コンピュータ・グラフィクス関連の最新論文紹介

～Modular Radiance Transfer法の解説～

田村　尚希
安田　廉

シリコンスタジオ株式会社

発表の流れ

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

1. 発表の概要
2. 技術紹介 + デモ
3. まとめ

発表の流れ

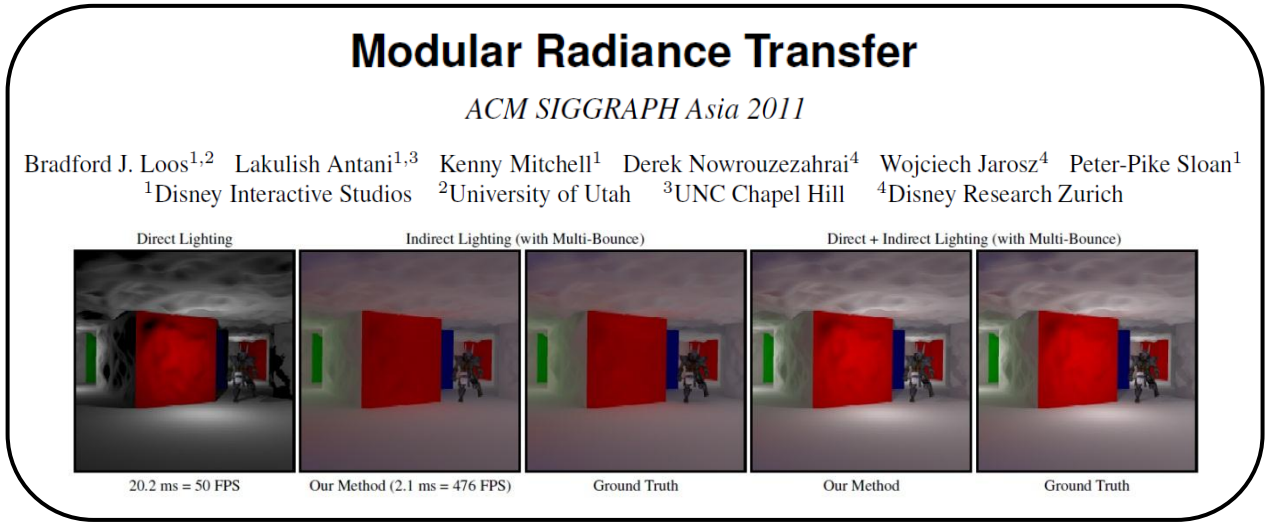
CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

1. 発表の概要
2. 技術紹介 + デモ
3. まとめ

発表の概要

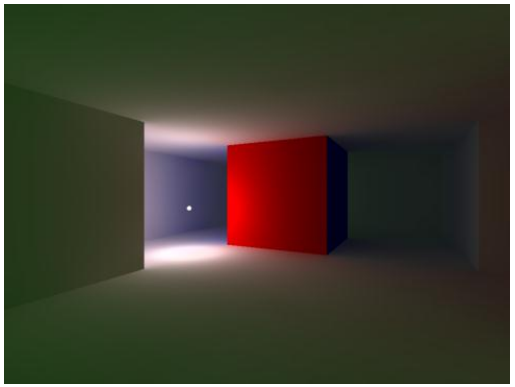
- 本セッションの目的
  - ゲーム業界にとって有望な技術の理解及び知識の共有
- 本セッションの内容
  - [**Modular Radiance Transfer法**]を利用した
    リアルタイムGI(Global Illumination)技術の解説



解説技術の特徴

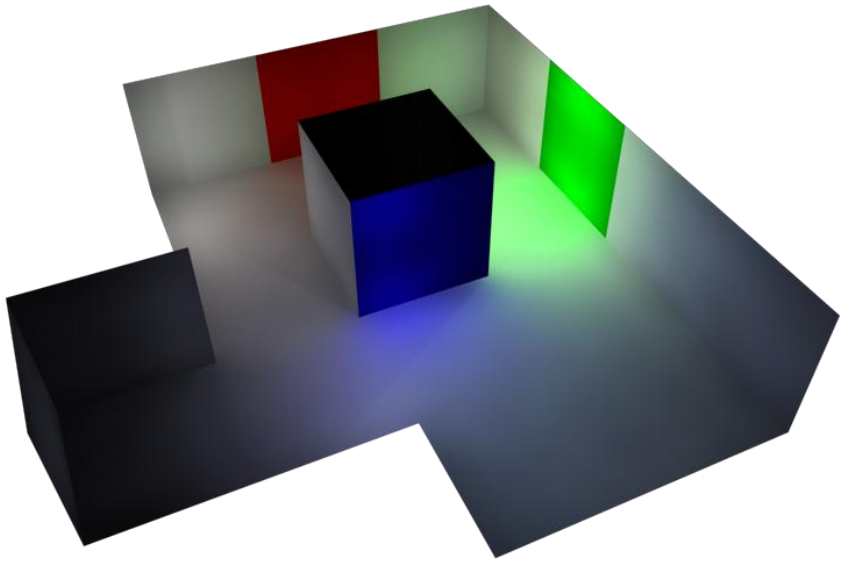
- 長所
  - とにかく高速
  - 多重反射を扱える
- 短所
  - 使い所を選ぶ
  - 多少込み入った前計算が必要

使いこなせれば
ワンランク上の表現が可能



注意点

- 元論文では、Modular (モジュール式)という概念が前面に押し出されており、立方体を組み合わせたシーン以外は適用しにくいような印象を受けます
- 本セッションでは、Modularの部分は抑えて説明しますので論文の流れと多少異なります



発表の流れ

1. 発表の概要
2. 技術紹介 + デモ
   - 間接光計算手法の解説（論文4節）
   - シーン分割による高速化の解説（論文5節）
3. まとめ

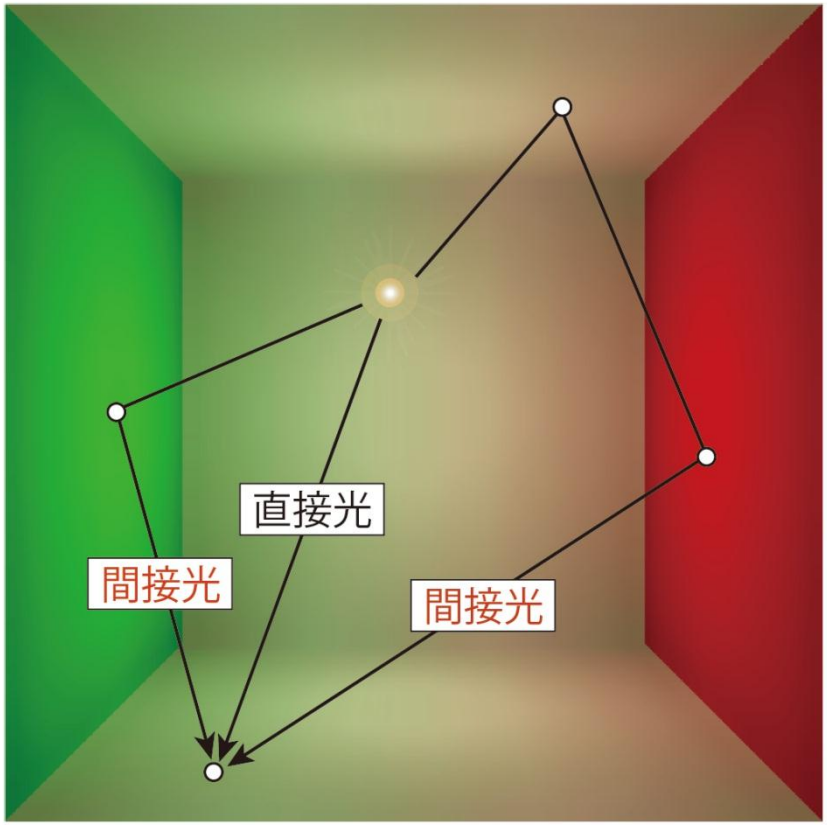
発表の流れ

1. 発表の概要
2. 技術紹介 + デモ
   - 間接光計算手法の解説（論文4節）
   - シーン分割による高速化の解説（論文5節）
3. まとめ

間接光の概要

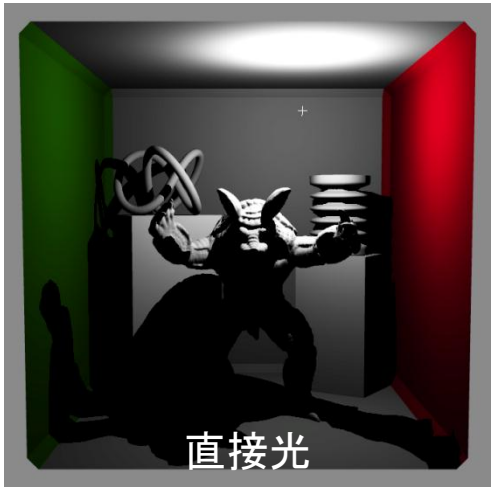
- そもそも間接光って何?

  ⇒ 光源から一度以上物体に反射して届く光



間接光の概要

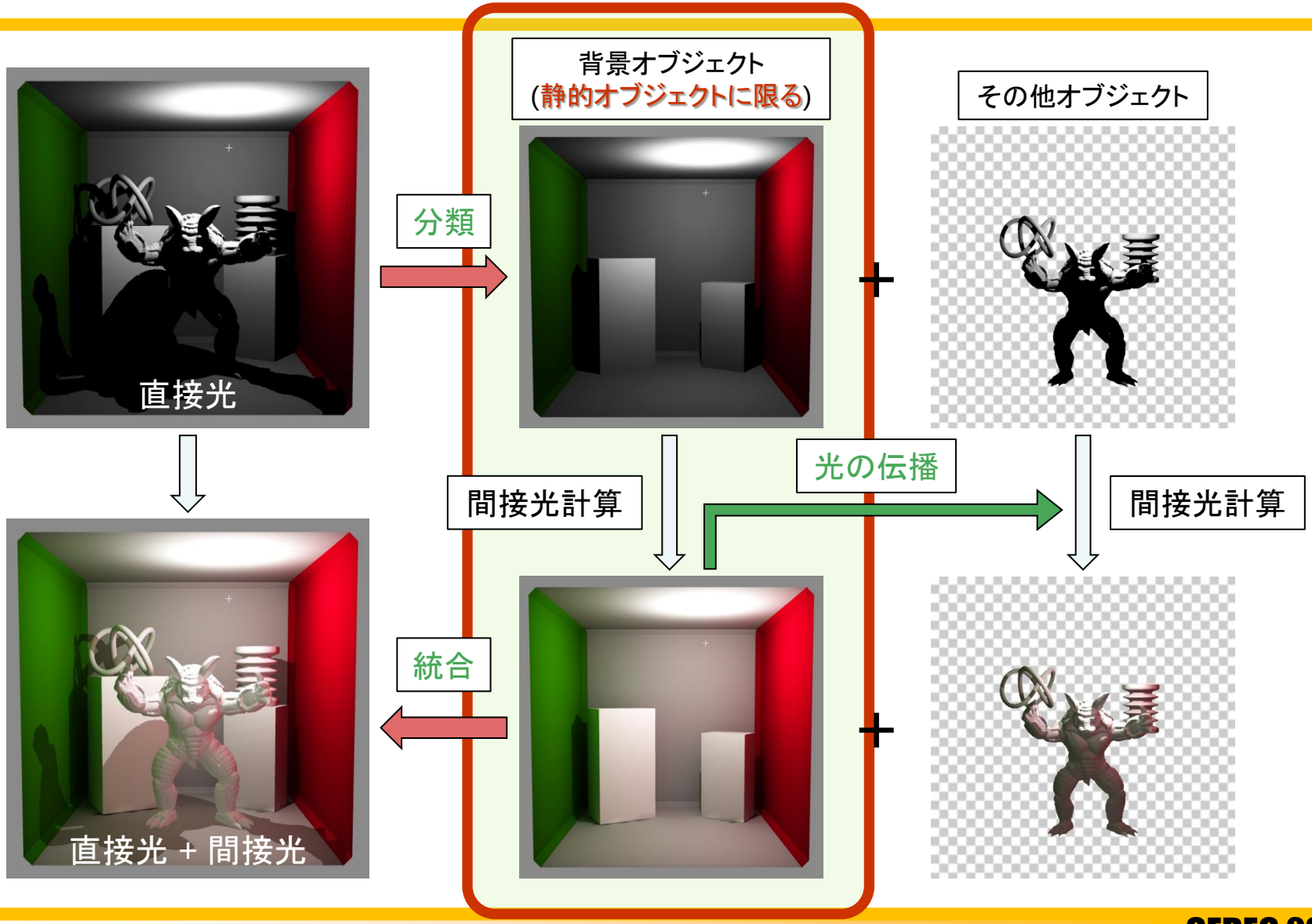
- 間接光があると何が良いのか
  ⇒ 柔らかなライティングを再現でき、シーン全体の説得力が増す



その分非常に計算負荷が高くリアルタイム処理は困難

解説手法は様々な高速化や制約を設けてリアルタイム処理を実現

解説手法の計算概要


背景オブジェクト
(静的オブジェクトに限る)
その他オブジェクト
直接光
分類
+
光の伝播
間接光計算
間接光計算
統合
+
直接光 + 間接光

間接光計算の解説手順

Step1. 背景オブジェクトの間接光計算

Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

間接光計算の解説手順

Step1. 背景オブジェクトの間接光計算

Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

間接光計算の解説手順

Step1. 背景オブジェクトの間接光計算

- 導入
- 手法の詳細解説
- ショートデモ

Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

間接光計算の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 背景オブジェクトの間接光計算

- 導入
- 手法の詳細解説
- ショートデモ

Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

間接光計算の解説手順

CEDEC 2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 背景オブジェクトの間接光計算

- 導入
  1. 間接光計算の考え方の説明
  2. 問題提起
  3. キーアイディアの説明
- 手法の詳細解説
- ショートデモ

Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

間接光計算の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 背景オブジェクトの間接光計算

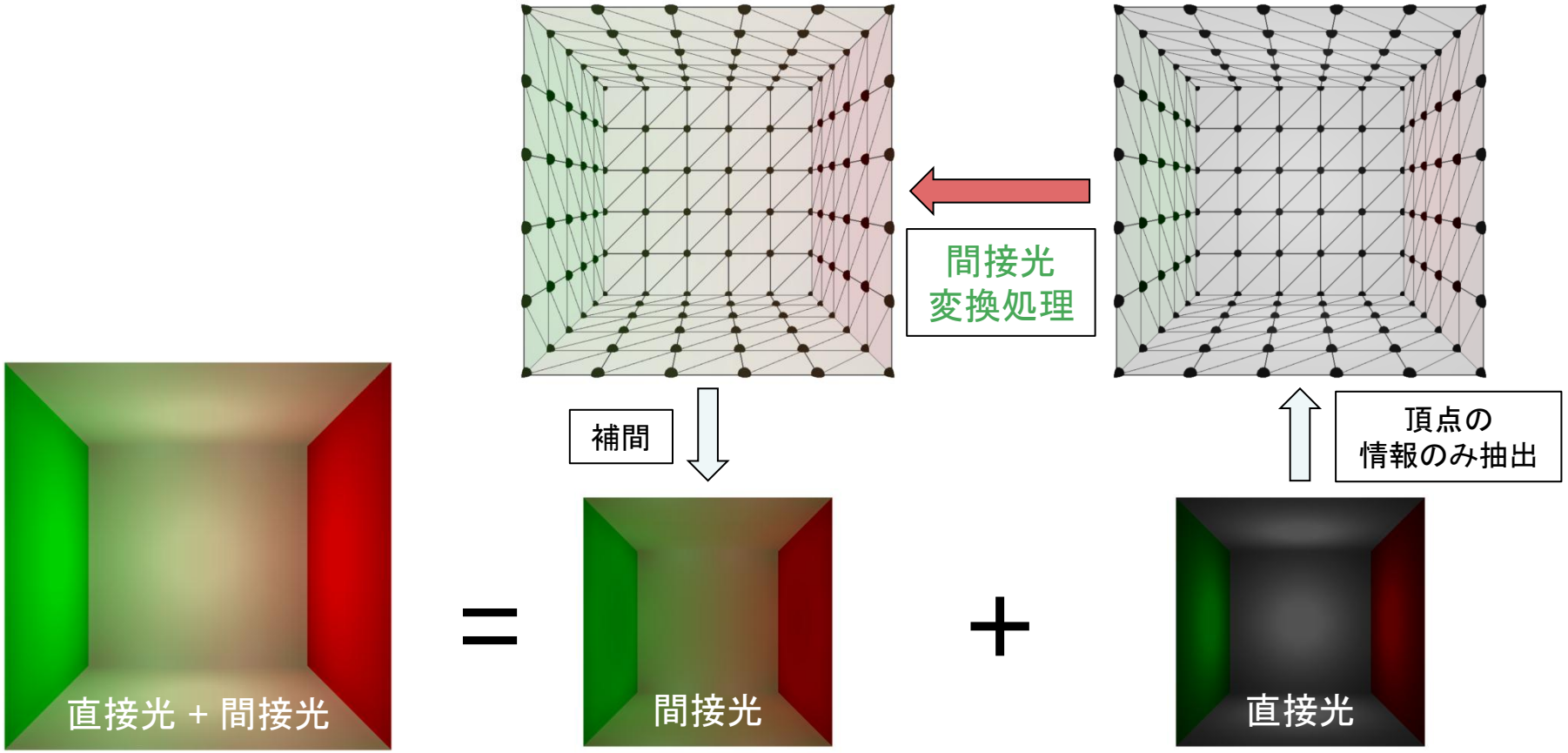
- 導入
  1. 間接光計算の考え方の説明
  2. 問題提起
  3. キーアイディアの説明
- 手法の詳細解説
- ショートデモ

Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

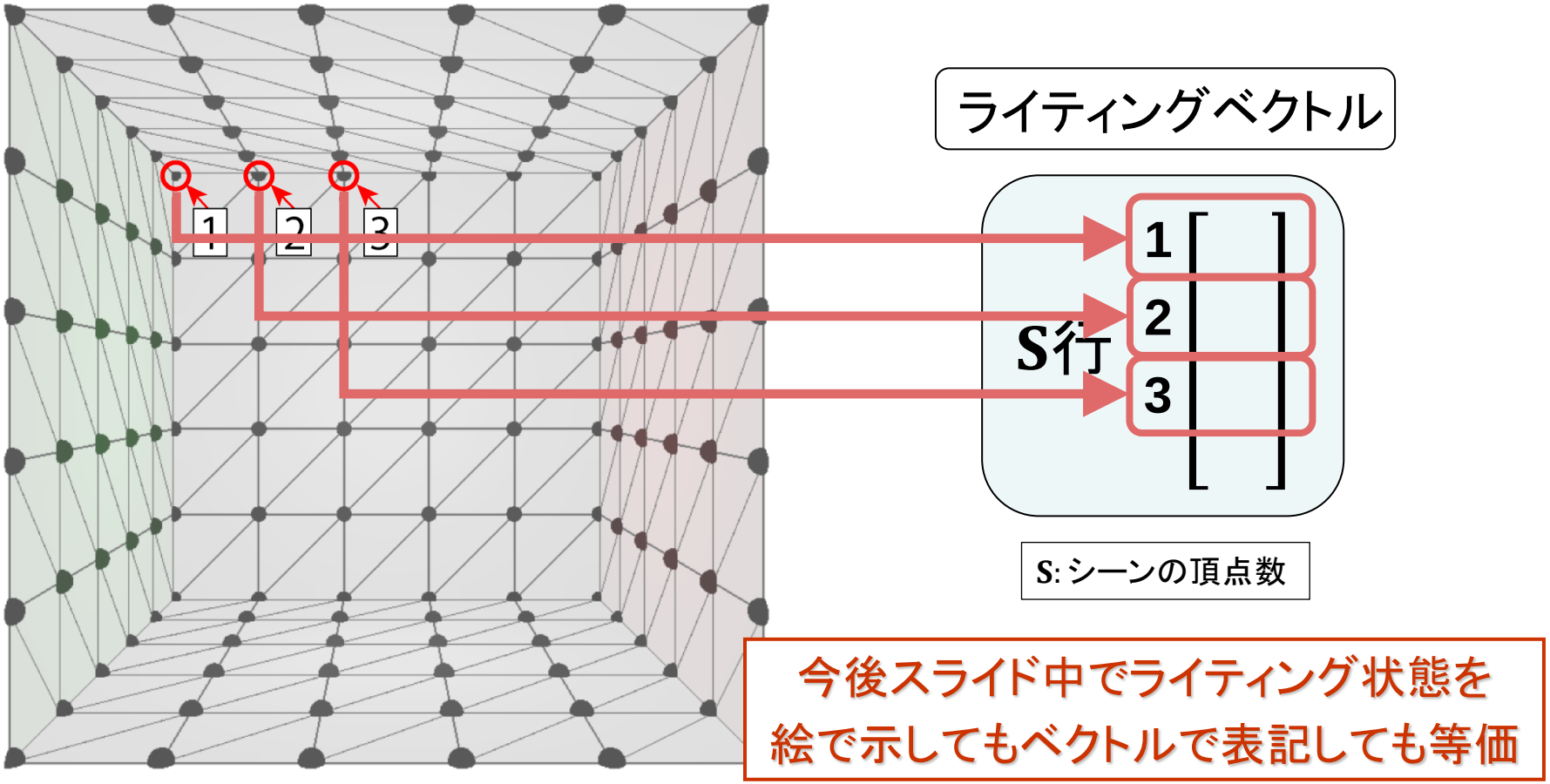
間接光計算の考え方

- 背景オブジェクトの間接光はメッシュ頂点で計算



間接光計算の考え方

- メッシュ頂点のライティング状態をベクトルで表記



間接光計算の考え方

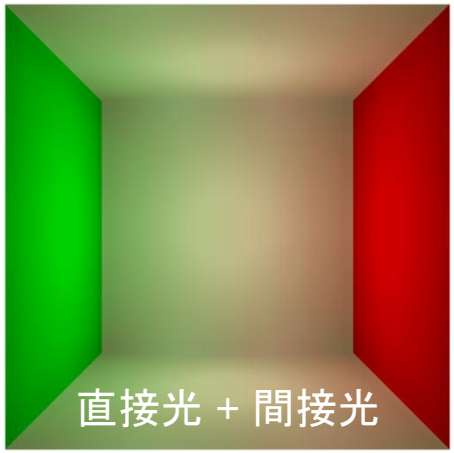
- 光の変換行列Fを用いた間接光計算

S: シーンの頂点数
l_d: サンプル点の直接光成分
l_{ind}: サンプル点の間接光成分

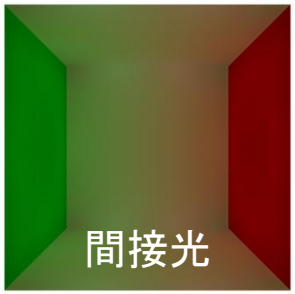
$$\begin{bmatrix} l_{ind\,1} \\ l_{ind\,2} \\ l_{ind\,3} \\ \vdots \\ l_{ind\,S} \end{bmatrix} = \begin{bmatrix} \\ \mathbf{F} \\ \\ \end{bmatrix} \times \begin{bmatrix} l_{d\,1} \\ l_{d\,2} \\ l_{d\,3} \\ \vdots \\ l_{d\,S} \end{bmatrix}$$

補間

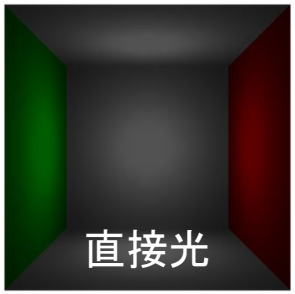
頂点の
情報のみ抽出



=

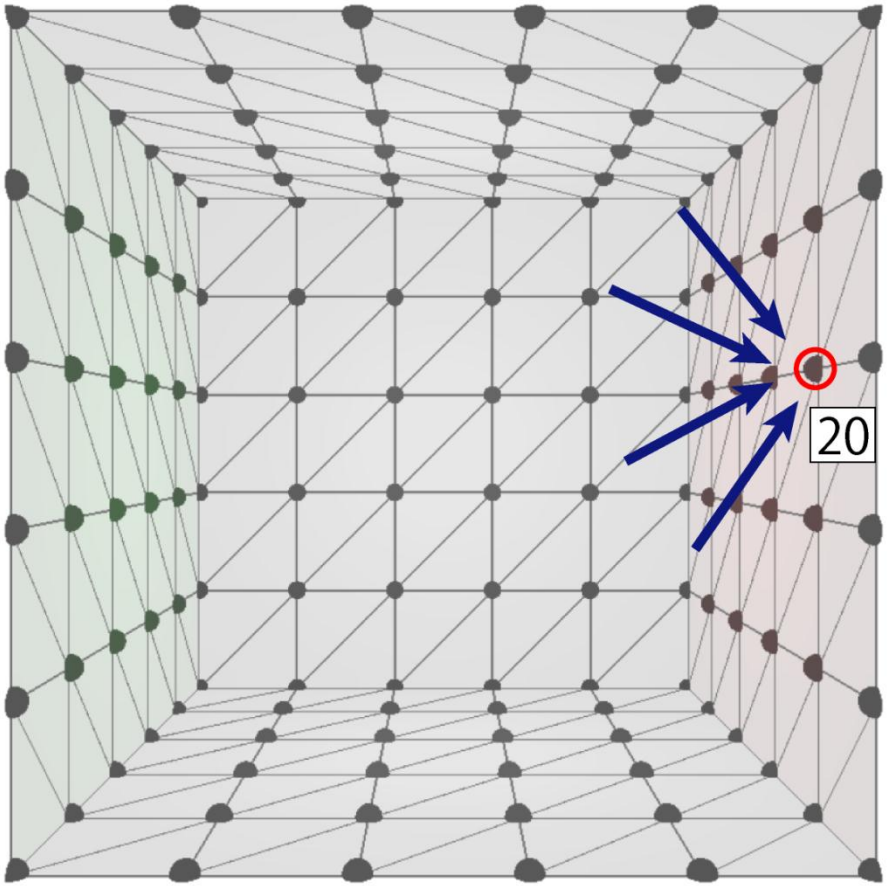


+



間接光計算の考え方

- 光の変換行列Fの計算



光の変換行列F

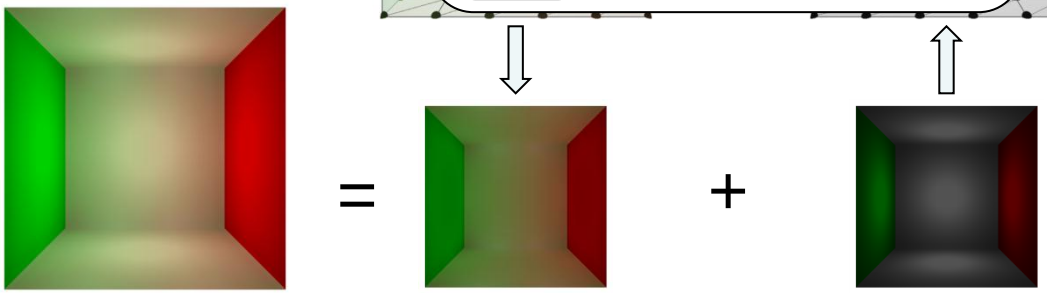
S: シーンの頂点数

行列Fは静的オブジェクトなら前計算可能

間接光計算の考え方

- 要点
  1. 背景オブジェクトの間接光はメッシュ頂点で計算
  2. ライティング状態はベクトルで表記可能
  3. 直接光は光の変換行列Fを用いて間接光に変換
  4. 光の変換行列Fは静的オブジェクトなら前計算可能

$$\begin{bmatrix} l_{ind\,1} \\ l_{ind\,2} \\ l_{ind\,3} \\ \vdots \\ l_{ind\,S} \end{bmatrix} = \begin{bmatrix} \\ \mathbf{F} \\ \\ \end{bmatrix} \times \begin{bmatrix} l_{d\,1} \\ l_{d\,2} \\ l_{d\,3} \\ \vdots \\ l_{d\,S} \end{bmatrix}$$



