Graphics with Processing

2014-13 レンダリング技術

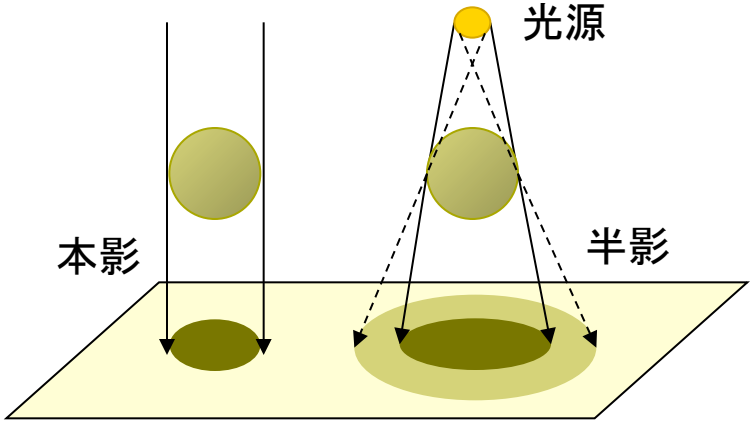
http://vilab.org

塩澤秀和

13.1 影付け

影の種類(p.136)

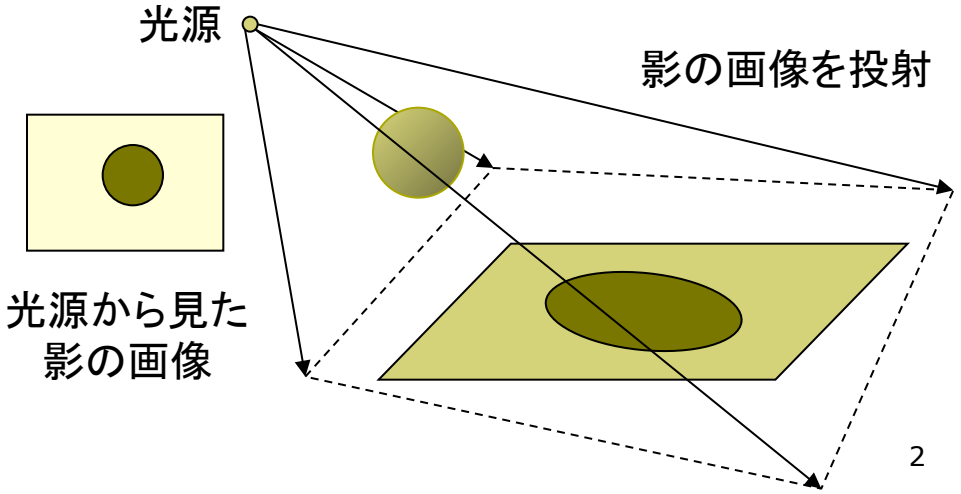
- 本影と半影
  - 点光源や平行光ではくっきりした影(本影)だけができる
  - 光源に広がりがあると，半影を含むソフトシャドウができる



