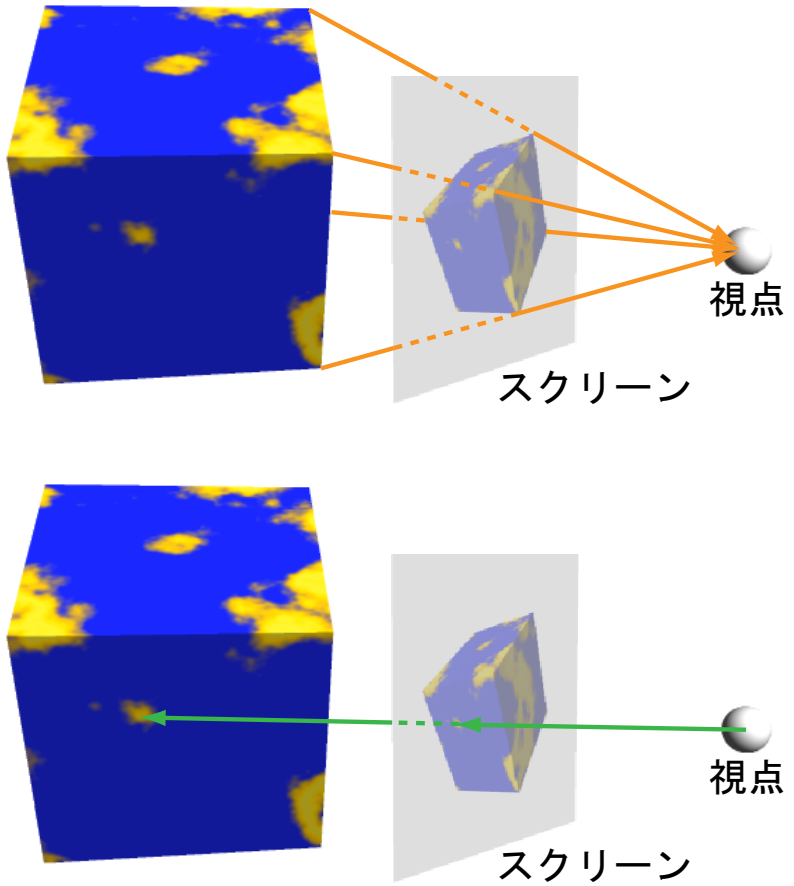
コンピュータグラフィックス

第14回：レイトレーシング

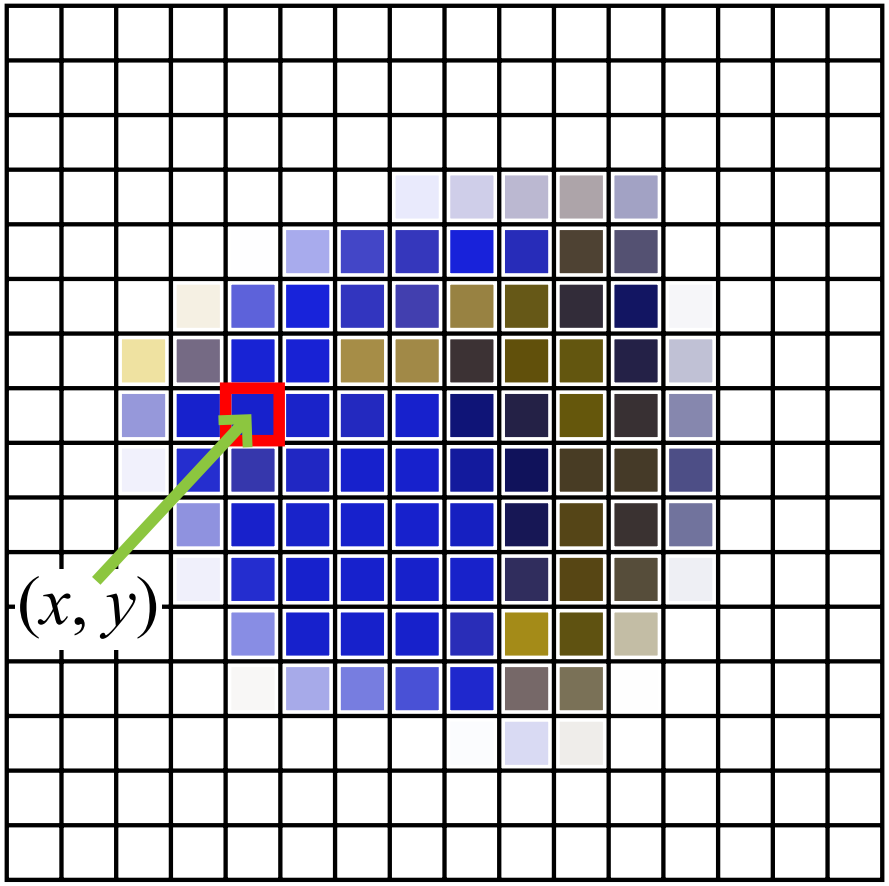
レンダリングの二つの方向

- 投影方式
  - 立体図形の各部分がスクリーン上のどの位置に表示されるか計算する
  - ラスタライズ
- サンプリング方式
  - 視点からスクリーン上のひとつの画素を通して何が見えるか調べる
  - レイトレーシング
    - 視線探索法
    - 光線追跡法



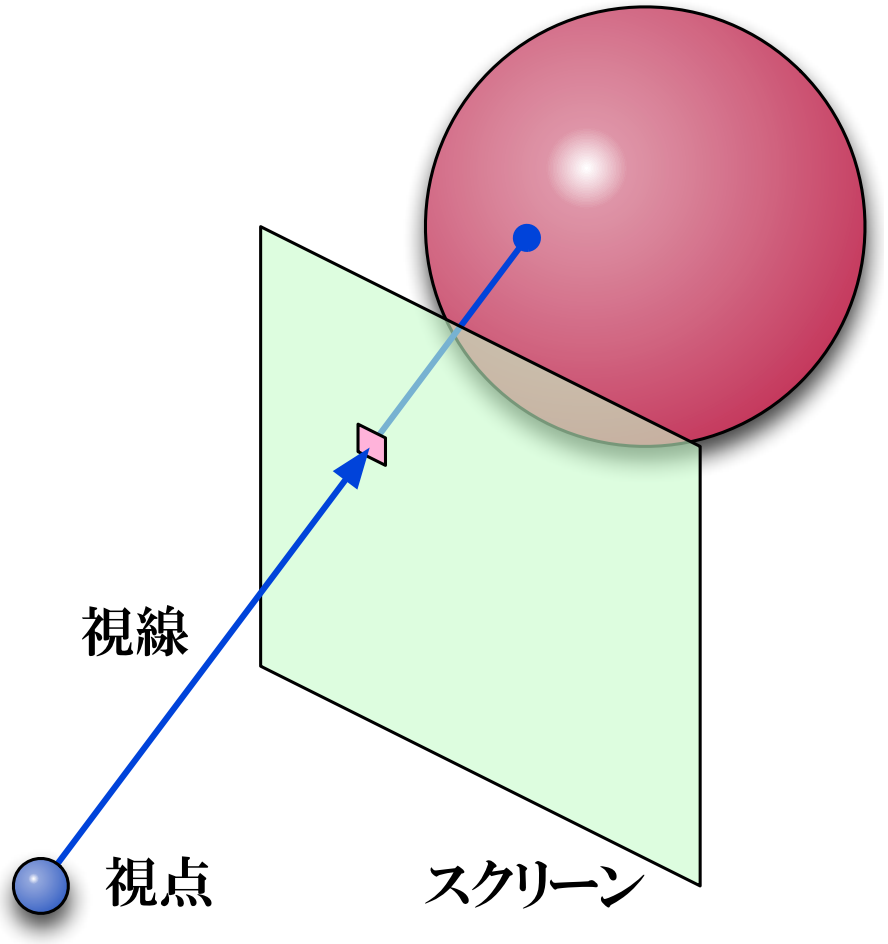
レイトレーシングの考え方

- 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

- 視線と球との交点を求める
  - 視線は視点から出発して色を決定したい画素のスクリーン上の位置を通る半直線
  - 球が複数あれば最も手前にある球との交点を求める
- 交点における球の色を求める
  - 交点の位置とその点における法線ベクトルをもとに陰影付けを行う
- 画素の色を球の色にする



