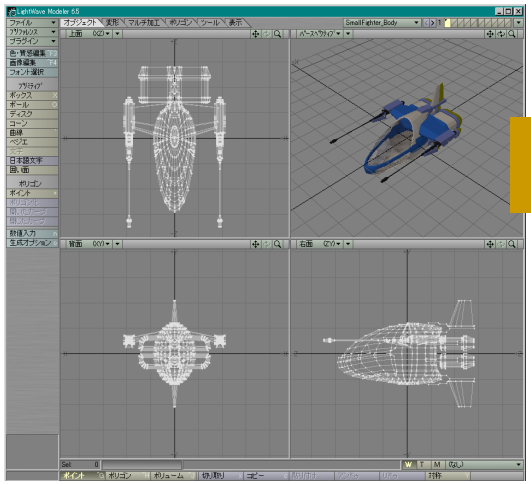
レンダリングとモデリング

メディアサイエンス基礎

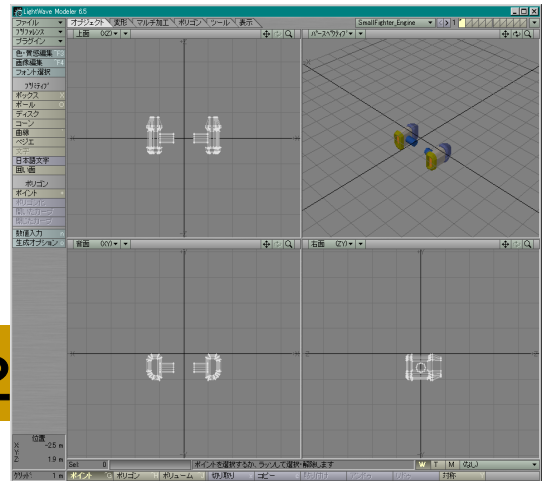
コンピュータグラフィックスの2つの要素

- モデリング
  - データを作成すること
- レンダリング
  - データから映像を生成すること

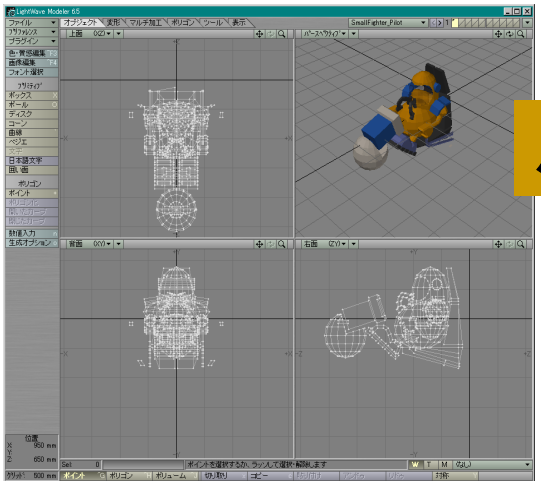
モデリング



パーツ1



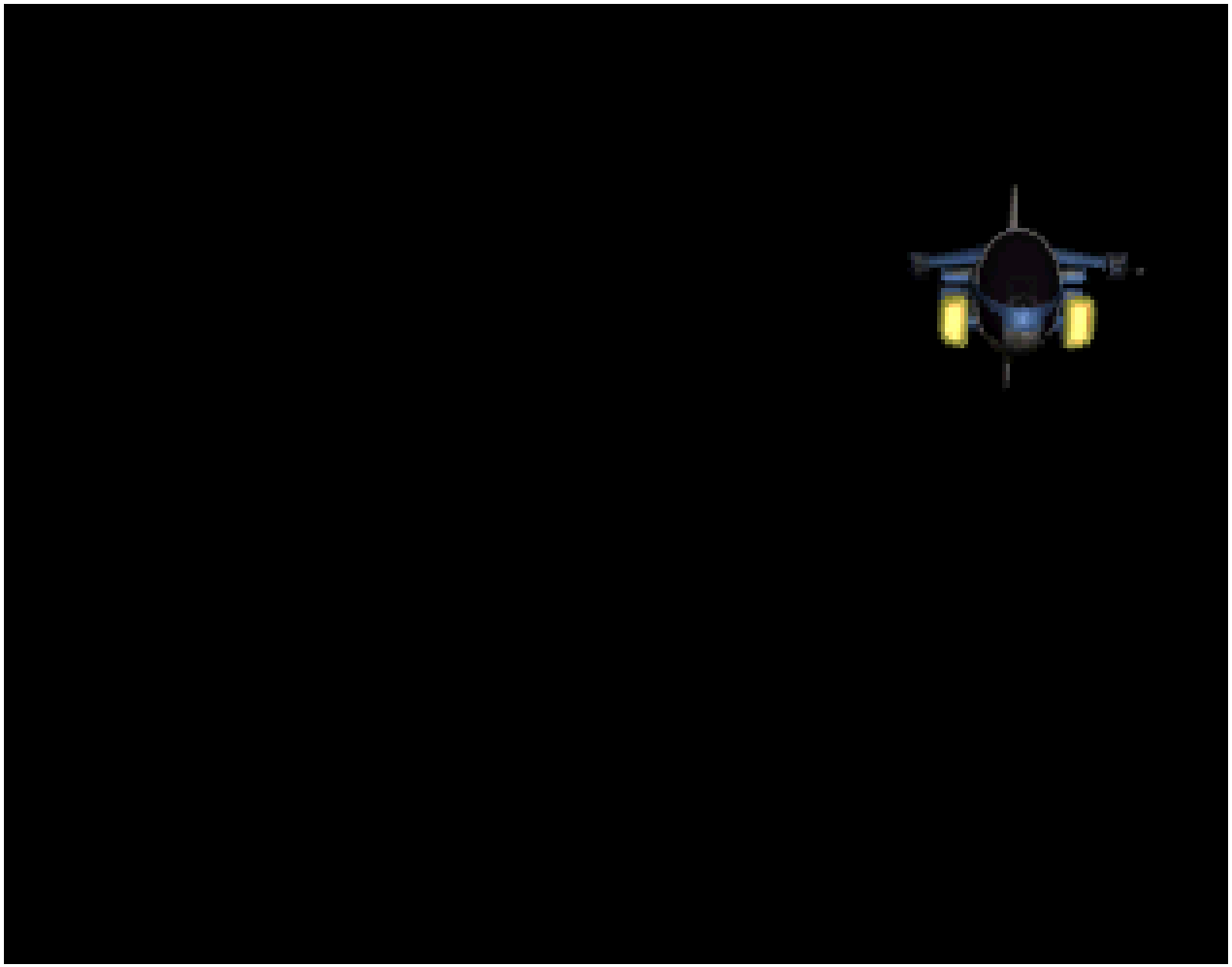
パーツ2



パーツ2

配置

レンダリング



レンダリングの2つの方向

- リアルな映像を作成する
  - 映画等
    - あらかじめ映像を生成しておく
      - プリレンダリング
    - 実写映像の置き換え
      - フォトリアリスティックレンダリング
- リアルタイムに映像を作成する
  - ゲーム等
    - 「その場」で映像を生成する
      - リアルタイムレンダリング

リアルな映像の作成

- 物に光を当てて初めて姿が見える
  - 照明効果は物理現象
    - 照明による陰影がリアルさの基本
- 形のリアルさ、動きのリアルさ
  - いずれも物理的なモデルから再現される
- 物理現象の厳密なシミュレーション
  - 非常に時間がかかる
    - 泊り込み
    - レンダリングファーム

リアルタイムに映像を作成

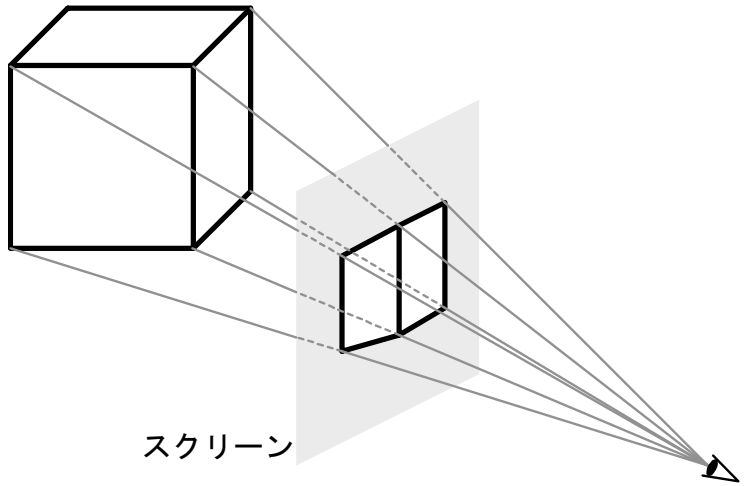
- 制限時間内に描画を完了する
  - 速度重視
    - 見かけのリアルさは2の次
      - ポリゴン（形のデータ）をできるだけ減らすなど
    - 厳密にシミュレーションを行わなくても、それらしく見えればいい
      - 陰影計算の手を抜くなど
    - テクスチャでごまかす
  - むしろ動きのリアルさが重視される
    - モーションキャプチャ

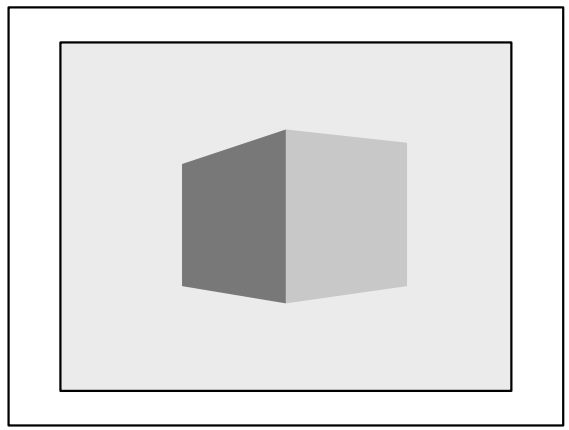
リアリティとリアルタイム性

- トレードオフが存在する
  - リアリティの追求
    - リアルタイム性の実現が難しくなる
  - リアルタイム性の追求
    - リアリティを損なう結果になり得る
- トレードオフの解消が求められている
  - ゲームのリアリズムの向上
    - 実写のような空間の再現
  - CGアニメーション映画のコストダウン
    - 制作にかかる時間はそのままコストに跳ね返る

陰影画像の生成

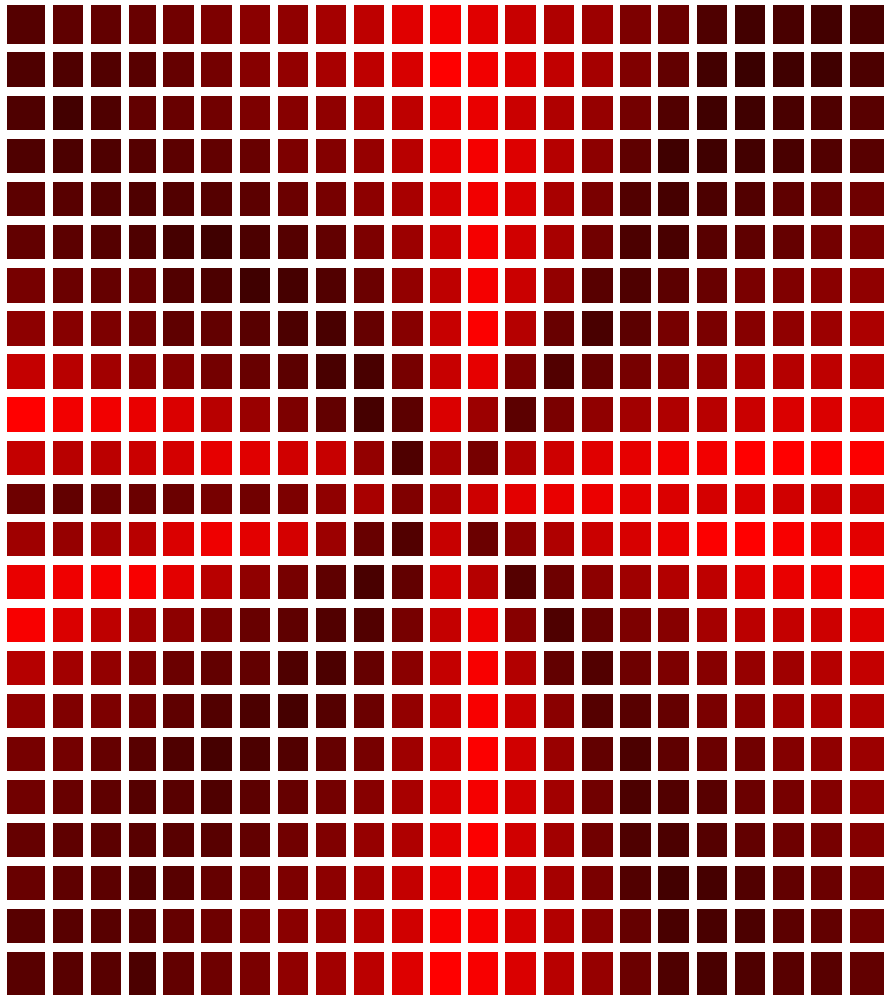
- ３次元形状がスクリーンに映る形（投影像）を求める
- その投影像の各部分のディスプレイ上での色を求める
- ディスプレイ全体について色を決定すれば目的の画像が得られる
  - 得られる画像はデジタル画像





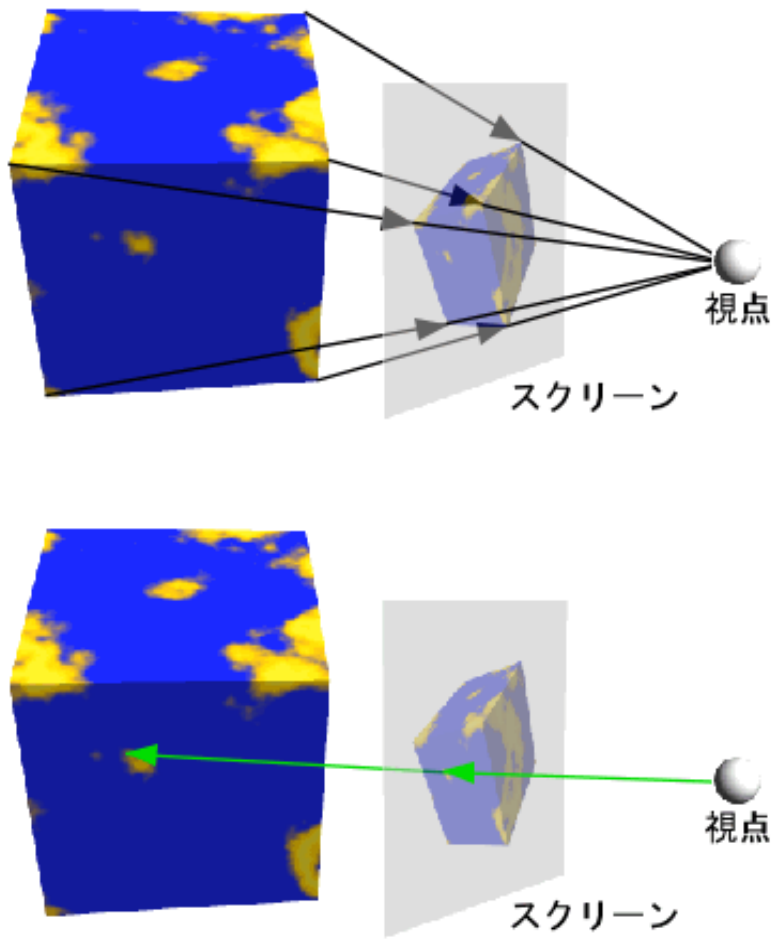
デジタル画像は光の点の集合

- 一つ一つの点の色を決めることで，図形や画像，映像を表現する
- この点のことを画素(pixel)と言う
- CGの制作は，この一つ一つの画素の色を決める作業



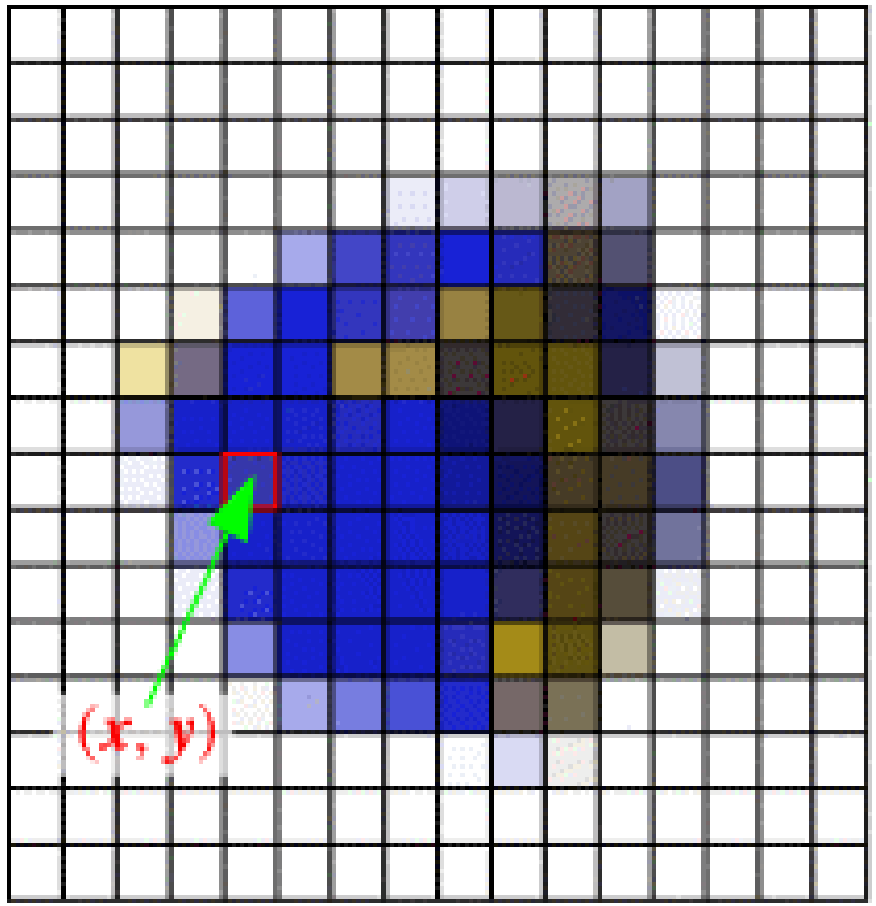
立体図形の投影像を求める方法

- 立体図形の各部分（頂点など）がスクリーン上のどの位置に表示されるか計算する方法
  - リアルタイムレンダリングなどで用いられる
- 視点からスクリーン上のひとつの画素を見て，そこを通して何が見えるか調べる方法
  - レイトレーシング



