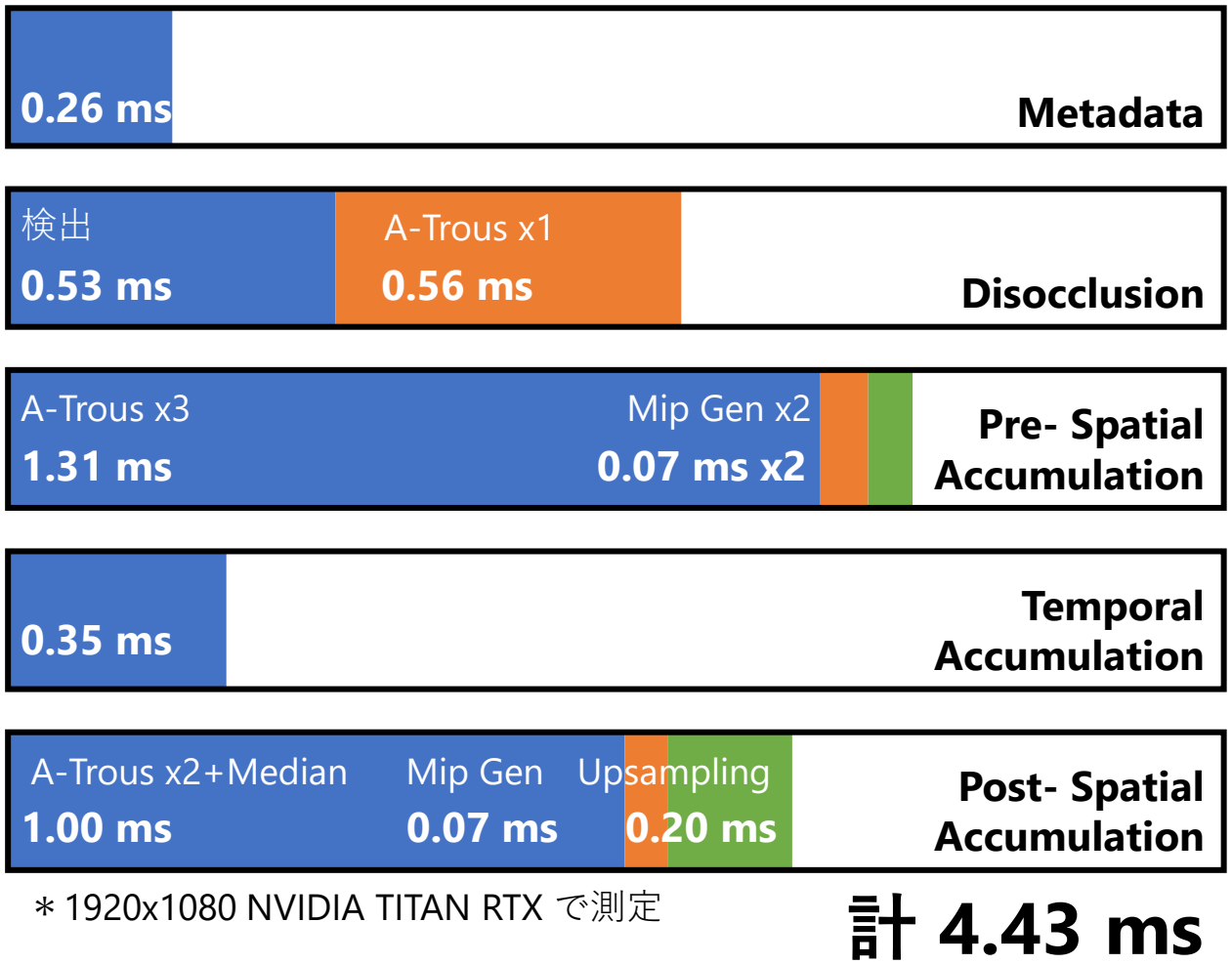
**計 4.43 ms**

©Silicon Studio Corp., all rights reserved.

# デノイザー設計とパフォーマンス

Silicon Studio
Ideas × Art × Technology

## 5.2. パフォーマンス




＊1920x1080 NVIDIA TITAN RTX で測定

**計 4.43 ms**

- Spatial Accumulation の改善は必須
  今回は検証用コードも含まれる
  小さいフィルタを再帰に含める Recurrent Blur の手法も有効か

- Disocclusion は TA 内で扱ってもよい

- Pre- SA は分散や Disocclusion の処理など有用な場面も多いが処理時間に見合わない
- Spatial Accumulation は影響範囲が重要
- Mip4 とA-Trous 3回が同じ影響範囲なので、

同時に使ってもサンプル数収集には非効率

- 実情としては Ray Tracing の負荷が支配的で数十 ms はかかってしまっている

©Silicon Studio Corp., all rights reserved.

# 6 デノイザーの未来とセッションまとめ

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 6.1. デノイザーの未来

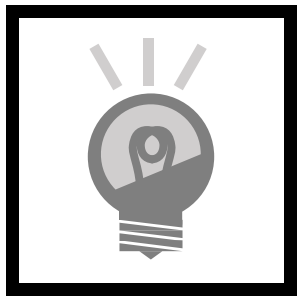
## 画像処理

・ライブラリ化されているものも含めて今の主流
・カスタマイズ性が高いのでさらに多様な手法が提案されていくと考えられる
・ 特に SVGF ベースの手法が多く発展しているが Recurrent Blur にも注目

## 機械学習

・オフライン向けはたくさん提案されている
・リアルタイム向けも確実に実用化に向かっていると思われる

**Adversarial Monte Carlo Denoising with Conditioned Auxiliary Feature Modulation [Xu et al. 2019]**



## 最新手法

・Spatiotemporal にパストレーシングのサンプルを重みづけてノイズを低減
・必ずしも画像処理的手法に頼る必要が無いことが示されている

**Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting [Bitterli et al. 2020]**

©Silicon Studio Corp., all rights reserved.

Silicon Studio
Ideas × Art × Technology

# 6.2. セッションまとめ

リアルタイムレイトレーシングにおけるデノイザーの実装について紹介
・Spatial Accumulation
・Temporal Accumulation
・デノイザーの設計
・種々のテクニックや考察点の紹介

今回は鏡面反射のみ
今後は半透明の扱いなども取り組んでいく

機械学習ベースの手法も今後のターゲット

©Silicon Studio Corp., all rights reserved.

Ideas × Art × Technology

技術力・表現力・発想力を兼ね備えたCGソリューションプロバイダー

©Silicon Studio Corp., all rights reserved.