  - 光源が複数ある場合，それぞれの光(影)を重ね合せればよい
  - リアルタイムな影生成では基本的に本影部分を扱う

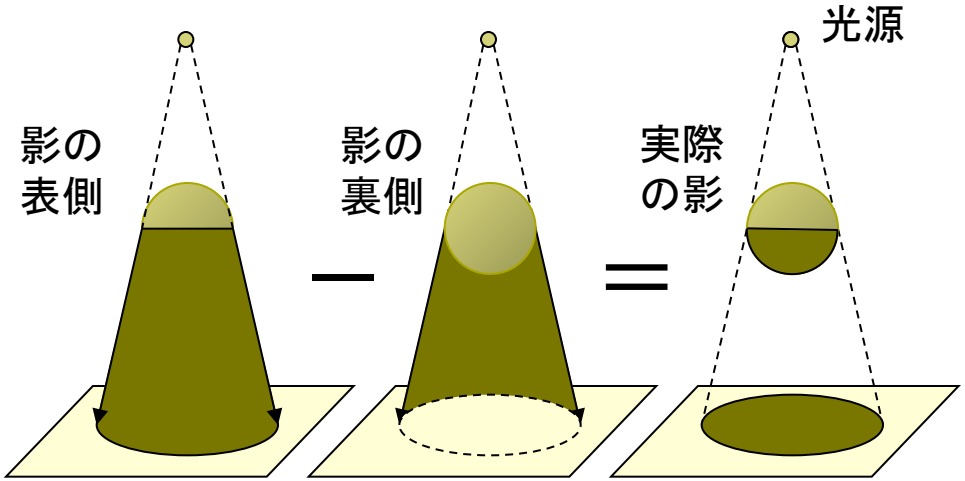
主な影付け方式

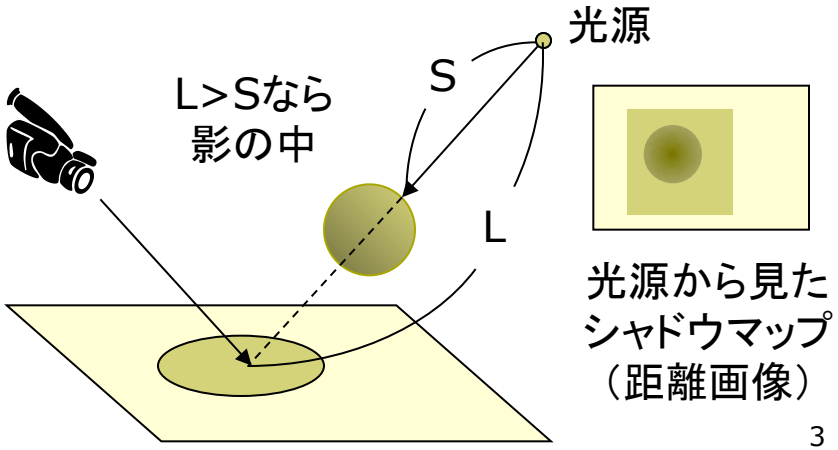
- 影の投影マッピング(p.137)
  - (物体空間における2段階法)
  - いったん視点を光源に置き，物体のシルエットを描画すると，光源から見たその物体の影になる
  - 視点は戻して，影の画像を光源の位置から物体の下の地面などに投影テクスチャマッピングする



13.2 影付け（続き）

- シャドウボリューム法（p.138）
  - 物体が光をさえぎってできる影の空間（シャドウボリューム）を囲う“影ポリゴン”を算出する
  - 視点から見て表を向いている影ポリゴンの像から，裏を向いている影ポリゴンの像を引くと，視点から見た影の形が分かる
  - 「ステンシルバッファ」を用いると，高速に実現できる
- シャドウマップ法（p.139）
  - （Zバッファを用いた2段階法）
  - 光源から見た場合のZバッファを構成すると，光の到達距離Sのマップ（シャドウマップ）ができる
  - 視点を戻し，レンダリングするオブジェクトから光源までの距離Lとシャドウマップ上の対応点の内容（S）を比較し，光がそこまで届いているか判定する





13.3 高品質なレンダリング

目的別レンダリング

- リアルタイムレンダリング
  - 3Dゲーム ← ユーザが操作
  - 毎秒10コマ以上の速度が必要
- 高品質レンダリング
  - 静止画，映画 ← 事前に“撮影”
  - やわらかい陰影やガラスの表現

⇒ レイトレーシング法＋大域照明

大域照明モデル
（Global Illumination: GI）

- 間接光まで含む照明計算
  - 単純な環境光モデルではなく，間接光をより精密に計算する
  - 特に室内の陰影がより自然
  - ラジオシティ，フォトンマッピング

フリーソフトによるレンダリングの例

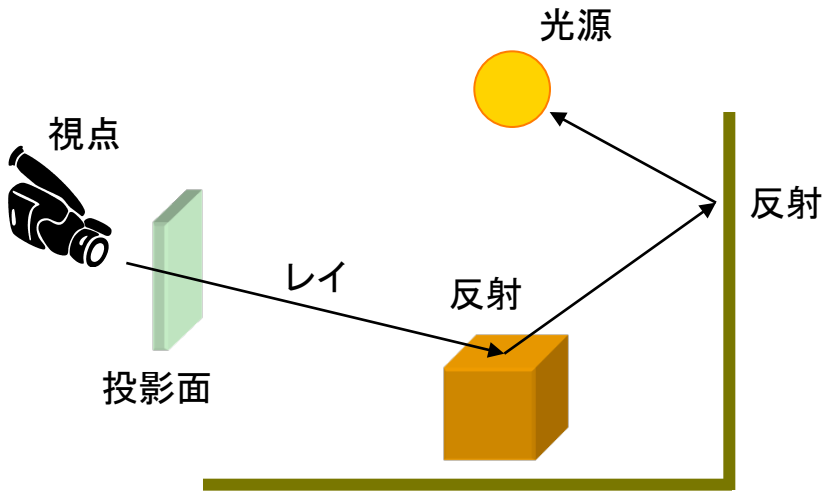
- POV-Ray
  http://www.povray.org
  → Hall of Fame
- Blender+Yafray
  http://www.blender.org
  → Feature & Gallery
  http://www.yafaray.org
  → Gallery
- Sunflow
  http://sunflow.sourceforge.net
  → Gallerly
- Art of Illusion
  http://www.artofillusion.org
  → Art Gallery