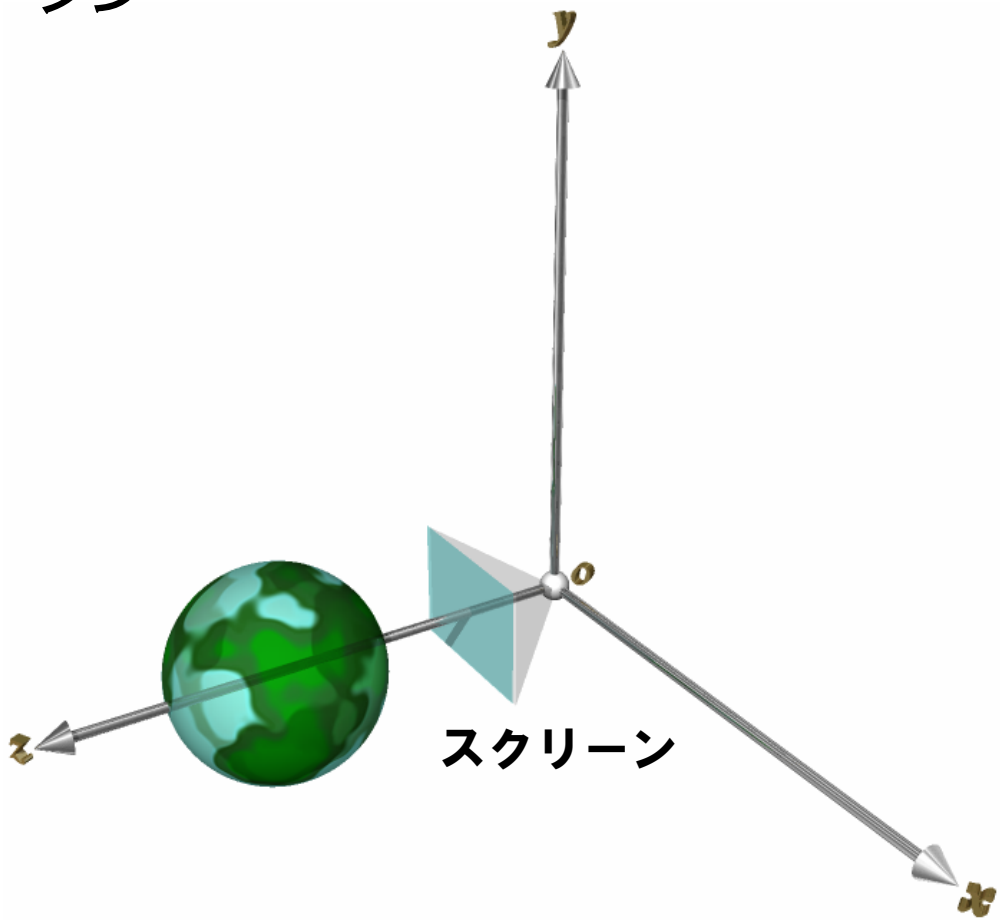
レイトレーシングの考え方

- その画素のところに何が見えるのかわかれば，画素の色が決定できる



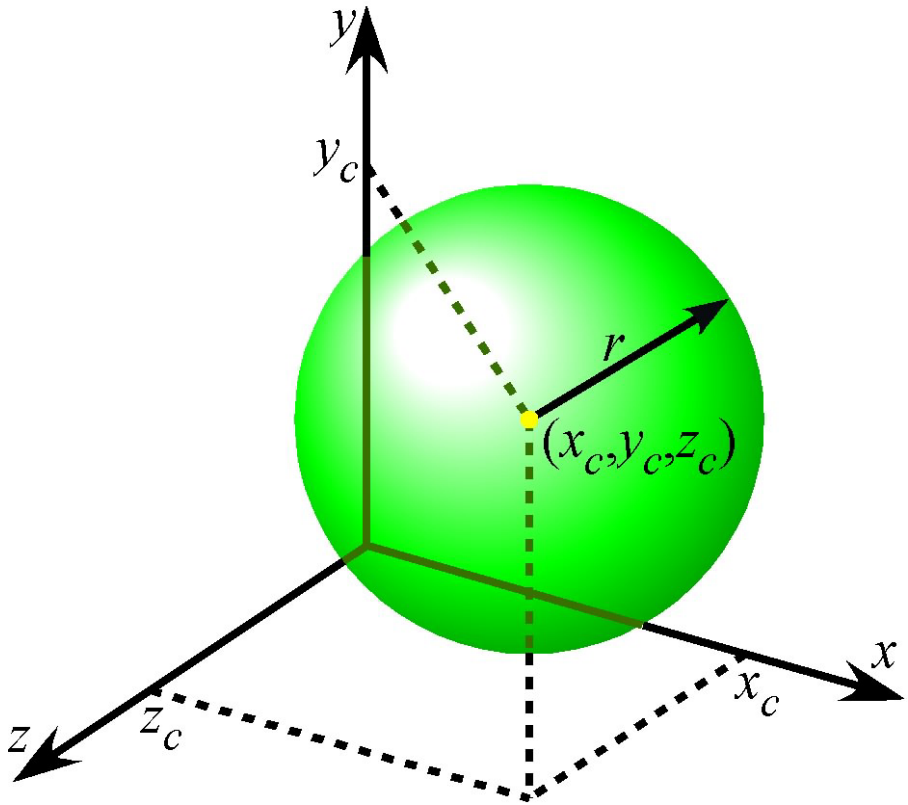
球のレイトレーシングによる表示

- 空間中の球のスクリーンへの投影像を，レイトレーシングにより作成する



球の方程式

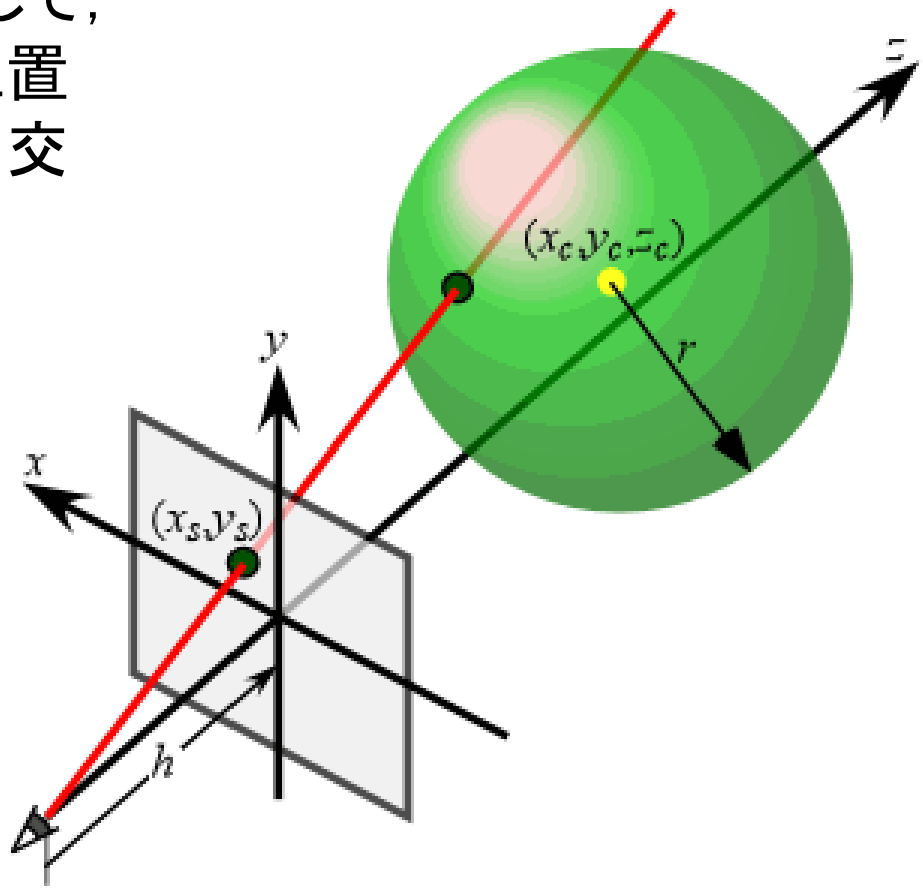
- 中心
  - (x_c, y_c, z_c)
- 半径
  - r



$$(x-x_c)^2+(y-y_c)^2+(z-z_c)^2=r^2$$

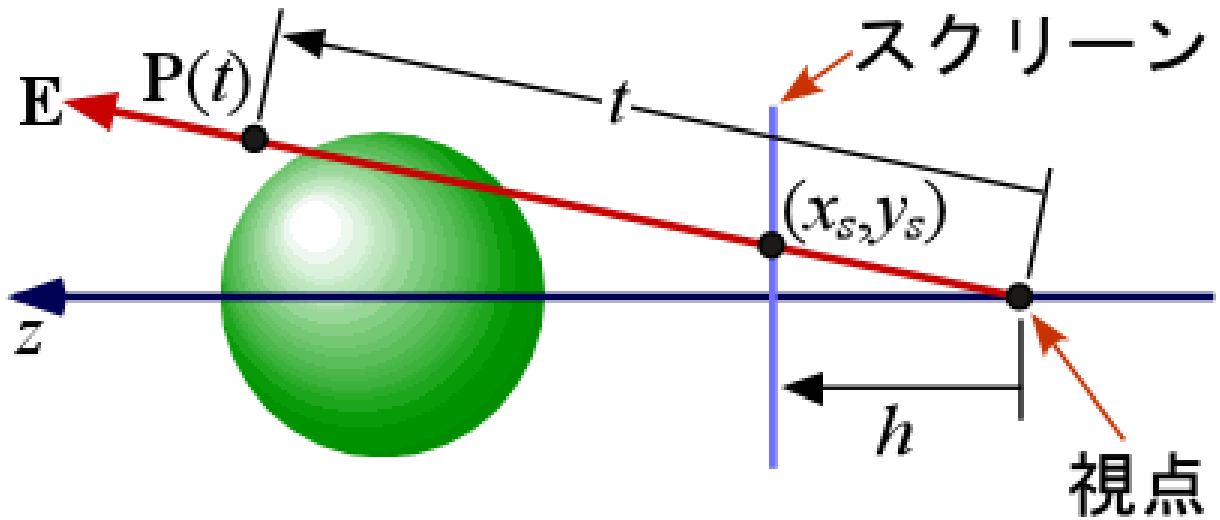
視線と球の交点を求める

- この球と，視点から出発して，色を決定したい画素の位置を通る半直線＝視線との交点を求める



視線の方程式

- 原点から出発し，そこから h の位置にあるスクリーン上の1点 (x_s, y_s) を通る半直線の方程式を媒介変数 t で表す



$$x = x_s t$$

$$y = y_s t$$

$$z = ht$$

視線を球の方程式に代入

$$At^2 - 2Bt + C = 0$$

$$A = x_s{}^2 + y_s{}^2 + h^2$$

$$B = x_s x_c + y_s y_c + h z_c$$

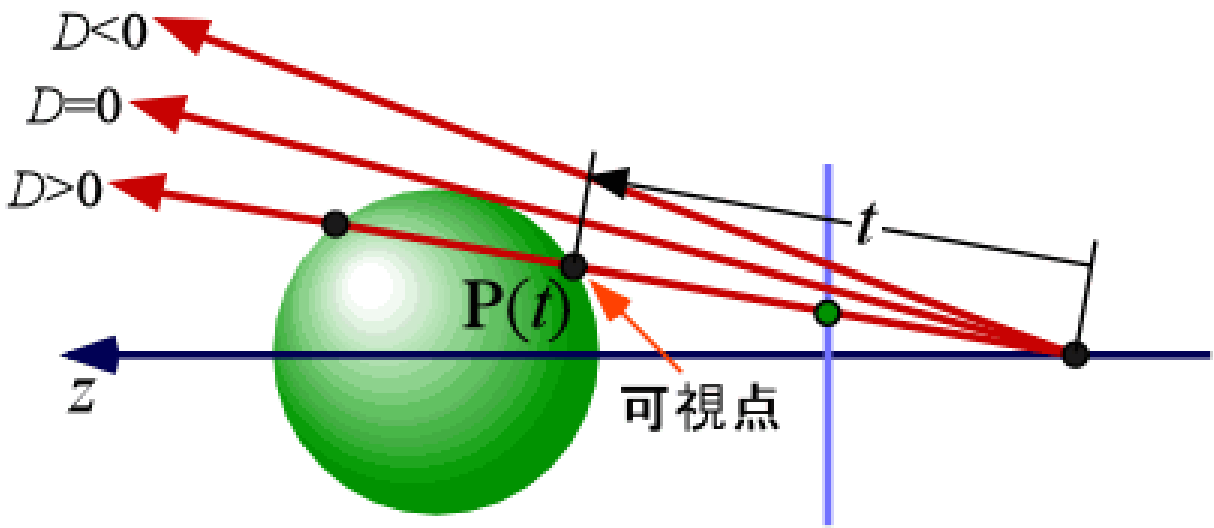
$$C = x_c{}^2 + y_c{}^2 + z_c{}^2 - r^2$$