間接光計算の解説手順

Step1. 背景オブジェクトの間接光計算

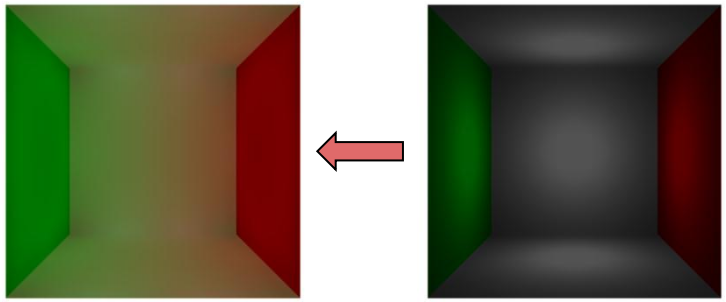
- 導入
  1. 間接光計算の考え方の説明
  2. 問題提起
  3. キーアイディアの説明
- 手法の詳細解説
- ショートデモ

Step2. その他オブジェクトへ光の伝播

Step3. デモ及び考察

問題提起

$$\begin{bmatrix} l_{ind\,1} \\ l_{ind\,2} \\ l_{ind\,3} \\ \vdots \\ l_{ind\,S} \end{bmatrix} = \begin{bmatrix} \\ \mathbf{F} \\ \\ \end{bmatrix} \times \begin{bmatrix} l_{d\,1} \\ l_{d\,2} \\ l_{d\,3} \\ \vdots \\ l_{d\,S} \end{bmatrix}$$



上記計算式で間接光は計算可能

ただし実行速度が不十分 + 頂点数の増加で処理量が爆発

間接光計算の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 背景オブジェクトの間接光計算

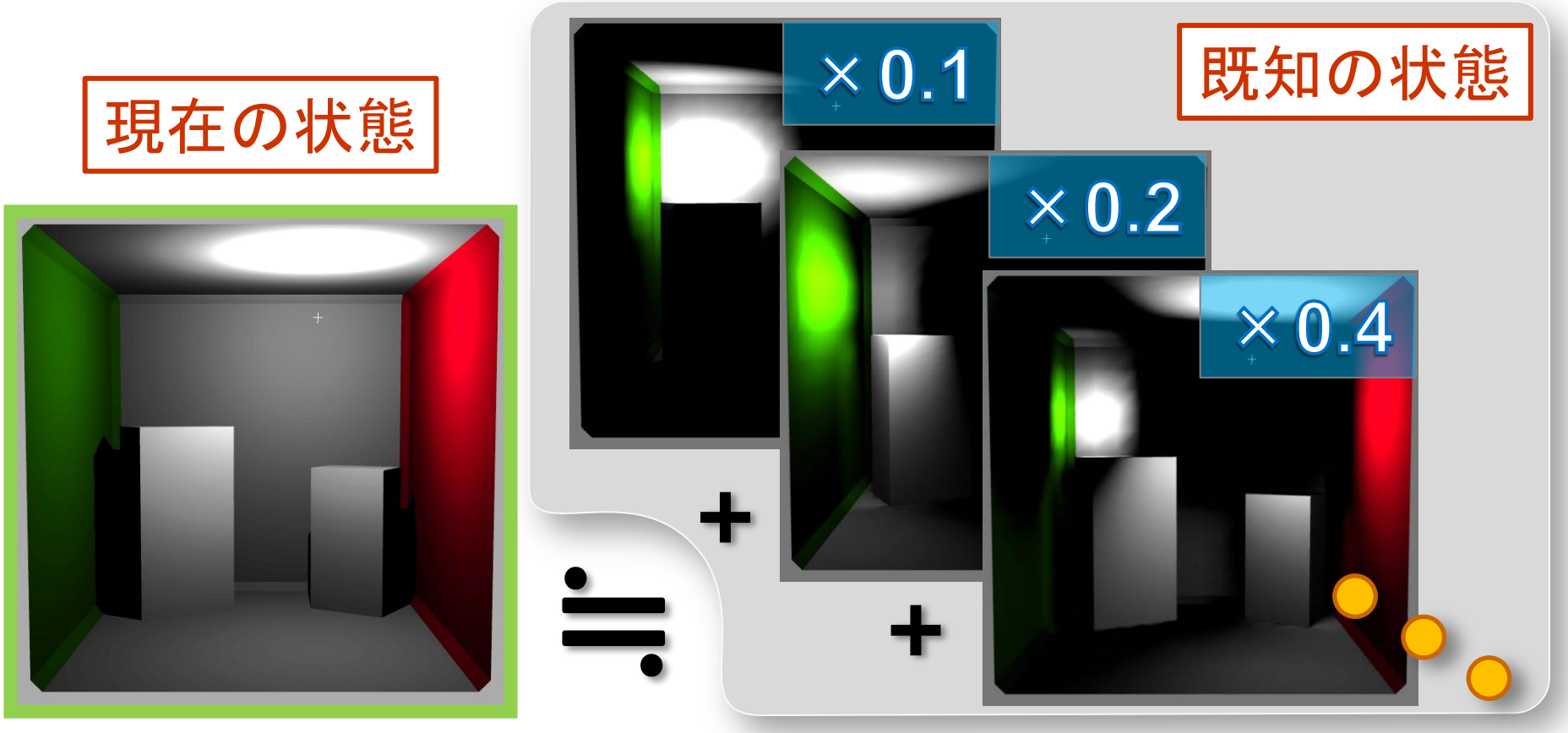
- 導入
  1. 間接光計算の考え方の説明
  2. 問題提起
  3. キーアイディアの説明
- 手法の詳細解説
- ショートデモ

Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

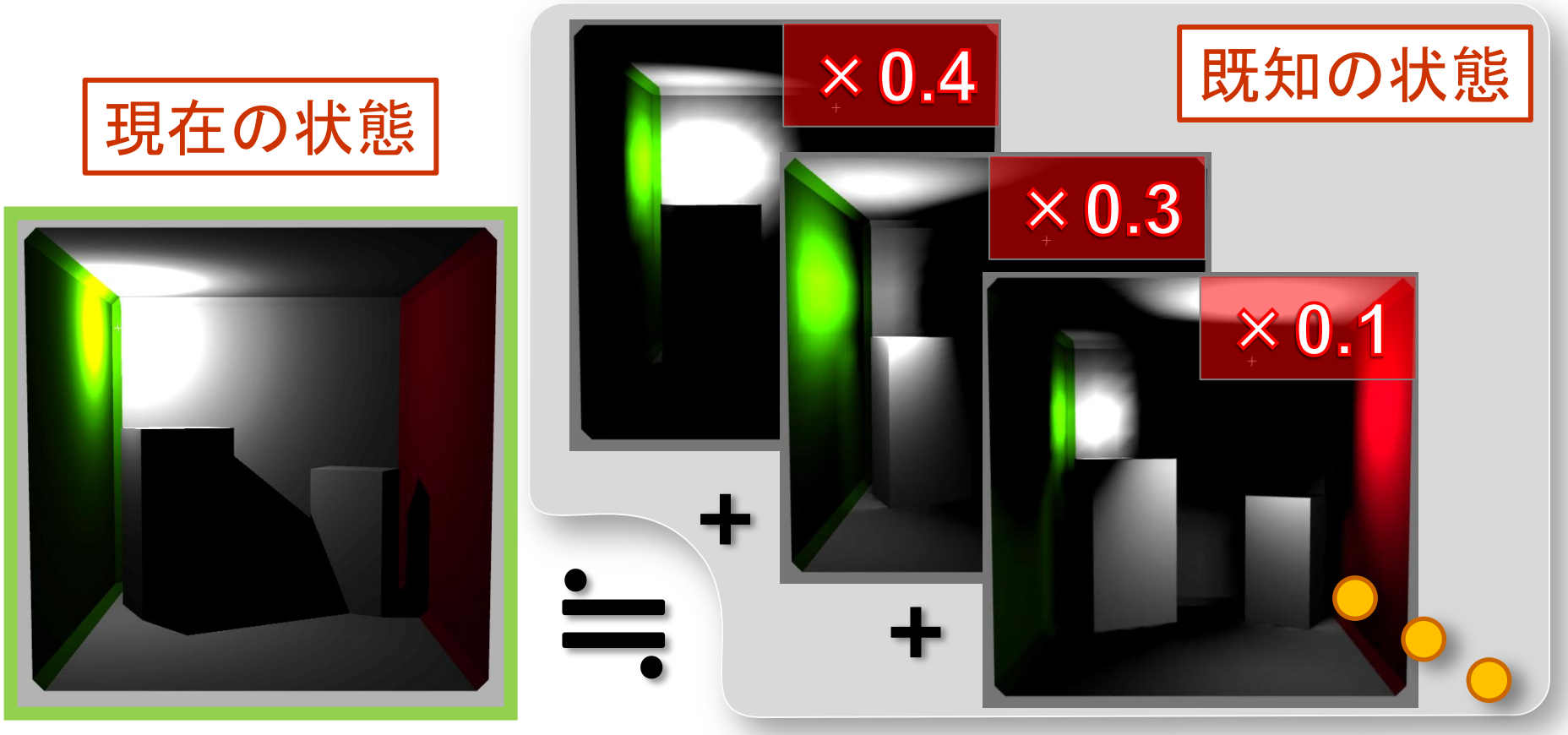
キーアイディア

- 全ての直接光は前計算した既知の状態の重ね合わせで近似できると仮定



キーアイディア

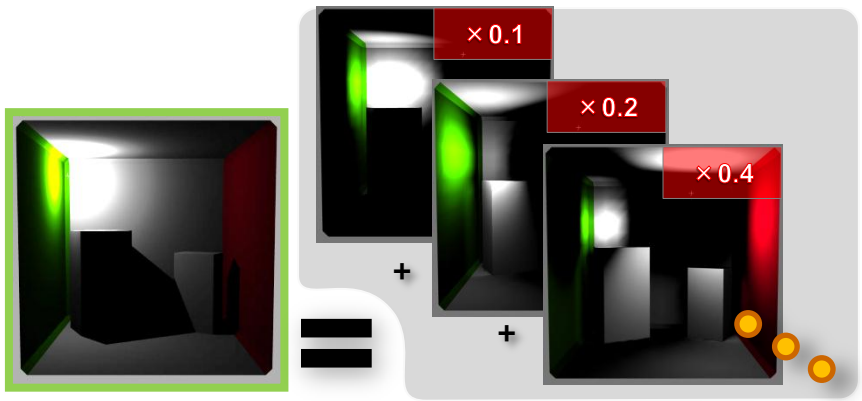
- 直接光が変化しても係数のみが変化
  - 既知の状態は変化しない



キーアイディア

- これはベクトルの足し合わせとして表現可能

$$\begin{bmatrix} l_{d\,1} \\ l_{d\,2} \\ \vdots \\ l_{d\,s} \end{bmatrix} \fallingdotseq w_1 \begin{bmatrix} B^1_{d\,1} \\ B^1_{d\,2} \\ \vdots \\ B^1_{d\,s} \end{bmatrix} + w_2 \begin{bmatrix} B^2_{d\,1} \\ B^2_{d\,2} \\ \vdots \\ B^2_{d\,s} \end{bmatrix} + w_3 \begin{bmatrix} B^3_{d\,1} \\ B^3_{d\,2} \\ \vdots \\ B^3_{d\,s} \end{bmatrix} \cdots$$

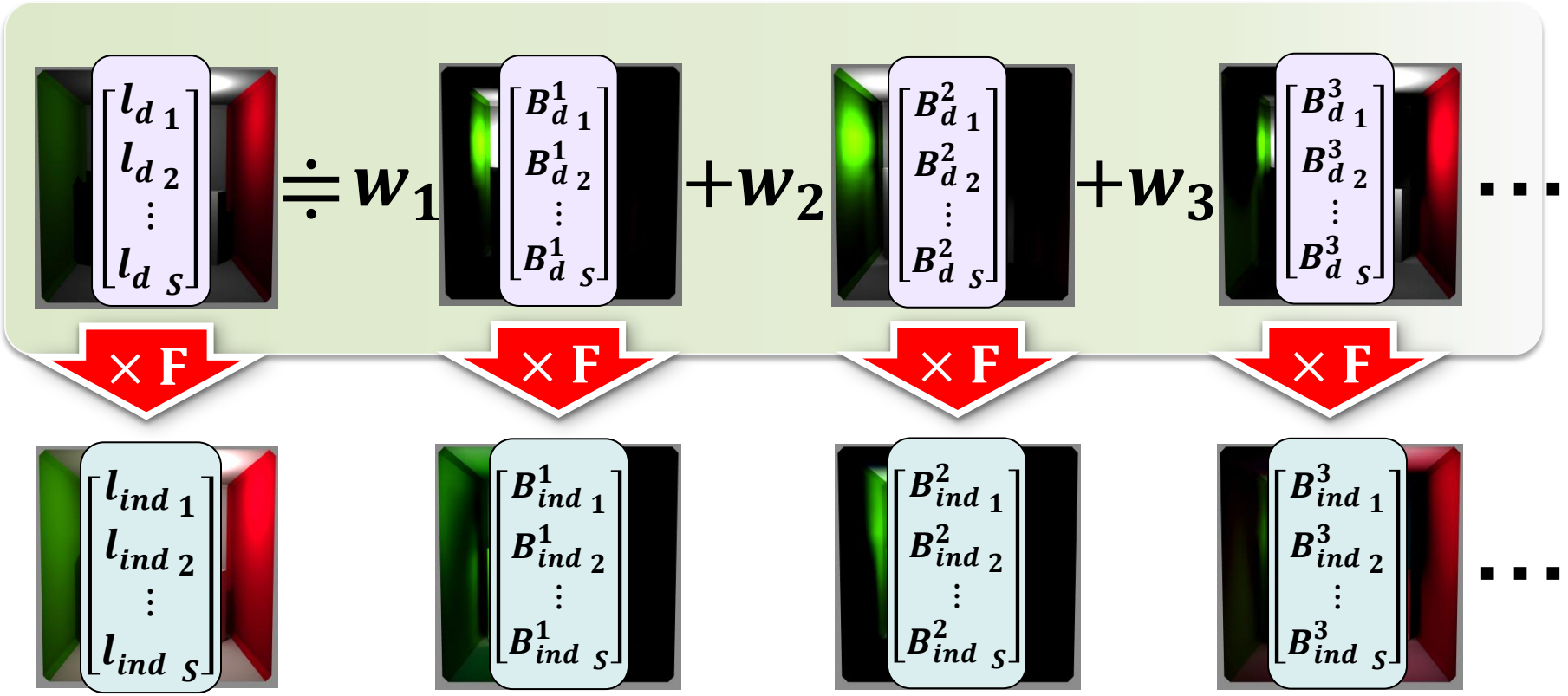


l_d: 現在の直接光の状態

$B^1_d \cdots B^n_d$: 既知の状態

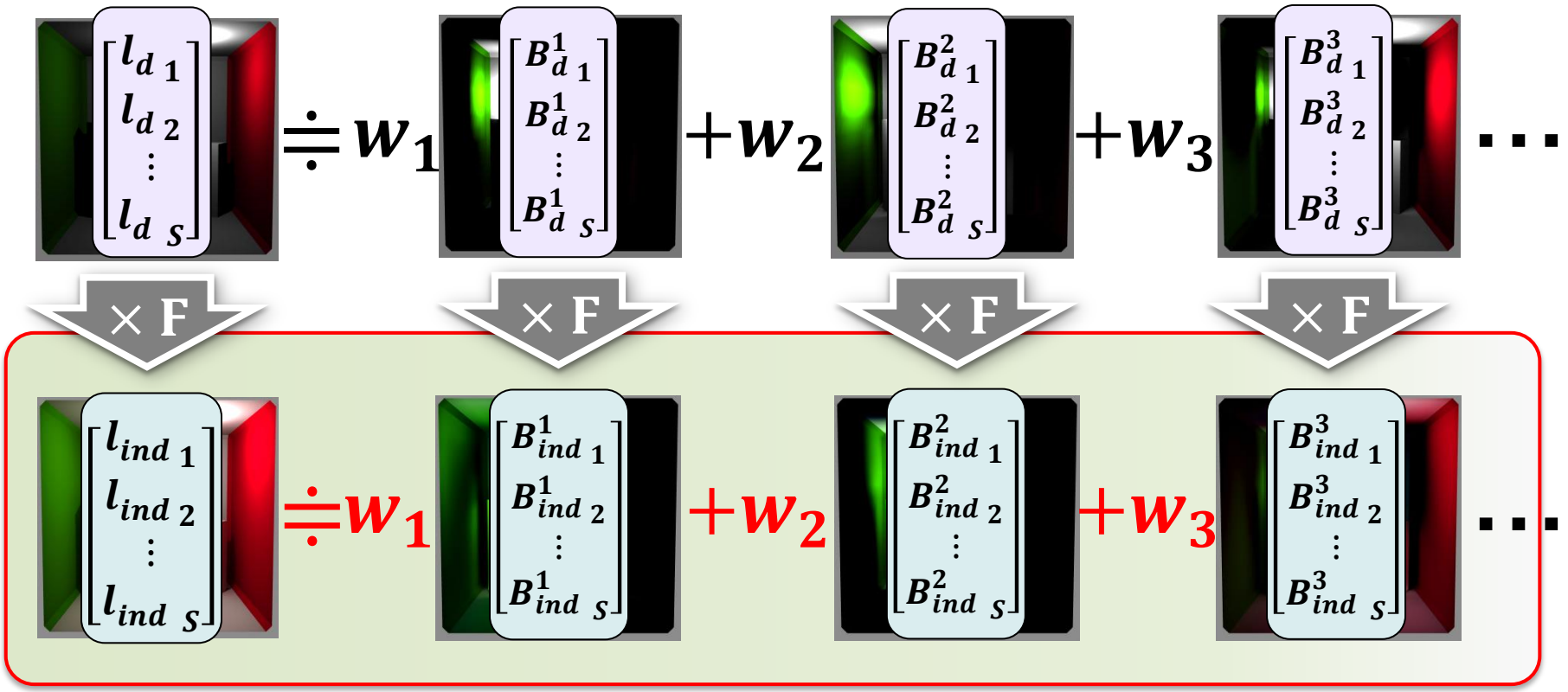
キーアイディア

- 各状態B_dには、対になる間接光の状態B_{ind}がそれぞれ存在



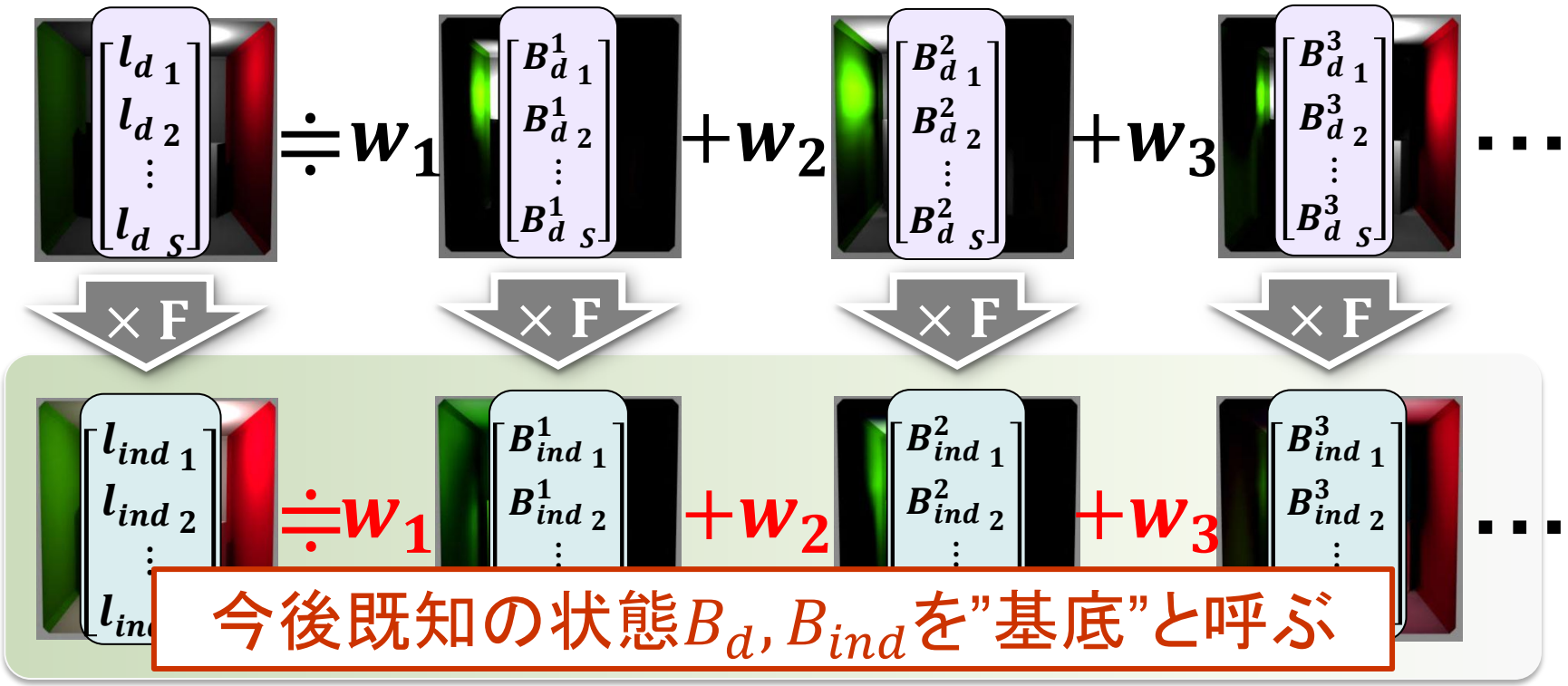
キーアイディア

- 直接光側と同じ重みで間接光側を足し合わす事でl_{ind}を近似可能



キーアイディア

- このロジックは下記条件を満たせば非常に高速
  1. 必要な係数が十分に少ない
  2. 係数が高速に計算可能



間接光計算の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 背景オブジェクトの間接光計算

- 導入
- 手法の詳細解説
  1. 最適な基底の計算方法
  2. 係数の計算方法
- ショートデモ

Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

間接光計算の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 背景オブジェクトの間接光計算

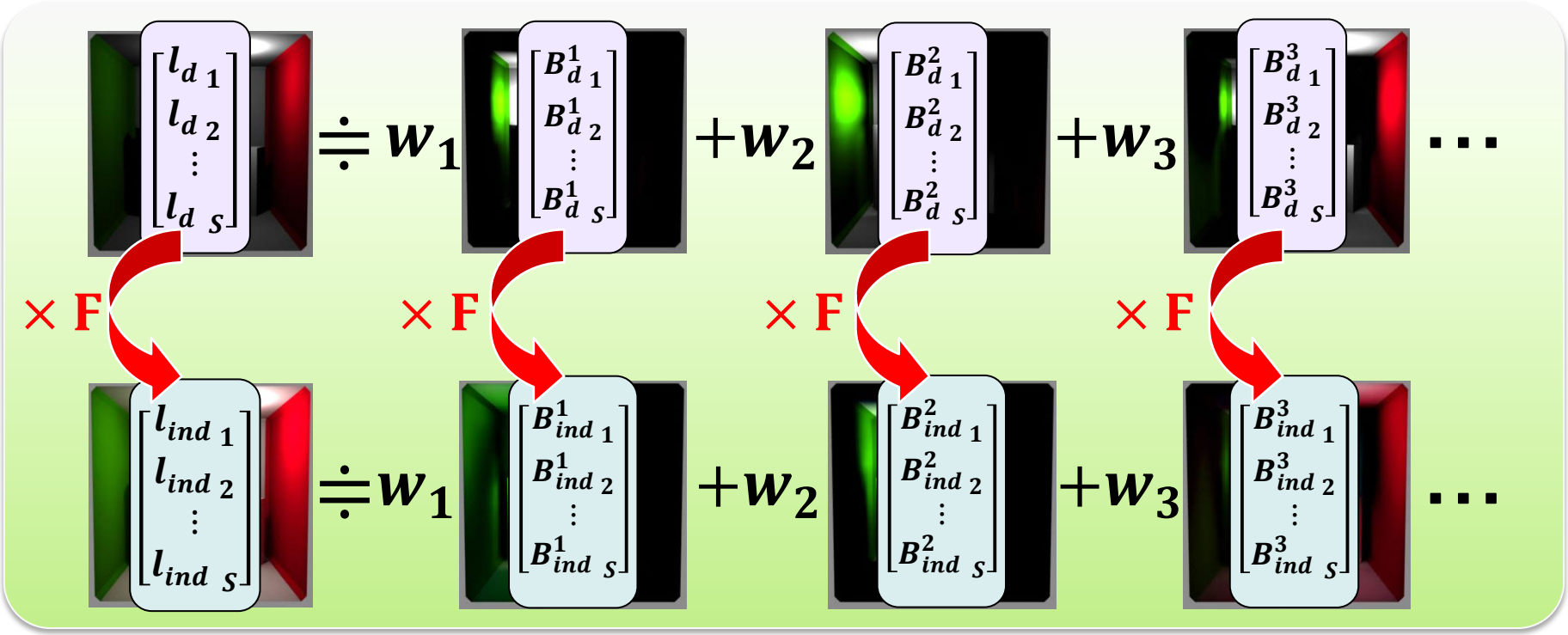
- 導入
- 手法の詳細解説
  1. 最適な基底の計算方法
  2. 係数の計算方法
- ショートデモ

Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

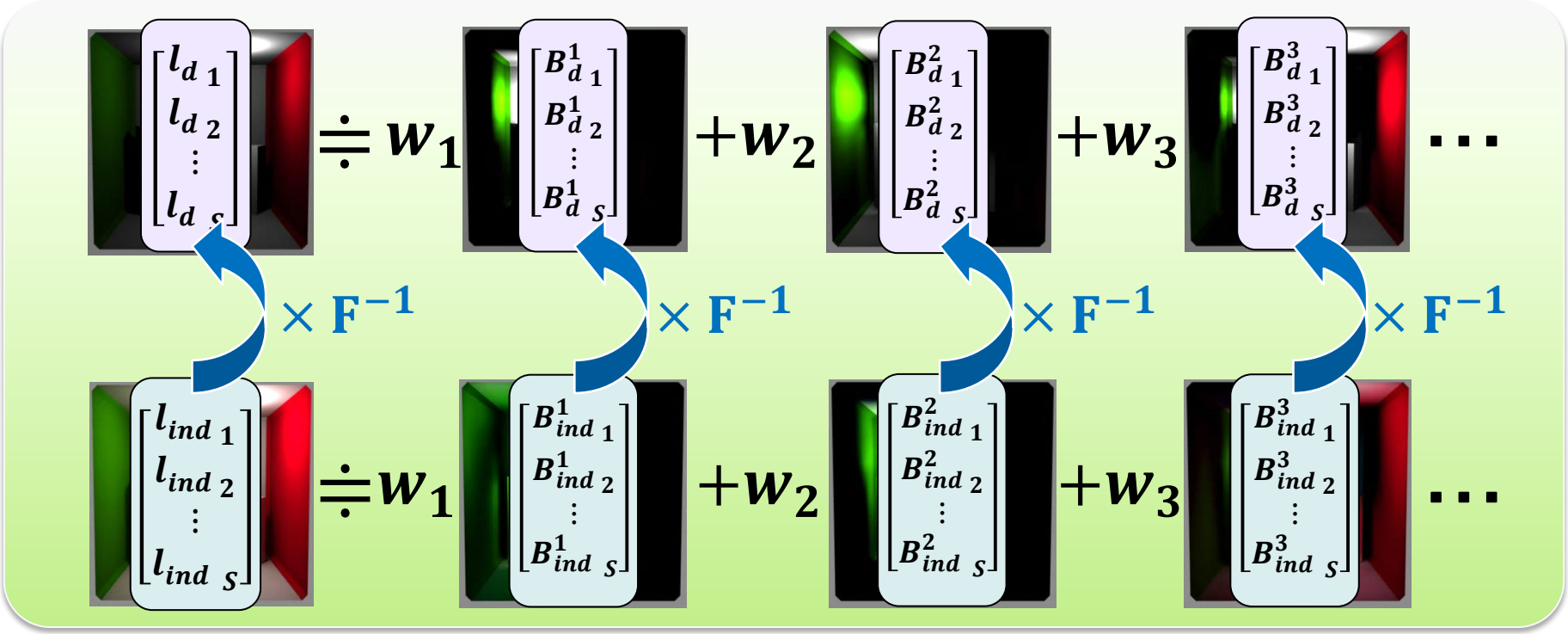
B_d, B_{ind}を求める

- 2方向のアプローチが考えられる
  - 直接光優先：B_dを先に求めて
    $B_{ind} = \mathrm{F}B_d$からB_{ind}を算出



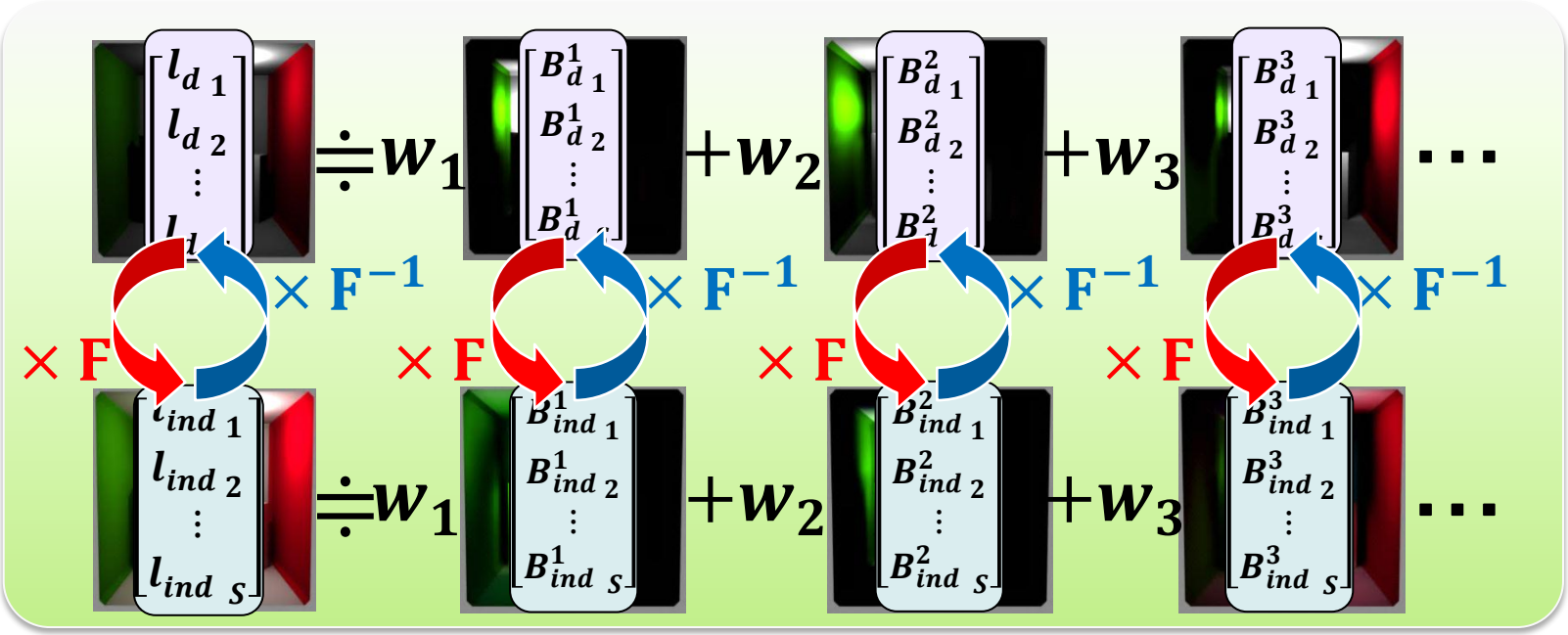
B_d, B_{ind}を求める

- 2方向のアプローチが考えられる
  - 間接光優先：B_{ind}を先に求めて
    $B_d = \mathrm{F}^{-1} B_{ind}$から$B_d$を算出



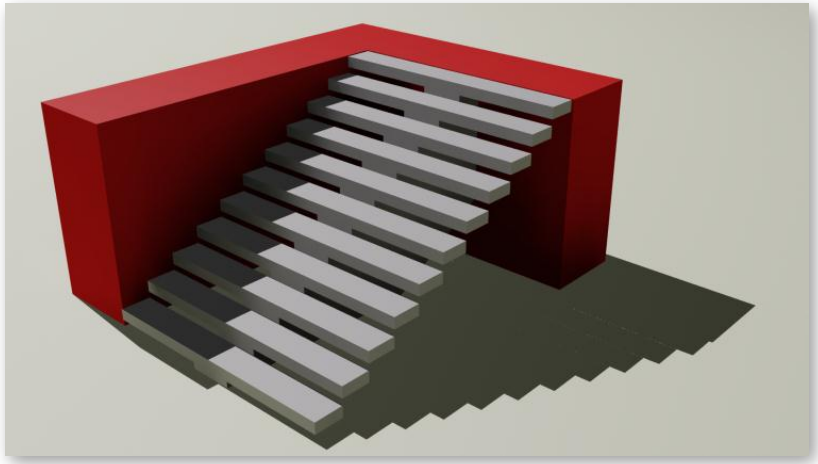
B_d, B_{ind}を求める

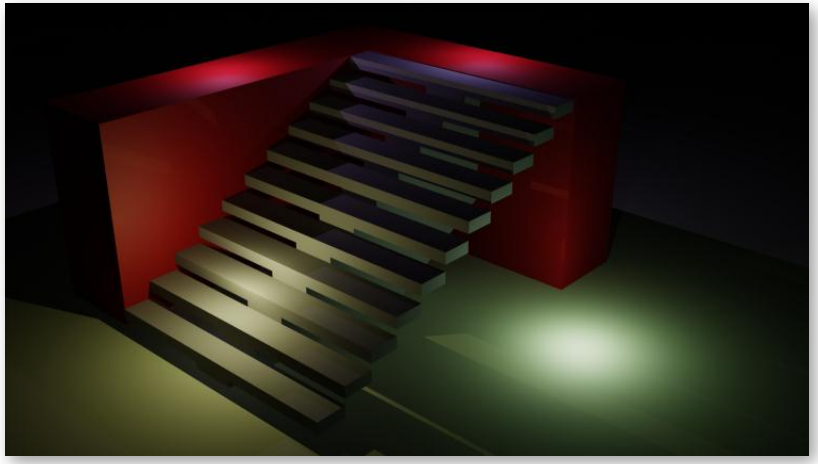
- 両方を並行して解説
  - 最終的には両方をハイブリッドしたアプローチで
    より良い基底を算出



最適な基底

- 最適な基底は同じシーンでも場合によって異なる
  - それぞれに特化した基底のほうが良い結果を得られる

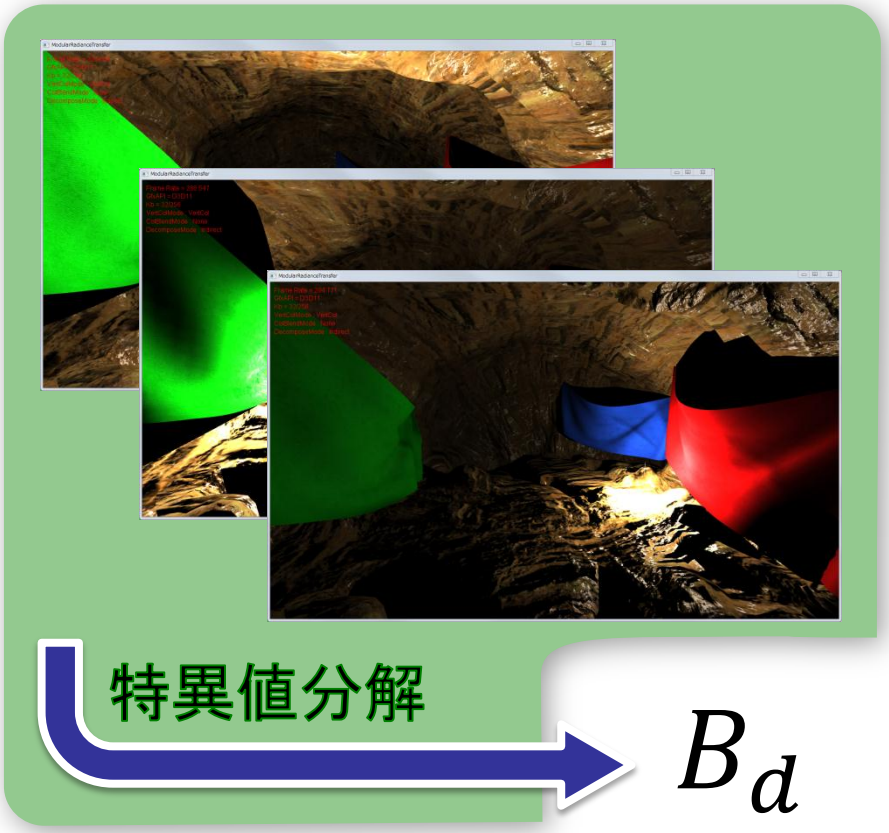


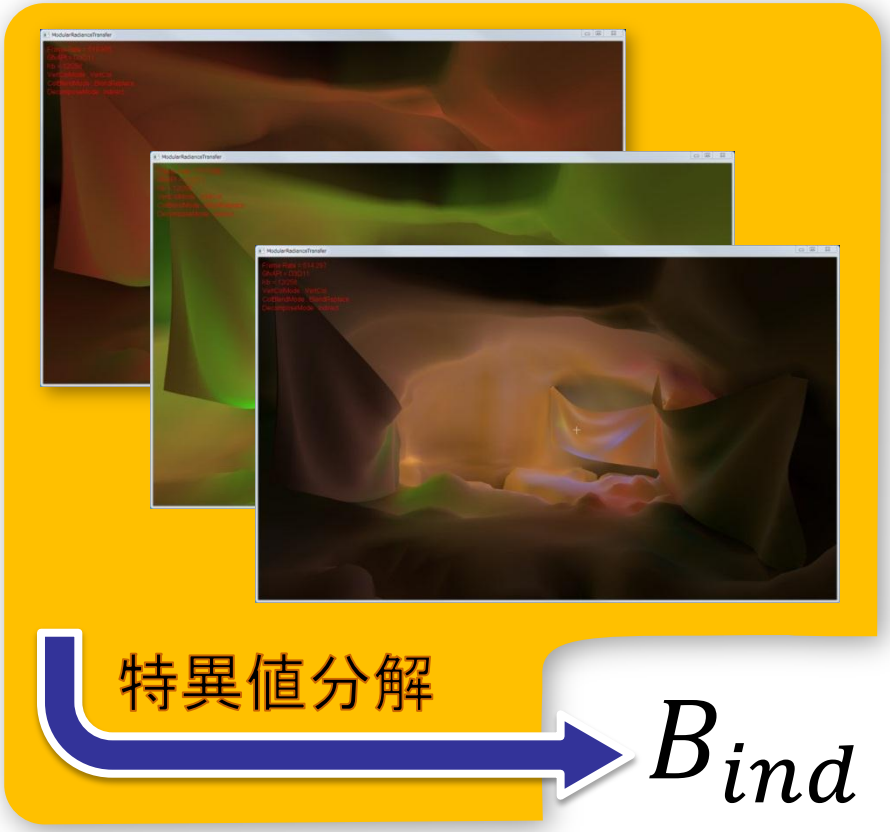


サンプリングベースの基底を採用

サンプリングベースの基底

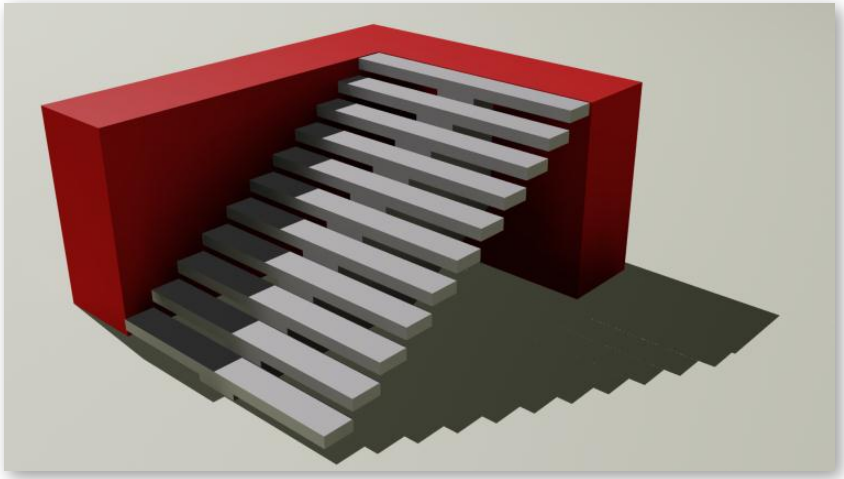
- 様々な状況でライティングした結果を集め、
  特異値分解を用いて基底を算出

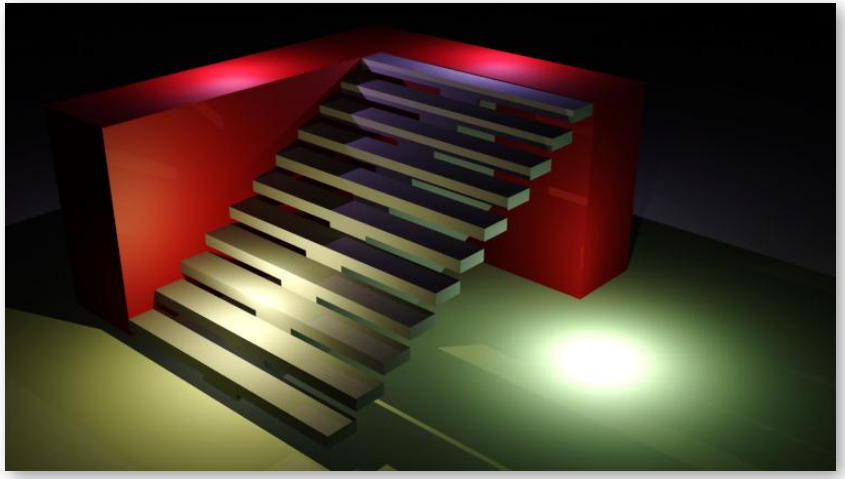




サンプルの選び方について

- 特異値分解で得られるのは
「全ての入力サンプルを良く近似する基底」

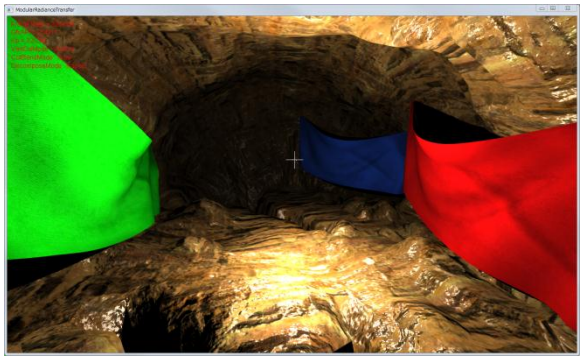




- サンプルで用いる光源のタイプ・方向・位置を
予め分かっている範囲で厳選するのが好ましい

サンプリング行列

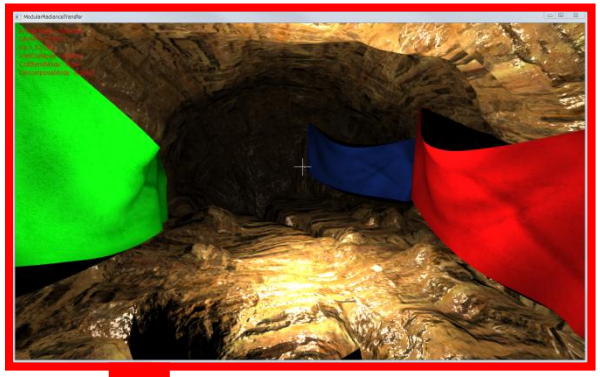
- 特異値分解を施すために、各サンプルの頂点の入射光を列に並べた行列を作成

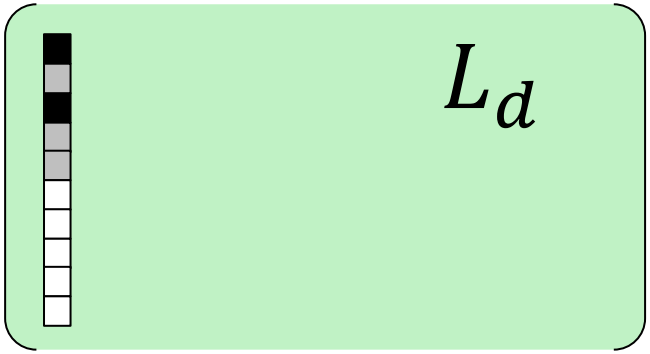


$$L_d$$

サンプリング行列

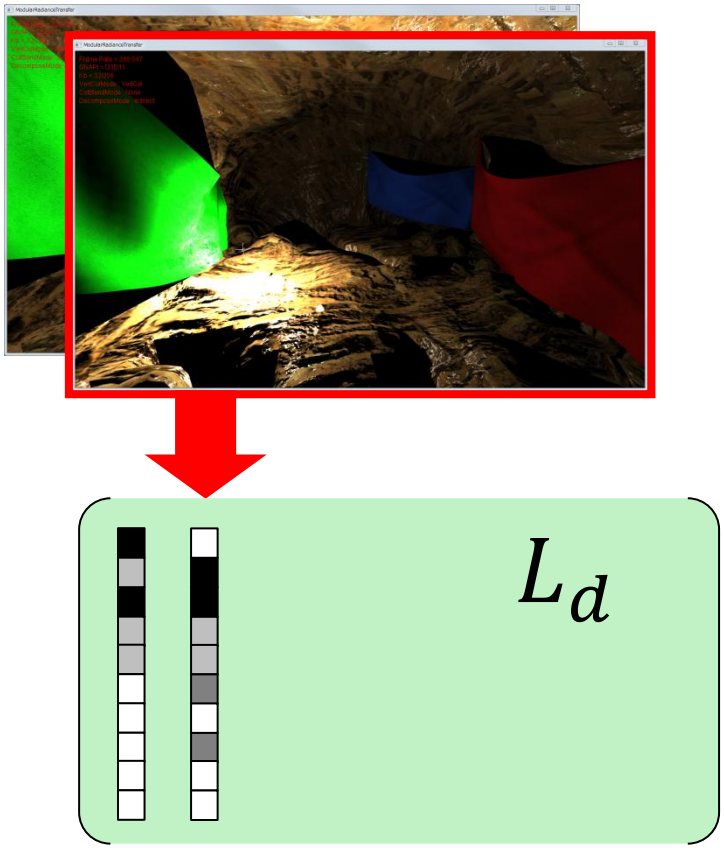
- 特異値分解を施すために、各サンプルの頂点の入射光を列に並べた行列を作成





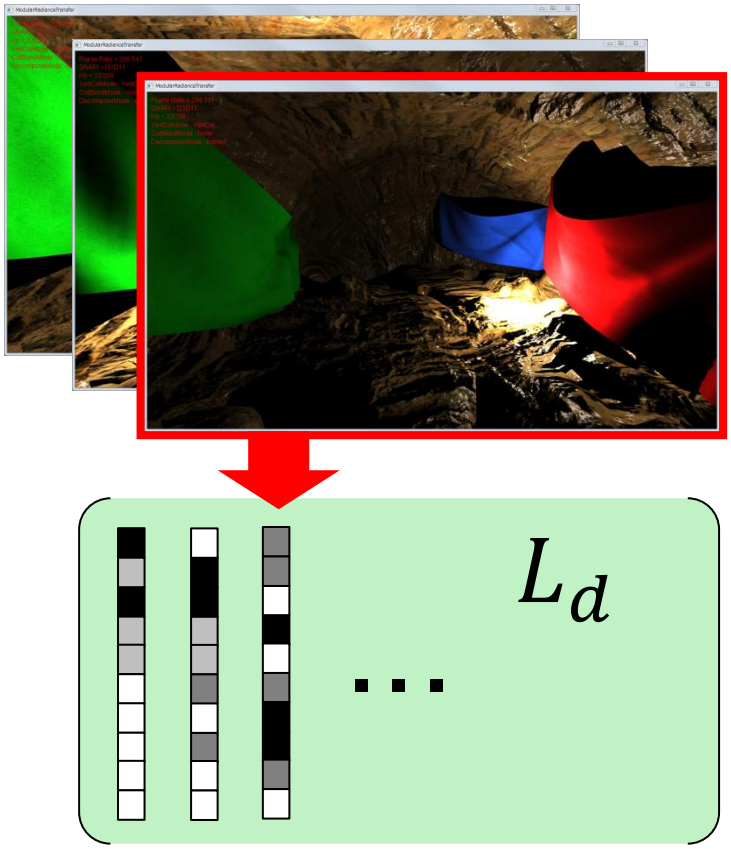
サンプリング行列

- 特異値分解を施すために、各サンプルの頂点の入射光を列に並べた行列を作成



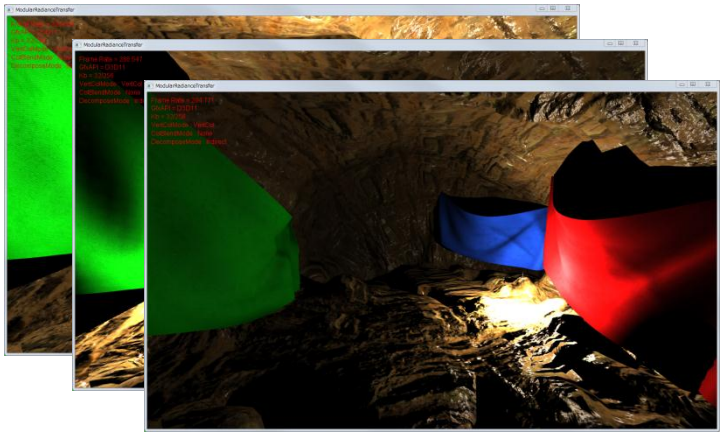
サンプリング行列

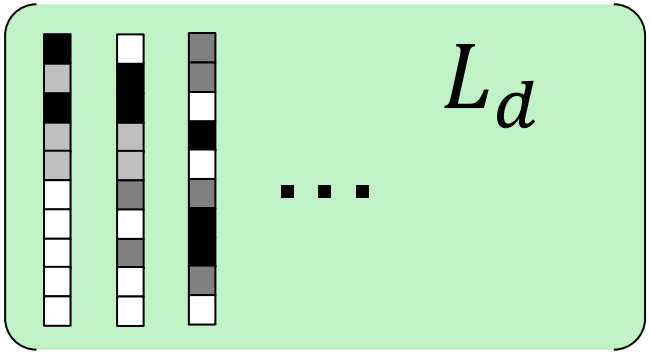
- 特異値分解を施すために、各サンプルの頂点の入射光を列に並べた行列を作成



サンプリング行列

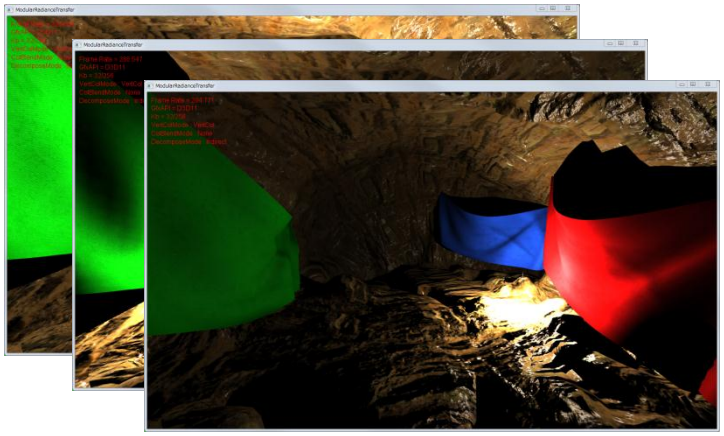
- 特異値分解を施すために、各サンプルの頂点の入射光を列に並べた行列を作成

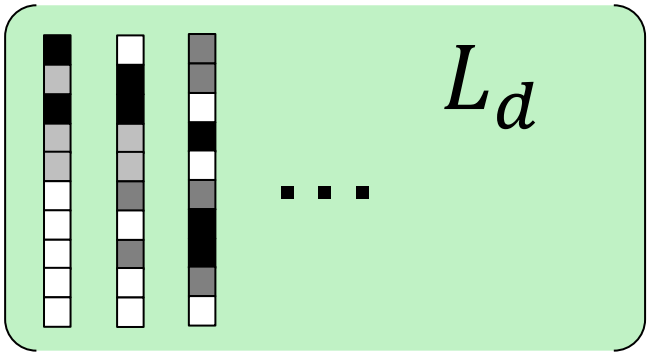




サンプリング行列

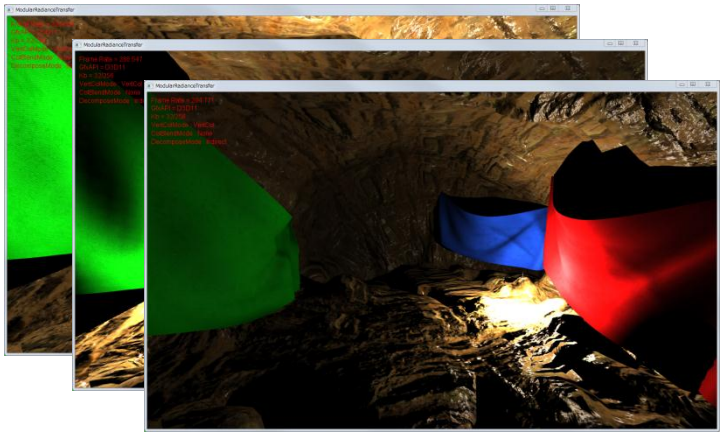
- 特異値分解を施すために、各サンプルの頂点の入射光を列に並べた行列を作成

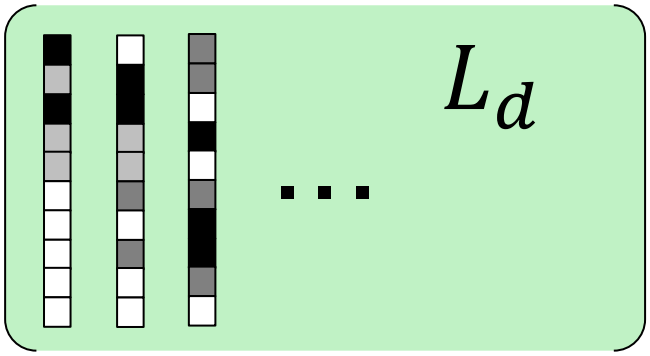


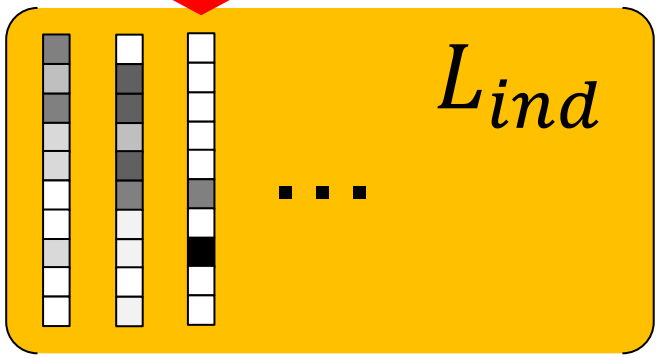


サンプリング行列

- 特異値分解を施すために、各サンプルの頂点の入射光を列に並べた行列を作成



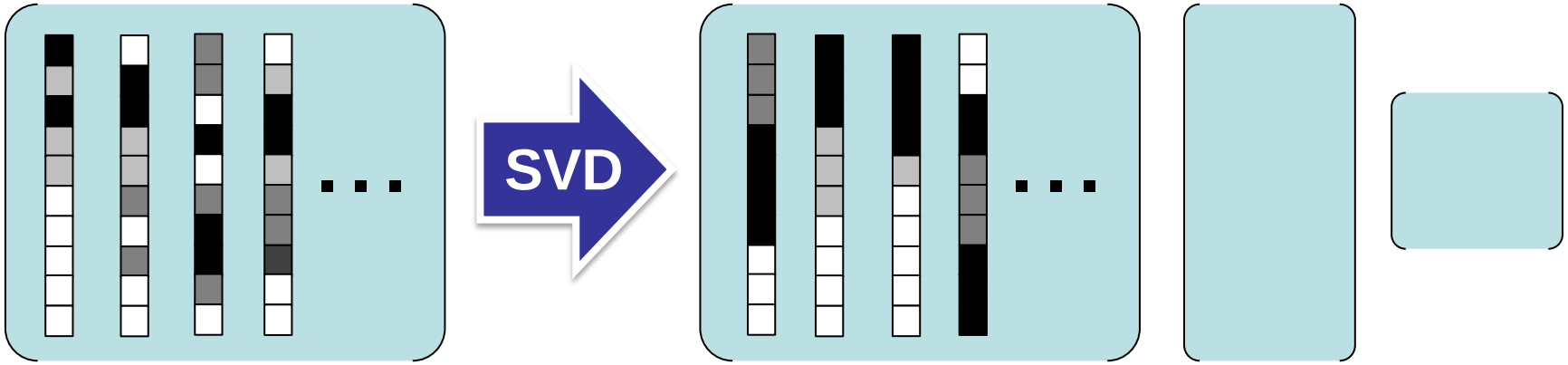




特異値分解(Singular Value Decomposition)