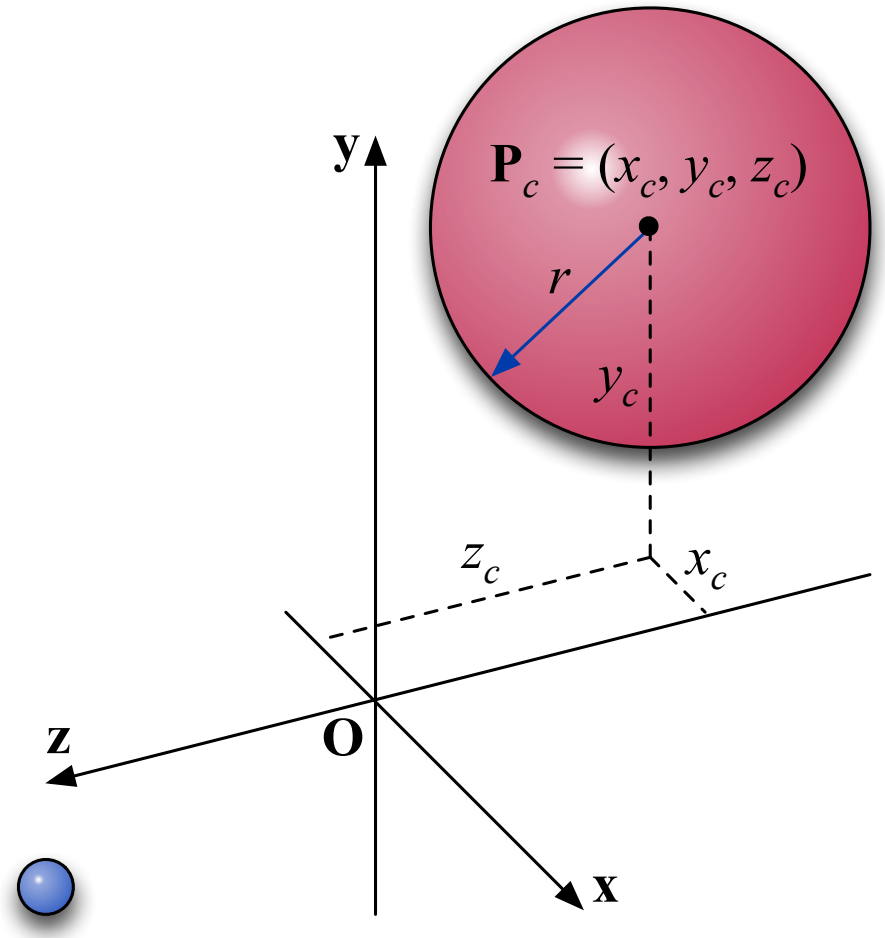
球の方程式

- 表面上の点の位置
  - $\mathbf{P} = (x, y, z)$
- 中心の位置
  - $\mathbf{P}_c = (x_c, y_c, z_c)$
- 半径
  - r
- 方程式

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

$$\Downarrow$$

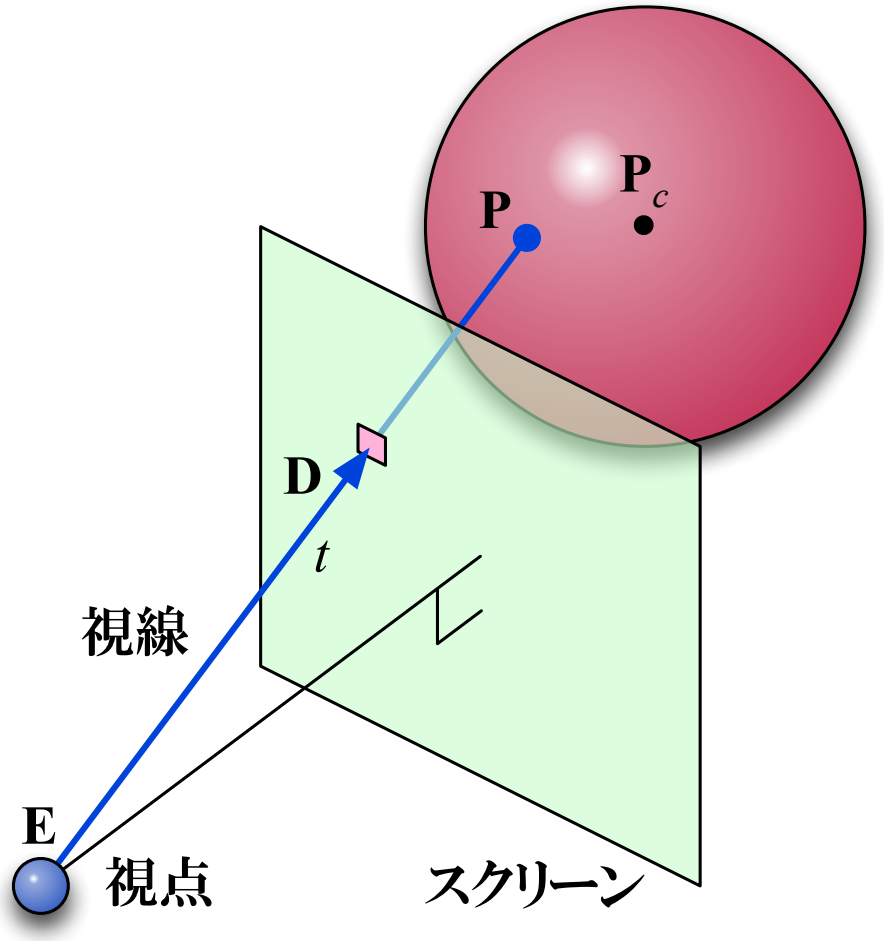
$$(\mathbf{P} - \mathbf{P}_c)^2 = r^2$$



視線の方程式

- 視点の位置
  - $\mathbf{E} = (x_e, y_e, z_e)$
- 視線の方向
  - $\mathbf{D} = (x_d, y_d, z_d)$
- パラメータ
  - t
- 方程式

$$\left.\begin{aligned} x &= x_e + x_d t \\ y &= y_e + y_d t \\ z &= z_e + z_d t \end{aligned}\right\} \Rightarrow \mathbf{P} = \mathbf{E} + \mathbf{D}t \qquad t > 0$$



球の方程式を視線の方程式に代入

- t の二次方程式になる

$$\{(\mathbf{E}+\mathbf{D}t)-\mathbf{P}_c\}^2 = r^2$$

$$\mathbf{D}^2t^2 + 2\mathbf{D}\cdot(\mathbf{E}-\mathbf{P}_c)t + \left\{(\mathbf{E}-\mathbf{P}_c)^2 - r^2\right\} = 0$$

$$\left.\begin{aligned} A &= \mathbf{D}^2 \\ B &= \mathbf{D}\cdot(\mathbf{E}-\mathbf{P}_c) \\ C &= (\mathbf{E}-\mathbf{P}_c)^2 - r^2 \end{aligned}\right\} \Rightarrow At^2 + 2Bt + C = 0$$