（x,y,z を消去）

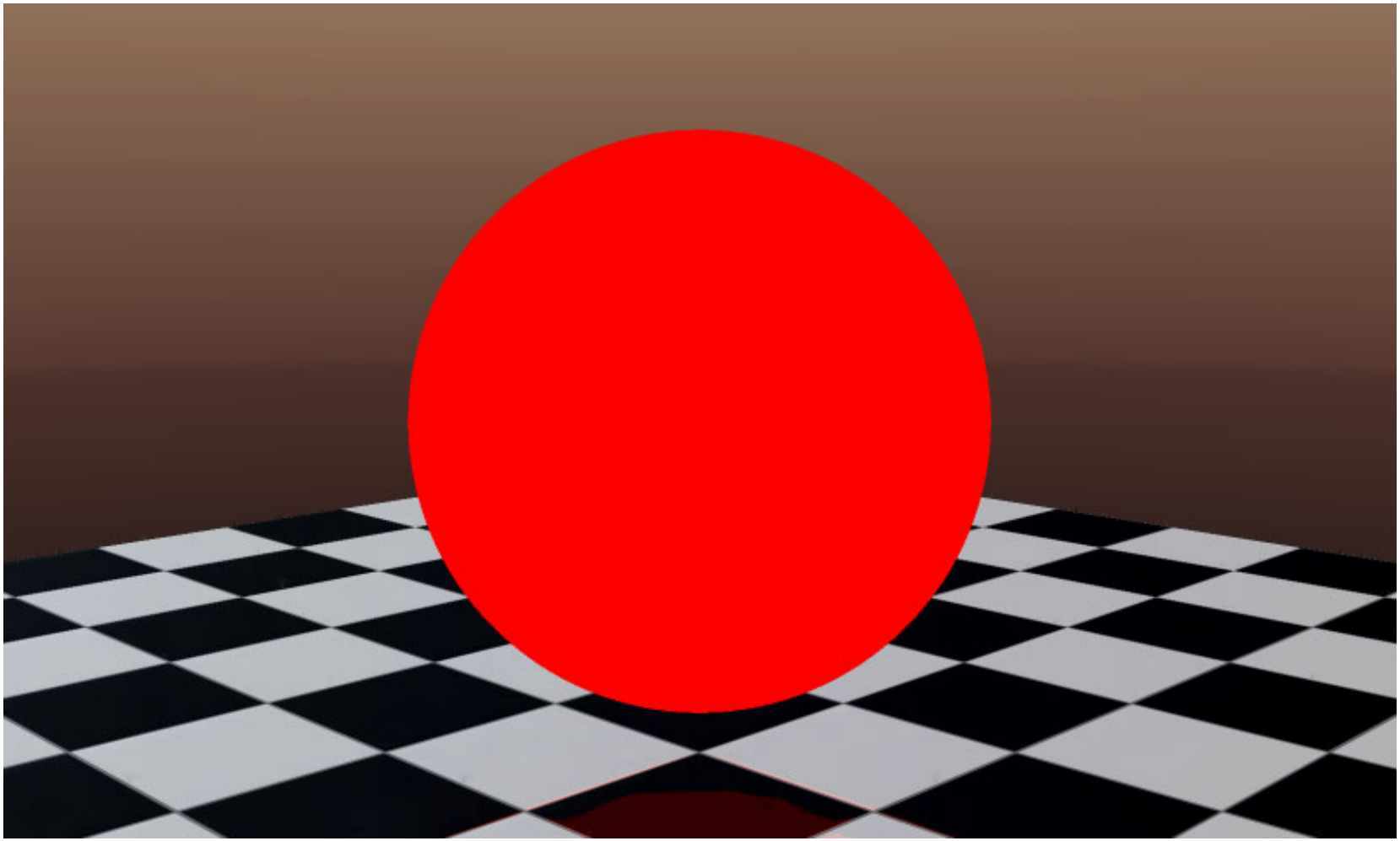
交差の検出

$$At^2 - 2Bt + C = 0$$

判別式 $D = B^2 - AC$

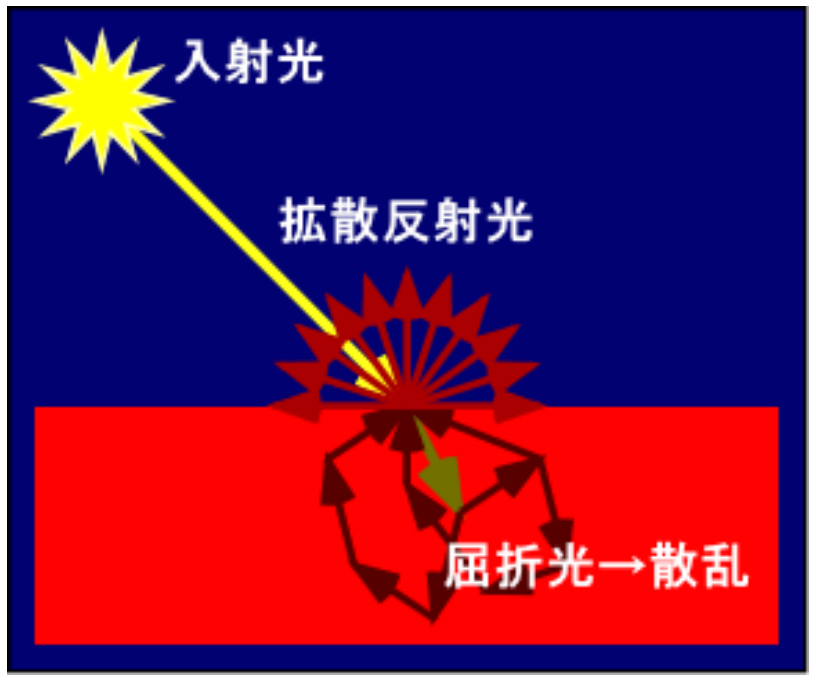


球と交差した画素を赤にしてみる

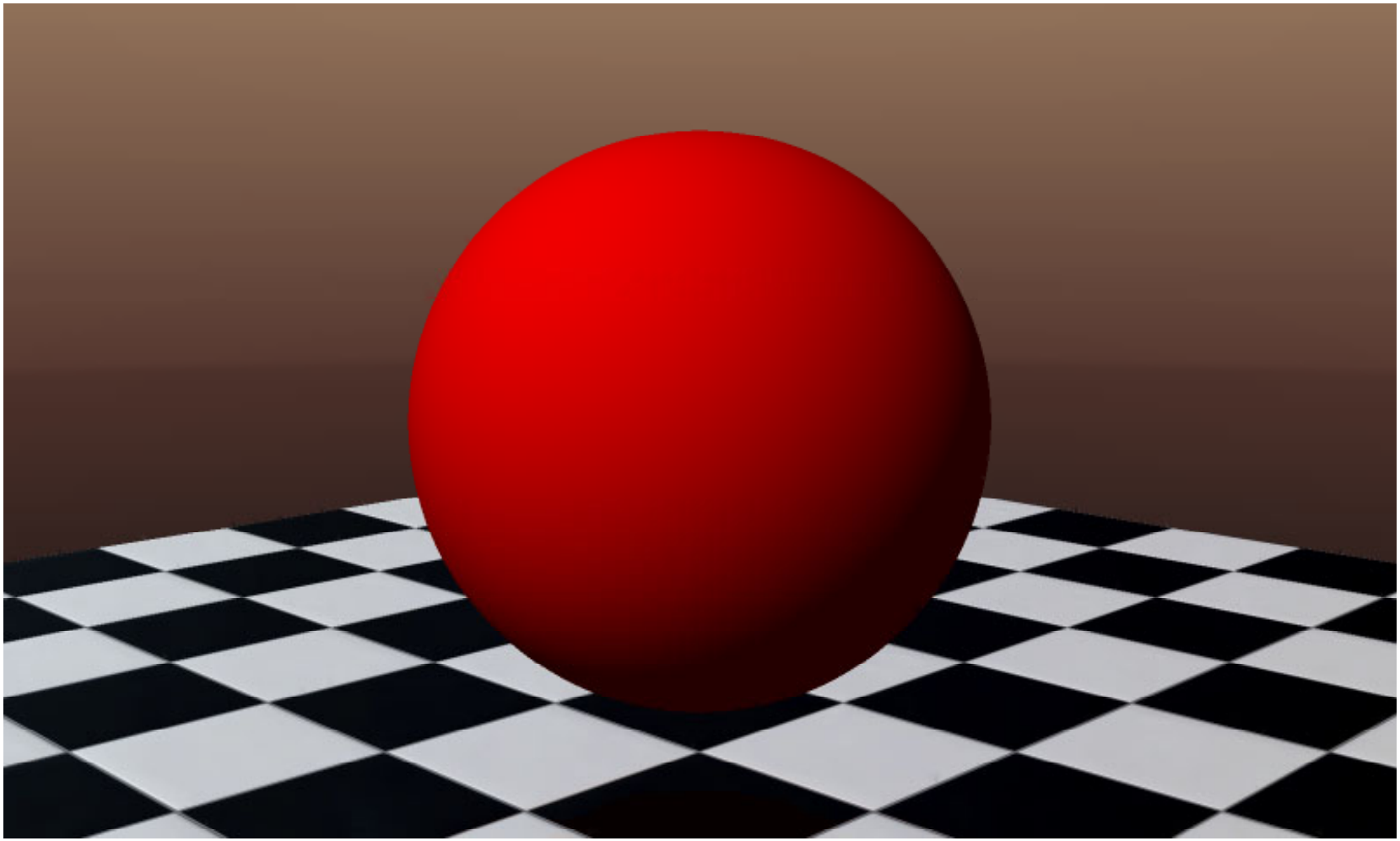


拡散反射光

- 完全拡散反射面
  - 全方向に対して均一に光を反射する
- 反射光強度
  - 入射光強度に比例
- 入射光強度
  - 入射光密度に比例

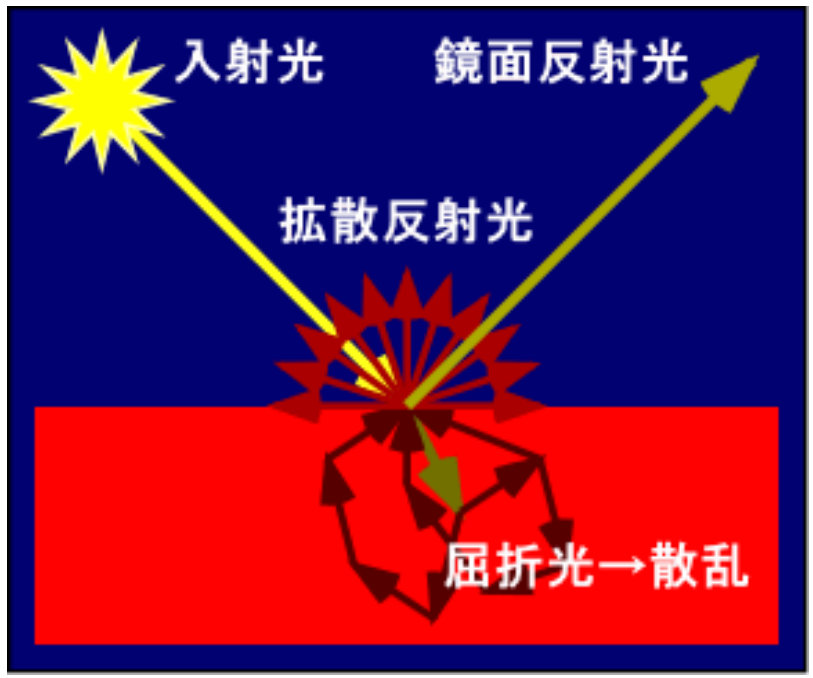


拡散反射光による陰影

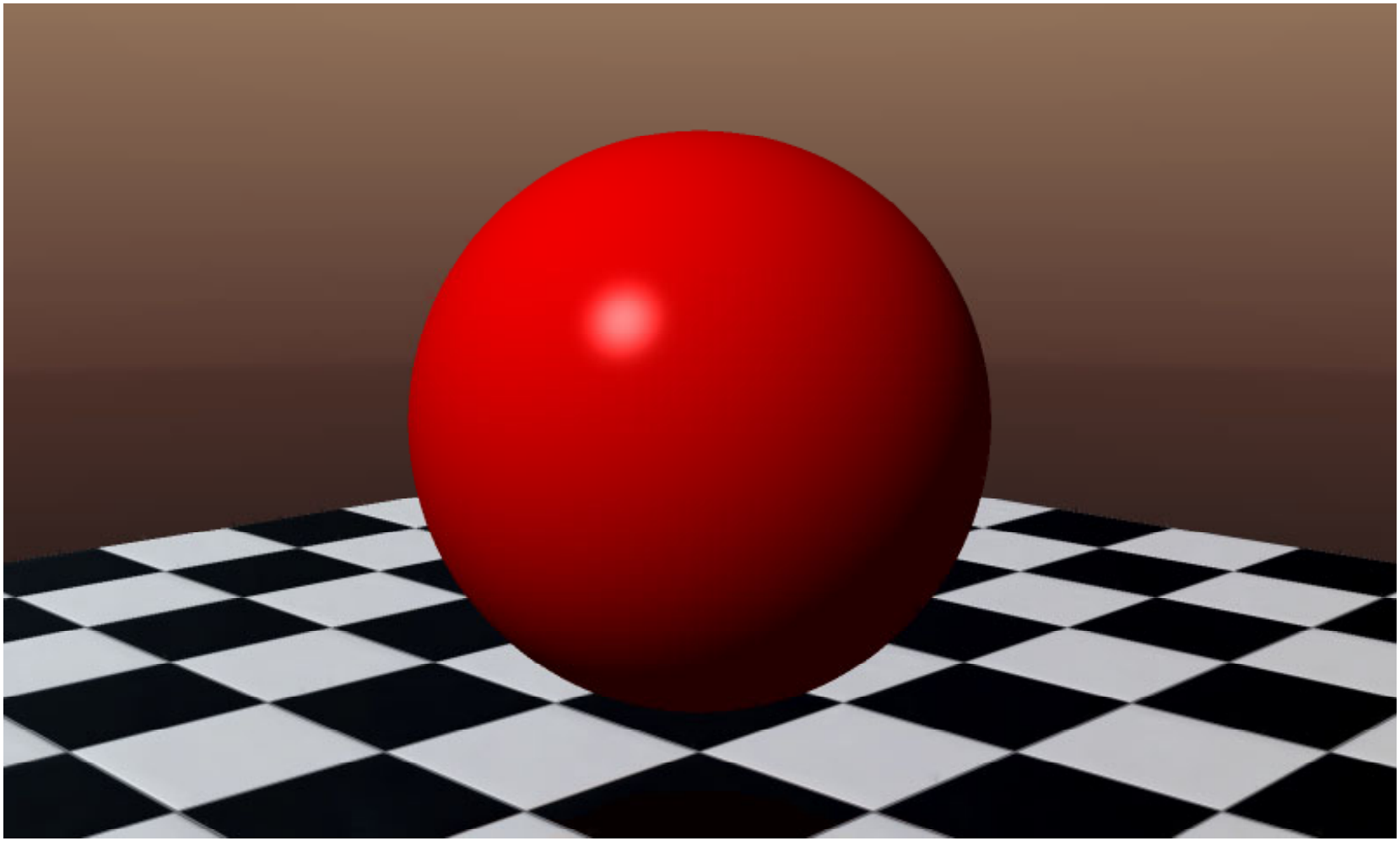


鏡面反射光

- 光源の映り込み
  - 入射光の正反射光
  - ハイライト

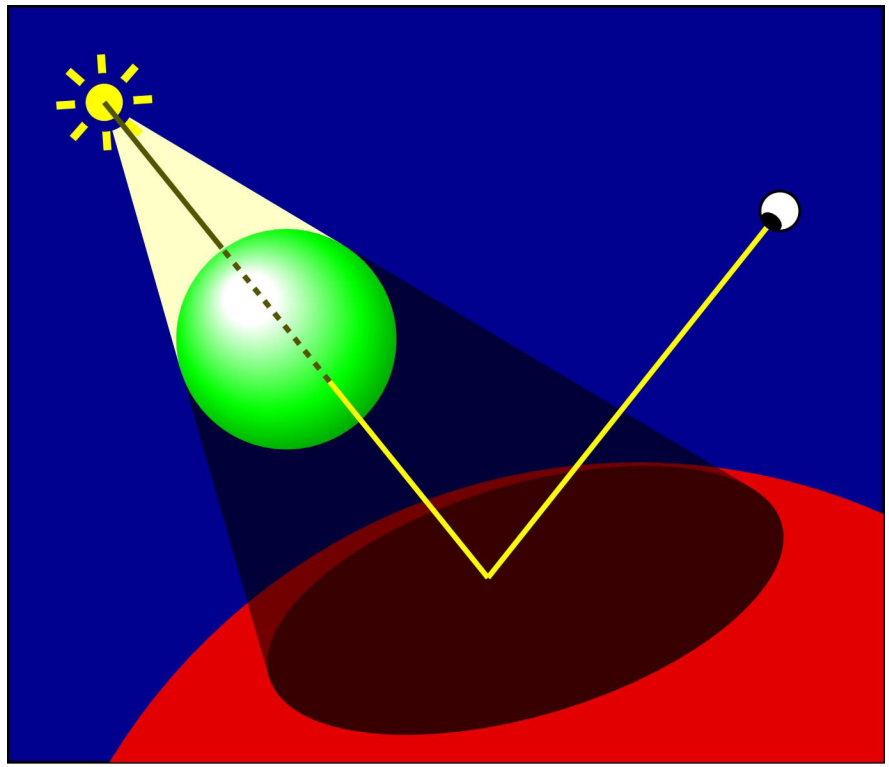


鏡面反射光によるハイライト

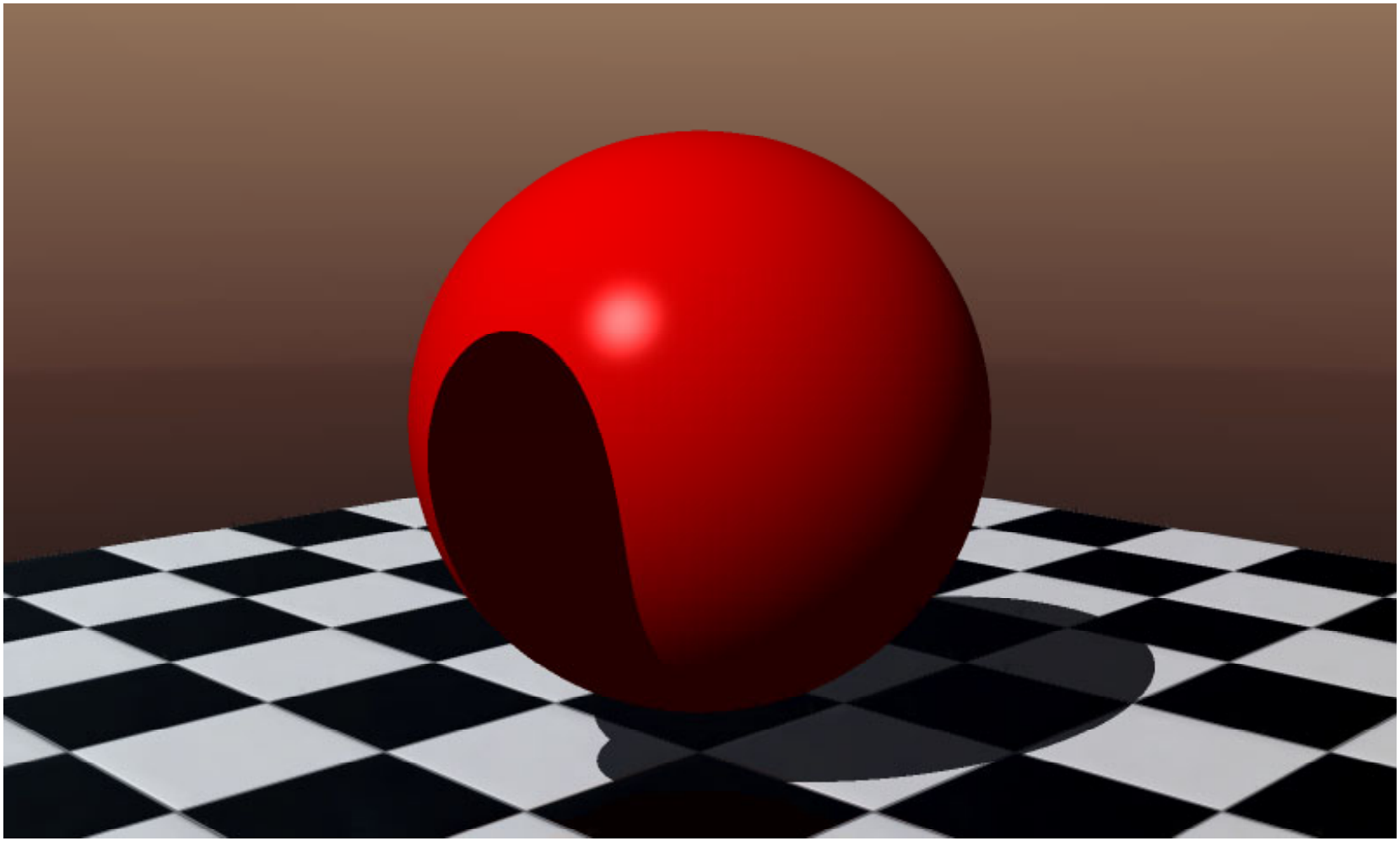


光源と物体の間にある遮蔽物

- 可視点と光源の間に別の物体があるかどうか調べる
- 別の物体がひとつでもあれば，その可視点は影