- 入力された行列MをU, Σ, V^Tという3行列に分解
  - 数多くの数学ライブラリがサポート

$$M = U * \Sigma * V^T$$

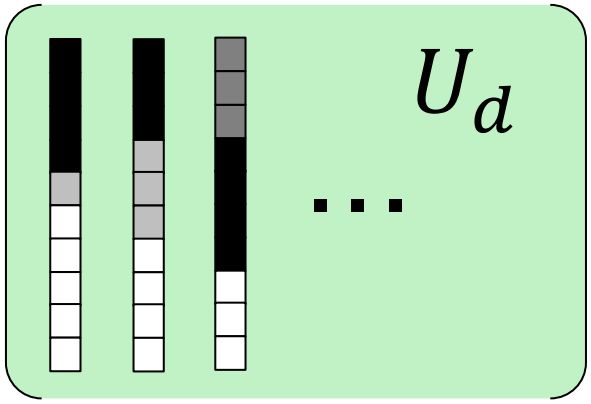


Mが異なれば当然U, Σ, V^Tも異なる

特異値分解(Singular Value Decomposition)

- L_d, L_{ind}に特異値分解を適用すると、行列UがB_d, B_{ind}となる

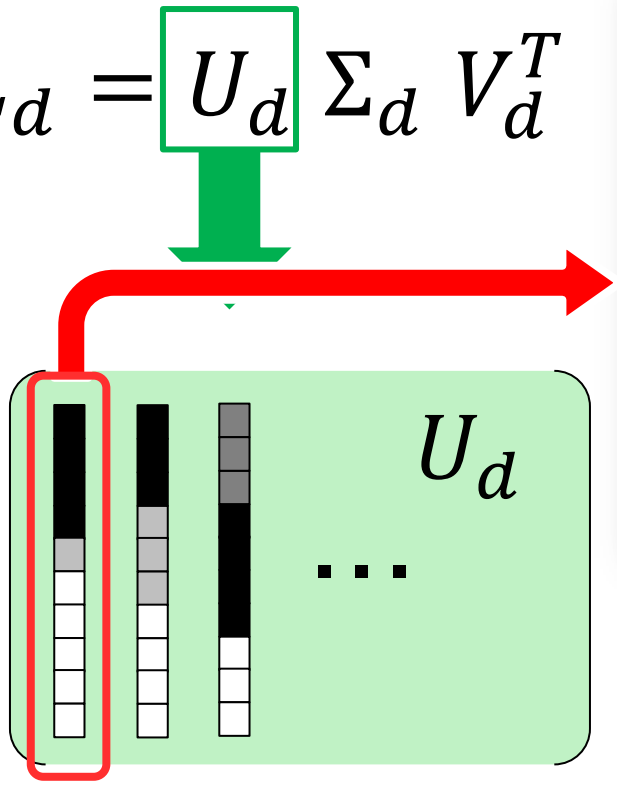
$$L_d = U_d \, \Sigma_d \, V_d^T$$

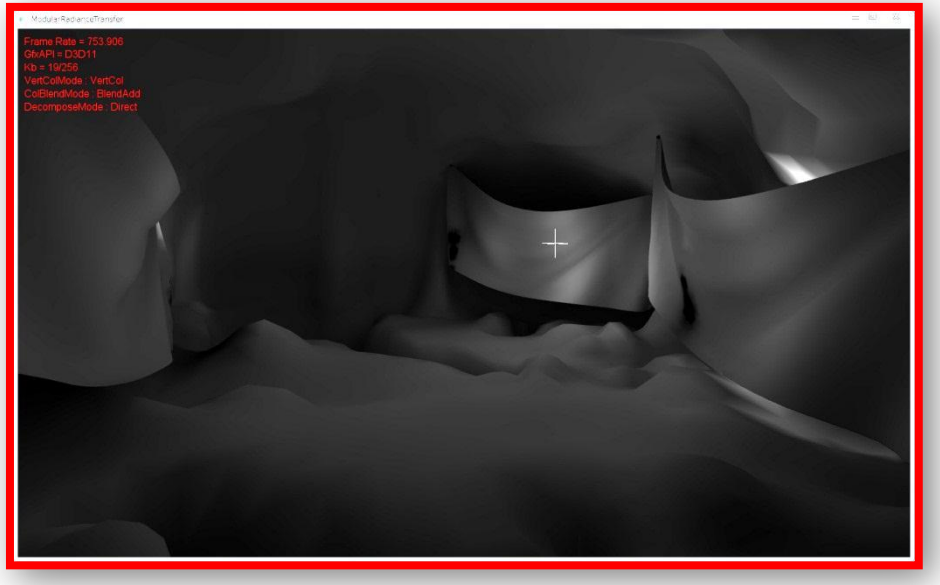


特異値分解(Singular Value Decomposition)

- L_d, L_{ind}に特異値分解を適用すると、行列UがB_d, B_{ind}となる

$$L_d = U_d \, \Sigma_d \, V_d^T$$

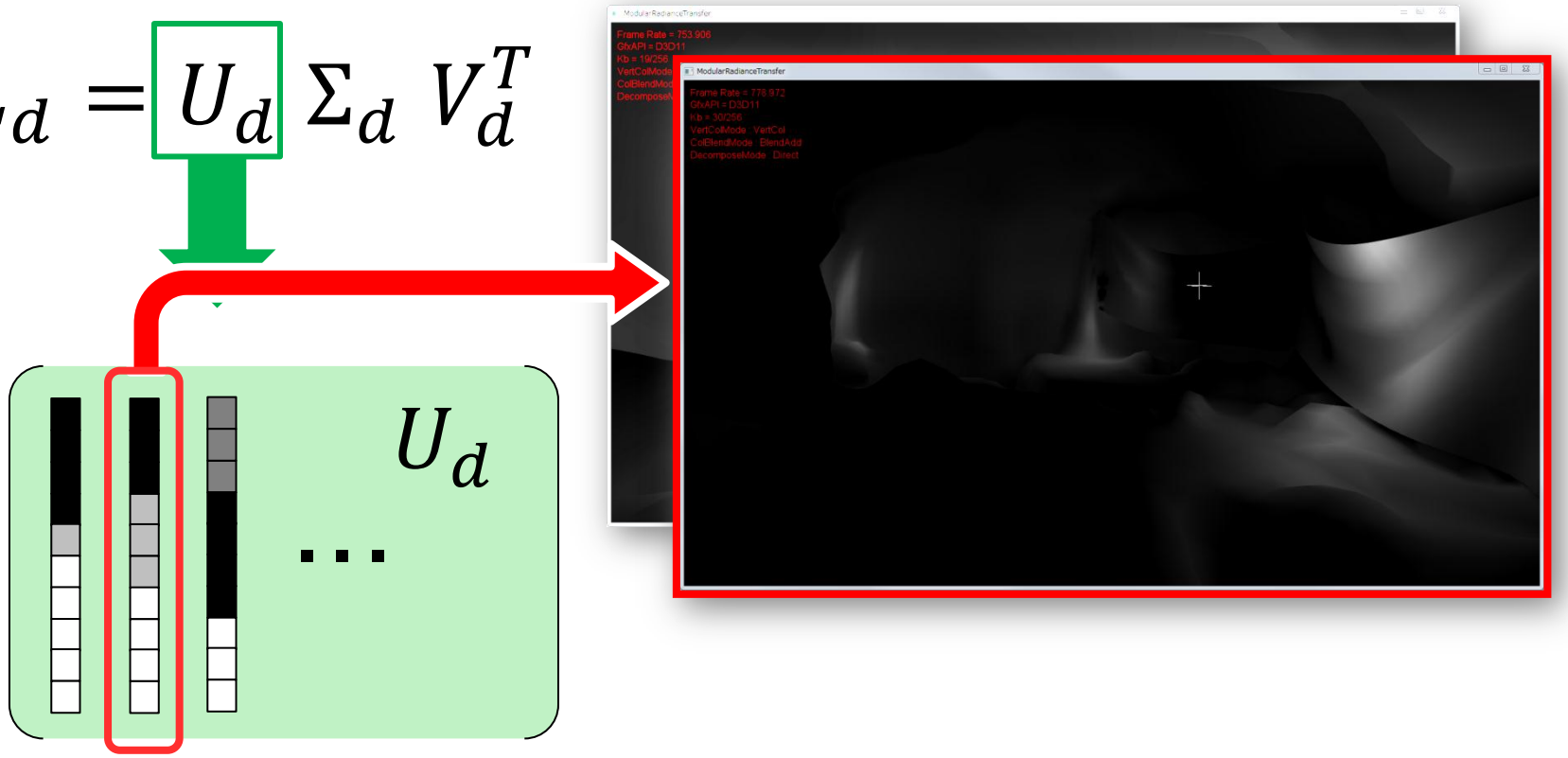




特異値分解(Singular Value Decomposition)

- L_d, L_{ind}に特異値分解を適用すると、行列UがB_d, B_{ind}となる

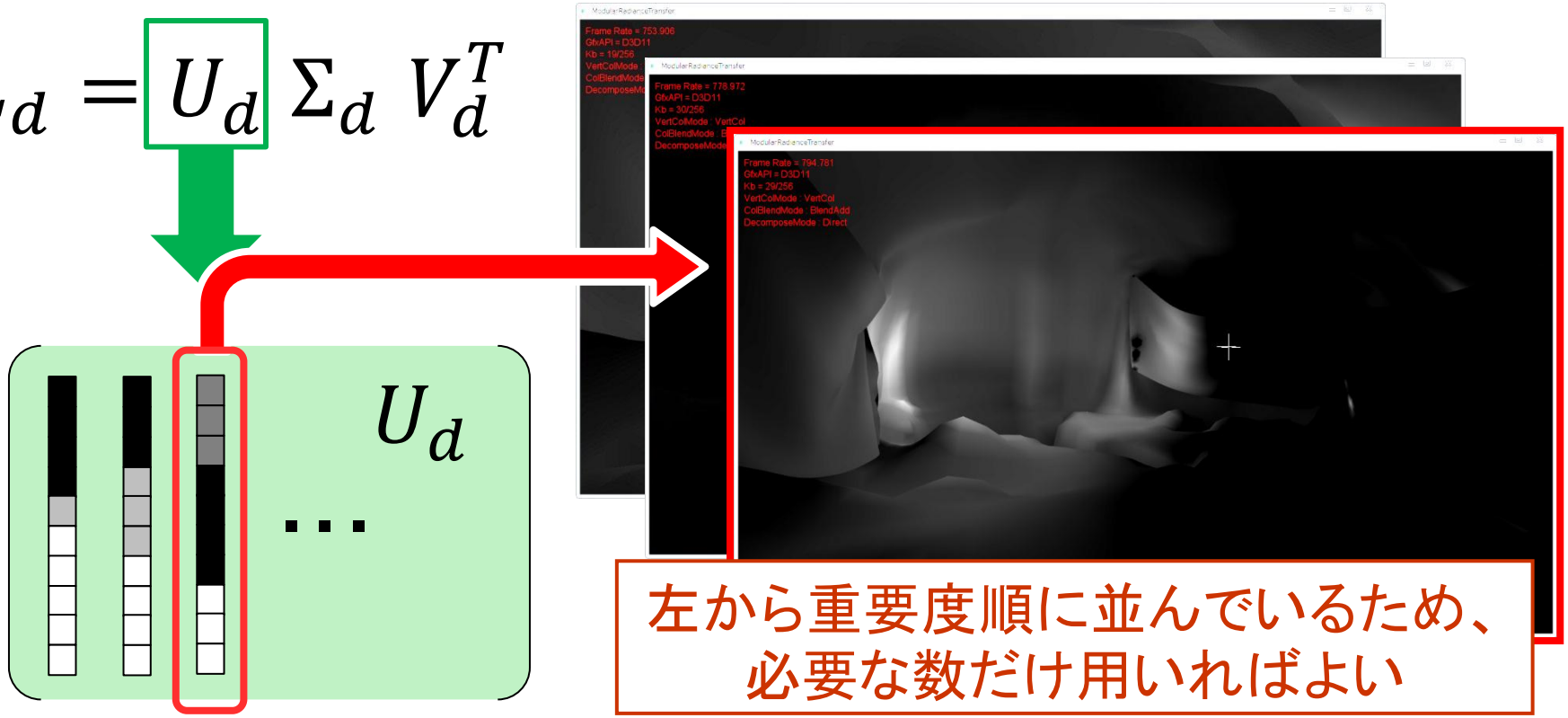
$$L_d = U_d \, \Sigma_d \, V_d^T$$



特異値分解(Singular Value Decomposition)

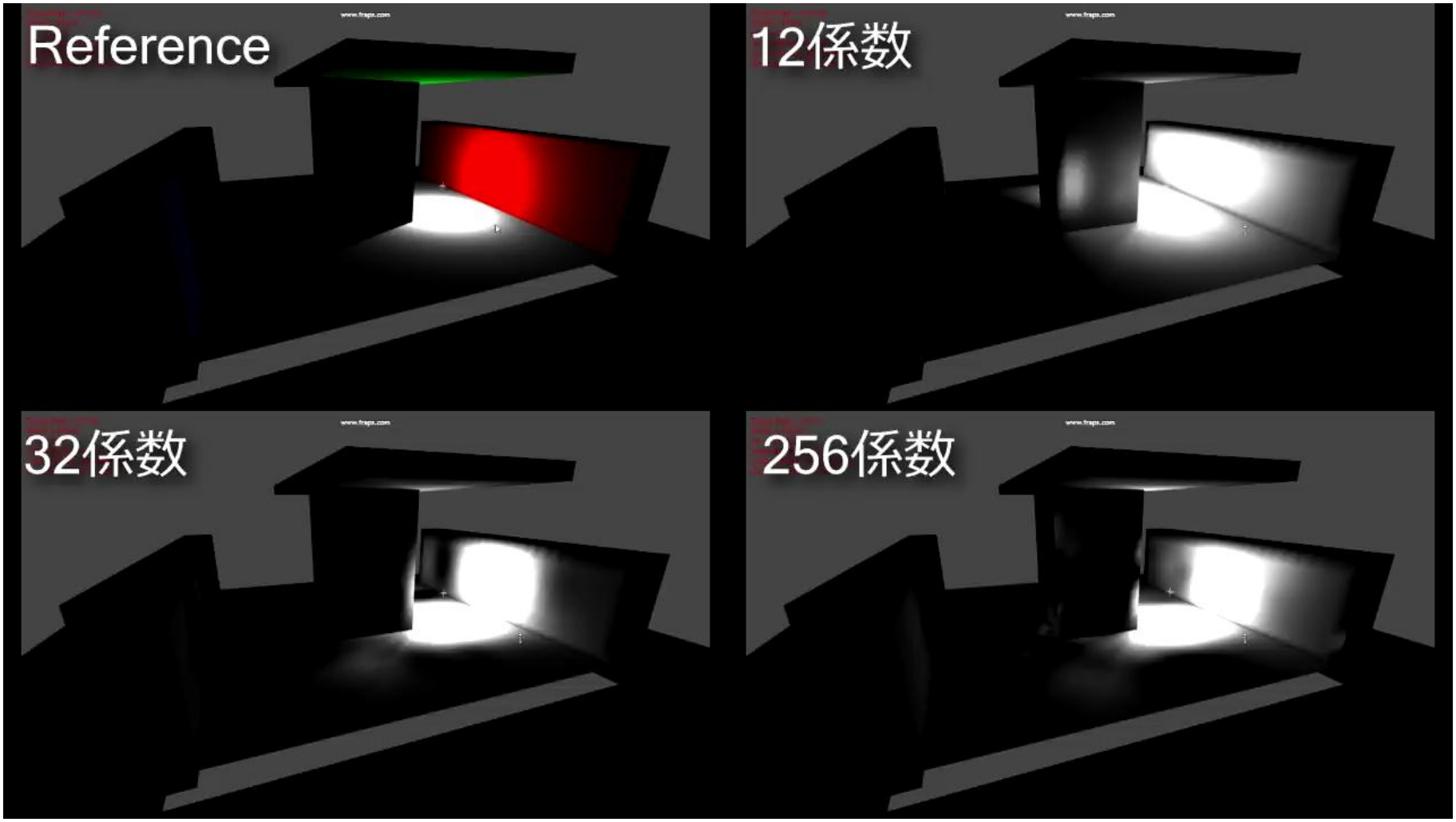
- L_d, L_{ind}に特異値分解を適用すると、行列UがB_d, B_{ind}となる

$$L_d = U_d \, \Sigma_d \, V_d^T$$



左から重要度順に並んでいるため、
必要な数だけ用いればよい

基底による直接光の近似



アプローチの比較

	直接光の近似精度	間接光の近似精度
直接光優先	○	△
間接光優先	△	○

- 優先してサンプリングした方の近似精度が高い
- 近似したいのは間接光なので間接光優先が好ましい

論文ではもう一歩進んだハイブリッドなアプローチを採用

ハイブリッド手法の計算手順

① $L_d \xrightarrow{SVD} U_d \, \Sigma_d \, V_d^T$

直接光優先アプローチ

② $L_{ind} = F U_d \Sigma_d$

①で求めた基底を元に
間接光サンプルを計算

③ $L_{ind} \xrightarrow{SVD} U_m \Sigma_m V_m^T$
$B_{ind} = U_m \Sigma_m$

間接光優先アプローチを
利用してB_{ind}を算出

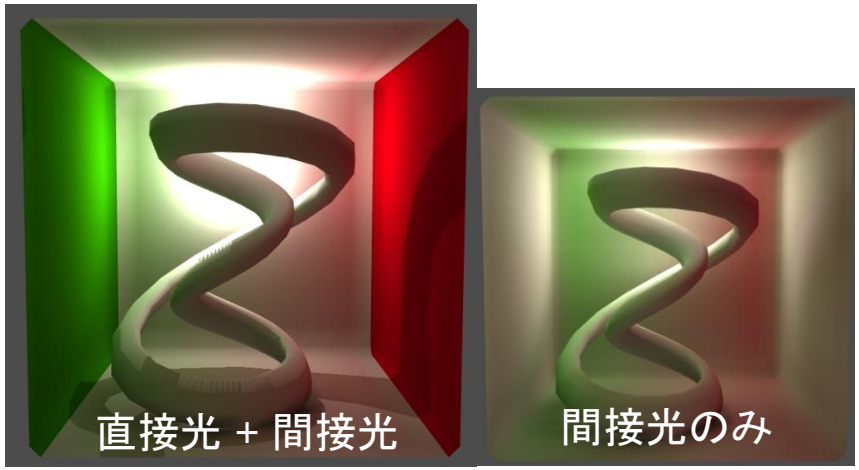
④ $L_{ind} = B_{ind} V_m^T = F U_d \Sigma_d$
$B_{ind} = F U_d \Sigma_d V_m = F B_d$
$B_d = U_d \Sigma_d V_m$

B_{ind}からB_dを計算

$B_{ind} = U_m \Sigma_m$, $B_d = U_d \Sigma_d V_m$

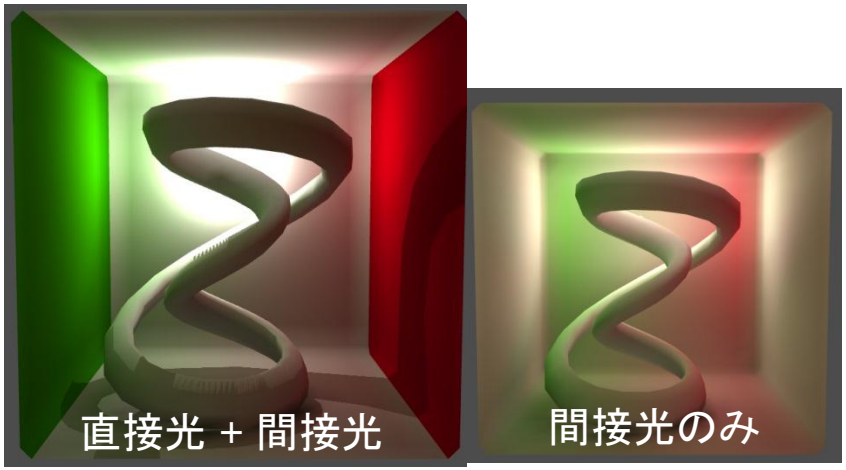
係数の数の比較

- 直接光優先



係数の数
32

- ハイブリッド



係数の数
4

ランタイムの計算処理

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 背景オブジェクトの間接光計算

- 導入
- 手法の詳細解説
  1. 最適な基底の計算方法
  2. 係数の計算方法
- ショートデモ

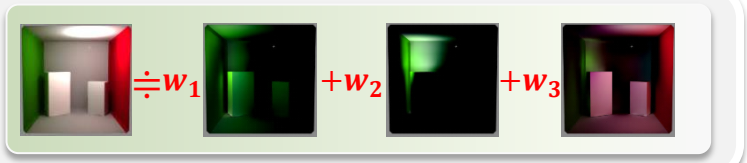
Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

係数の計算方法

- ランタイムで必要な計算は

$$l_{ind} = B_{ind}\boldsymbol{w}$$



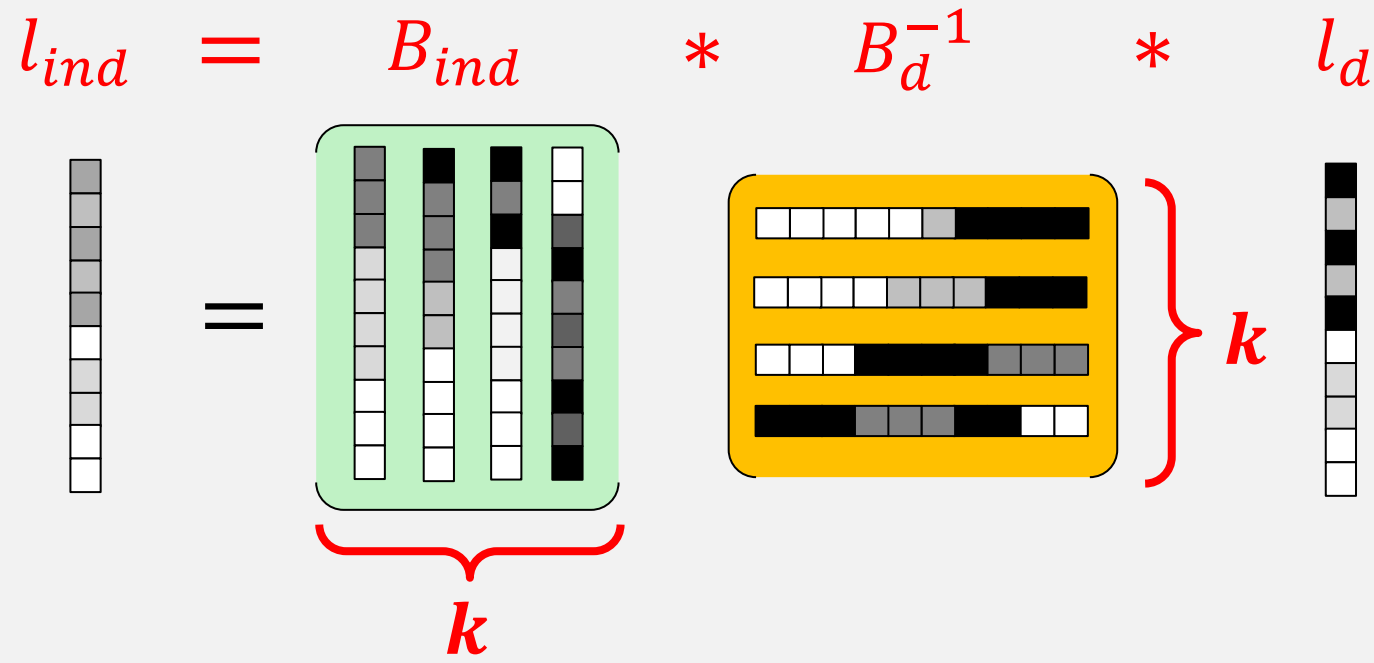
ここで

$$\boldsymbol{w} = B_d^{-1} l_d$$

- B_d^{-1}が必要だが、ハイブリッド手法の場合

$$B_d^{-1} = V_m^T \Sigma_d^{-1} U_d$$

ランタイムの計算処理



- ここでkは使用する係数の数
  - 自分たちのデモでは$k = 4\sim32$
  - 頂点数に比べ非常に少ない→高速

間接光計算の解説手順

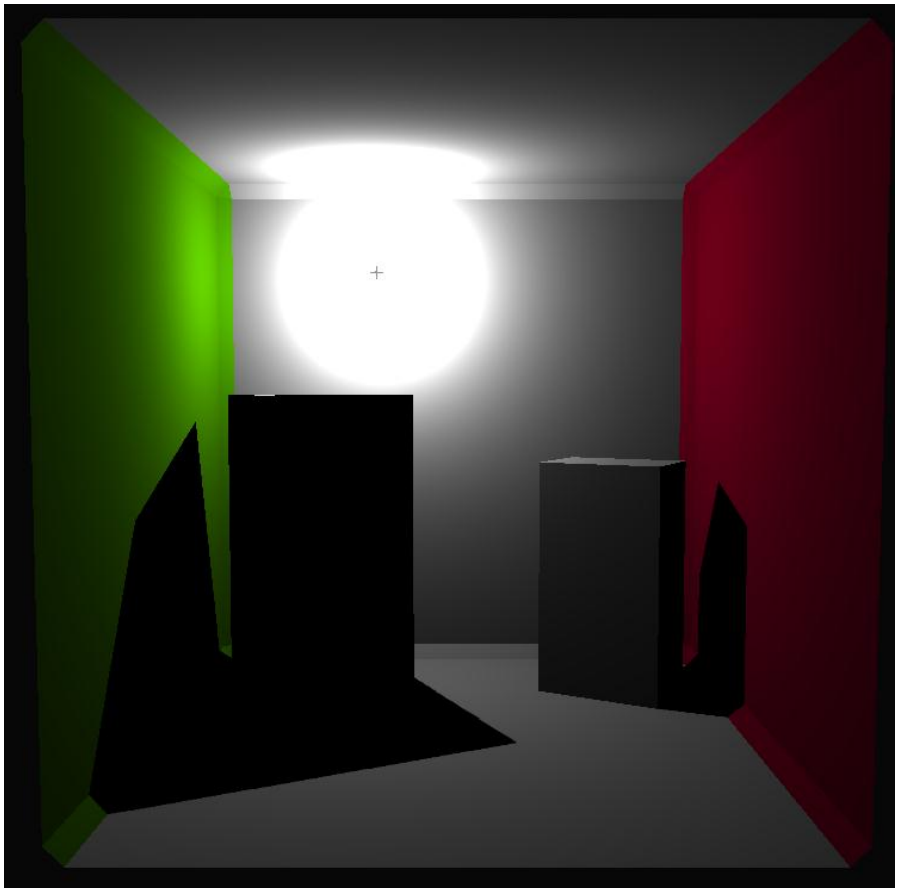
Step1. 背景オブジェクトの間接光計算

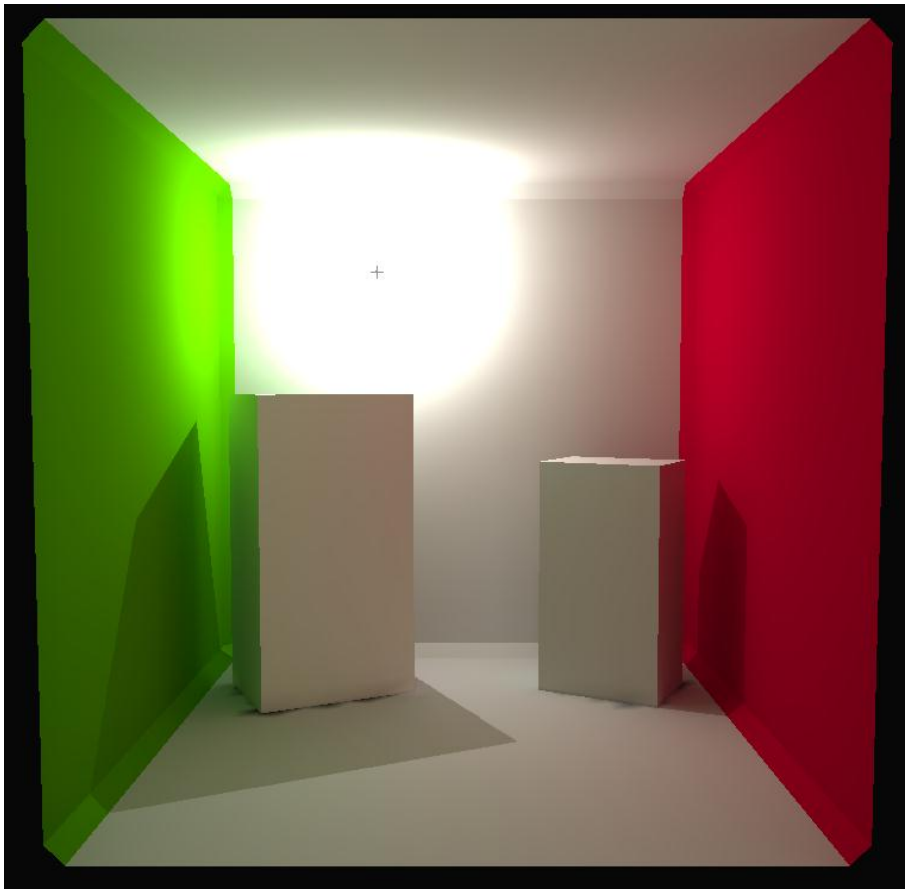
- 導入
- 手法の詳細解説
- ショートデモ

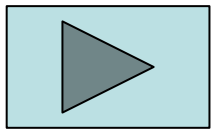
Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