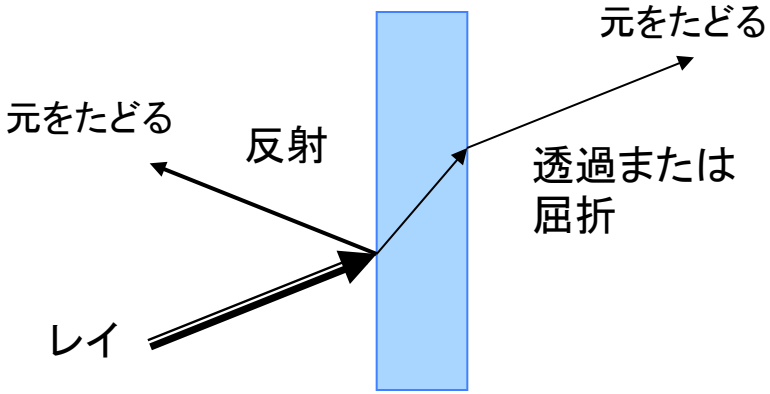
13.4 レイトレーシング (p.110)

レイトレーシング法

- 概要
  - Ray Tracing＝光線追跡
  - 各ピクセルに届く光の軌跡を，視点から光源にさかのぼるように追跡するレンダリング技術
- 高品質
  - 3DCGの初期からあるが，より正しく光学現象を再現するように研究され続けている
  - 原理的に隠面消去される
  - 透明，影，レンズも自然に表現
- 用途
  - リアルだが時間がかかるので，まだゲームなどには向かない
  - 映像作品（映画）製作で一般的

- 光線追跡の概念図



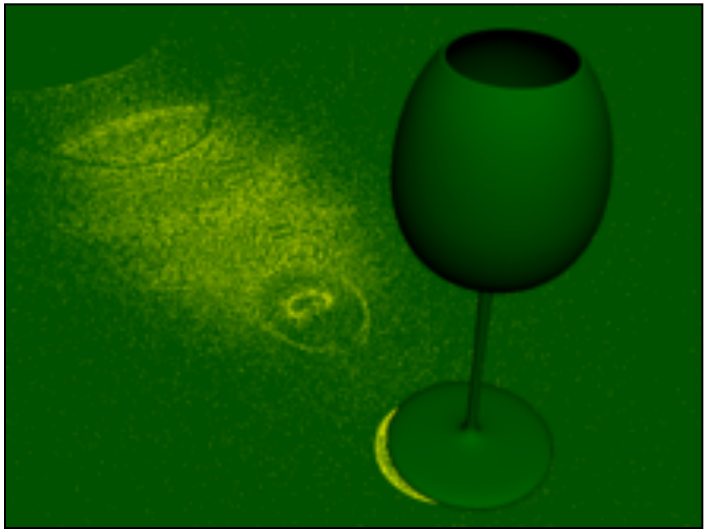


13.5 フォトンマッピング（p.142）

フォトン（Photon）マッピング

- 概要
  - 光源から出る大量の光子を考え，その軌跡をシミュレーションする
  - すると，シーン全体の光の分布（間接光）が概算できる
  - この間接光を環境光の代わりにして，レイトレーシングを行う
- 特徴
  - レンズなどの集光現象（コースティックス）が表現できる
  - 逆方向のレイトレーシングといえ，レイトレーシング法と相性が良い
  - 着想は簡単だが，アルゴリズムは複雑で膨大な時間がかかる