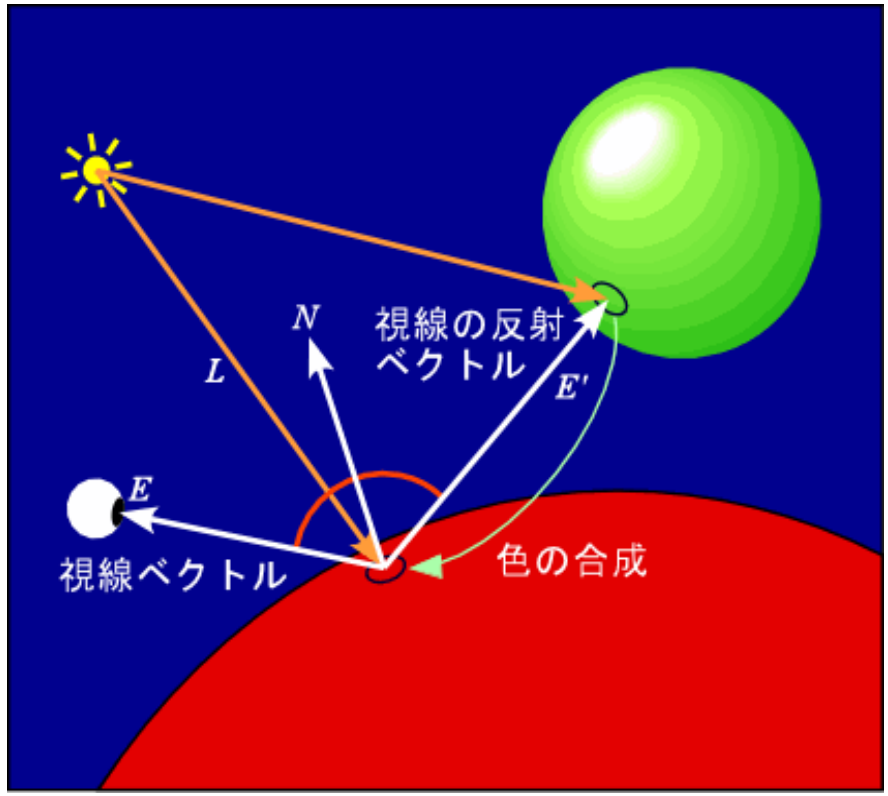


遮蔽物による影

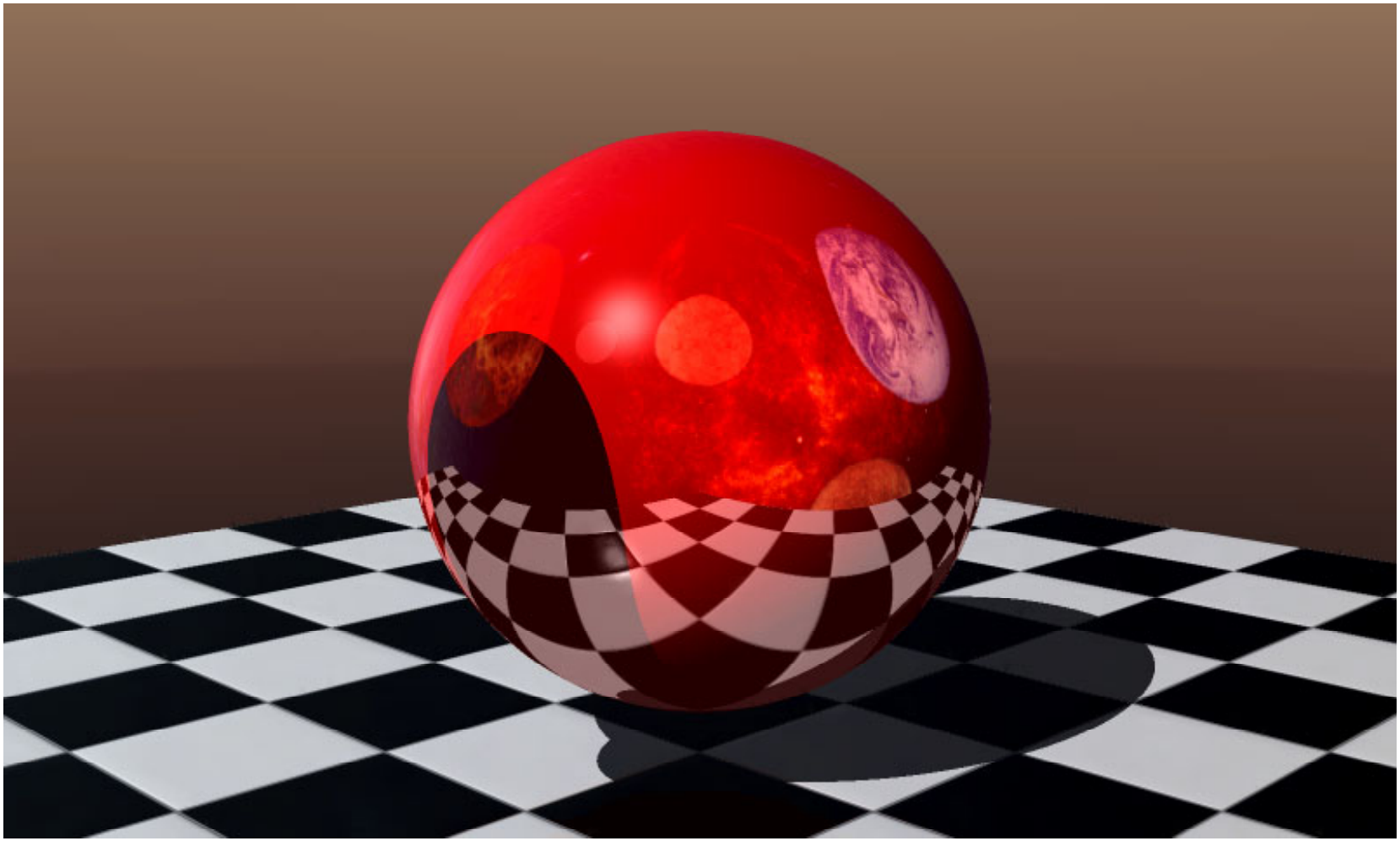


視線の反射ベクトルを追跡

- 可視点を視点として，反射方向に何が見えるか調べる
- 反射方向に見えた色を可視点の色と合成する

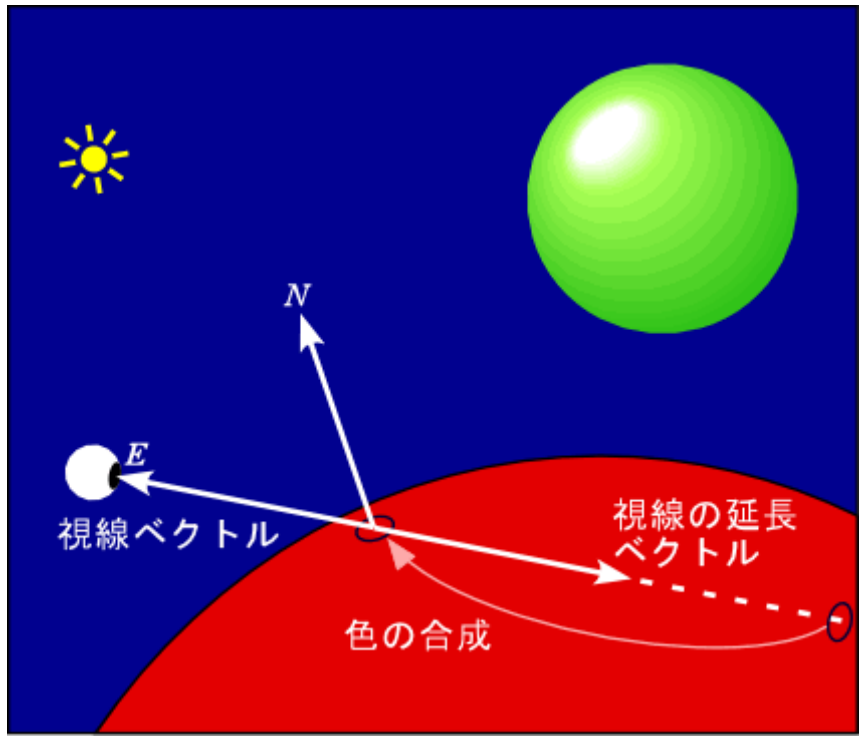


映り込み

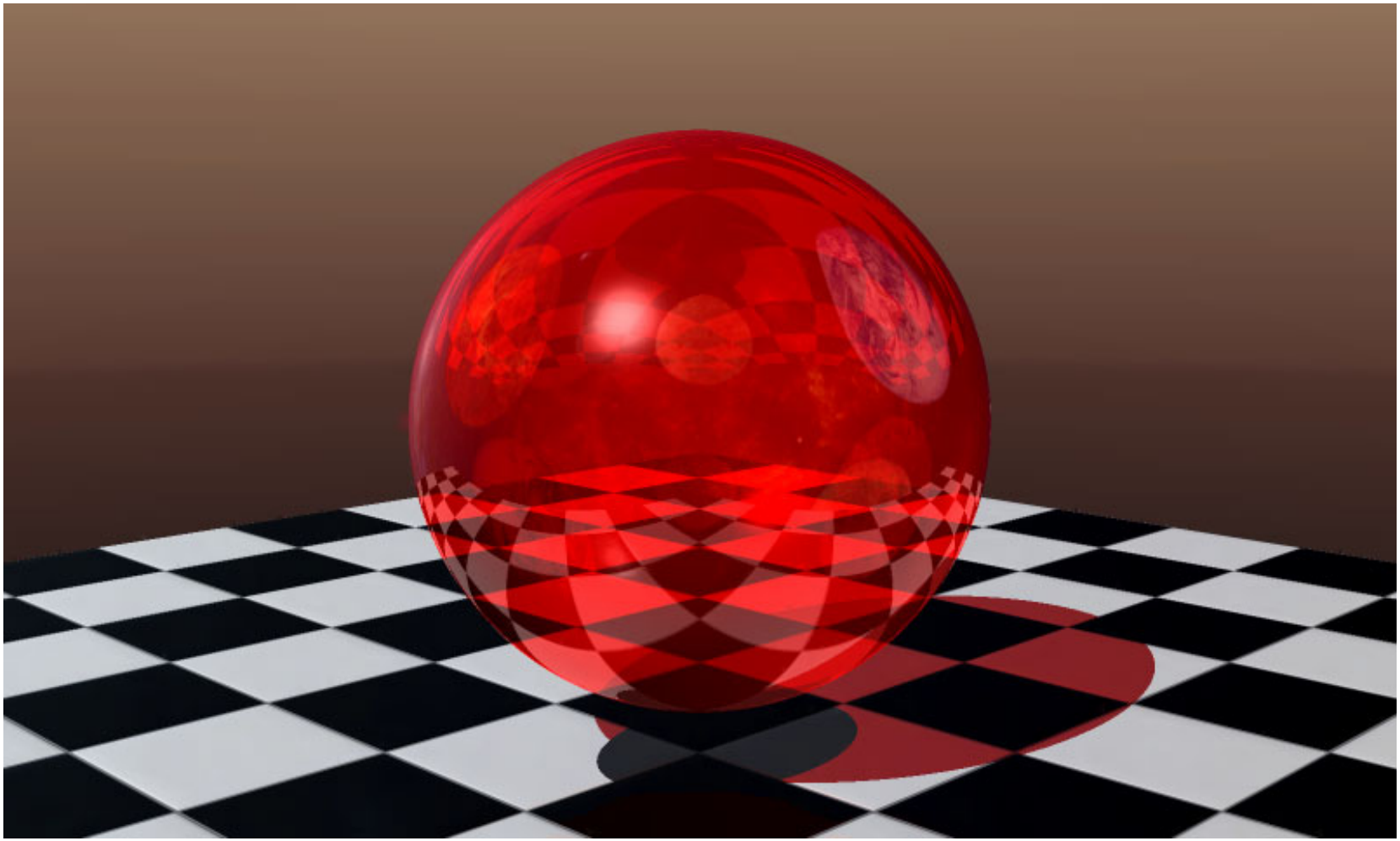


視線の延長ベクトルを追跡

- 可視点を視点として，延長方向に何が見えるか調べる
- 延長方向に見えた色を可視点の色と合成する



透明

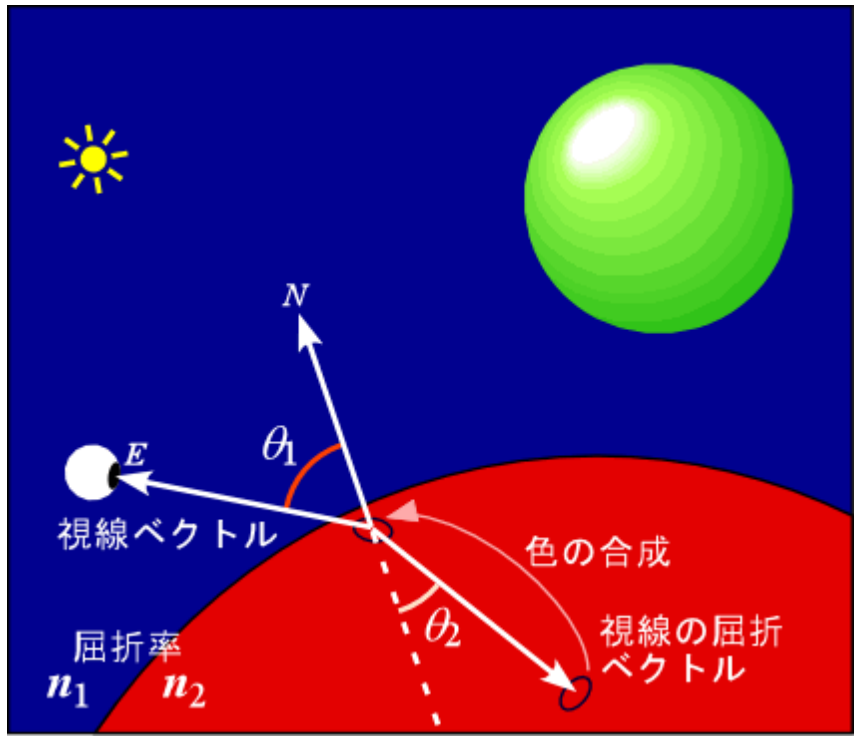


屈折率を考慮する

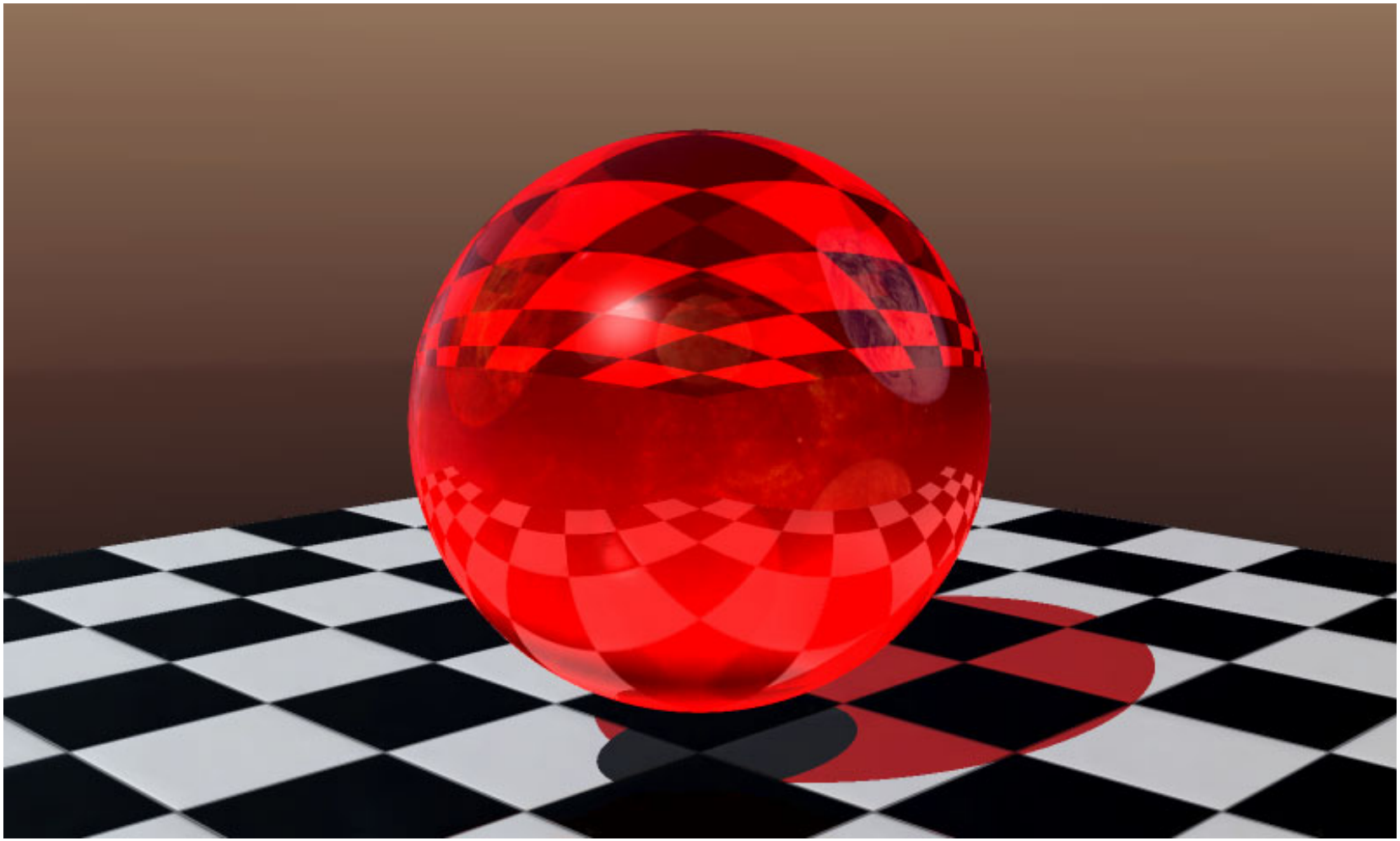
- 二つの媒質の屈折率から屈折方向を求める
  - スネルの法則

$$n_1(\lambda)\sin\theta_1 = n_2(\lambda)\sin\theta_2$$

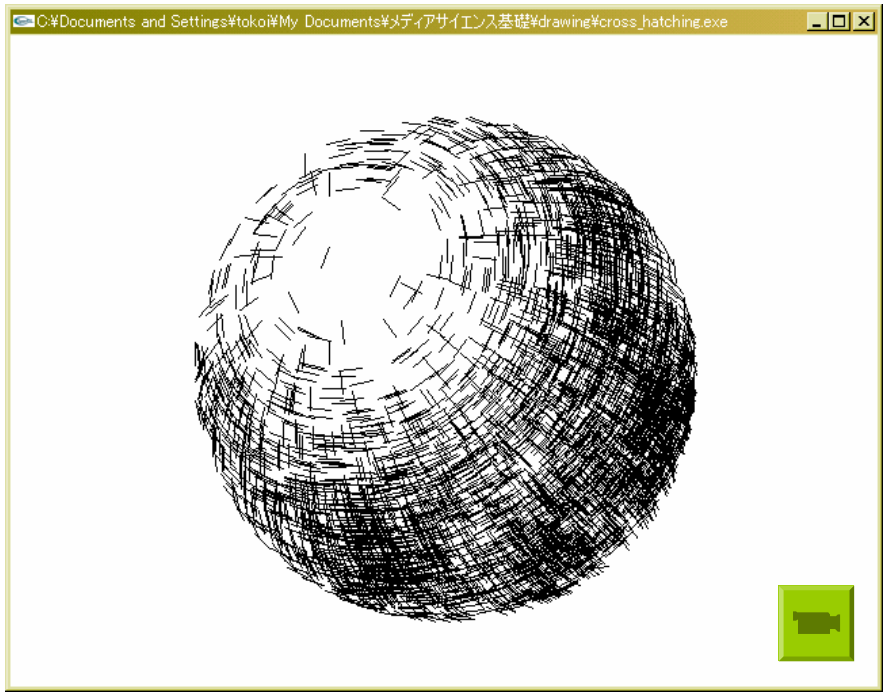
- 屈折方向に見えた色を可視点の色と合成する