ショートデモ







間接光計算の解説手順

Step1. 静的オブジェクトの間接光計算

Step2. その他オブジェクトへの光の伝播

- 手法概要の説明
- データ更新時における高速化の解説

Step3. デモ及び考察

間接光計算の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 静的オブジェクトの間接光計算

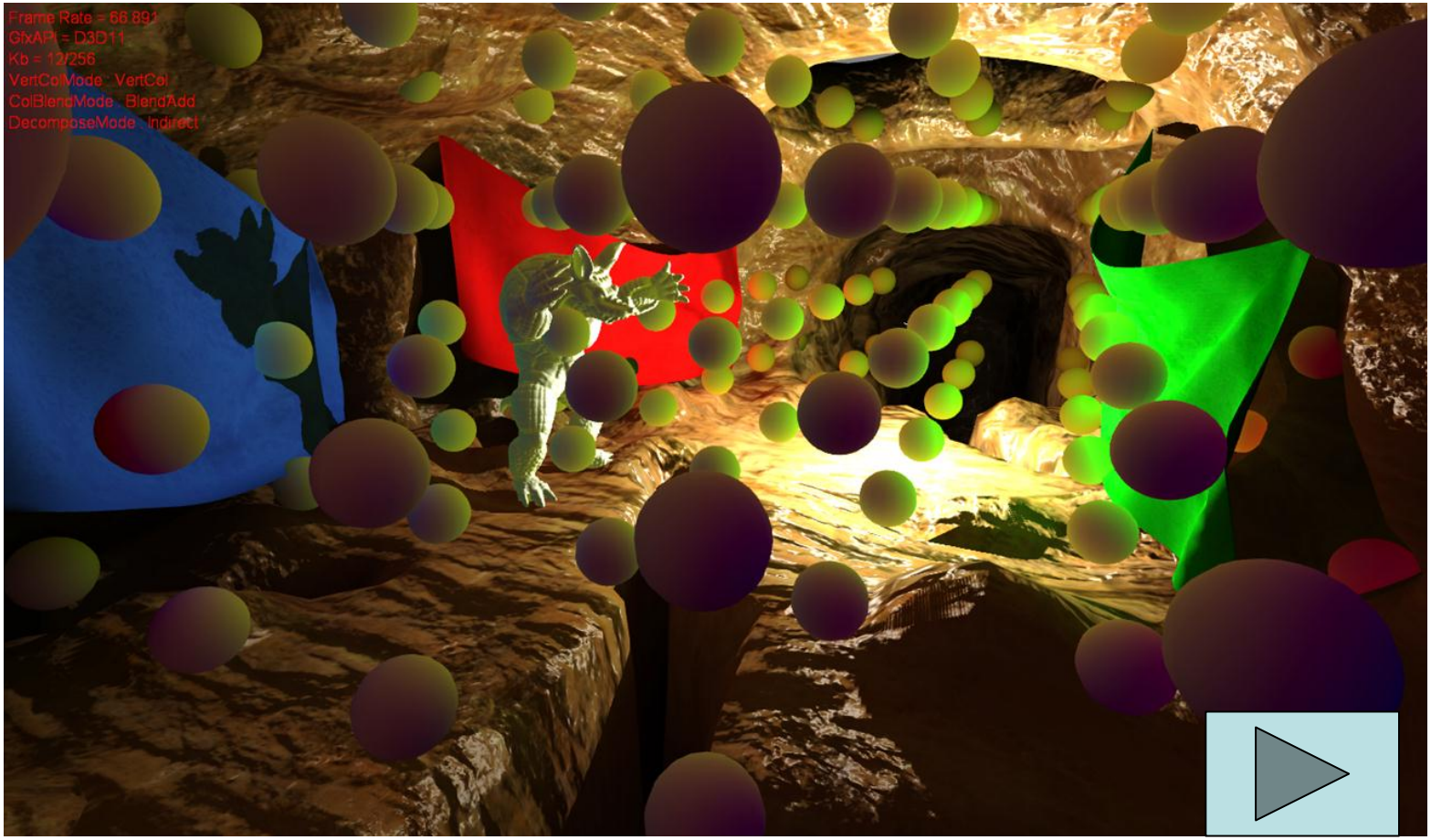
Step2. その他オブジェクトへの光の伝播

- 手法概要の説明
- データ更新時における高速化の解説

Step3. デモ及び考察

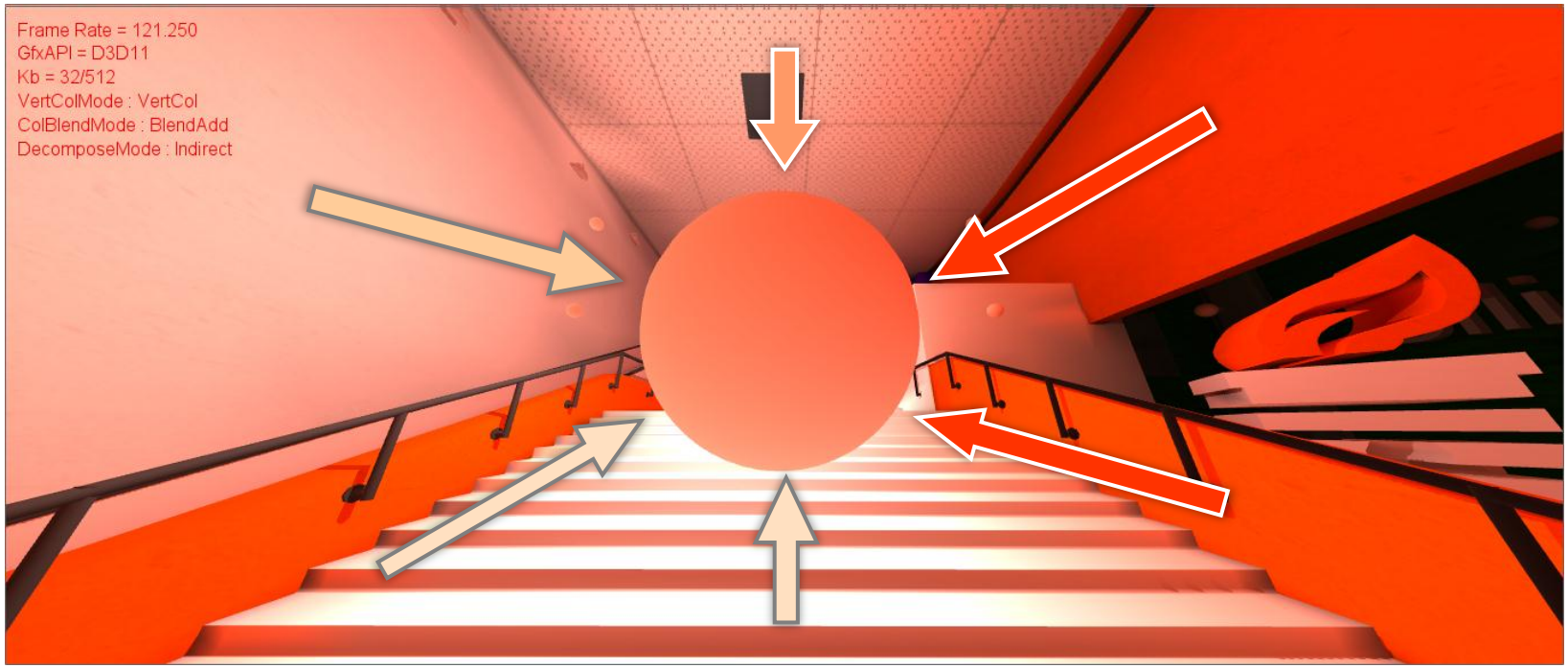
Light Probe

- 背景以外のオブジェクトにはLight Probeを用いる



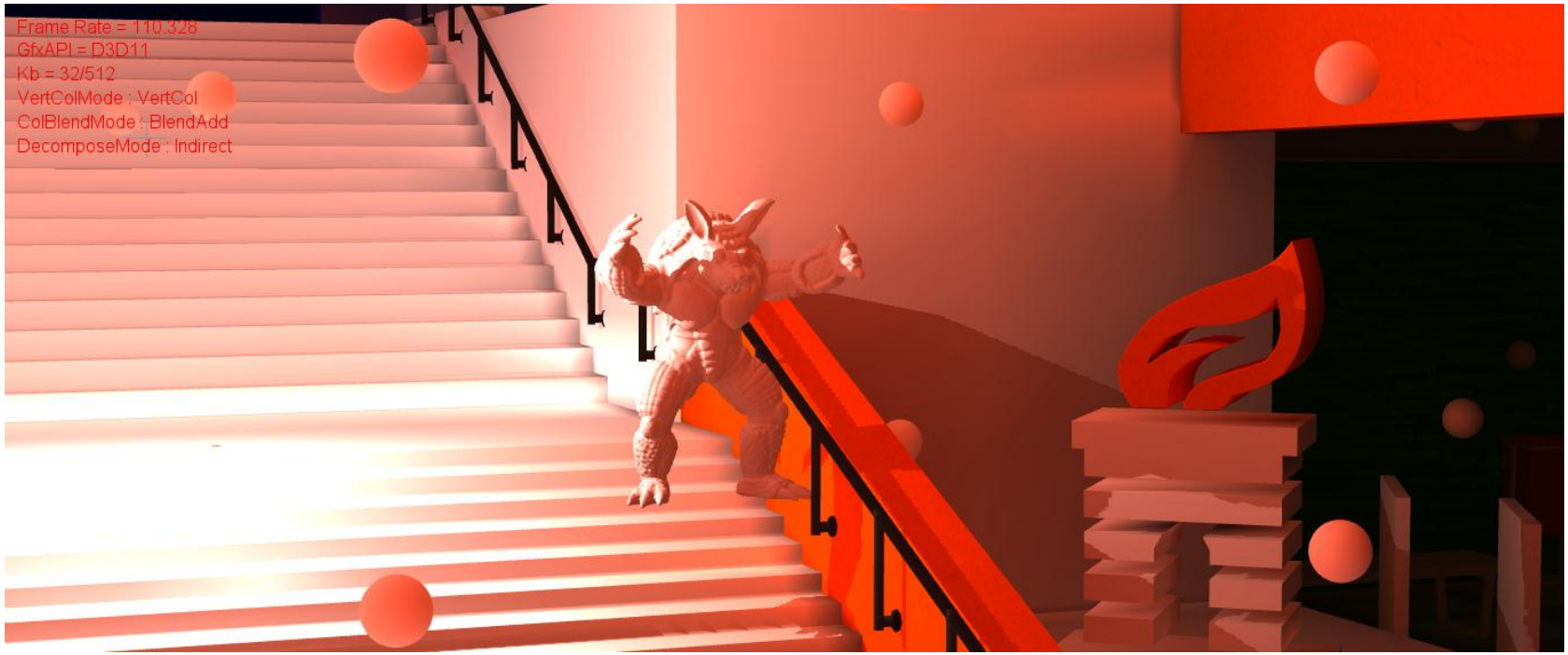
Light Probe

- Light Probeとは
  - 全方位の入射光を保持する仮想的な球
  - 今回は球面調和関数(SH)を使用



Light Probeによるライティング

- Light Probeを離散的に配置
  - シェーダで近傍のLight Probeの値を補間して間接光の表現が可能



Light Probeによるライティング

- 時間の都合上、詳細は割愛
  詳しくは下記書籍をご参照下さい
  - Chapter8「KILLZONE2」: Light Probe自体の解説
  - Chapter9「バイオハザード5」: Light Probeによるライティング

西川善司著
「3Dゲームファンのための
グラフィックス講座」

ここまでは、あくまで既存の手法の話

間接光計算の解説手順

Step1. 静的オブジェクトの間接光計算

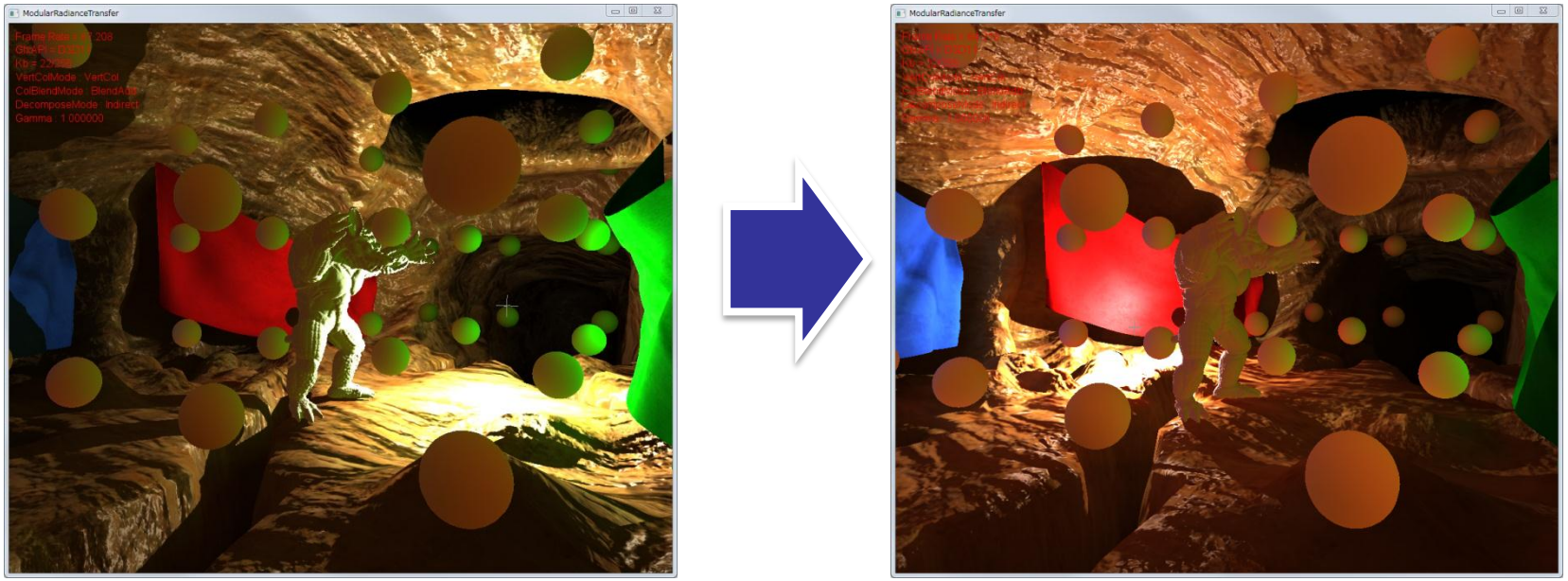
Step2. その他オブジェクトへの光の伝播

- 手法概要の説明
- データ更新時における高速化の解説

Step3. デモ及び考察

Light Probeの計算

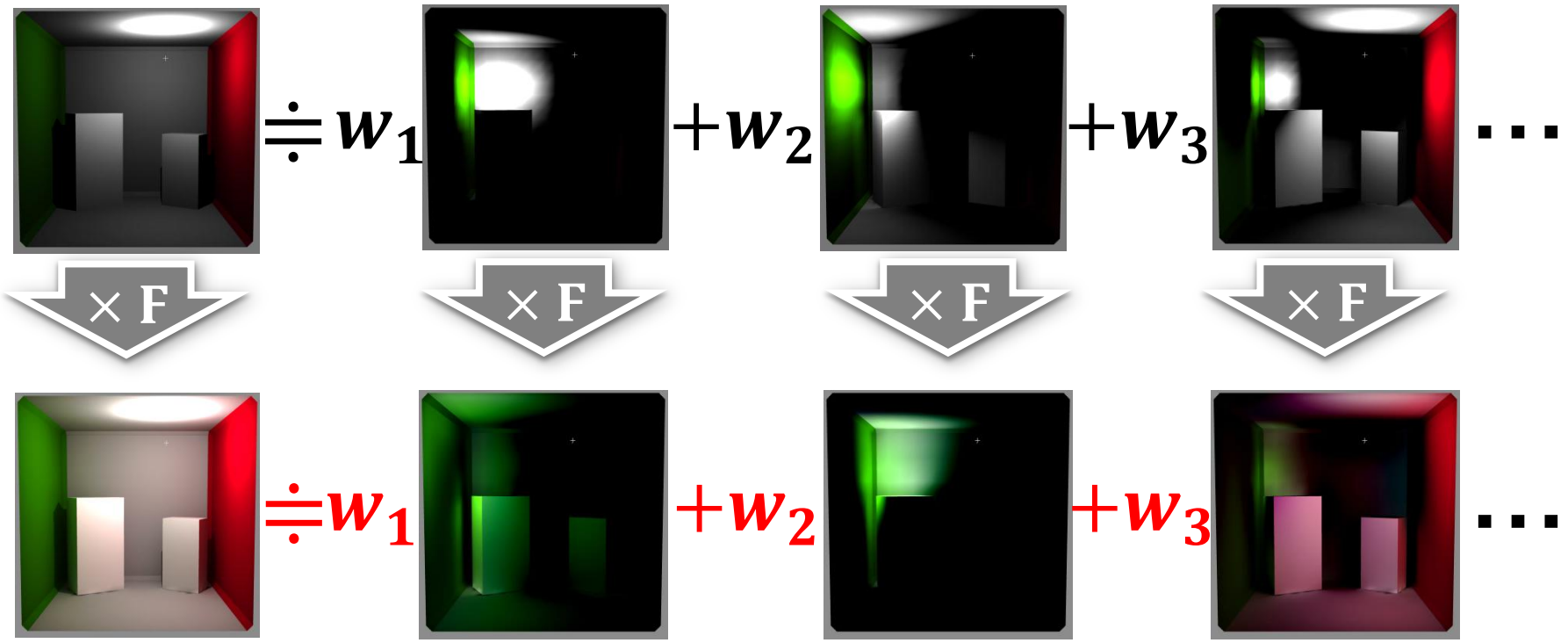
- 通常Light Probeは前計算した後更新しない
  - Light Probeの計算は負荷が高い



今回は、動的にLight Probeを更新する必要がある

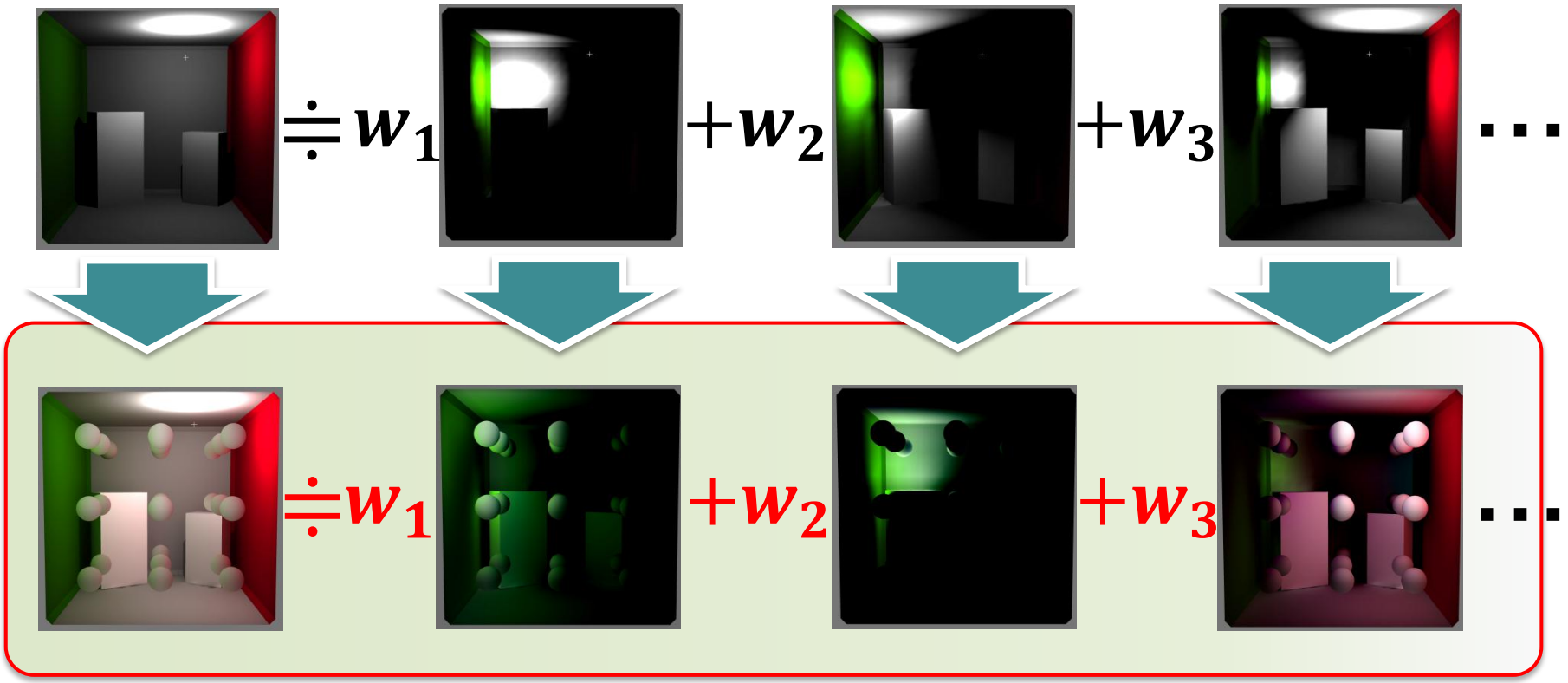
動的なLight Probe

- 今回、シーンのライティング結果は既知の状態の組み合わせに分解される



動的なLight Probe

- 間接光と同様に、それぞれに対応したLight Probeを計算しておけば足し合わせで高速に計算可能



間接光計算の解説手順

Step1. 背景オブジェクトの間接光計算

Step2. その他オブジェクトへの光の伝播

Step3. デモ及び考察

デモ

デモ詳細

- 背景オブジェクト
  - 頂点数 10800 (頂点)
  - 基底の係数 22 (係数)
- Light Probe
  - Probe総数 500 (個)
  - 基底の係数 12 (係数)
  - 球面調和関数の係数 9 (係数)

パフォーマンス

	影あり	影なし	影なし・GIなし
Geforce GTX 680	500 fps	800 fps	1500 fps
Geforce GTX 560	360 fps	440 fps	750 fps
Radeon 6630 M	80 fps	120 fps	280 fps

- 解像度は1400*800で測定
- 実装にはCompute Shaderを使用
- 前計算約45分
  - 2520光源サンプル
  - 20回反射まで考慮
  - 最適化で10倍以上の高速化の余地あり

実装のTipsはAppendixに記載

まとめ

- 良い点
  - 非常に高速＆計算負荷が一定
  - 反射回数が無制限(行列Fに含められる)
  - 高速なLight Probe計算
  - 様々な拡張
- 悪い点
  - Lightの位置・方向を含めた前計算
  - 動的すぎるシーンは不可(→Delta Radiance Transfer)
  - 頂点カラーを念頭に置いたメッシュ分割＆頂点配置
  - 複雑すぎない形状