- 判別式

$$D = B^2 - AC$$

交差判定

- 判別式 $D > 0$ のとき

$$t = \frac{-B \pm \sqrt{D}}{A}$$

- 視点に近いのは小さい方の t

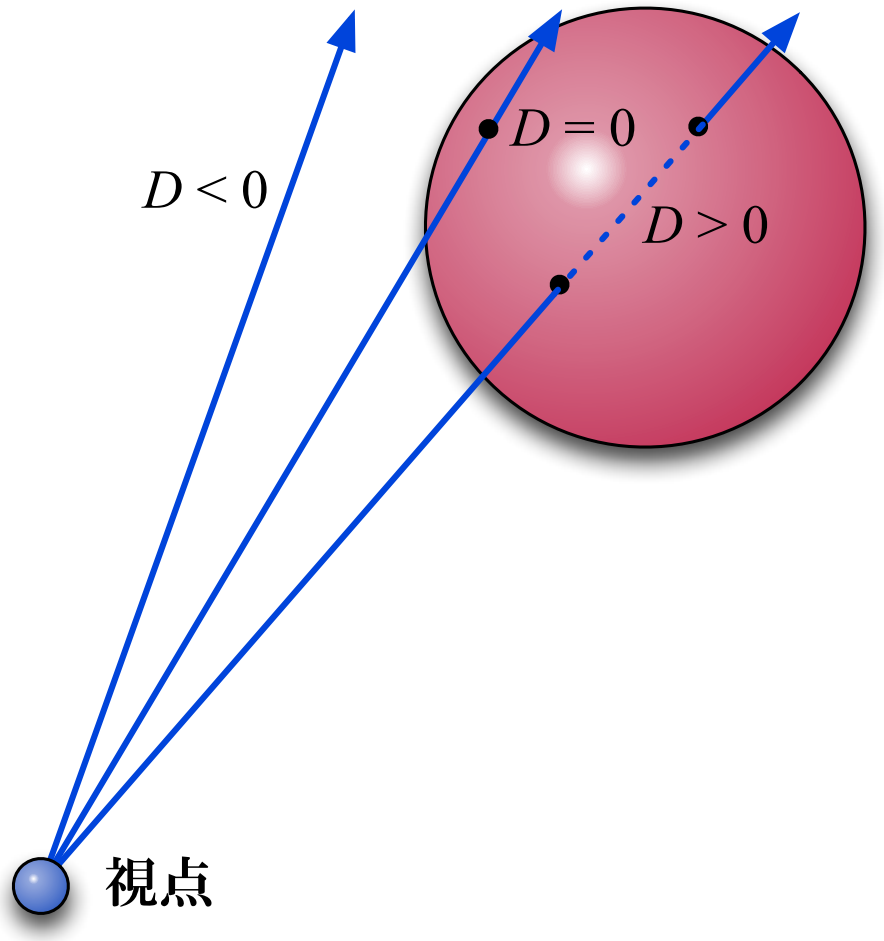
$$t = \frac{-B - \sqrt{D}}{A}$$

- 交点の位置

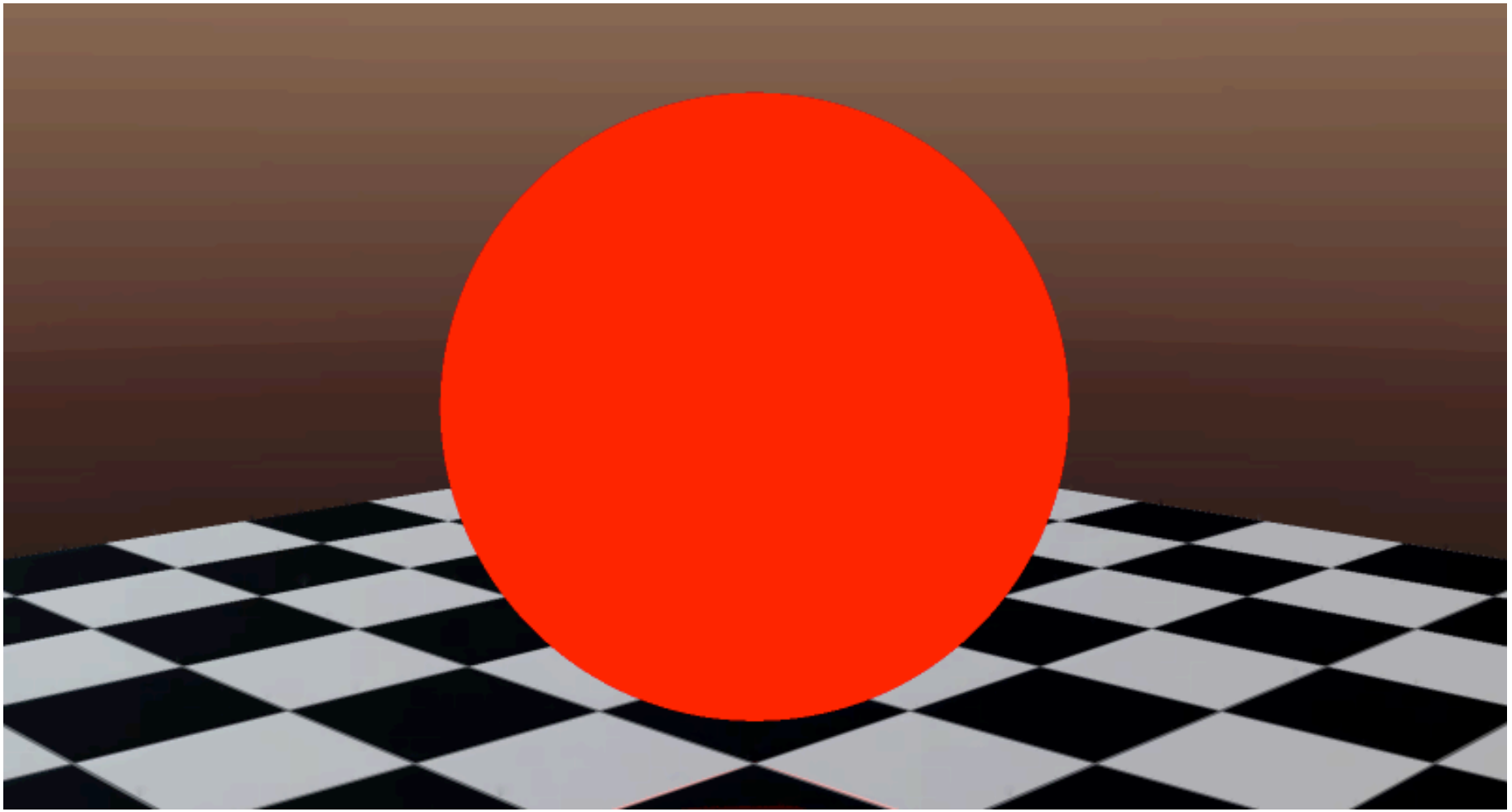
$$\mathbf{P} = \mathbf{E} + \mathbf{D}t$$

- 法線ベクトル

$$\mathbf{N} = \mathbf{P} - \mathbf{P}_c$$



球と交差した画素を赤にしてみる

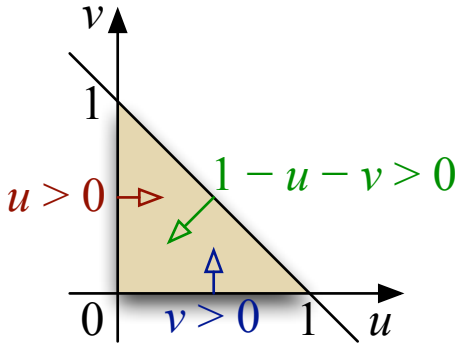


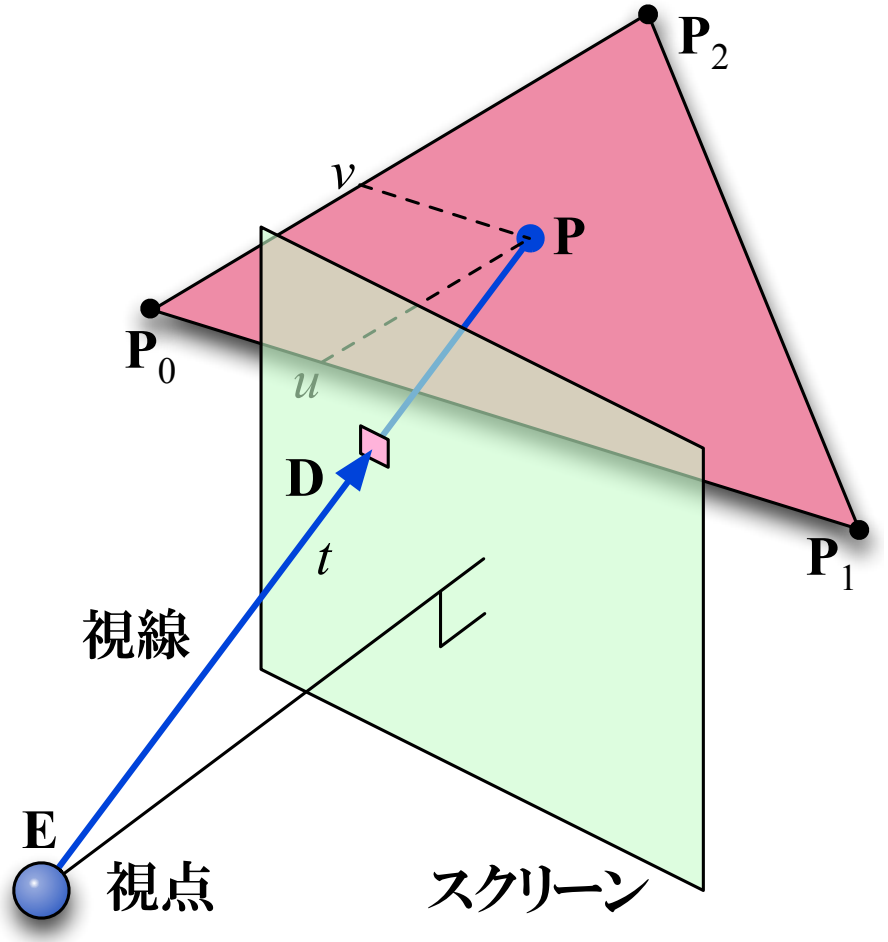
三角形の場合

- 三角形の頂点
  - $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$
- 三角形上の点

$$\mathbf{P} = (1-u-v)\mathbf{P}_0 + u\mathbf{P}_1 + v\mathbf{P}_2$$

$$\begin{cases} u > 0 \\ v > 0 \\ 1-u-v > 0 \end{cases}$$





視線と三角形の交点 (1)

- 視線上の点と三角形上の点が一致する

$$\mathbf{P} = (1-u-v)\mathbf{P}_0 + u\mathbf{P}_1 + v\mathbf{P}_2 = \mathbf{E} + \mathbf{D}t$$

$$\Downarrow$$

$$-\mathbf{D}t + (\mathbf{P}_1 - \mathbf{P}_0)u + (\mathbf{P}_2 - \mathbf{P}_0)v = \mathbf{E} - \mathbf{P}_0$$