屈折

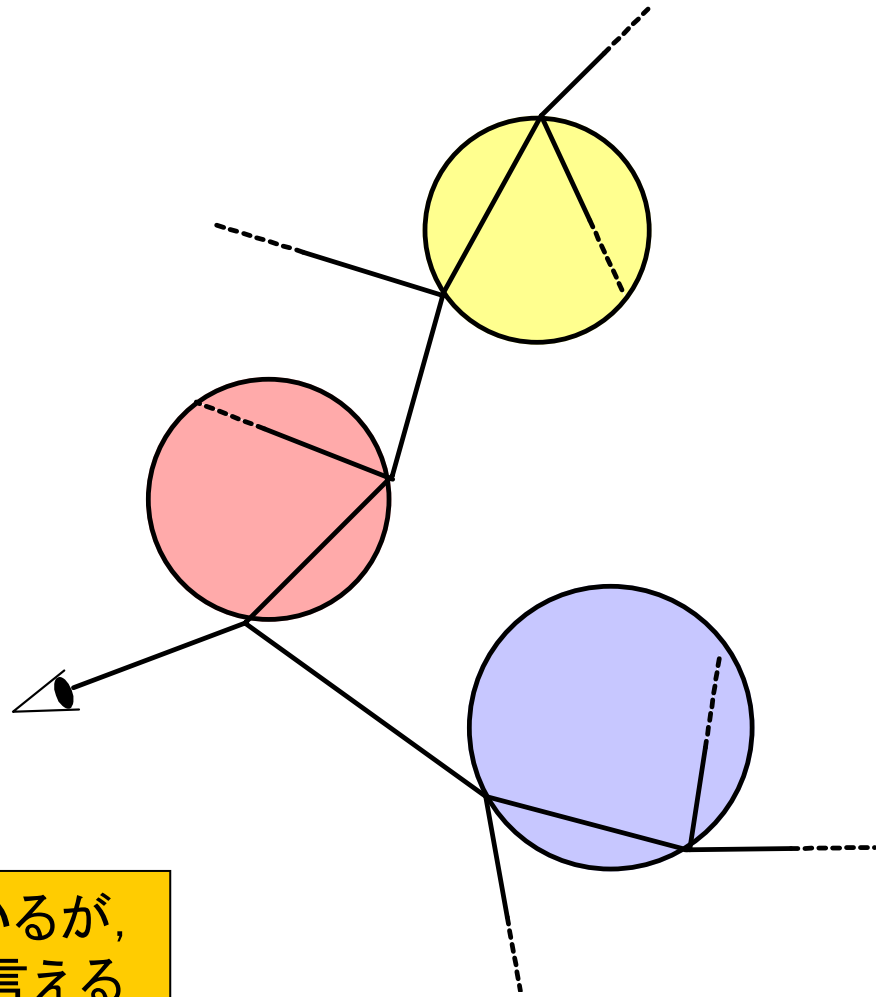


リアルさを追求しない場合もある



レイトレーシングの課題

- 物体が複数あれば，その中のどれが見えているのか探すのに時間がかかる
- 透明な物体が複数あれば，反射や屈折した先にある物体を探さなければならない
- 反射や屈折によって視線が枝分かれしていくので，反射や屈折を繰り返すたびにレイの数がどんどん増える



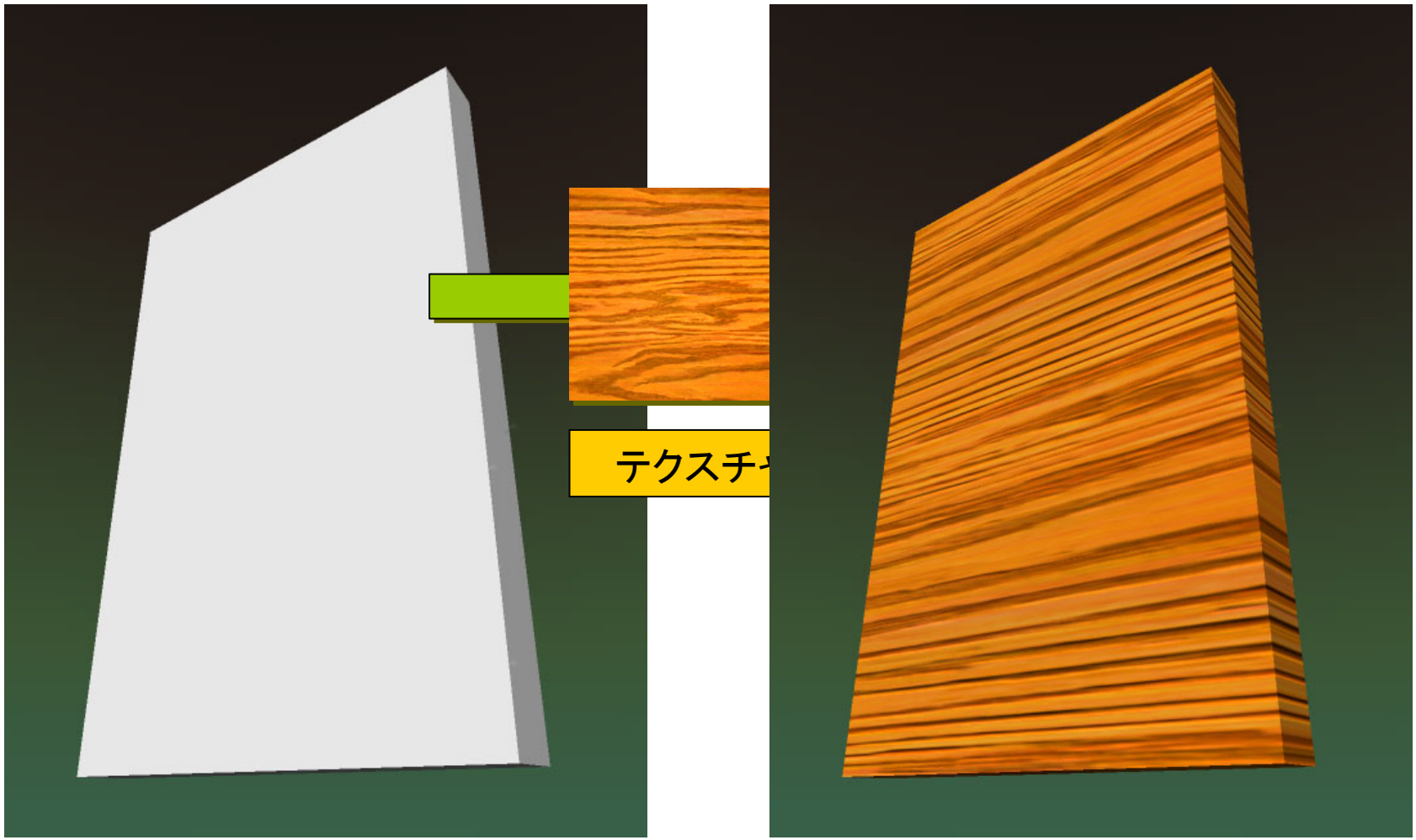
さまざまな高速化手法が提案されているが，本質的に時間のかかる方法であると言える

リアルタイム処理

- 制限時間内に処理が完了できる必要がある
  - リアルタイムグラフィックス（ゲーム等）では，画面の書き換えタイミング等
- レイトレーシングは空間をサンプリングして画像を作る
  - どういう画像ができるのか，完成までにどれだけ時間がかかるのかを事前に見積もることが難しい
- レイトレーシングはリアルタイム処理に向かないと考えられる

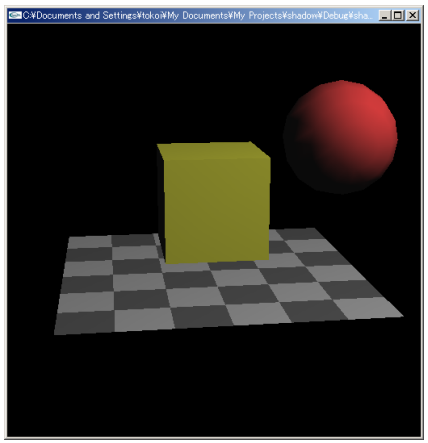
リアルタイム処理を目指したものではないが，映画制作などで使われる
PIXAR の開発した RenderMan は，極力レイトレーシングを使わない