発表の流れ

1. 発表の概要
2. 技術紹介 + デモ
   - 間接光計算手法の解説（論文4節）
   - シーン分割による高速化の解説（論文5節）
3. まとめ

シーン分割による高速化の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 問題提起

Step2. 小領域間の光の伝播の計算

Step3. 手法の拡張

Step4. デモ及び考察

シーン分割による高速化の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 問題提起

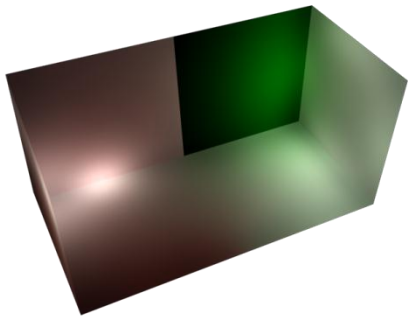
Step2. 小領域間の光の伝播の計算

Step3. 手法の拡張

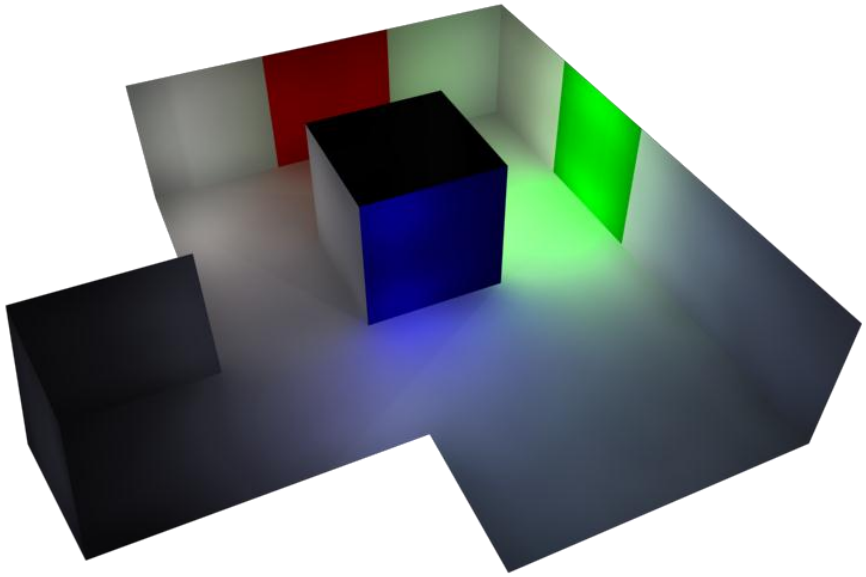
Step4. デモ及び考察

問題提起（1/4）

- 対象とする問題：
  シーンが複雑になった場合の計算負荷の増大



シンプルなシーン

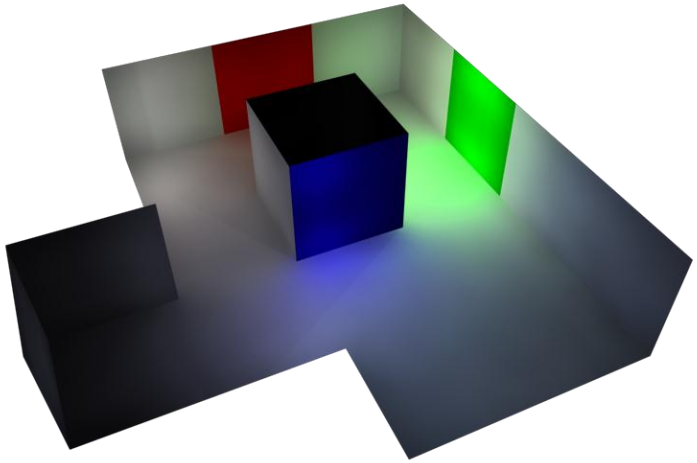


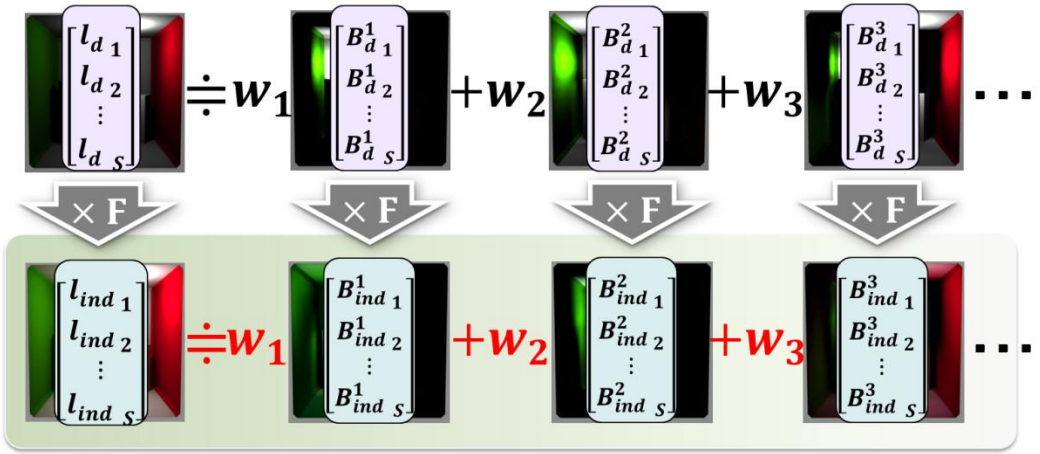
より複雑なシーン

問題提起（2/4）

- 複雑なシーンでは：
  1. メッシュ頂点数が増加
  2. 必要な係数の数の増大

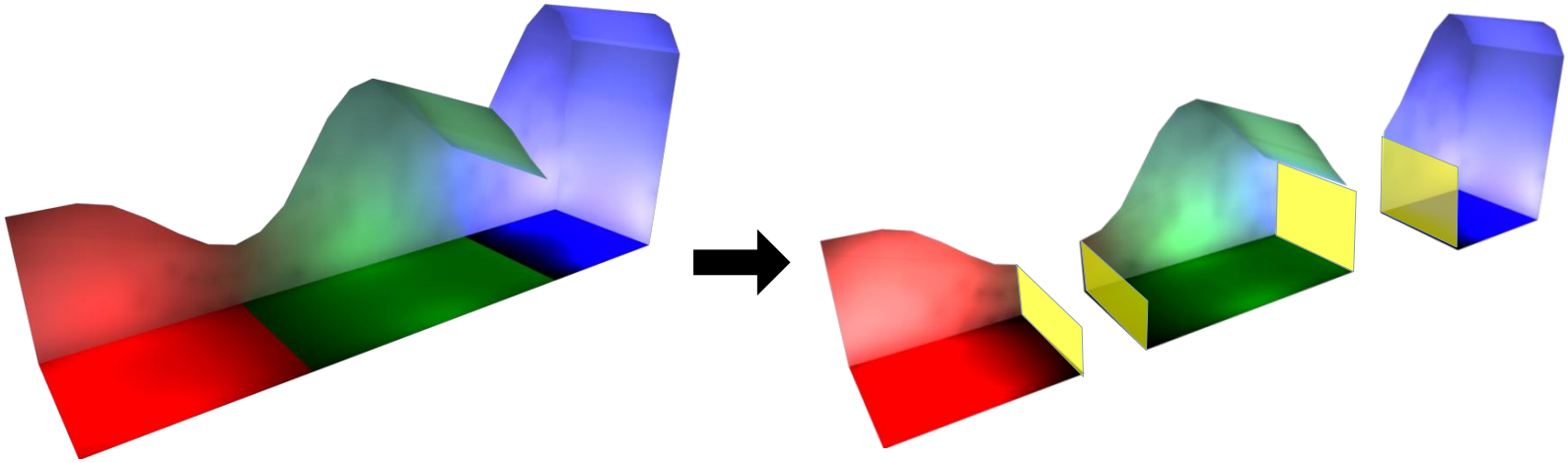
  } ⇒ 計算量の増大





問題提起（3/4）

- 解決策：シーンを複数の小領域に分割

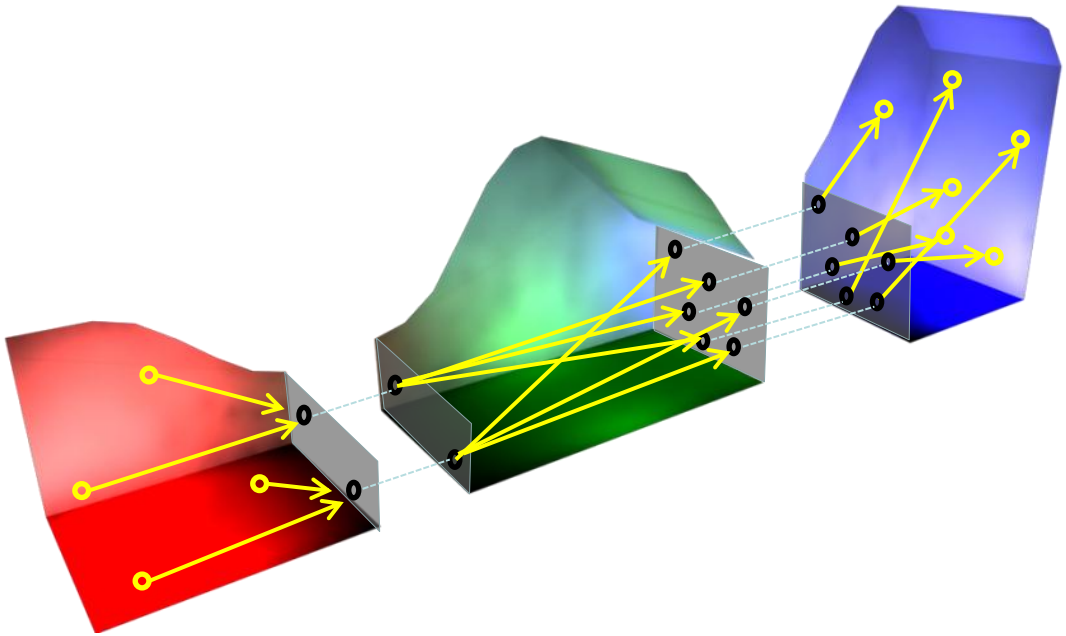


⇒ 適切に小領域を設定すればシーンの複雑さを抑える事が可能

- 小領域の分割方法はユーザーが決定

問題提起（4/4）

- 各小領域内の間接光は、前述の方法で計算可能
- 小領域間の光の伝播を別途考慮する必要がある



小領域間の光の伝播の高速な計算方法を解説

シーン分割による高速化の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 問題提起

Step2. 小領域間の光の伝播の計算

- キーアイディアの説明
- 小領域間の伝播行列の計算

Step3. 手法の拡張

Step4. デモ及び考察

シーン分割による高速化の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 問題提起

Step2. 小領域間の光の伝播の計算

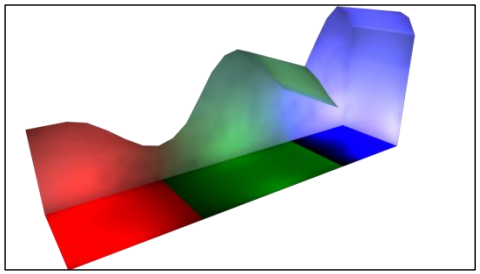
- キーアイディアの説明
- 小領域間の伝播行列の計算

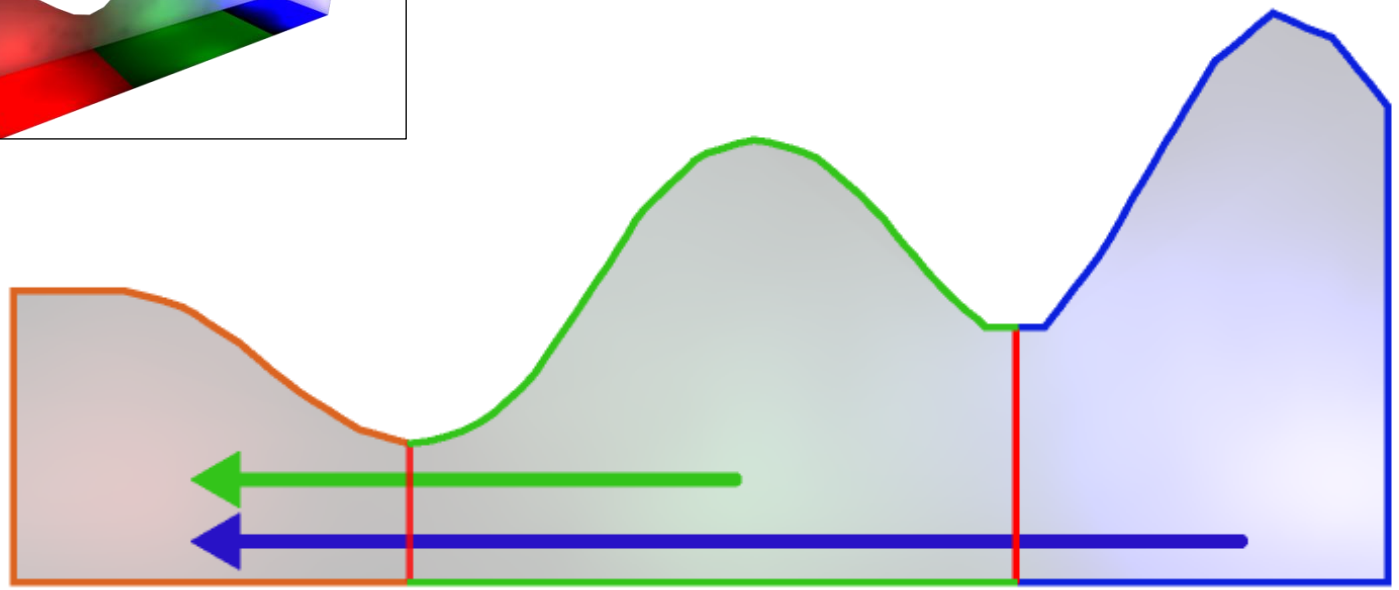
Step3. 手法の拡張

Step4. デモ及び考察

キーアイディア（1/6）

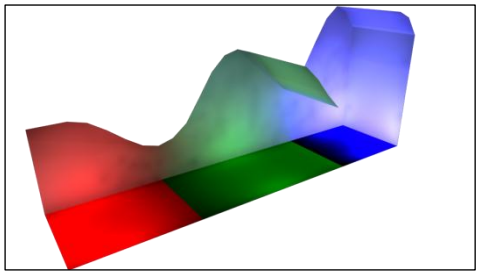
- 小領域間の光の伝播を個別に計算

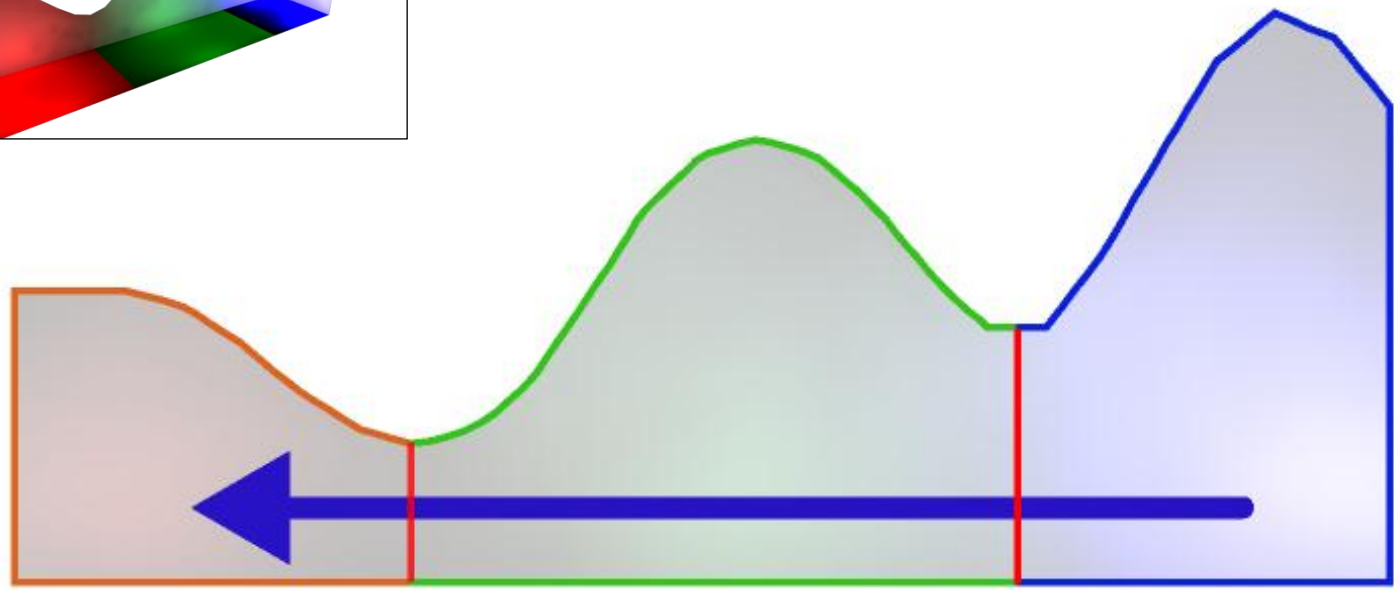




キーアイディア（2/6）

- 例として右⇒（中央）⇒左へ伝播する光を考える

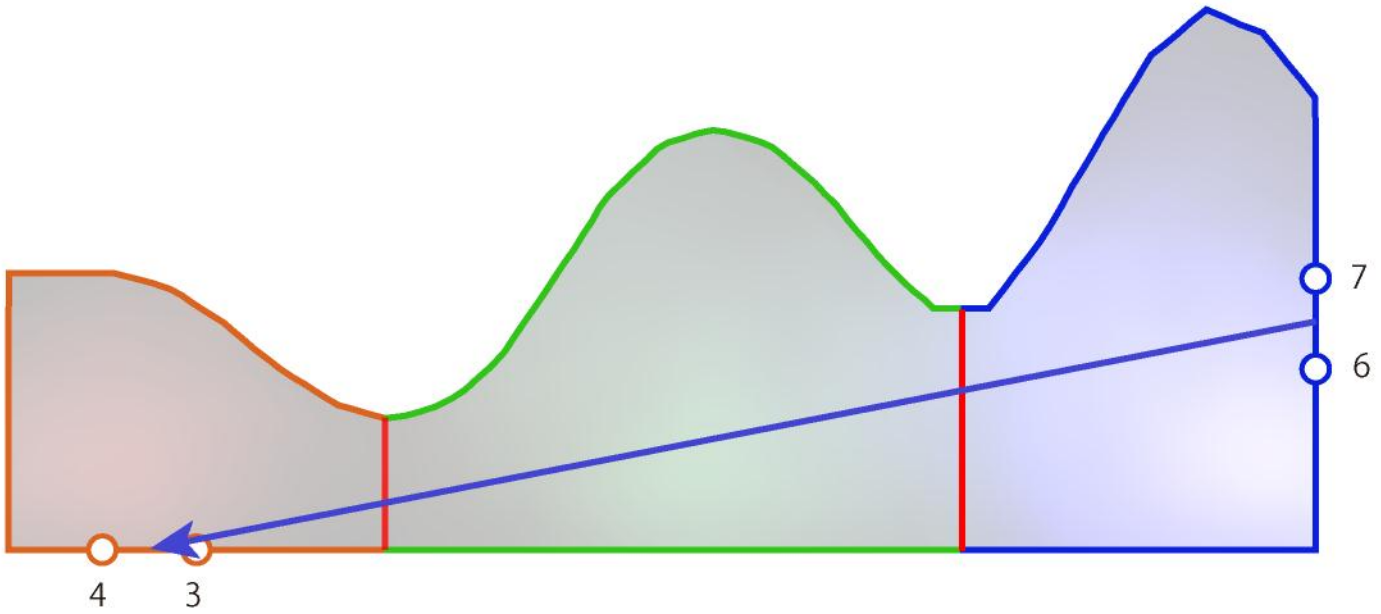




キーアイディアでは多重反射を考慮しない光の伝播を考える

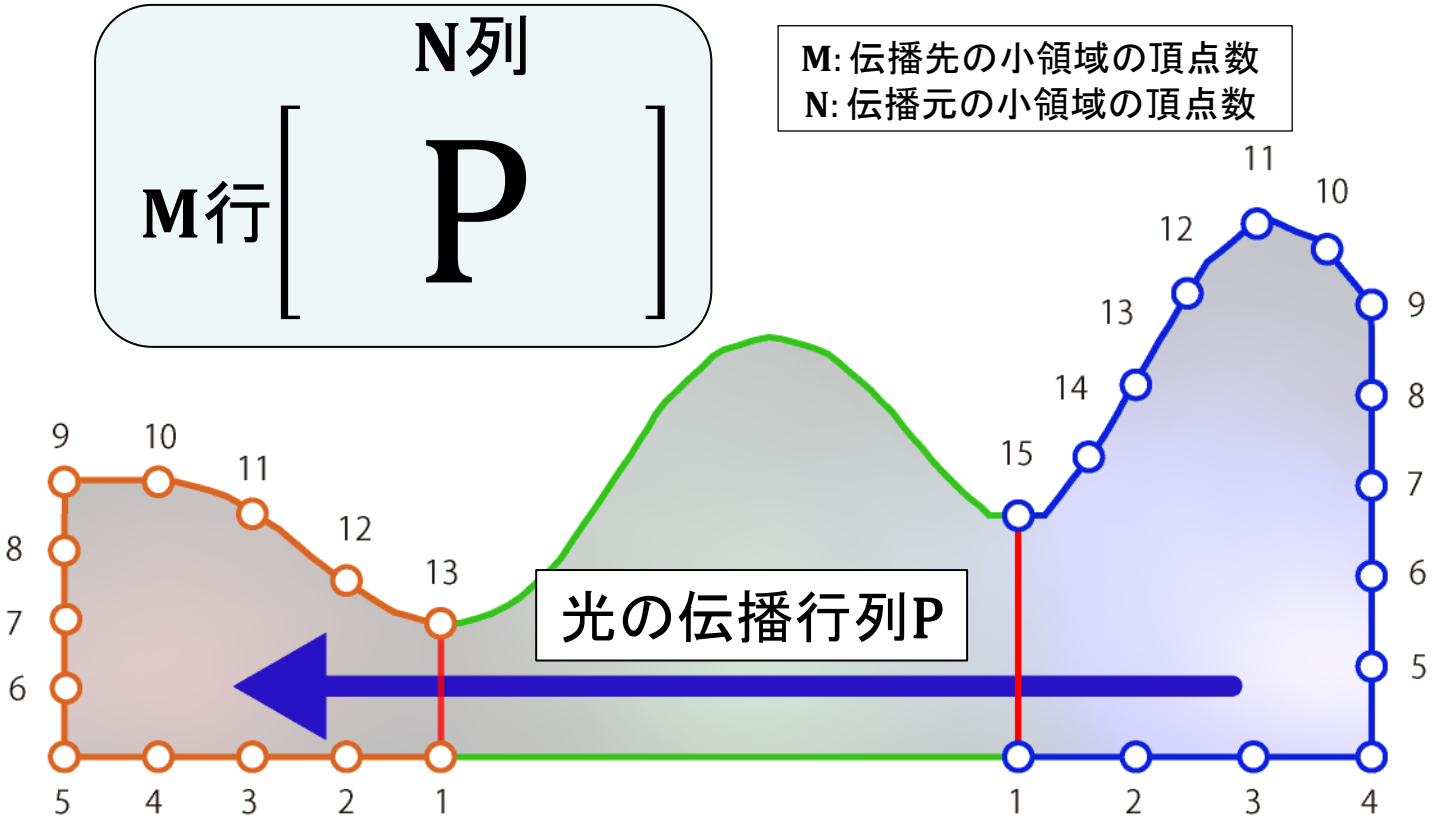
キーアイディア（3/6）

- 小領域間の光の伝播は頂点間の影響に帰着



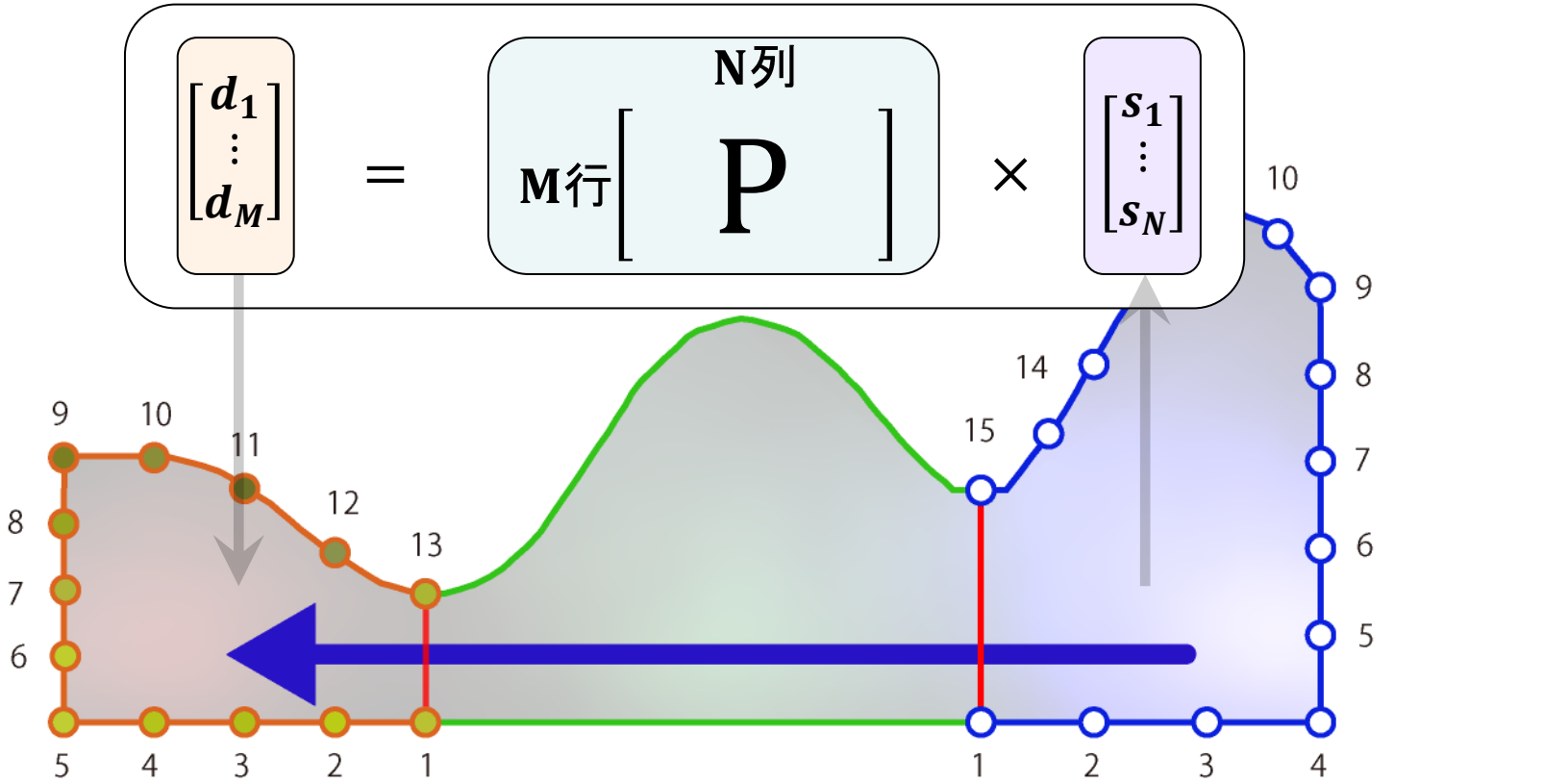
キーアイディア（4/6）

- 小領域間の光の伝播行列Pを前計算



キーアイディア（5/6）

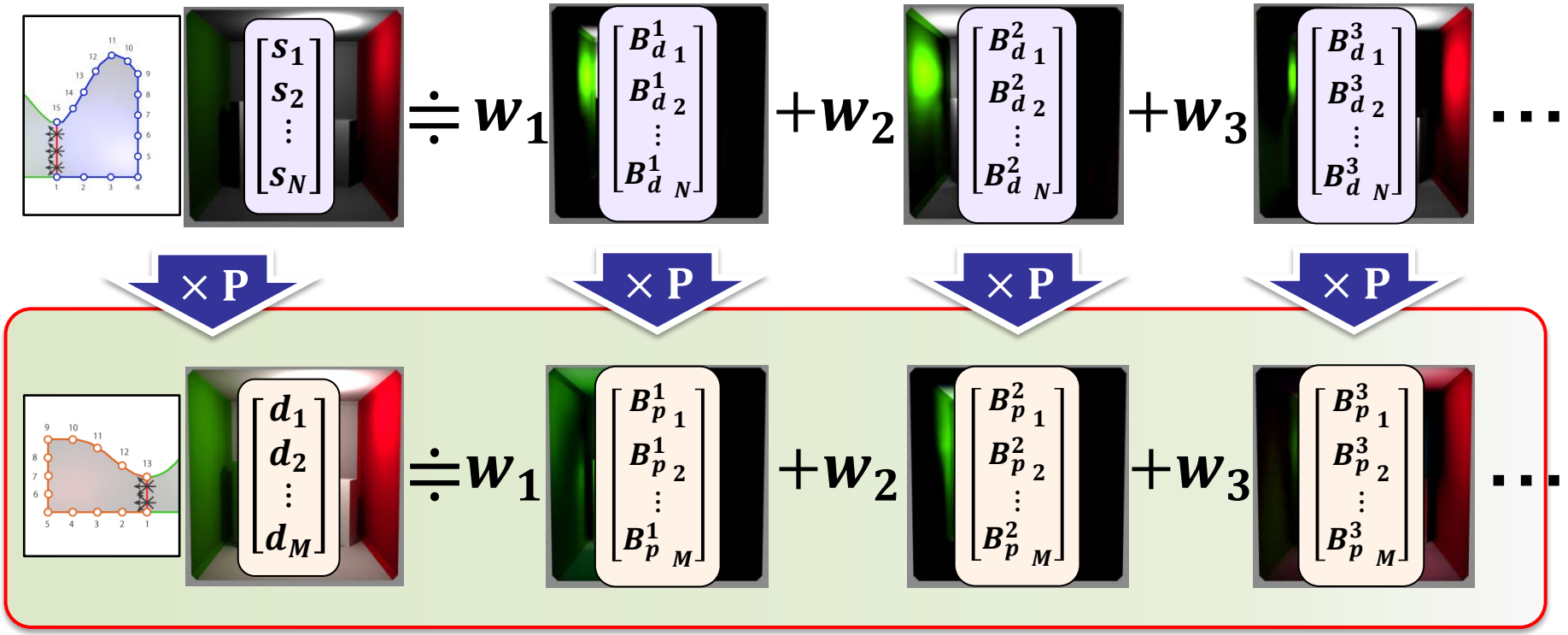
- 光の伝播行列Pを利用して光の伝播を高速に計算



d: 伝播先の小領域の頂点の色	M: 伝播先の小領域の頂点数 N: 伝播元の小領域の頂点数	s: 伝播元の小領域の頂点の色

キーアイディア（6/6）

- 光の伝播を係数からの変換に修正



シーン分割による高速化の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 問題提起

Step2. 小領域間の光の伝播の計算

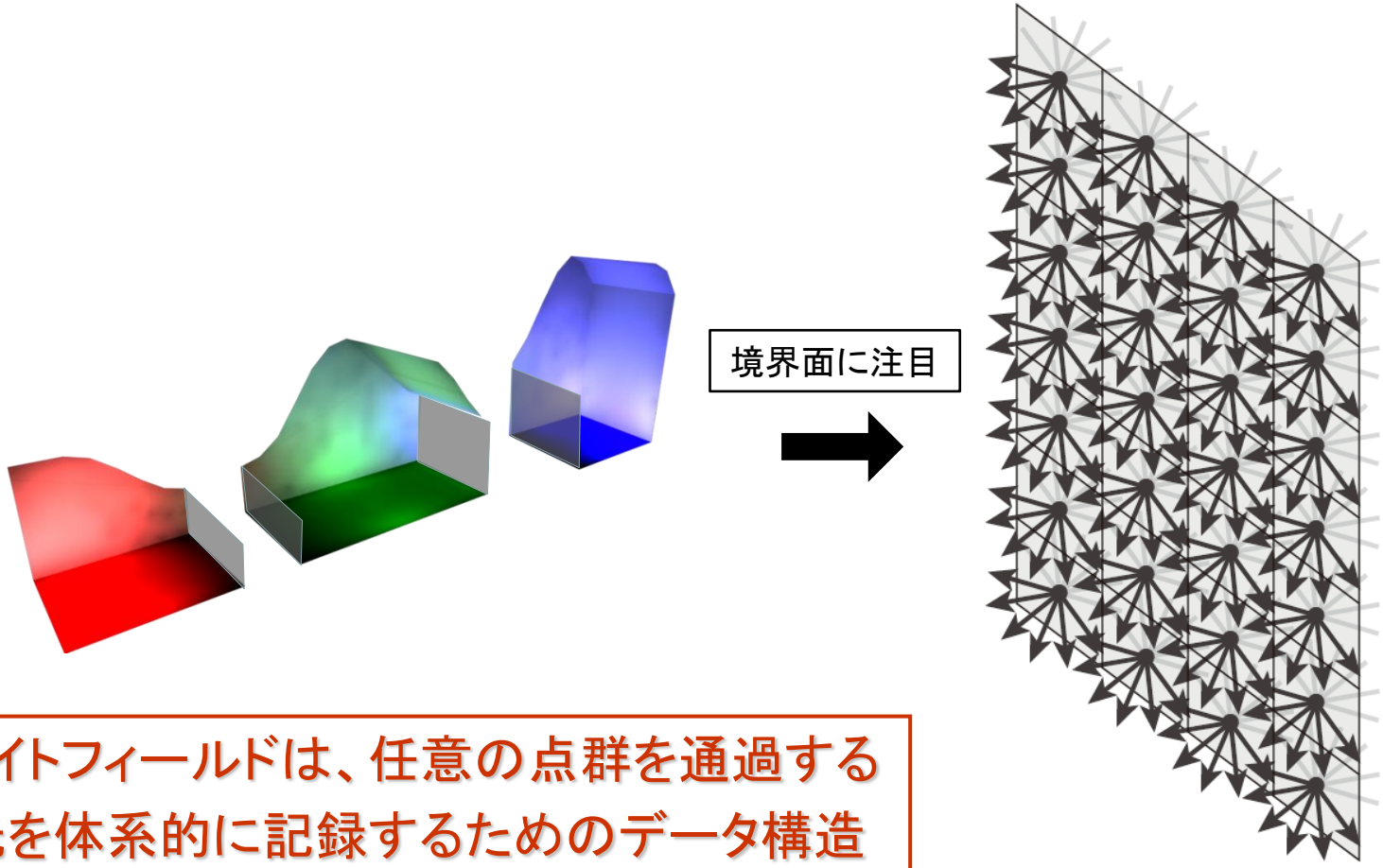
- キーアイディアの説明
- 小領域間の伝播行列の計算

Step3. 手法の拡張

Step4. デモ及び考察

光の伝播行列の計算（1/7）