  - $\mathbf{V}_1 = \mathbf{P}_1 - \mathbf{P}_0$, $\mathbf{V}_2 = \mathbf{P}_2 - \mathbf{P}_0$, $\mathbf{T} = \mathbf{E} - \mathbf{P}_0$ とおけば

$$-\mathbf{D}t + \mathbf{V}_1 u + \mathbf{V}_2 v = \mathbf{T}$$

$$\Downarrow$$

$$\begin{pmatrix} -\mathbf{D} & \mathbf{V}_1 & \mathbf{V}_2 \end{pmatrix} \begin{pmatrix} t \\ u \\ v \end{pmatrix} = \mathbf{T}$$

視線と三角形の交点 (2)

- これを t, u, v について解く

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{\begin{vmatrix} -\mathbf{D} & \mathbf{V}_1 & \mathbf{V}_2 \end{vmatrix}} \begin{pmatrix} \begin{vmatrix} \mathbf{T} & \mathbf{V}_1 & \mathbf{V}_2 \end{vmatrix} \\ \begin{vmatrix} -\mathbf{D} & \mathbf{T} & \mathbf{V}_2 \end{vmatrix} \\ \begin{vmatrix} -\mathbf{D} & \mathbf{V}_1 & \mathbf{T} \end{vmatrix} \end{pmatrix}$$

  - $|\mathbf{A}\ \mathbf{B}\ \mathbf{C}| = -(\mathbf{A}\times\mathbf{C})\cdot\mathbf{B} = -(\mathbf{C}\times\mathbf{B})\cdot\mathbf{A}$（スカラー三重積）

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{(\mathbf{D}\times\mathbf{V}_2)\cdot\mathbf{V}_1} \begin{pmatrix} (\mathbf{T}\times\mathbf{V}_1)\cdot\mathbf{V}_2 \\ (\mathbf{D}\times\mathbf{V}_2)\cdot\mathbf{T} \\ (\mathbf{T}\times\mathbf{V}_1)\cdot\mathbf{D} \end{pmatrix} = \frac{-1}{(\mathbf{V}_1\times\mathbf{V}_2)\cdot\mathbf{D}} \begin{pmatrix} (\mathbf{V}_1\times\mathbf{V}_2)\cdot\mathbf{T} \\ (\mathbf{T}\times\mathbf{D})\cdot\mathbf{V}_2 \\ -(\mathbf{T}\times\mathbf{D})\cdot\mathbf{V}_1 \end{pmatrix}$$

法線ベクトルN

Tomas Möller and Ben Trumbore, “Fast, minimum storage ray-triangle intersection,” *Journal of Graphics Tools*, 2(1):21−28, 1997.

視線と三角形の交差判定

- u, v が以下の条件のとき交差する

$$(\mathbf{V}_1 \times \mathbf{V}_2) \cdot \mathbf{D} \neq 0$$

三角形は視線と平行でない

$$\begin{cases} u > 0 \\ v > 0 \\ 1 - u - v > 0 \end{cases}$$

交点は三角形の内部にある

u, v はスムースシェーディングの際やテクスチャ座標の補間に使える

$$t > 0$$

交点は視点より前方にある

- 交点の位置

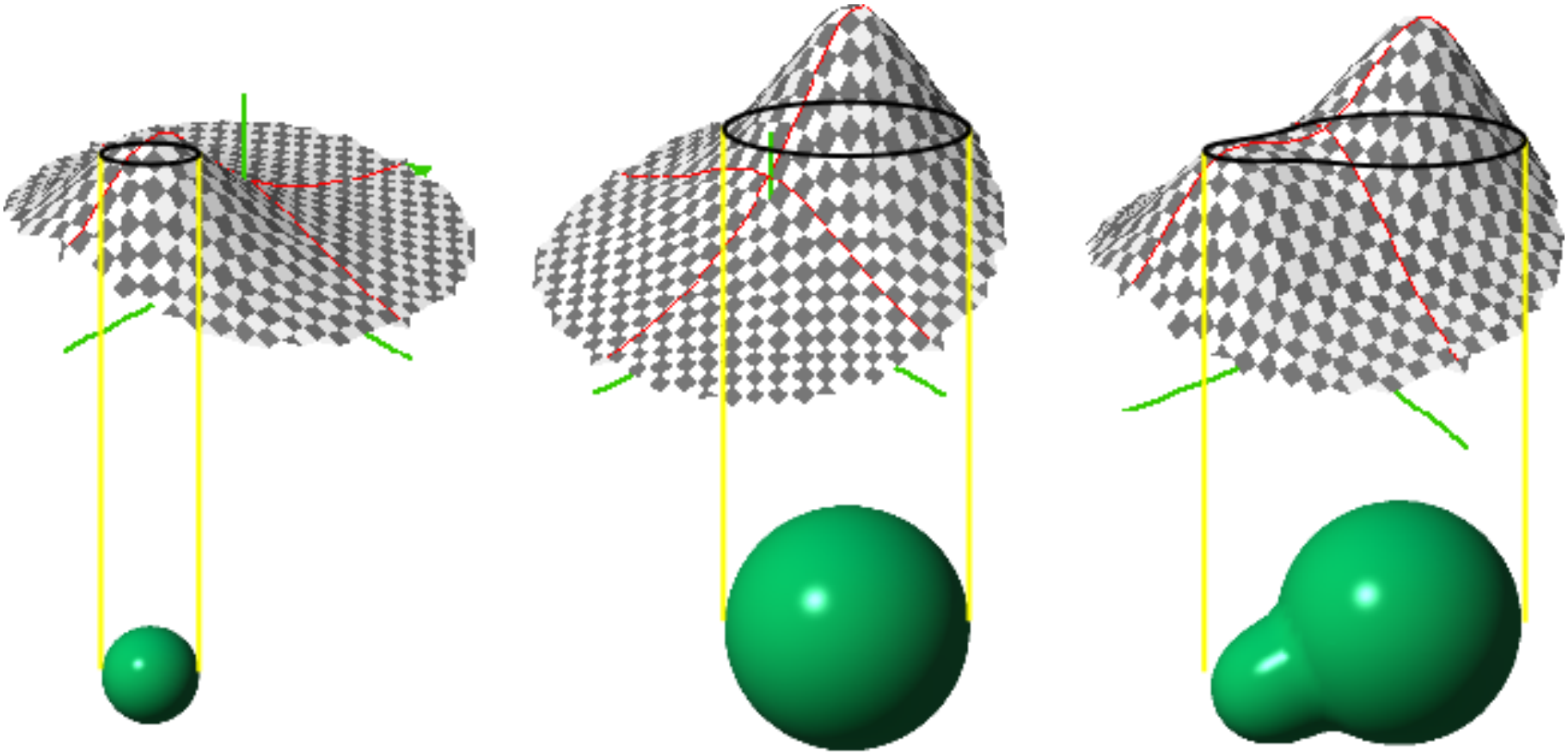
$$\mathbf{P} = \mathbf{E} + \mathbf{D}t$$

面の法線ベクトル $\mathbf{N} = \mathbf{V}_1 \times \mathbf{V}_2$ は既に計算している
あるいは頂点の法線ベクトルを (u, v) で補間する