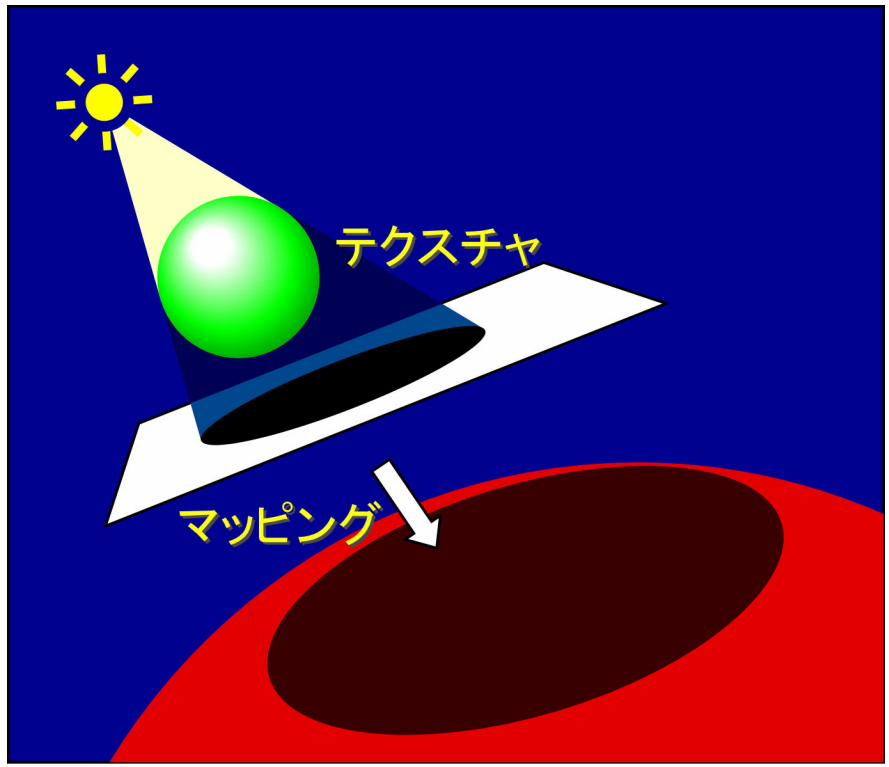
テクスチャマッピング



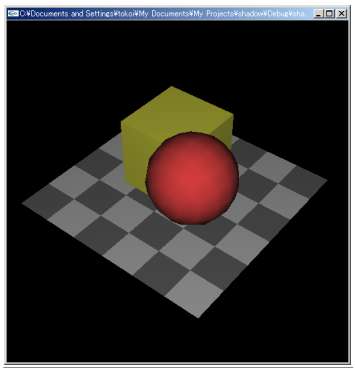
シャドウマッピング

- 影を落とす物体のシルエット画像をテクスチャにする
- シルエット画像のテクスチャを影が落ちる物体の表面にマッピングする



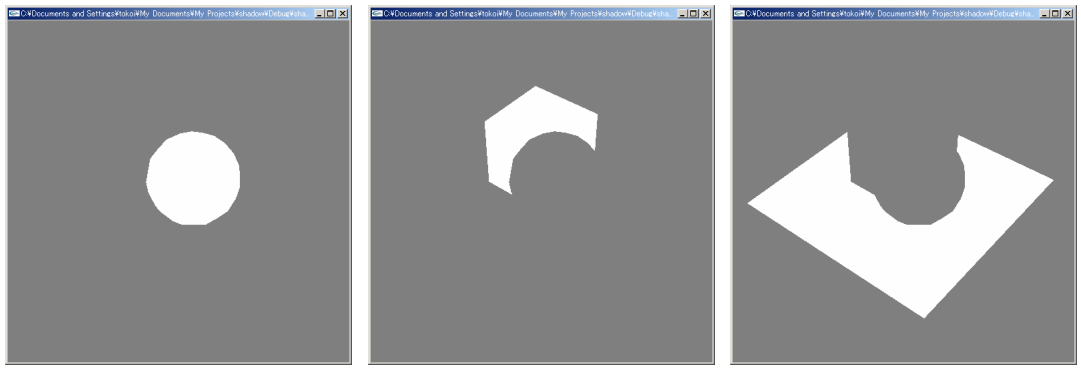


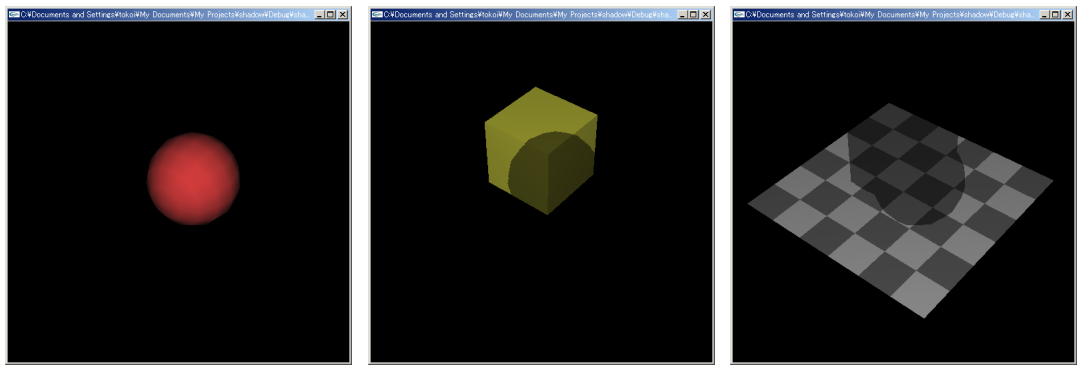
簡単なシャドウマッピングの手順



光源位置から
見たシーン

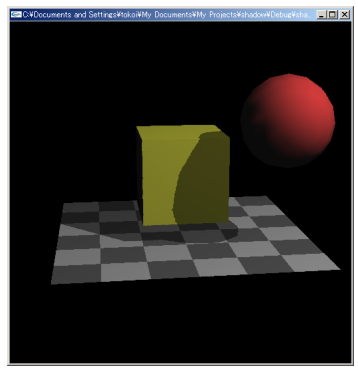
シルエット画像



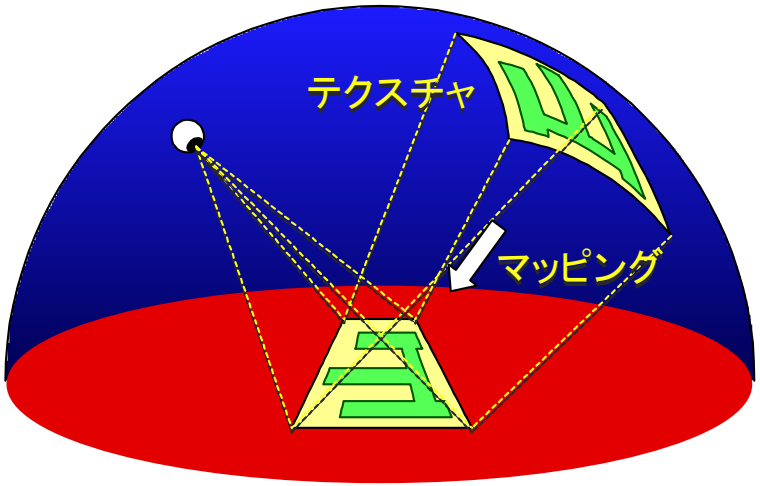


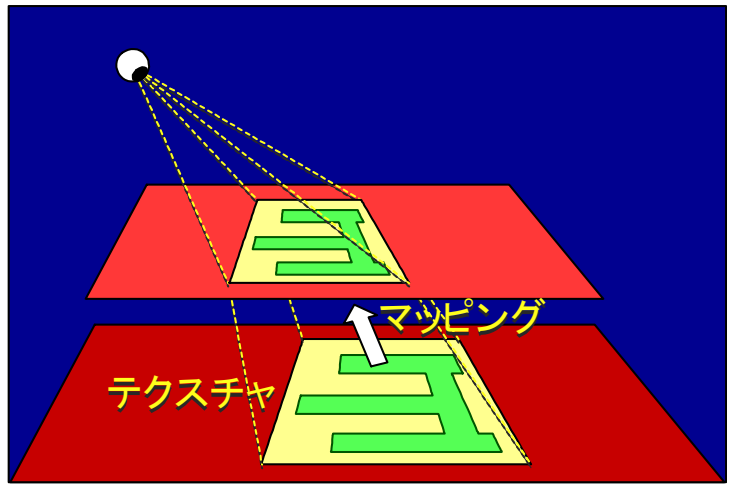
シルエット画像を物体にマッピング

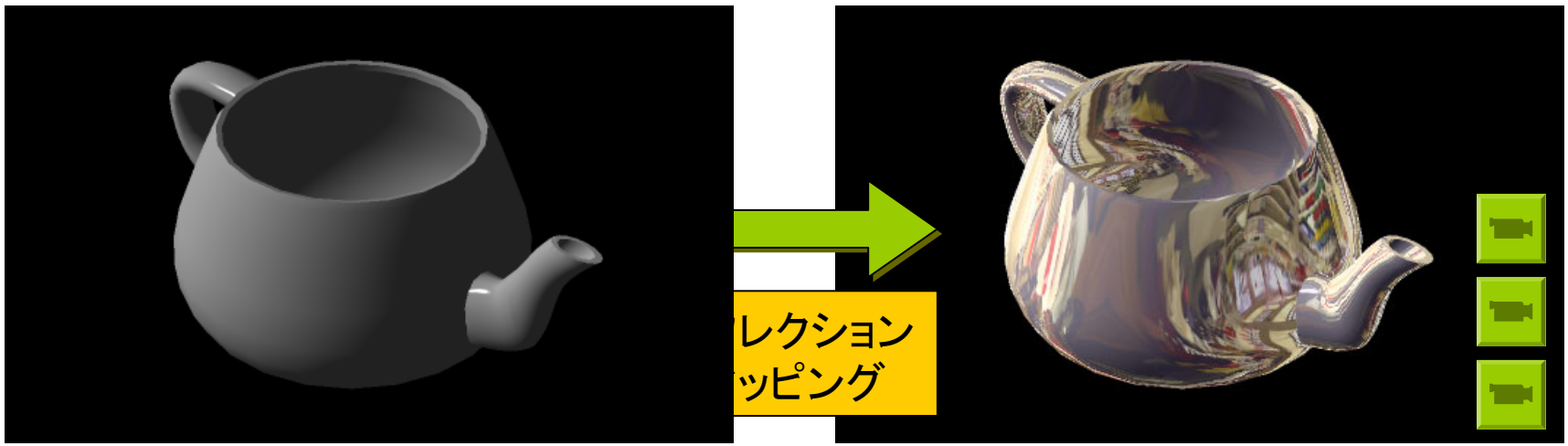
完成シーン



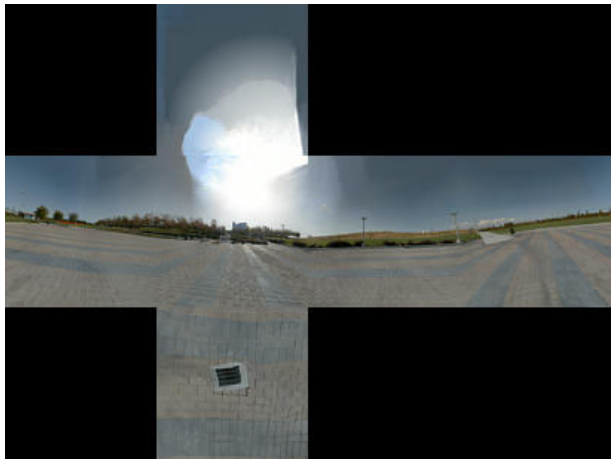
リフレクション（反射）マッピングと リフラクション（屈折）マッピング

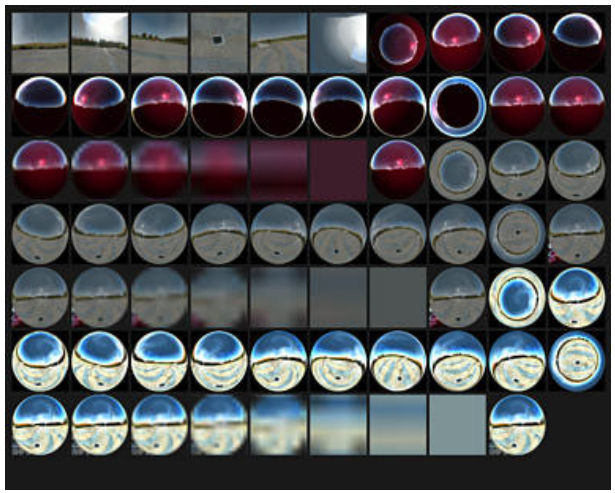






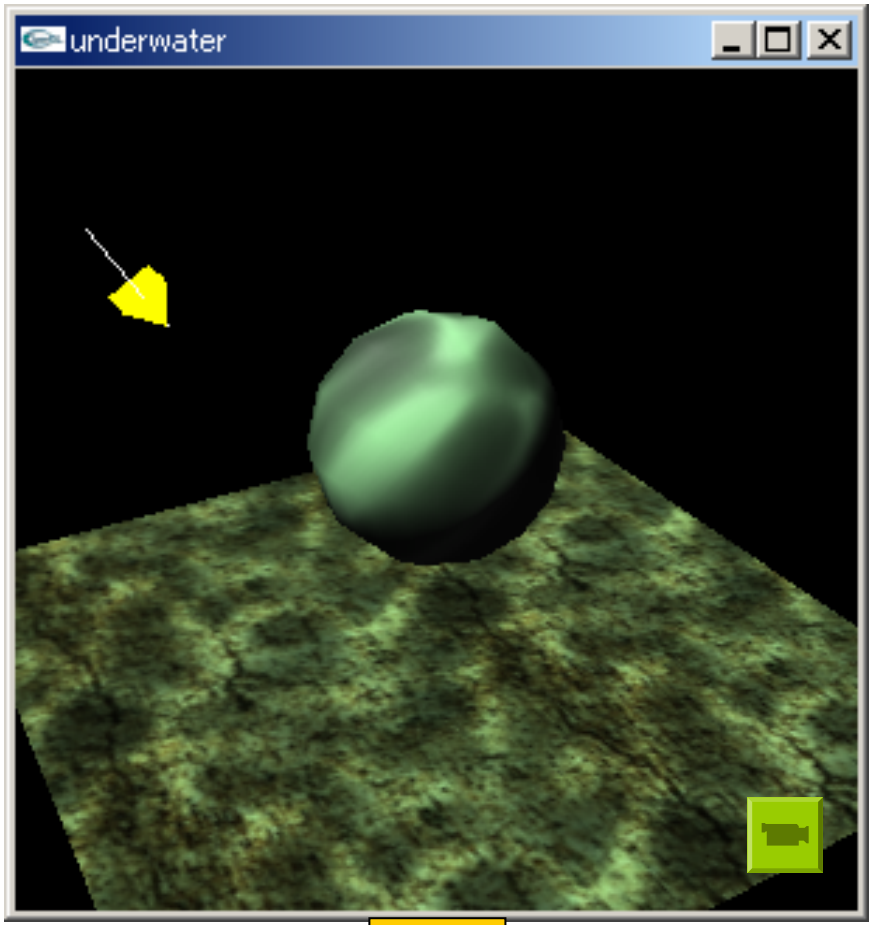
Reflection Space Image Based Rendering





Cabral, Olano, Nemec (1999)

そのほかの応用



水

火

プリレンダリングと リアルタイムレンダリングの将来

- プリレンダリング手法の高速化の必要性
  - 精度やリアリティの向上には際限がない
  - リアルさをどこに求めるかは時代とともに変わる
  - 時間のかかる処理は常に存在する
- プリレンダリング手法が高速化すればリアルタイムレンダリングは必要ないのか
  - むしろ，リアルタイムレンダリングの手法が，それまでのプリレンダリング手法を置き換えていく

モデリング

- コンピュータの中に創る「舞台空間」の演出
  - 物体の形を定義する
  - 物体の色を設定する
  - 物体を配置する
  - 光源を設定する

形状モデリング

- 人手によるモデリング
  - 工業製品のデザイン
  - 人物キャラクタ
  - クリーチャ
  - ・・・
- コンピュータによるモデリング
  - 自然の造形
  - 物理現象で形が決まるもの
  - ・・・

形状データの作成

- 計測による方法
  - デジタイザやレンジファインダ(形状計測装置)を使う
    - 実物やクレイモデルなどからデータを作成する
- 人手による方法
  - コンピュータで対話的に形状を定義する
    - 工業製品のデザイン，人物キャラクタ，クリーチャなど
- プログラムによる方法
  - 形状を表す「手続き」を書く
    - 自然の造形や物理現象によって決まる形はシミュレーションによって求める

形状の表現

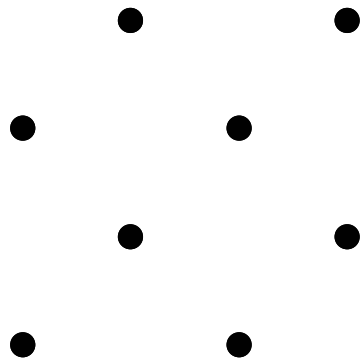
- どのような方法で形を表現すればよいか
  - 境界による表現
    - いちばん一般的、表現可能な形状に制限が少ない
  - 立体の組み合わせによる表現
    - 積み木など、利用者にとって理解しやすい
  - 生成過程の組み合わせによる表現
    - 形を作る「手順」そのものを形状表現に使う
  - 密度や濃度による表現
    - 雲とか煙とか水とか、「境界面」が明確に決められないもの