- ライトフィールドという概念を導入

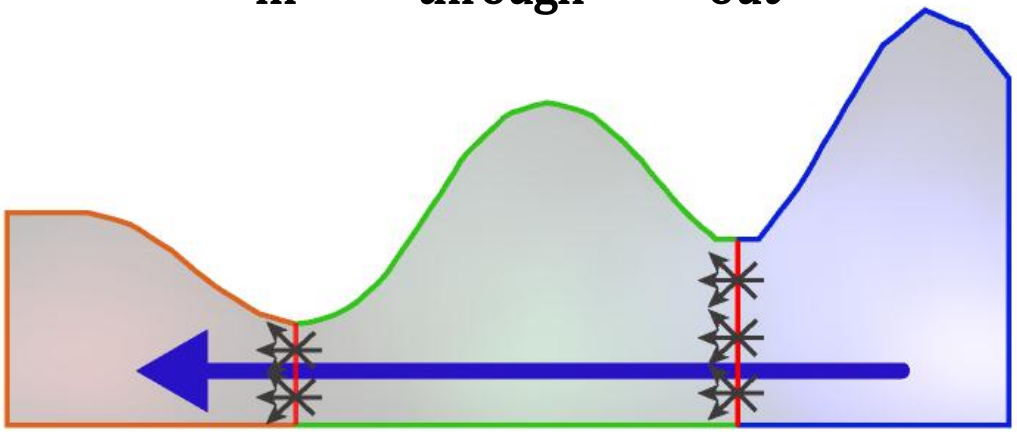


ライトフィールドは、任意の点群を通過する
光を体系的に記録するためのデータ構造

光の伝播行列の計算（2/7）

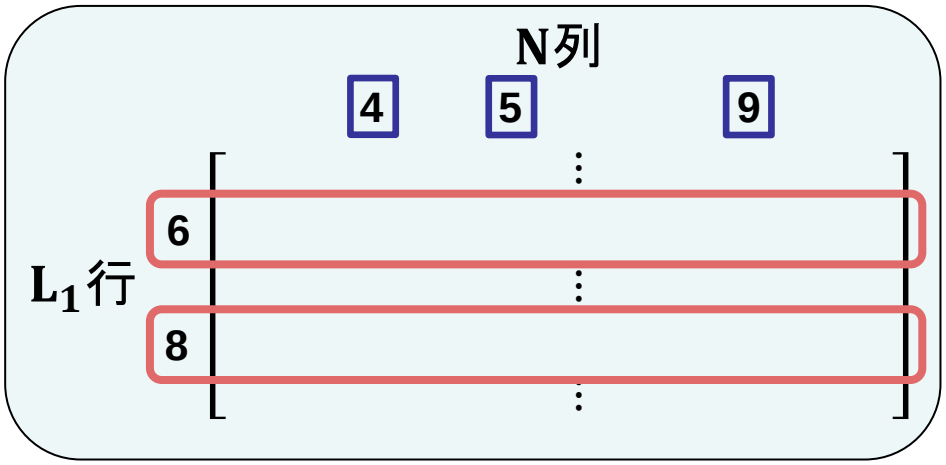
- 光の伝播行列Pを三要素に分割
  1. 光の出射行列 P_{out} ：右の小領域
  2. 光の通過行列 $P_{through}$ ：中央の小領域
  3. 光の入射行列 P_{in} ：左の小領域

$$P = P_{in} \times P_{through} \times P_{out}$$

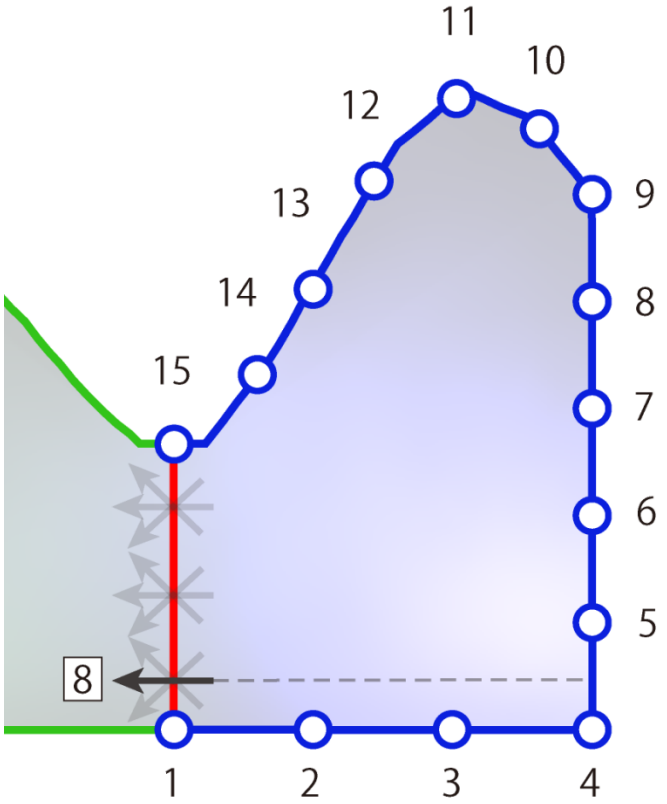


光の伝播行列の計算（3/7）

- 光の出射行列P_{out}
  - 小領域 ⇒ ライトフィールド

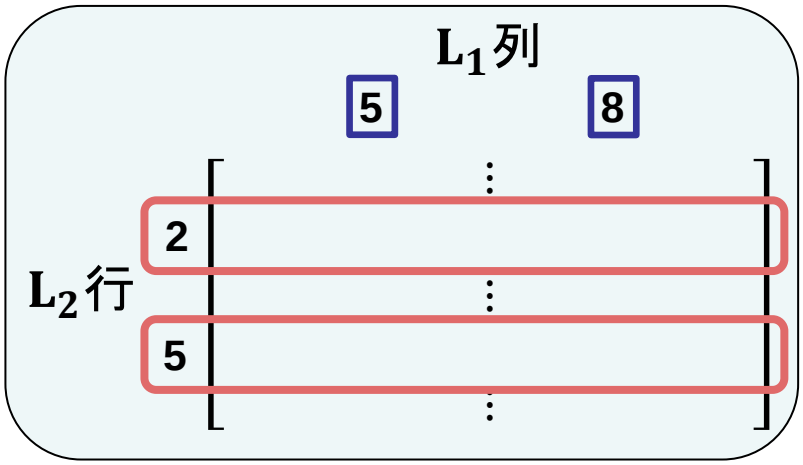


L_1: 出射ライトフィールドのサンプル数
N: 伝播元の小領域の頂点数

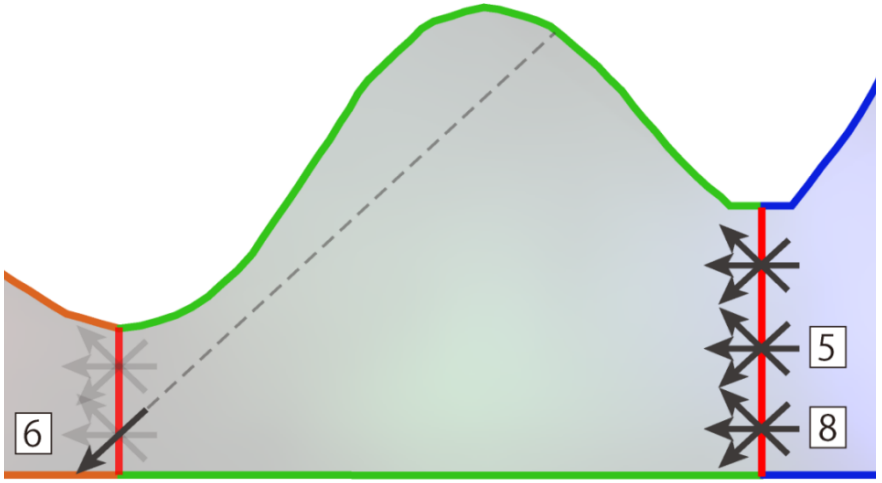


光の伝播行列の計算（4/7）

- 光の通過行列$P_{through}$
  - ライトフィールド ⇒ ライトフィールド

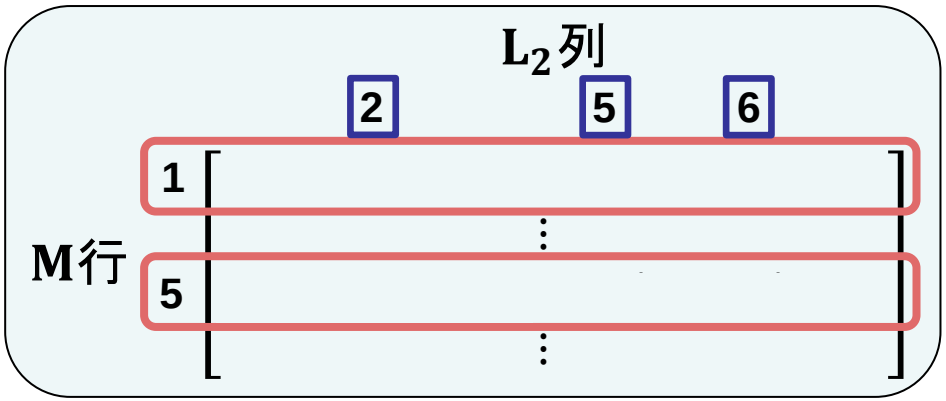


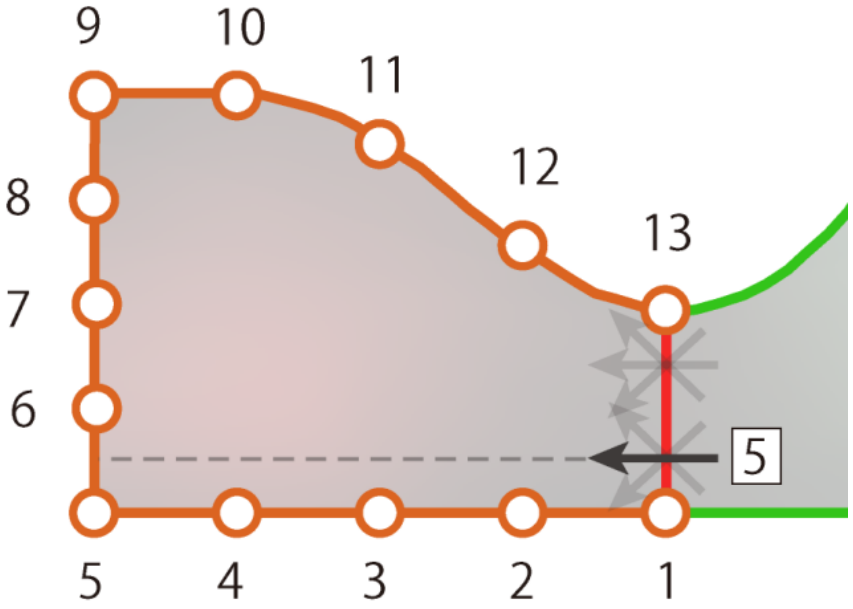
L_2: 出射ライトフィールドのサンプル数
L_1: 入射ライトフィールドのサンプル数



光の伝播行列の計算（5/7）

- 光の入射行列P_{in}
  - ライトフィールド ⇒ 小領域



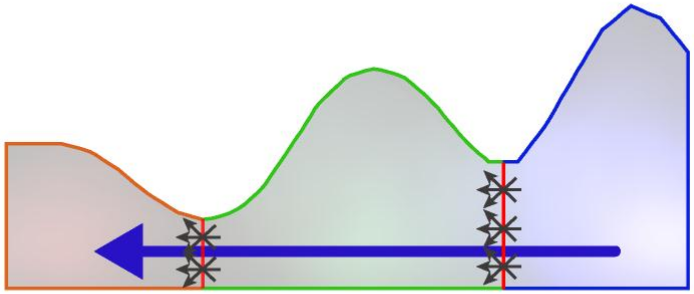


M: 伝播先の小領域の頂点数
L_2: 入射ライトフィールドのサンプル数

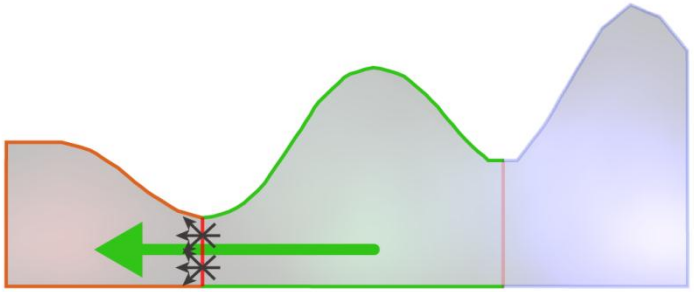
P_{in} 計算時にヘミキューブ法などを用いて立体角を考慮

光の伝播行列の計算（6/7）

- 全ての小領域間の光の伝播は三種類の行列の組み合わせで計算可能



$$P = P_{in} \times P_{through} \times P_{out}$$

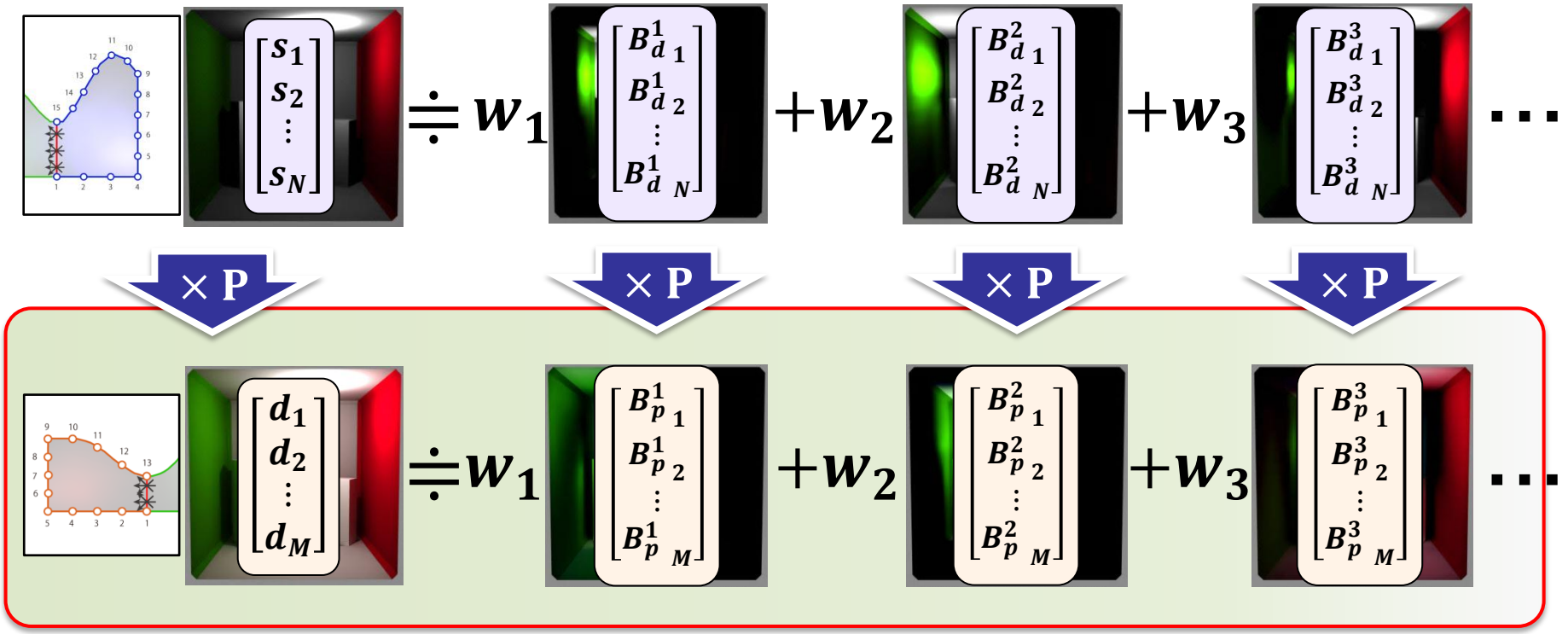


$$P = P_{in} \times P_{out}$$

上記例ではP_{out}は再利用不可能だがP_{in}は共通

光の伝播行列の計算（7/7）

- 光の伝播の計算のまとめ



シーン分割による高速化の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 問題提起

Step2. 小領域間の光の伝播の計算

Step3. 手法の拡張

- 多重反射への拡張
- モジュール導入による更なる効率化

Step4. デモ及び考察

シーン分割による高速化の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 問題提起

Step2. 小領域間の光の伝播の計算

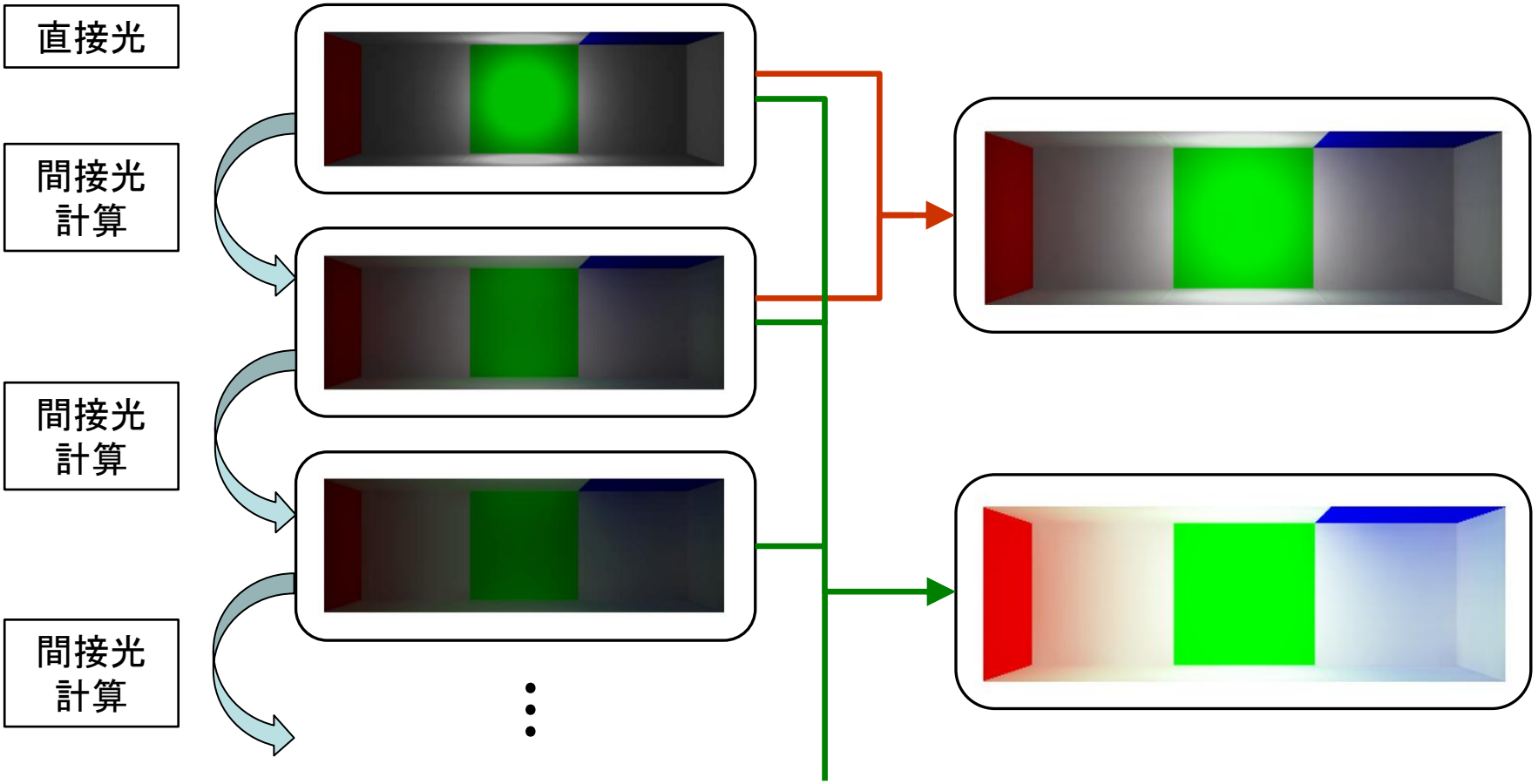
Step3. 手法の拡張

- 多重反射への拡張
- モジュール導入による更なる効率化

Step4. デモ及び考察

多重反射への拡張

- 間接光計算を反復実行



シーン分割による高速化の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

Step1. 問題提起

Step2. 小領域間の光の伝播の計算

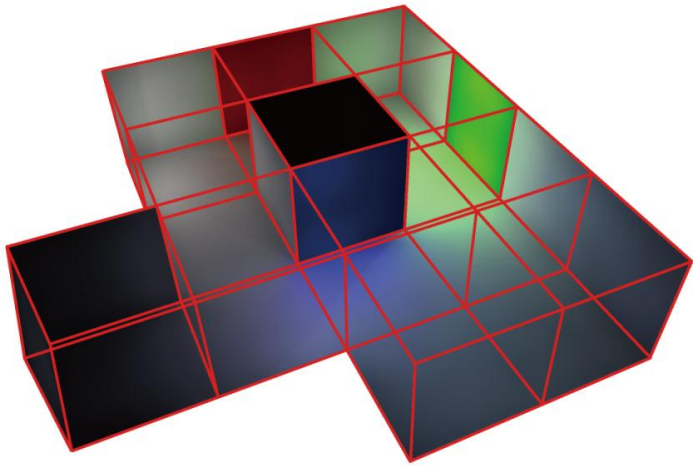
Step3. 手法の拡張

- 多重反射への拡張
- モジュール導入による更なる効率化

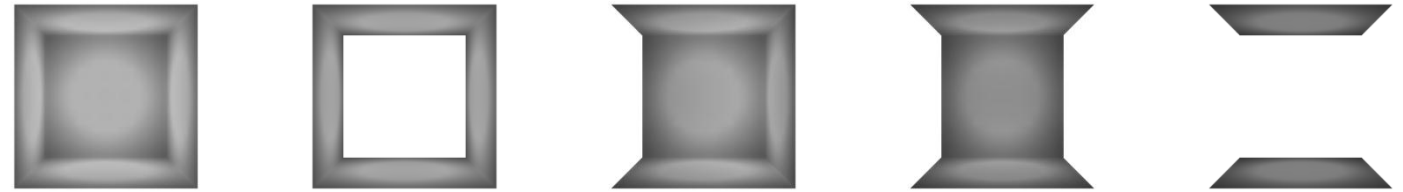
Step4. デモ及び考察

モジュールの導入（1/3）

- モジュール(=再利用可能な汎用小領域)の導入

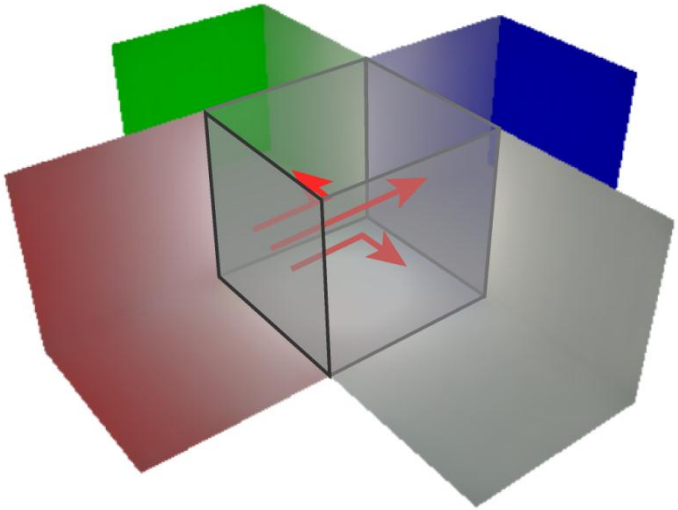


- 上記のようなシーンは「回転」も考慮すれば
  5種の立方体モジュールの組合わせで構成可能

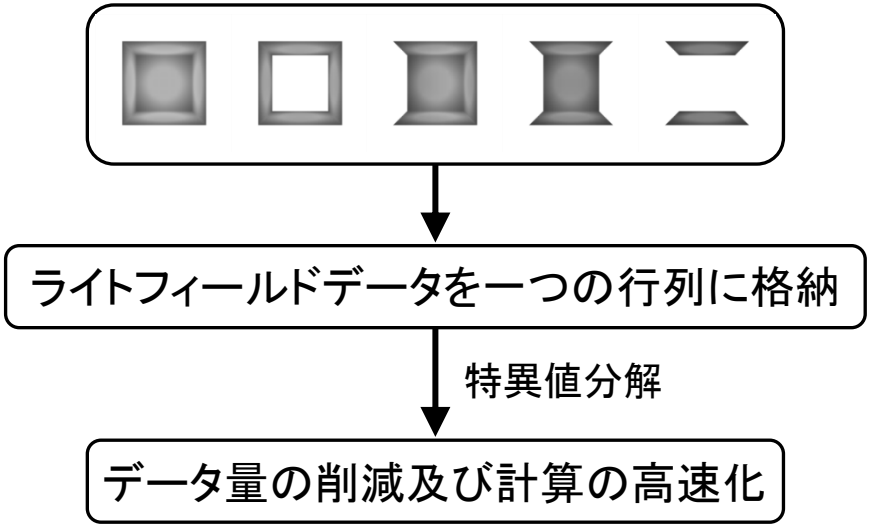


モジュールの導入（2/3）

- 立方体モジュールのメリット
  1. データ量及び前計算時間の削減
  2. 再利用可能な光の通過行列が導入可能
  3. ライトフィールド用の基底が導入可能