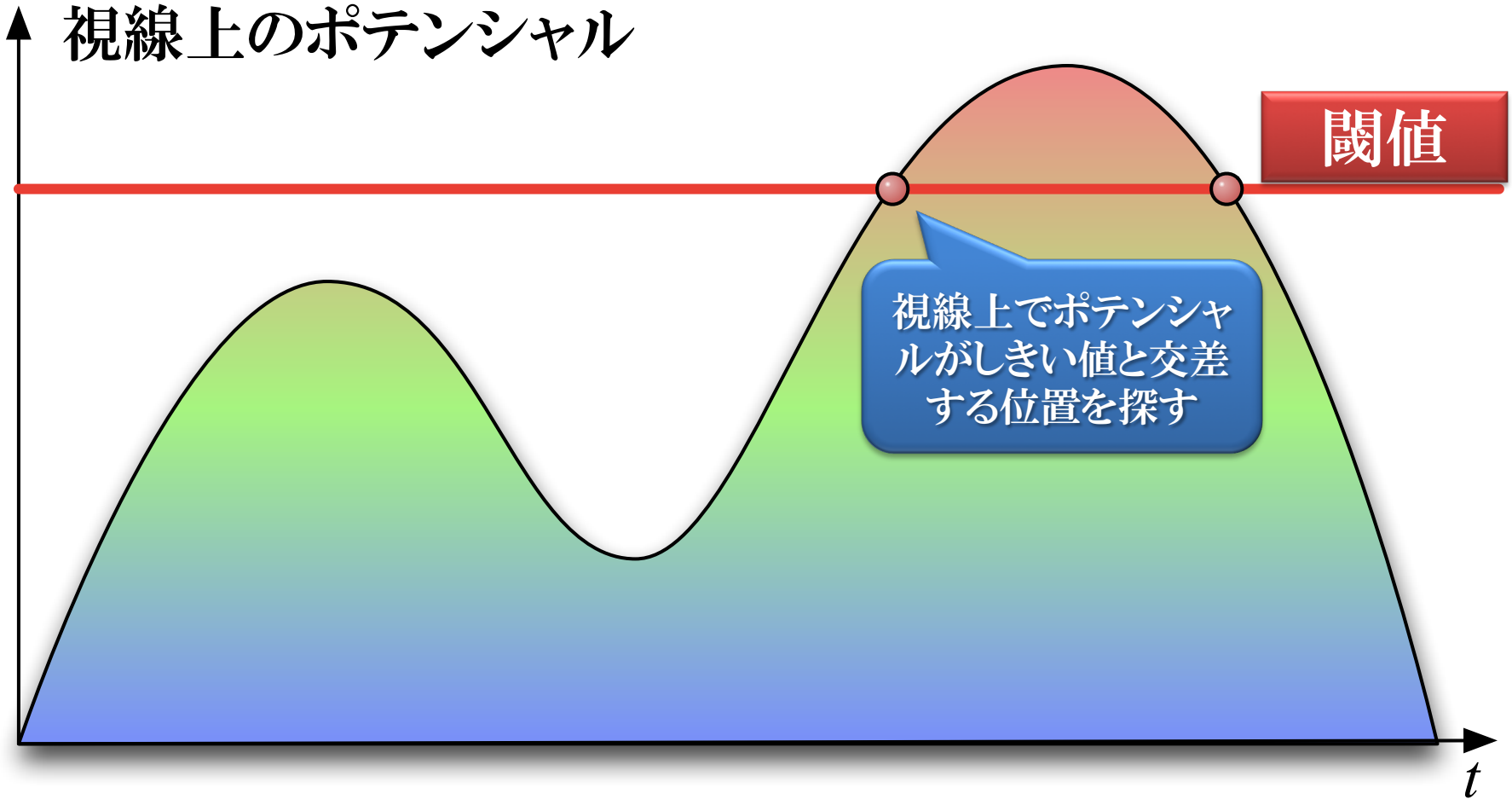
レイトレーシングに必要な要素

- 交差判定
  - 物体形状は直線との交差の有無を判定できるものである必要がある
- 交点位置の算出
  - 物体形状は直線との交点の位置を求めることができるものである必要がある
- 法線ベクトルの算出（陰影付けを行う場合）
  - 物体形状は表面上の一点における法線ベクトルを求められるものである必要がある