境界による表現

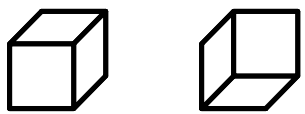
- 頂点・稜線・面などの情報で形を表現する
  - 立体の内部と外部を隔てる境界
- ワイヤフレームモデル
  - 簡易で高速⇔情報量は不十分
- サーフェースモデル
  - ハーフトーン画像の生成が可能
- ソリッドモデル
  - 立体同士の演算等が可能

ワイヤーフレームモデル

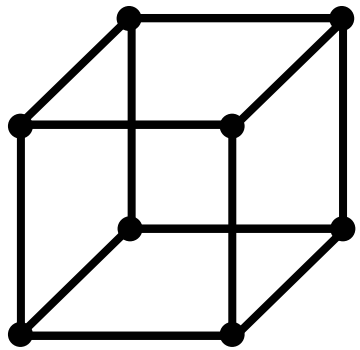
- 点だけで形を表してみる
  - なんだかよくわからない



- **点と点を線で結んでみる**
  - **だいたい形はわかる**

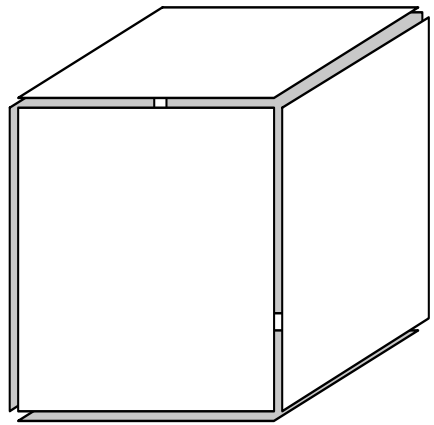


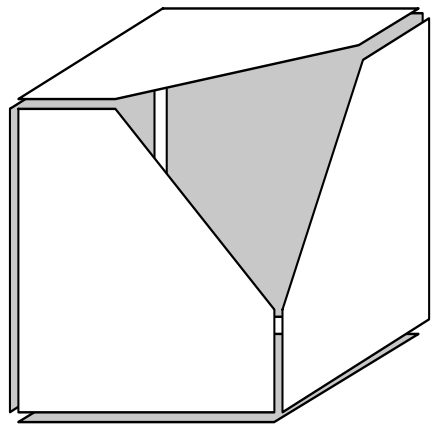
でもどっちだかわからない



サーフェースモデル

- 面を構成する稜線（あるいは頂点）をひとまとめにして，面のデータとして保持する
  - 内部と外部の区別が無い
  - 立体は閉じていなくても良い
  - 隠面消去可能

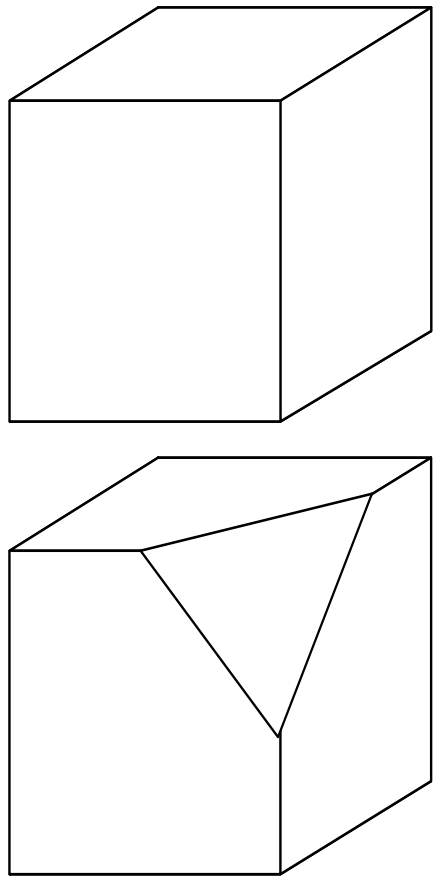




切ると内側が見える

ソリッドモデル

- 立体を構成する面をひとまとめにして，立体のデータとして保持する
  - 面は向き付けされている
  - 中身は詰まっている
  - 内部と外部が区別できる
  - 集合演算が可能
  - マスプロパティの算出が可能

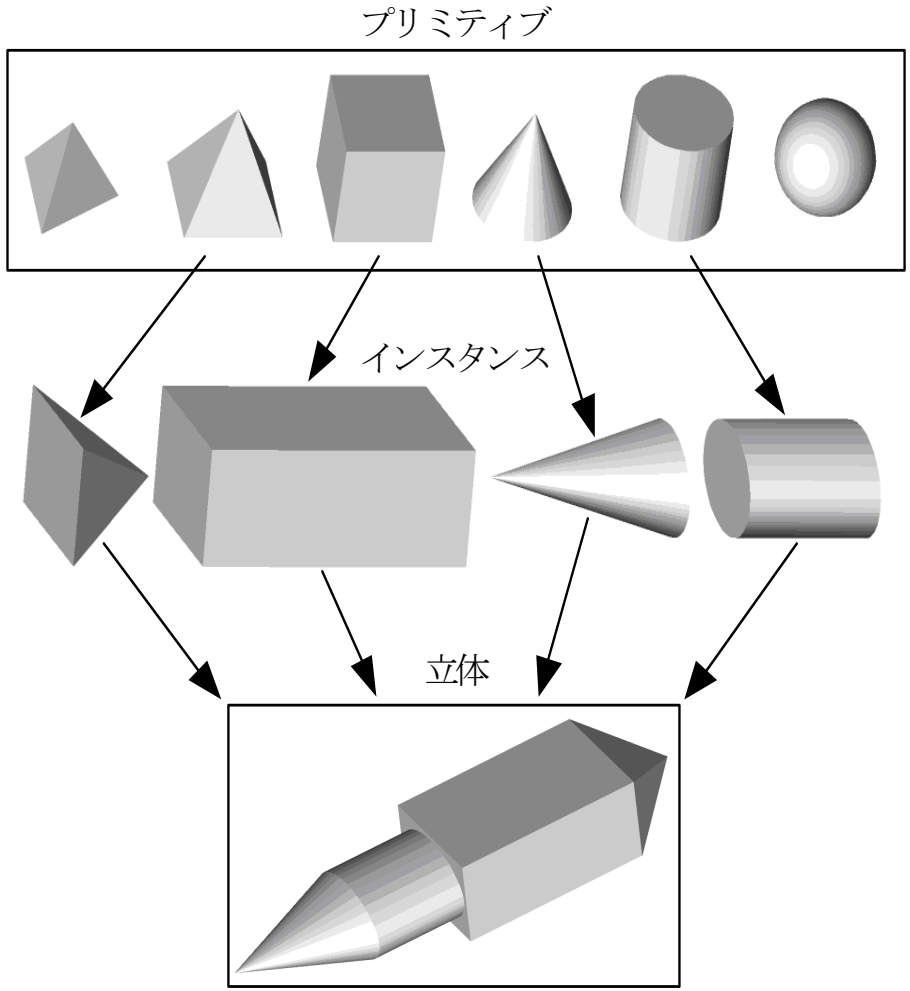


切ると中身が詰まっている

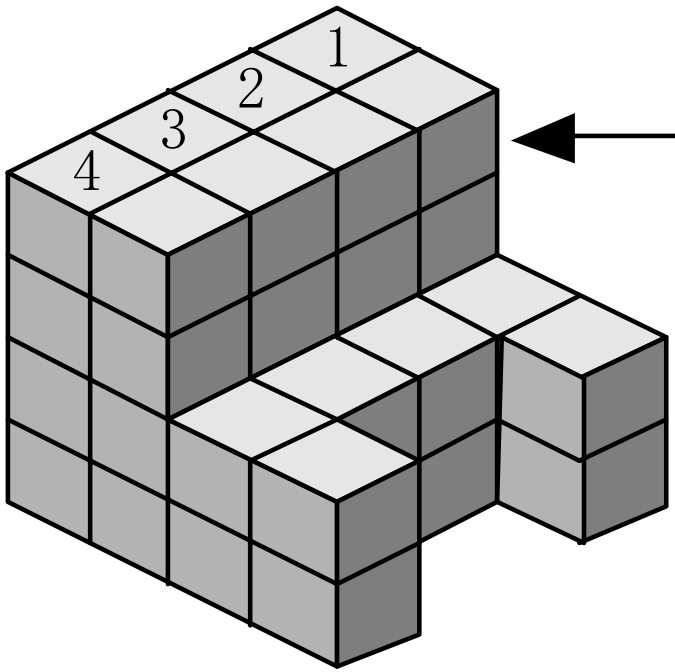
立体の組み合わせによる表現

- プリミティブインスタンス法
  - 積み木（パーツは拡大・縮小可能）
- Voxel表現
  - 同じ大きさのブロックの集合
- Octree表現
  - Voxel8個をひとつの単位にまとめて階層的に表現

プリミティブインスタンス法



Voxel 表現



○ その空間は空

● その空間は詰まっている

1 2 3 4

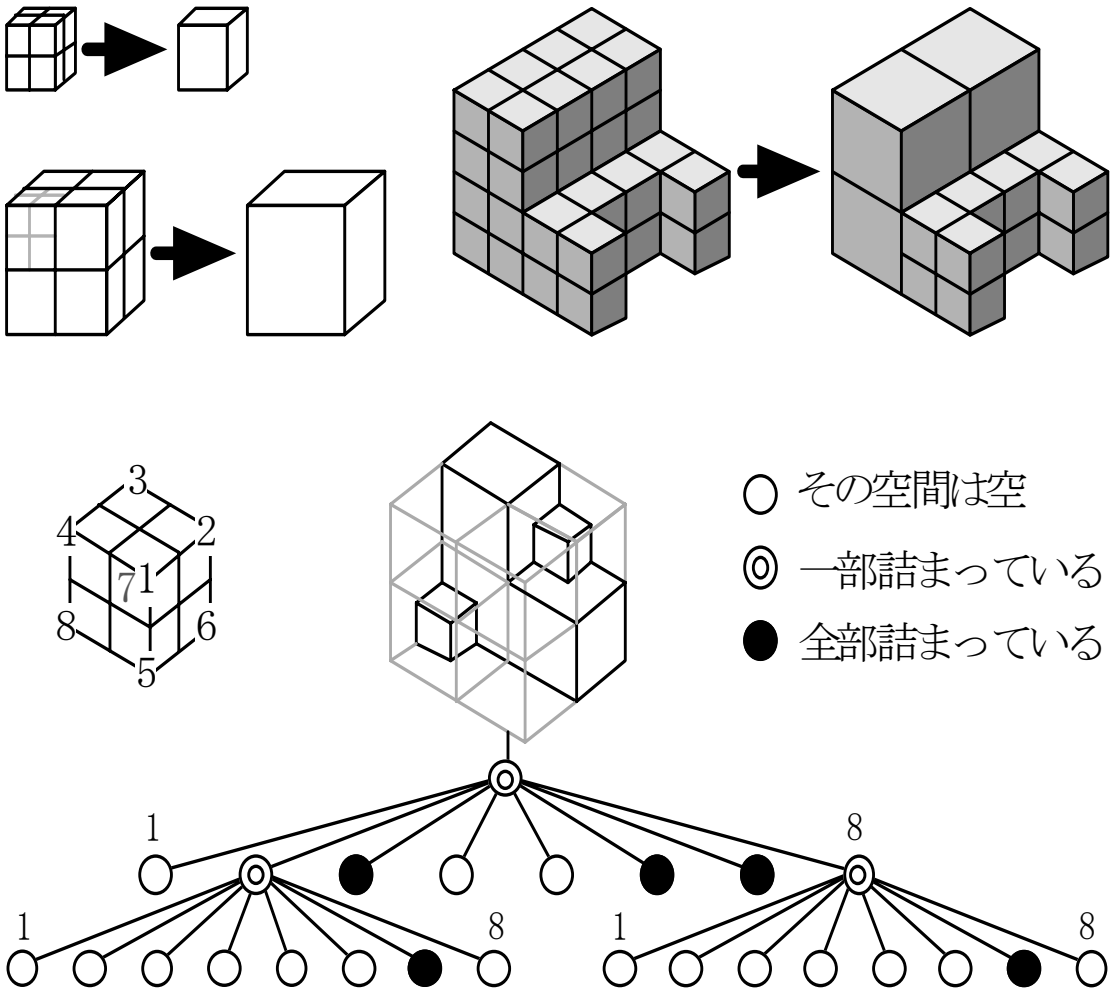
●●●● ●●●● ○○○○ ○○○○

●●●● ●●●● ○○○○ ○○○○

●●●● ●●●● ●●●● ●○○●

●●●● ●●●● ●●●● ●○○●

Octree（8分木）表現

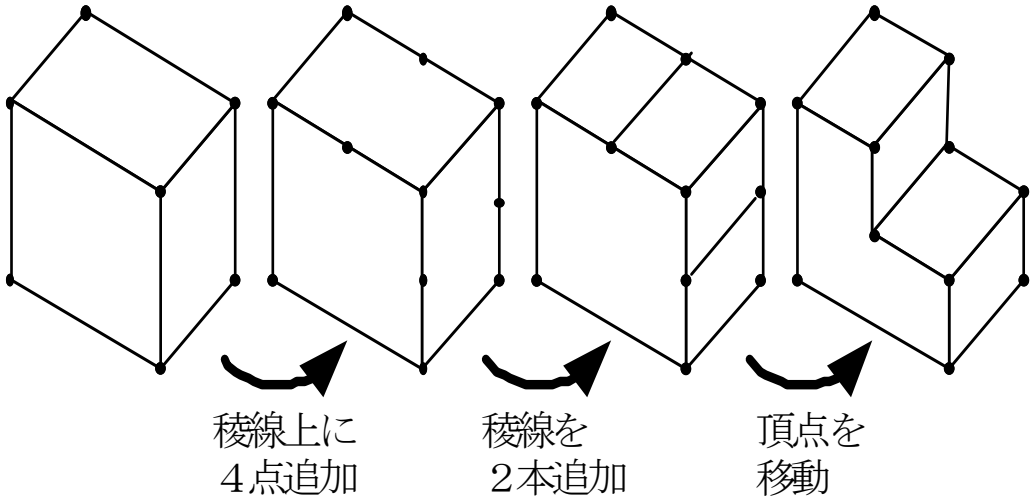


生成過程による表現

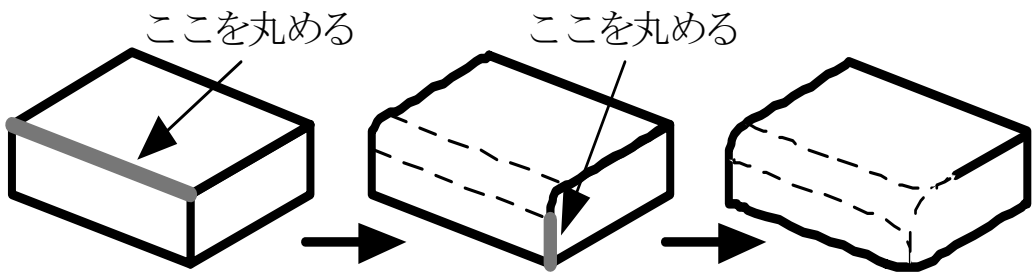
- 局所操作の組み合わせによる表現
  - 形状データの変更手順そのものを形状データに使う
- CSG (Constructive Solid Geometry)
  - プリミティブインスタンス法においてプリミティブ間の論理演算を行うもの
- スイープ
  - 掃引体（平面図形を経路に沿って移動したときにできる空間）