Wikipedia



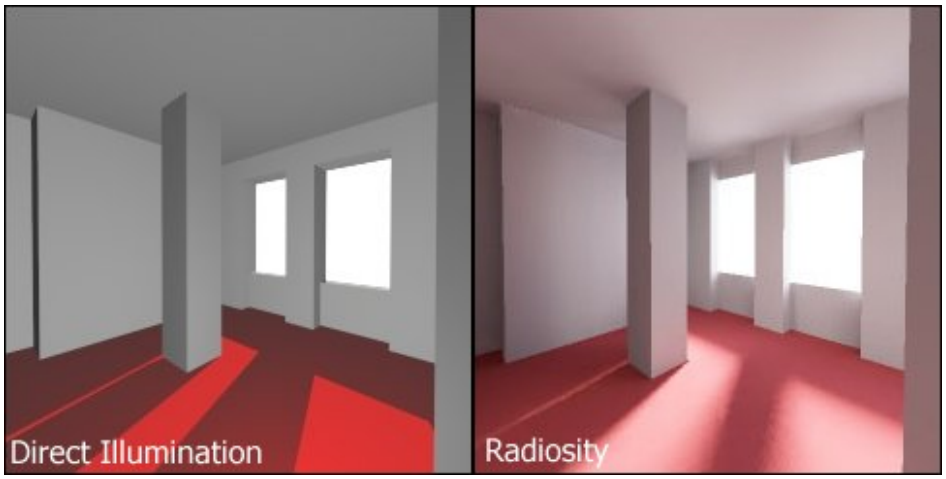
計算された光子の分布

13.6 ラジオシティ法（p.141）

ラジオシティ（Radiosity）法

- 概要
  - ポリゴンをパッチ（断片ポリゴン）に分割する
  - 2つのパッチの位置と向きの関係から，光の相互伝達率（フォームファクタ）を計算する
  - 全パッチ間での光エネルギーの放射発散の平衡状態を求める

Wikipedia



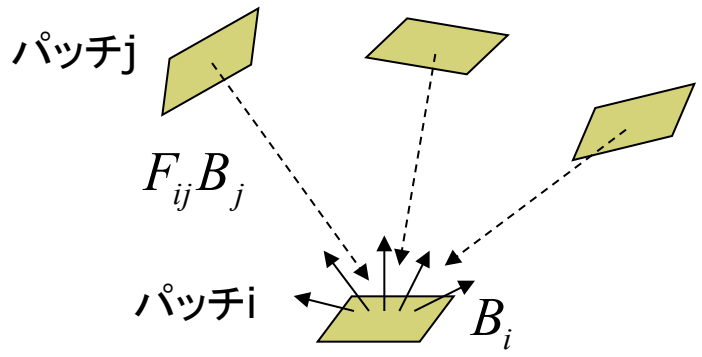
柔らかい影や壁の色の影響が表現されている

- ラジオシティ方程式（p.158）

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$

n	シーン全体のパッチ数
Bi	パッチiの光の放射量（ラジオシティ）
Ei	パッチiの発光量
ρi	パッチiの反射率
Fij	フォームファクタ（Fij=Fji）

- 本質的には「連立一次方程式」
  ⇒ ガウス・ザイデル法など



13.7 その他のレンダリング技術

ぼかし(ボケ)系

- CG画像の違和感
  - すべてがはっきりくっきりしすぎ
  - 現実感を出すために,「はっきり見えなくする」ことも必要
- アンチエイリアシング(p.210)
  - ドットのギザギザが目立たないように,輪郭を中間色でぼかす
- フォグ(霧)
  - 水蒸気やチリなどによる空気の「濁り」を再現する
  - 遠くにあるものがかすんでいき,色が落ちていく効果を与える
- 被写界深度(DOF)
  - レンズの効果を再現し,ピントが合っていないところをぼかす
- モーションブラー(p.167)
  - 速く動くものに見える残像(ボケ)をわざと表示する
  - 軌跡の画像を重ね合わせる

ノンフォトリアリスティック(非写実的)レンダリング(NPR)(p.249)

- 概要
  - 現実のマネではないレンダリング
  - 例) 油絵風,手書きタッチの再現,製図風,2次元アニメ,芸術作品
- 背景
  - 写実的(フォトリアリスティック)なCG技術はかなり完成
  - 漫画・アニメーションでの利用
  - 芸術などへのCG利用の広がり

13.8 演習

Processingでレイトレーシング

- joons-renderer
  - Processing から Sunflow を利用するライブラリ
  - https://github.com/joonhyublee/joons-renderer/
  - 最新版を入手して展開し，jonesrenderer フォルダを Processing\libraries の中にコピーしてインストール
  - 使い方は examples を参照
  - jr.background(0);は削除？
- 自由課題
  - 右のプログラムを変更して様々な図形をレンダリングしてみよ
  - 今回は提出はない

```
import joons.JoonsRenderer;
JoonsRenderer jr;

void setup() {
  size(800, 600, P3D);
  jr = new JoonsRenderer(this);
}

void draw() {
  jr.beginRecord();

  camera(0, 0, 120, 0, 0, -1, 0, 1, 0);
  perspective(PI/4, 4.0/3.0, 5, 100000);

  jr.background("cornell_box", 100, 100, 100);
  jr.background("gi_instant");

  jr.fill("diffuse", 255, 255, 255);
  translate(0,10,-10);
  rotateY(-PI/8); rotateX(-PI/8);
  box(20);

  jr.endRecord();
  jr.displayRendered(true);
}

void keyPressed() {
  if (key == 'r' || key == 'R') jr.render();
}
```

レンダリング結果を保存

Rキーでレンダリング開始