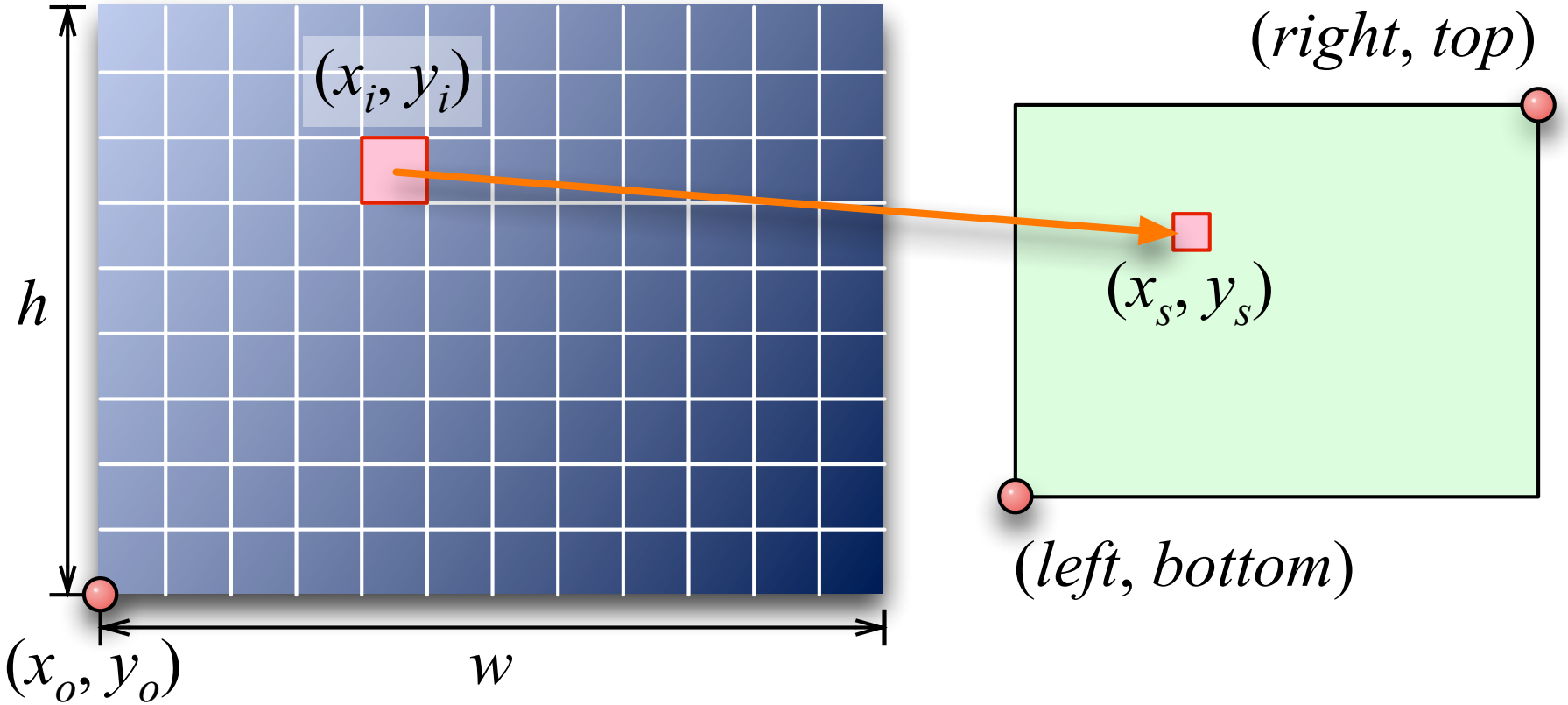
メタボール



メタボールの交差判定



ビューポートとスクリーン



ビューポート
（ディスプレイ上の表示領域）

スクリーン
（シーン中の投影面）

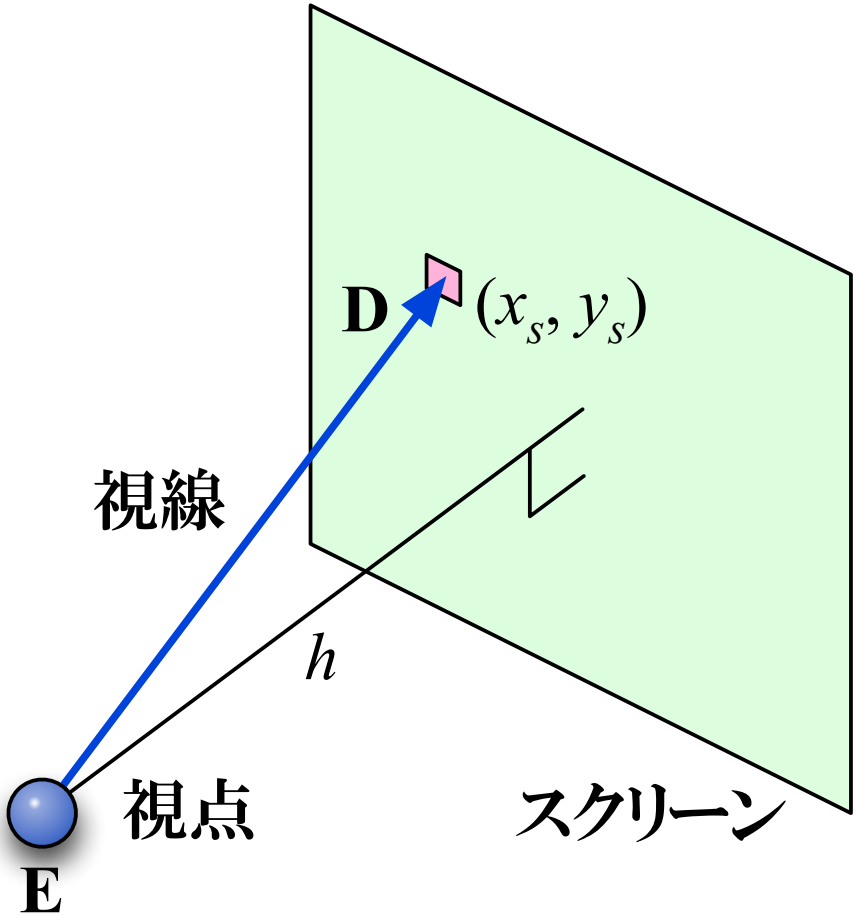
スクリーンマッピング

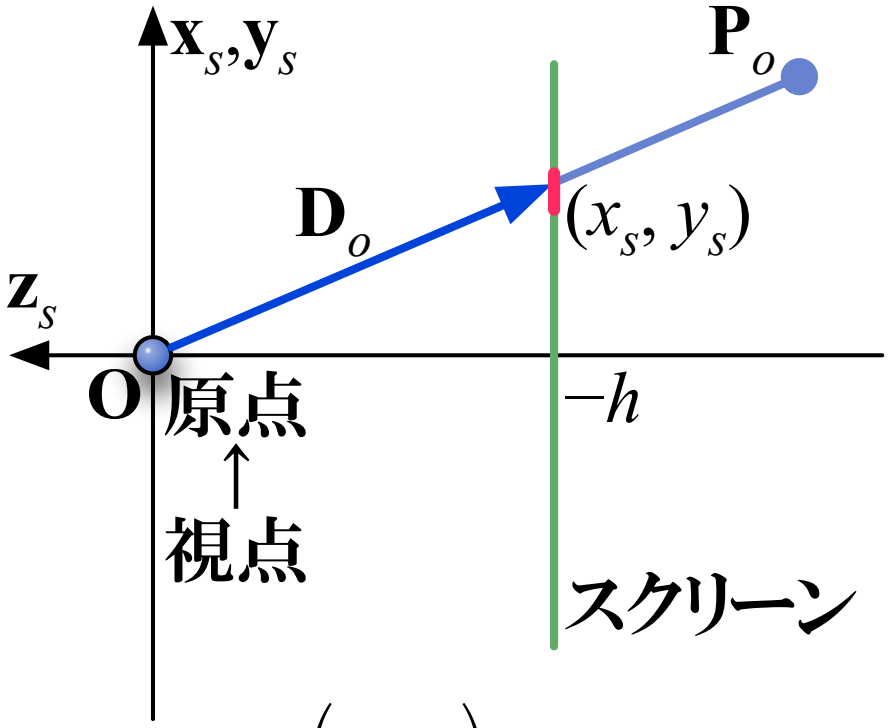
$$x_s = \frac{right - left}{w}\left(x_i - x_o\right) + left$$

$$y_s = \frac{top - bottom}{h}\left(y_i - y_o\right) + bottom$$

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} \frac{right - left}{w} & 0 & left - \frac{right - left}{w} x_o \\ 0 & \frac{top - bottom}{h} & bottom - \frac{top - bottom}{h} y_o \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

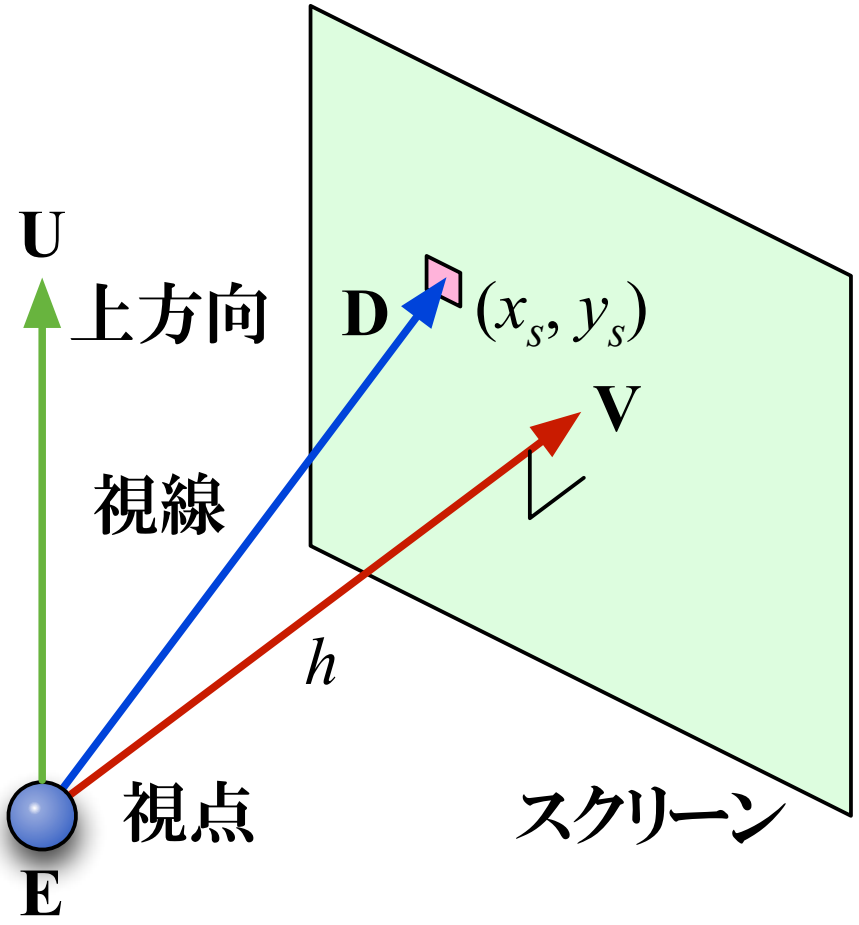
視線の発生





$$\mathbf{D}_o = \begin{pmatrix} x_s \\ y_s \\ -h \end{pmatrix}, \quad \mathbf{P}_o = \mathbf{D}_o t$$

視野変換



- 視線を回転する変換 $\mathbf{R}^T$
  - 第7回参照

$$\mathbf{z}' = -\frac{\mathbf{V}}{|\mathbf{V}|}$$

$$\mathbf{x}' = \frac{\mathbf{U} \times \mathbf{z}'}{|\mathbf{U} \times \mathbf{z}'|}$$

$$\mathbf{y}' = \mathbf{z}' \times \mathbf{x}'$$

$$\mathbf{R} = \begin{pmatrix} \mathbf{x}' & \mathbf{y}' & \mathbf{z}' \end{pmatrix}$$

物体ではなく視線を回転する

$$\mathbf{D} = \mathbf{R}\mathbf{D}_o$$

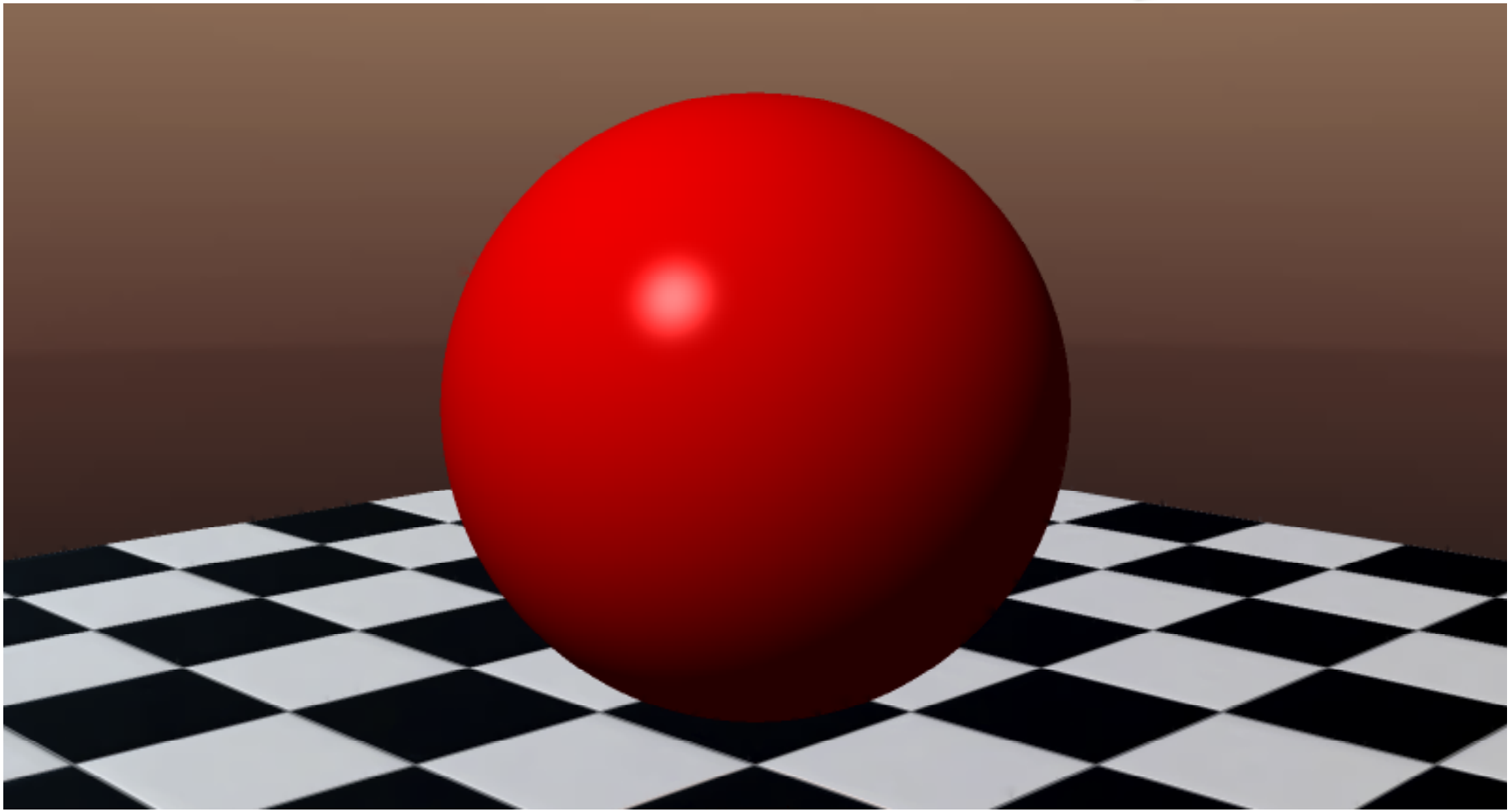
$$\mathbf{P} = \mathbf{E} + \mathbf{D}t$$