局所操作（ローカルオペレーション）

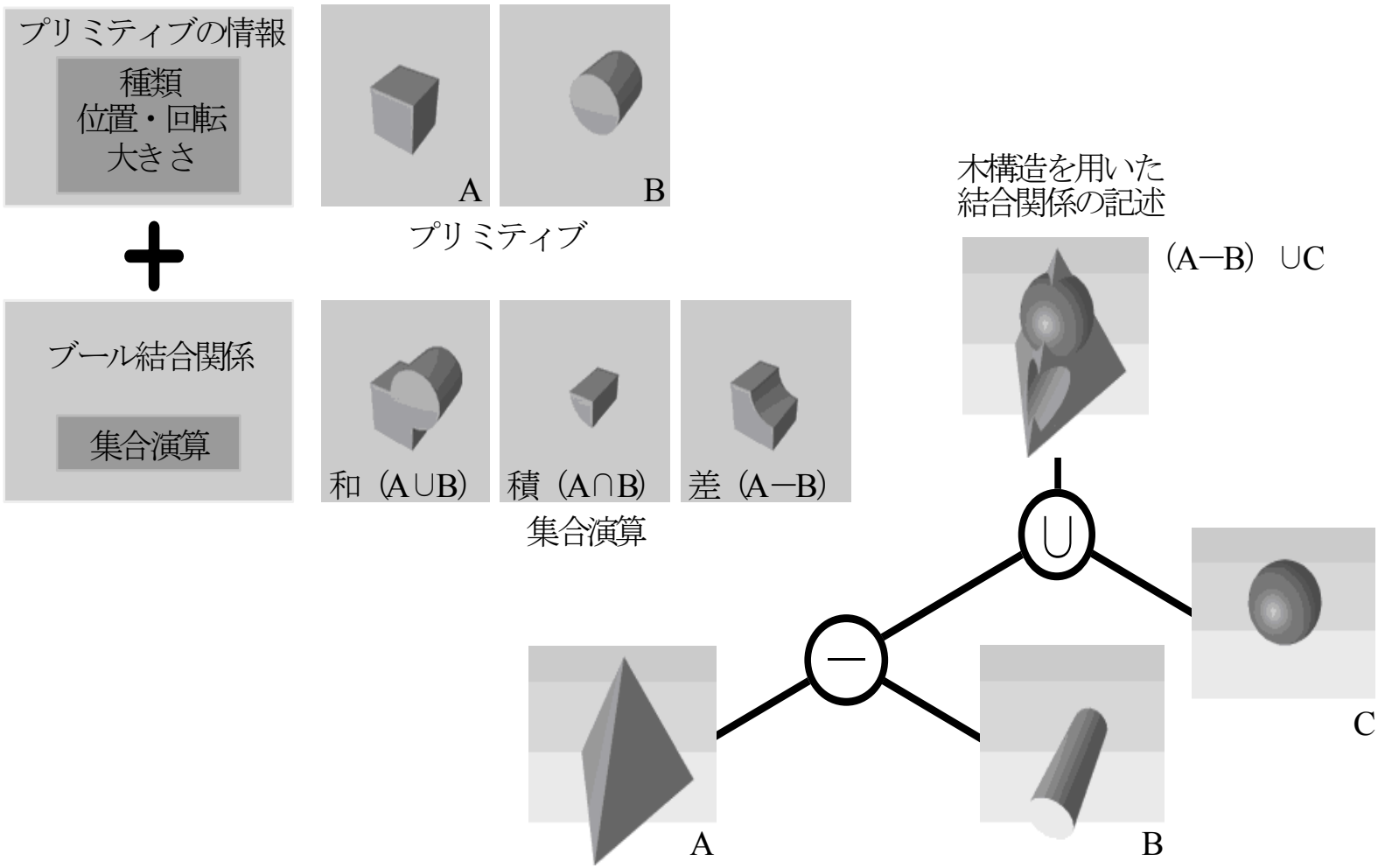
局所変形操作



丸め変形操作

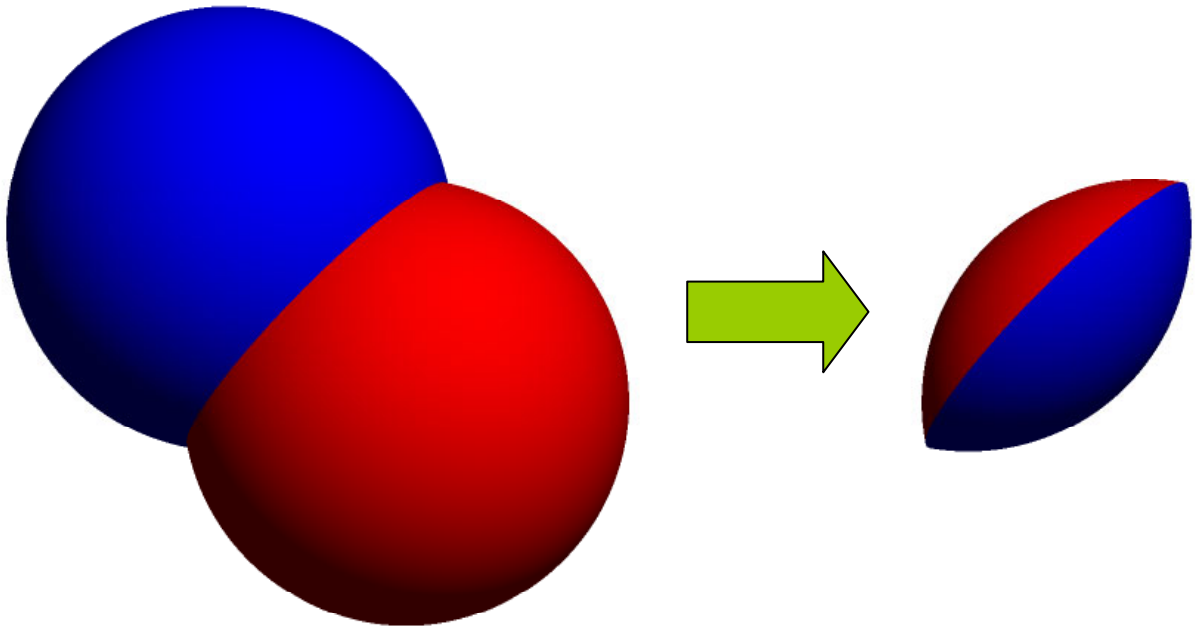


CSG(Constructive Solid Geometry)

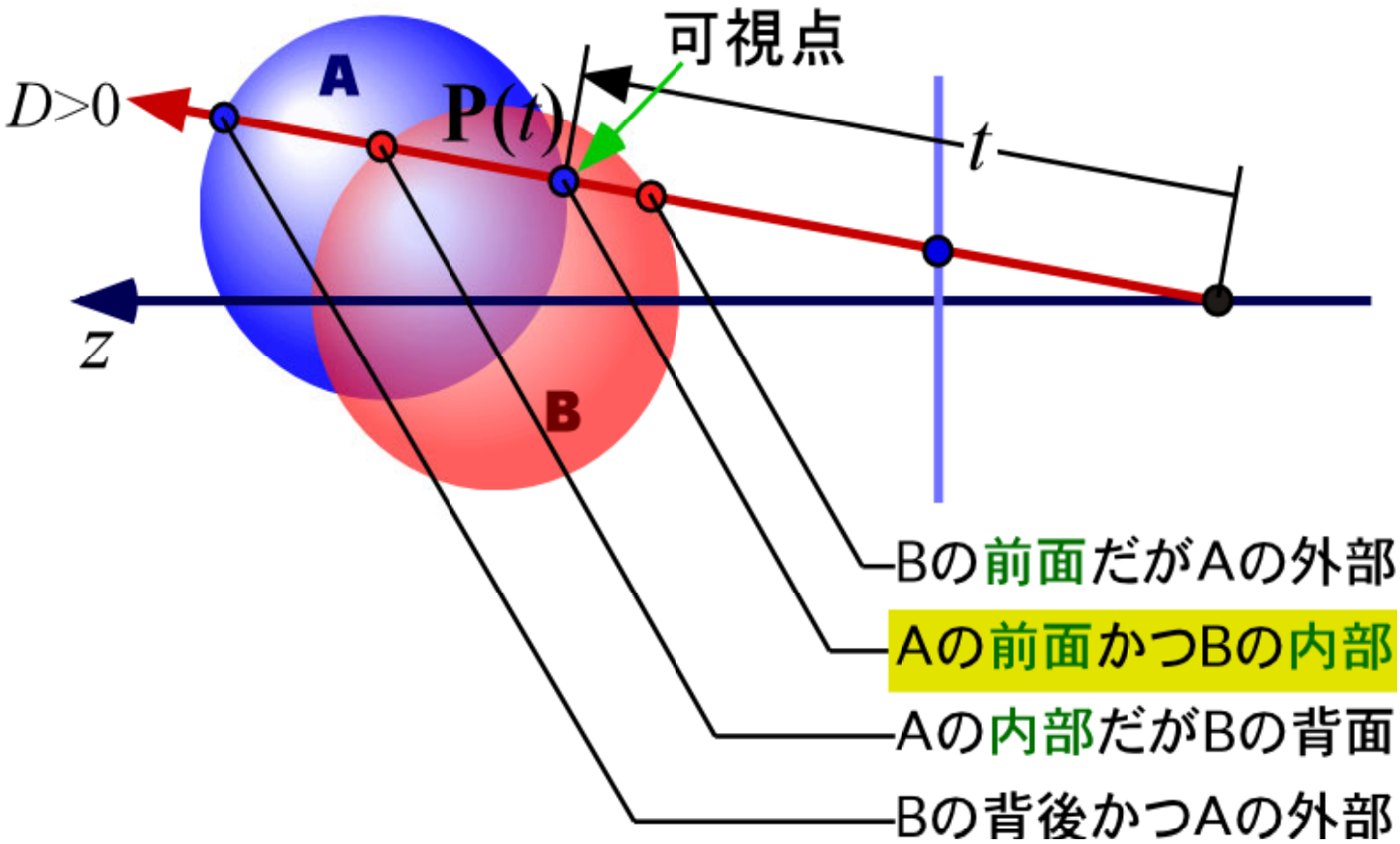


複数の球の組み合わせ

- 二つの球の共通部分だけを表示したい
  - 集合演算処理



視線上の点


可視点
D>0
A
P(t)
t
z
B
Bの前面だがAの外部
Aの前面かつBの内部
Aの内部だがBの背面
Bの背後かつAの外部

点の内外判定

- 球の場合
  - 球の方程式

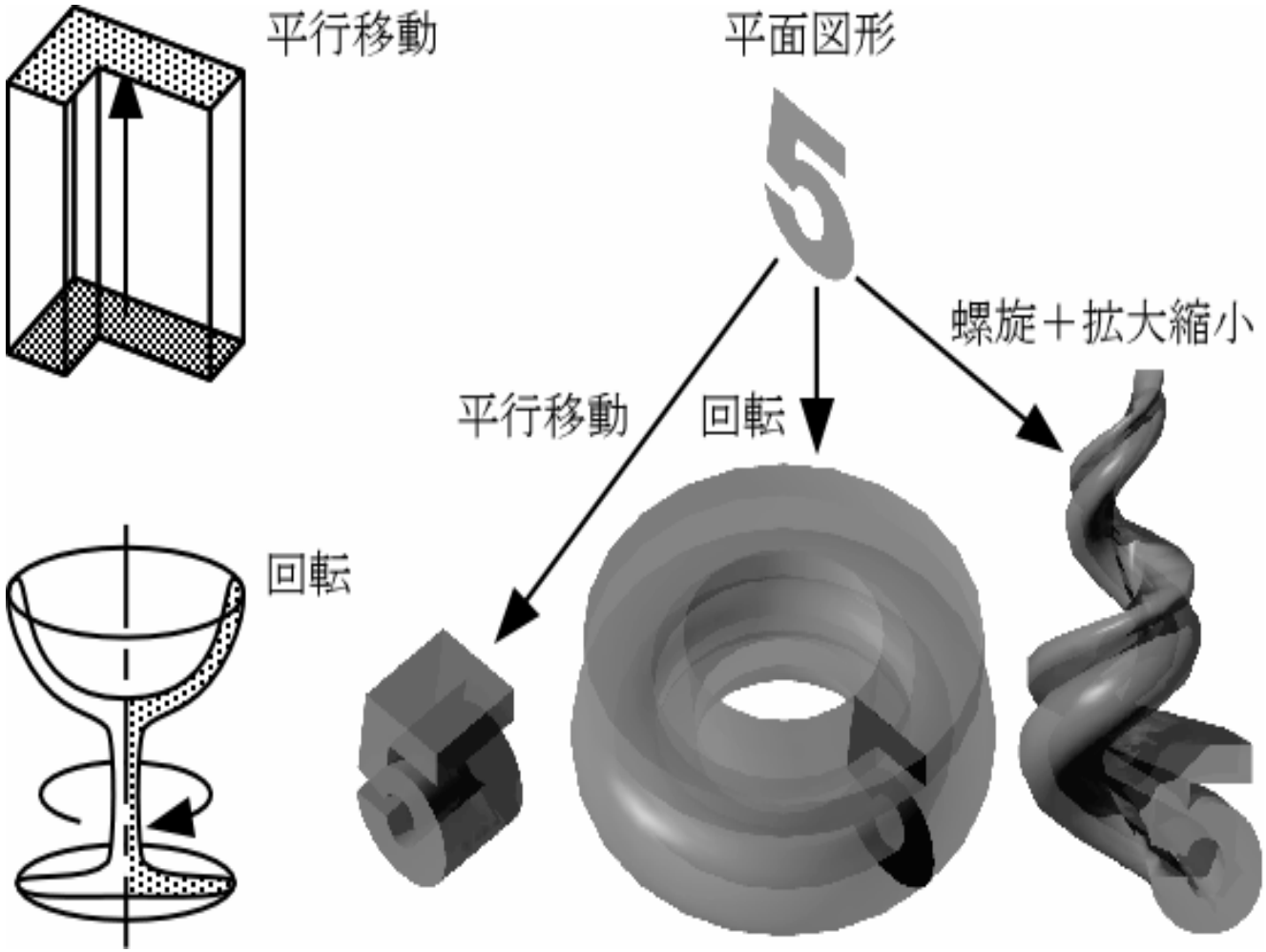
$$(x-x_c)^2+(y-y_c)^2+(z-z_c)^2=r^2$$

  - 内外判定の判別式

$$d=r^2-(x-x_c)^2-(y-y_c)^2-(z-z_c)^2$$

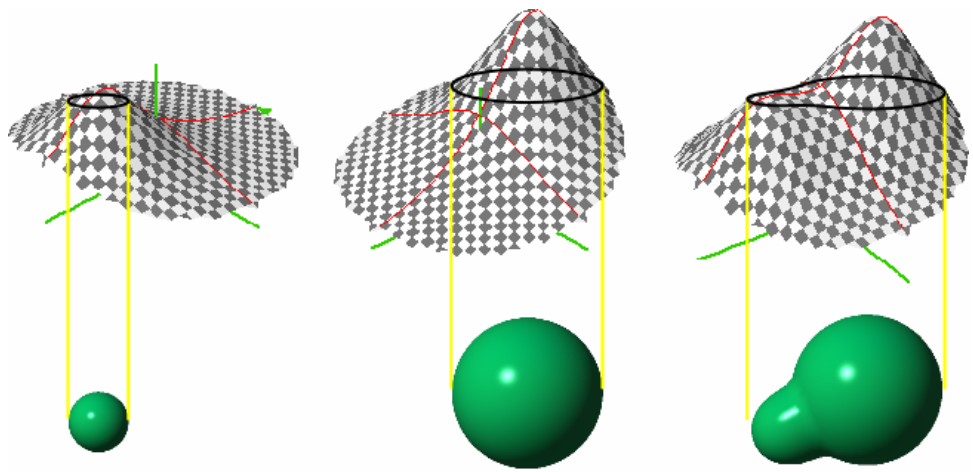
$d>0$：内部
$d=0$：表面
$d<0$：外部

スイープ



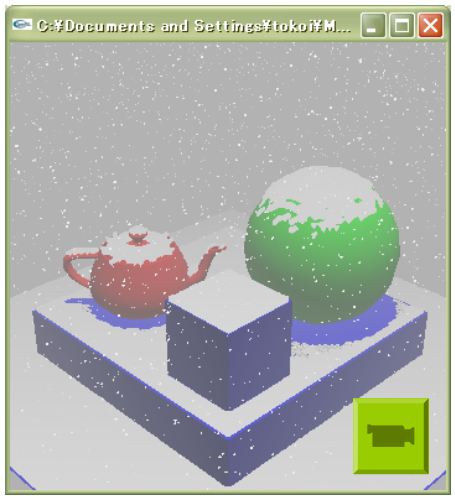
密度や濃度による表現

- パーティクル
  - 微粒子の集合体
    - 煙、雲、雪など
      明確な形を持たないもの
- メタボール
  - 等値面を用いて
    形状を表す



積雪形状のリアルタイムモデリング

- 複雑なシーン上に積もる雪の形状の時間変化の表現
  - 積雪景観の表現



クロスシミュレーション

- クロス（布）が作る形や動きの生成
  - 着衣形状のモデリング
- 糸シミュレータ 
- 網シミュレータ 
- 布シミュレータ 
- ダイナミックに動く布 

最近の研究

- Stable but Responsive Cloth 
  - Choi, Ko (2002)
  - ダイナミックなドレープの表現
- Robust Treatment of Collisions, Contact and Friction for Cloth Animation 
  - Bridson, Fedkiw, Anderson (2002)
  - 衝突や摩擦，接触の表現
- Untangling Cloth 
  - Baraff, Witkin, Kass (2003)
  - 布同士の干渉の処理（もつれない布）