通過行列は（直進/右折/左折）の三種類



詳しくは論文5.1節参照

モジュールの導入（3/3）

- 立方体モジュールのメリット
  1. データ量及び前計算時間の削減
  2. 再利用可能な光の通過行列が導入可能
  3. ライトフィールド用の基底が導入可能

メリットは多いがシーンの自由度とのトレードオフ

モジュールの使い所は要検討

シーン分割による高速化の解説手順

CEDEC2012 Computer Entertainment Developers Conference

Silicon Studio

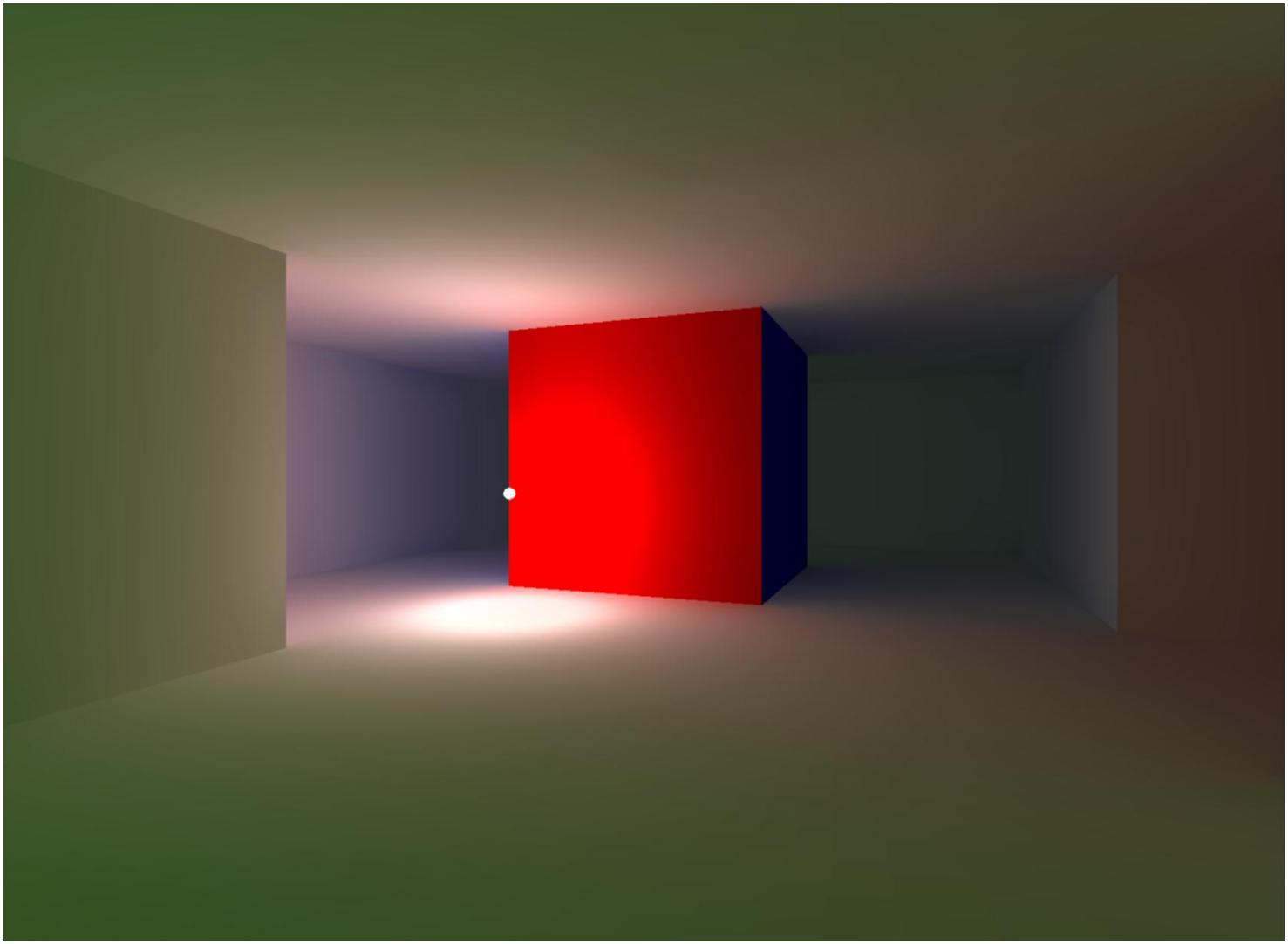
Step1. 問題提起

Step2. 小領域間の光の伝播の計算

Step3. 手法の拡張

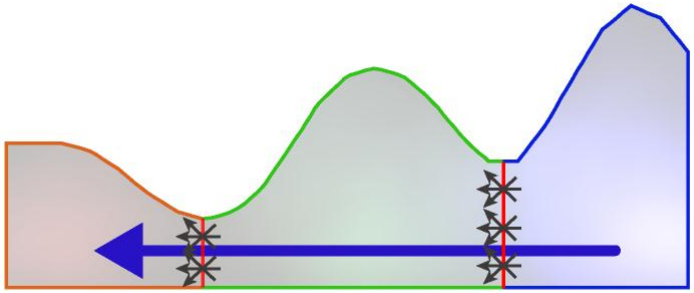
Step4. デモ及び考察

デモ



考察

- ライトフィールドは本当に必要か？



$$P = P_{in} \times P_{through} \times P_{out}$$

- 小領域間の位置関係が固定ならば（恐らく）不要

  ⇒ レイトレーシング法で伝播行列Pを直接計算した方が効率的

- 小領域を動的に入れ替える場合は必要

発表の流れ

1. 発表の概要
2. 技術紹介 + デモ
3. まとめ

発表内容のまとめ

- Modular Radiance Transfer法を解説
  - 長所
    - とにかく高速
    - 多重反射を含んだ間接光が扱える
    - 技術のコア部分は汎用的で応用の余地が高い
  - 短所
    - 前計算を含め、使い方に癖がある
      (実験を通してノウハウを蓄積すれば余り問題にならない?)

著作権について

本スライド内で引用した論文の図の
著作権は全て論文著者に帰属します

その他の図・スライド本体・挿絵等の著作権は
シリコンスタジオ株式会社に帰属します

最後に

CEDEC 2012 Computer Entertainment Developers Conference

Silicon Studio

御清聴ありがとうございました

スライド・デモは弊社HPで近日公開

Silicon Studio

http://www.siliconstudio.co.jp/presentations/index.html

問い合わせ先: 田村 nk-tamura@siliconstudio.co.jp
安田 re-yasuda@siliconstudio.co.jp

シリコンスタジオは常に人材を求めています
ご興味ある方は弊社HPをご覧ください

制作協力: 河野 駿介・辻 俊晶・古家 亙・後藤 晋也・京田 文人・三谷 健一郎

Appendix A. 実装について

- 実装自体は様々なアプローチが可能
- ここではデモの実装でのアプローチを紹介

Appendix A. 前計算について

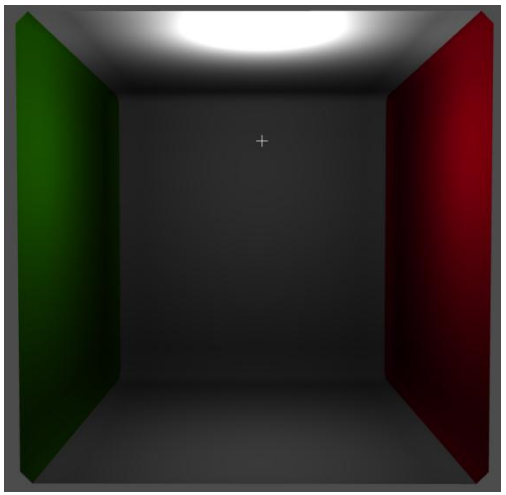
- Fの計算
  - メッシュの全頂点からキューブマップにレンダリング
  - ピクセルには頂点IDとウェイトを出力

- 直接光サンプルの計算
  - 頂点カラーをStream Outでバッファに出力
  - 頂点シャドウあり

- Light Probeの計算
  - 各Probeの中心からキューブマップにレンダリング
  - B_dとB_{ind}の対応する基底ベクトル同士を足し合わせたベクトル(直接光+間接光)を頂点カラーとして使用

Appendix A. ランタイム: l_d

- 直接光の結果をStream Outでバッファに出力
  - 頂点シャドウあり
  - カラーバッファへの書き込みOFF
  - 複数オブジェクトの場合は単一バッファにオフセットを変えて出力

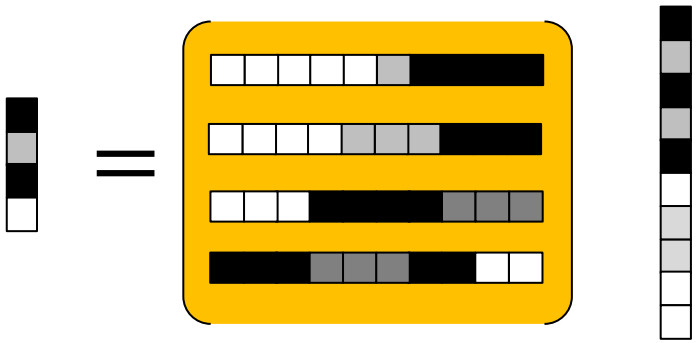


l_d

Appendix A. ランタイム: 係数$\boldsymbol{w}$

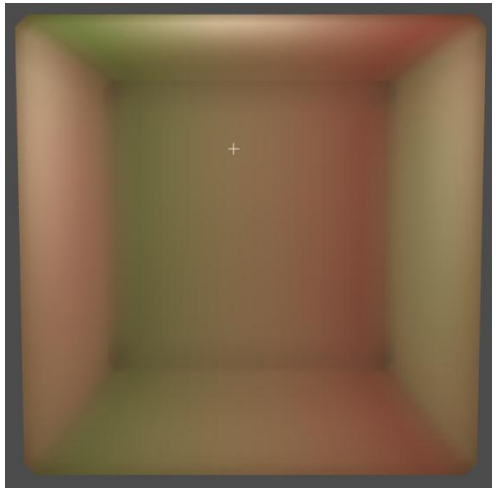
- Compute Shaderで$\boldsymbol{w} = B_d^{-1} l_d$, を計算
  - B_d^{-1}は極めて横長の行列(行: ~100程度, 列:頂点数)
  - $B_d^{-1} * l_d$はGPUが少々苦手な処理ではあるもののタイリングベースの実装で高速に処理可能

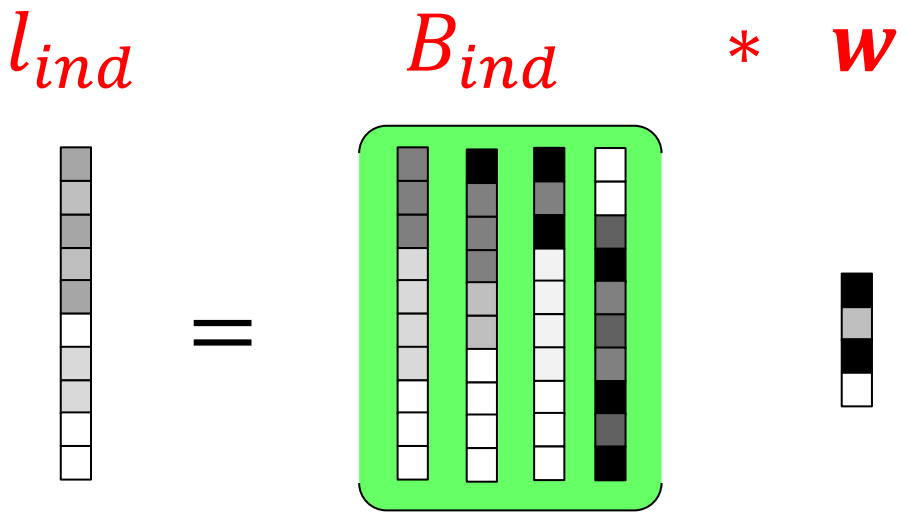
$$\boldsymbol{w} \qquad B_d^{-1} \quad * \quad l_d$$



Appendix A. ランタイム: l_{ind}

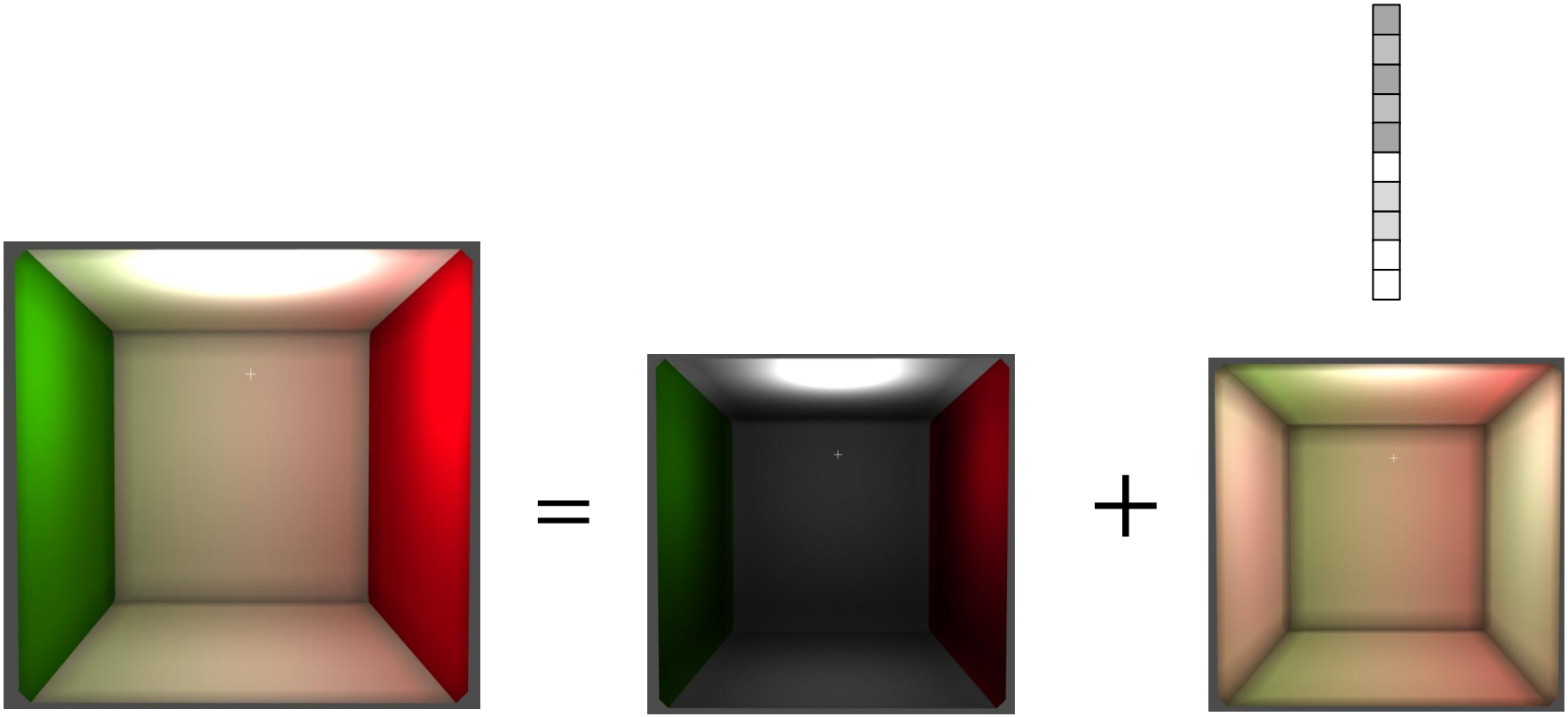
- Compute Shaderで $l_{ind} = B_{ind}\boldsymbol{w}$ を計算
  - B_{ind}は極めて縦長(B_dの転置と同サイズ)なので素直に実装可能





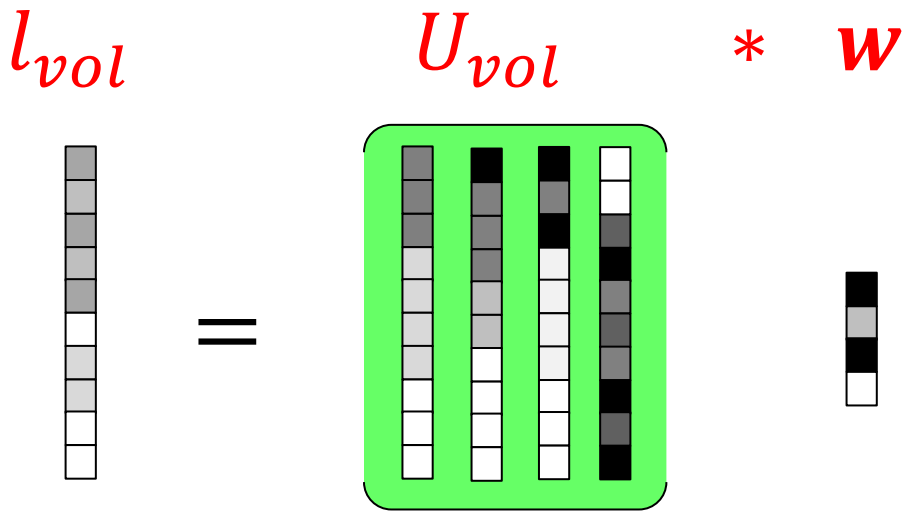
Appendix A. ランタイム: 最終パス

- l_{ind}を頂点カラーとして入力として、ピクセル単位の直接光のシェーディングと合成して最終結果を得る



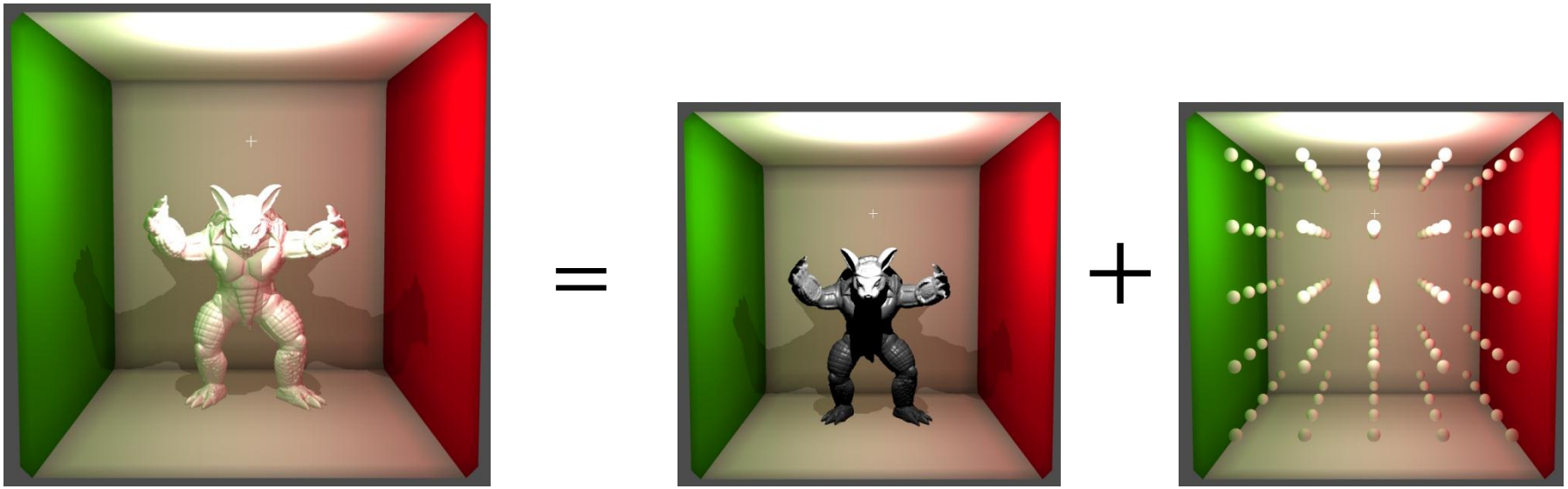
Appendix A. ランタイム: Light Probe(1)

- Compute Shaderで$U_{vol} * \boldsymbol{w}$を計算
  - U_{vol} は縦長(行:頂点数×SH係数の数, 列:~100程度)



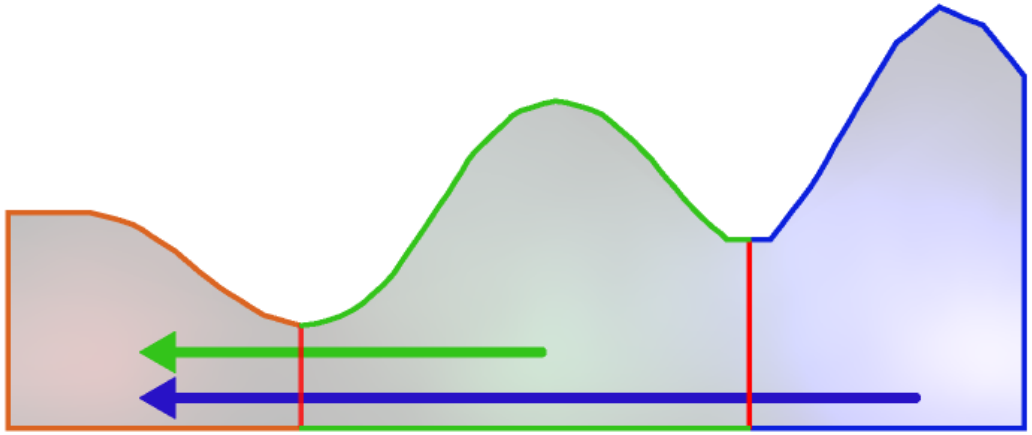
Appendix A. ランタイム: Light Probe(2)

- Probeを用いてDynamic Objectsをライティング
  - 各Light ProbeがSHを9係数で保持するため、3D TextureではなくStructured-Bufferで保持
  - Vertex Shader内で自力Tri-Linear補間



Appendix B. 後半の補足

- 大規模なシーンでの光の伝播
- 小領域間の光の伝播は個別に総当たりで計算

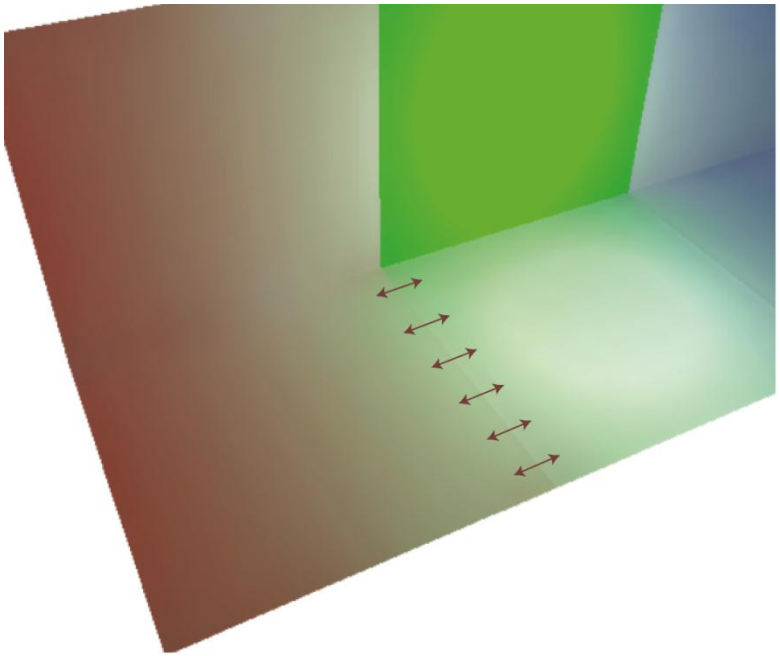


⇒小領域間の光の伝播は距離によって大きく減衰

実際は大規模シーンでも概ね二つ隣の小領域まで考慮すれば十分

Appendix B. 後半の補足

- 小領域の境界



- 小領域の境界部分が不連続になる

  ⇒ 小領域の境界部分は補間の必要がある

Appendix C. 記号の表記

- スライドと論文で表記が違う記号があります

スライドでの表記	論文での表記
B_d	L_{imp}
B_{ind}	U_b
B_d^{-1}	$T_{d \to b}$
$\boldsymbol{w}$	b