レンダリング手順

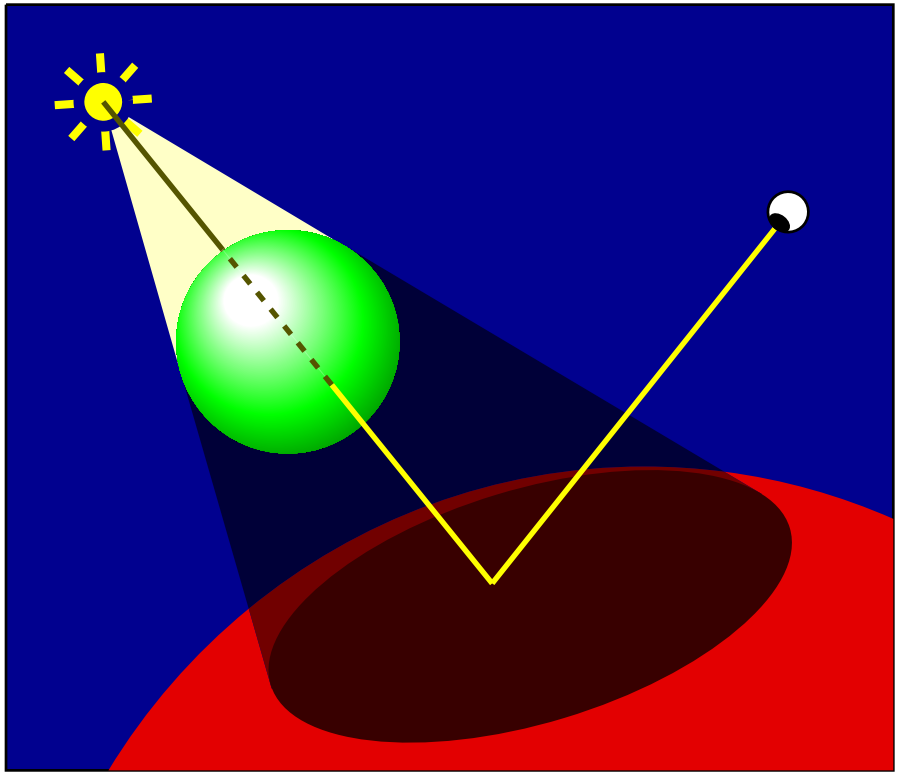
- 表示領域のすべての画素について
  - スクリーンマッピング
    - 画素位置 (x_i, y_i) からスクリーン上の位置 (x_s, y_s) を求める
  - 視線の生成
    - 原点を視点とした視線ベクトル $\mathbf{D}_o = (x_s, y_s, -h)$ を視野変換行列 $\mathbf{R}$ で回転して視線方向 $\mathbf{D} = \mathbf{R}\mathbf{D}_o$ を求める
  - 交差判定
    - 視線 $\mathbf{P} = \mathbf{E} + \mathbf{D}t$ とすべての物体との交差の有無を調べる
  - 可視判定
    - 交点のうち最も視点に近い (t が最小の) 可視点を見つける
  - 陰影付け
    - 可視点における法線を求めて陰影を計算し画素の色とする

陰影付け

ちくわ大明神

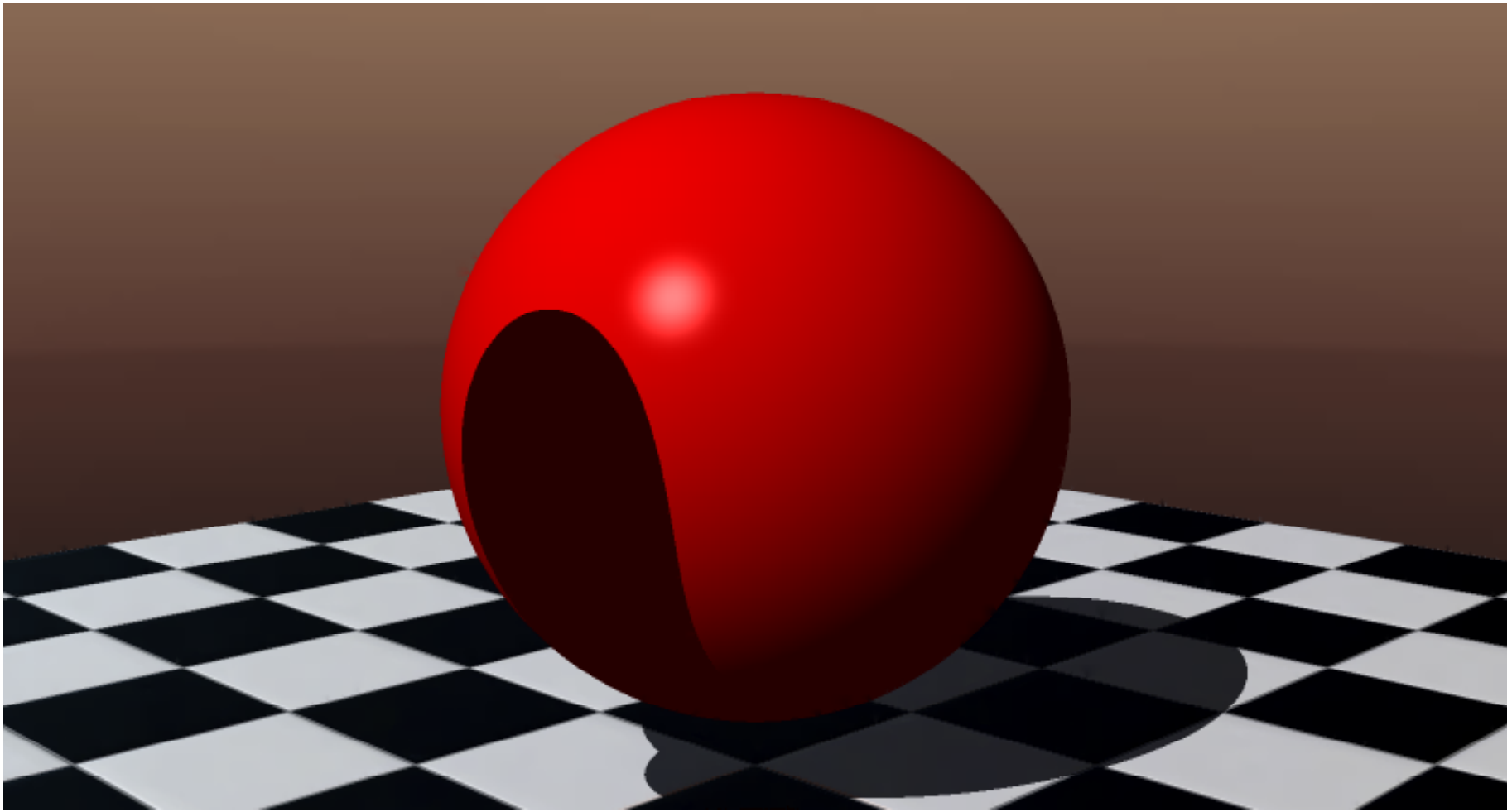


光源と物体の間に遮蔽物がある時

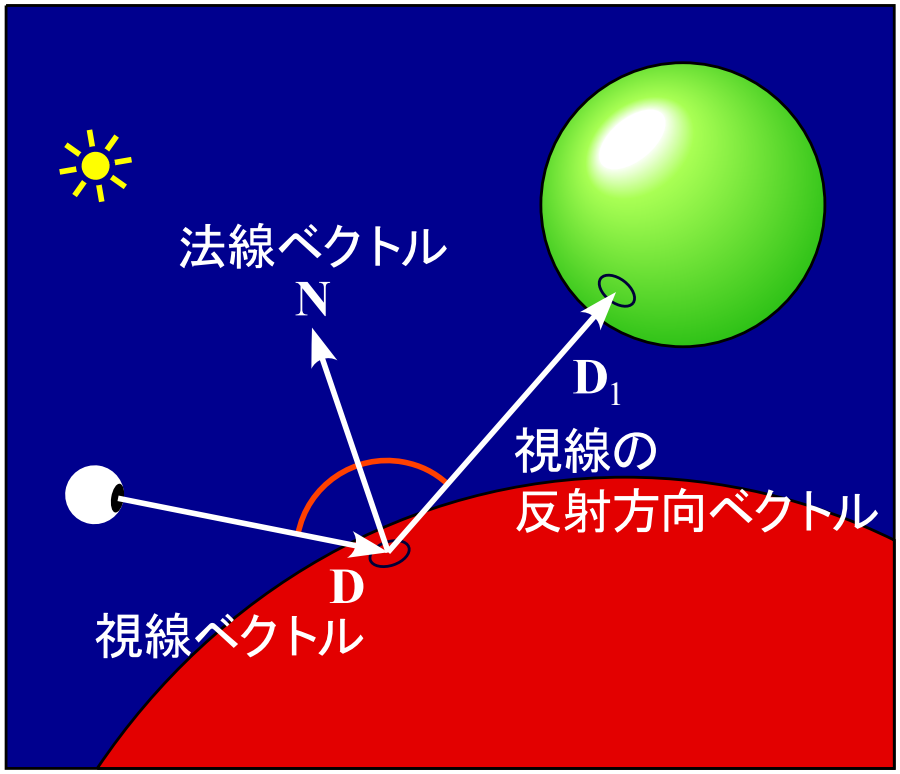


- 可視点と光源の間に別の物体があるかどうか調べる
  - 物体と視線との交点から光源に向かう半直線を新たな視線として交差判定を行う
- 別の物体がひとつでもあれば，その可視点は影
  - 光源の明るさを0（大域環境光のみ）にして陰影を求める

交点と光源の間の遮蔽物による影



視線の反射方向ベクトルを追跡

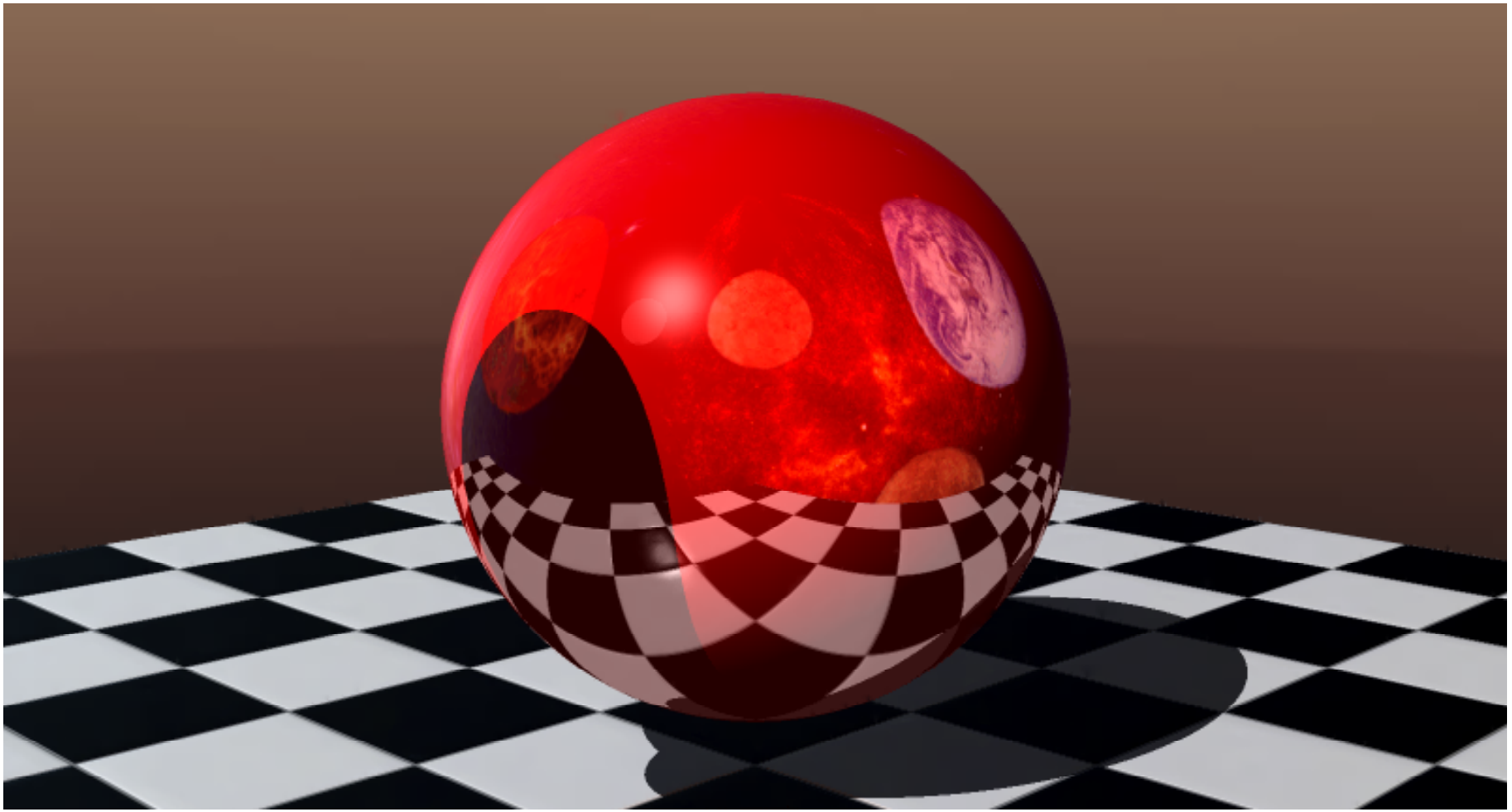


- 可視点を視点として反射方向に何が見えるか調べる
  - 視線の反射方向ベクトル

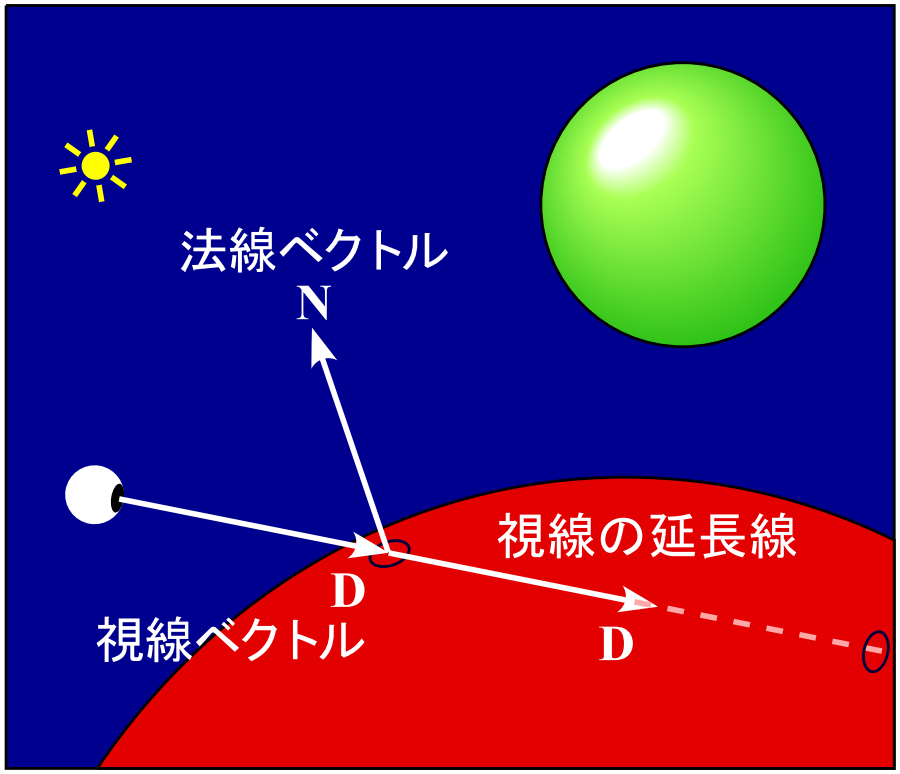
$$\mathbf{D}_1 = \mathbf{D} - 2(\mathbf{D}\cdot\mathbf{N})\mathbf{N}$$

- 反射方向に見えた色を可視点の色と合成する
  - 反射方向にある交点で陰影付けを行い，その反射光強度を光源強度として陰影付けする

映り込み

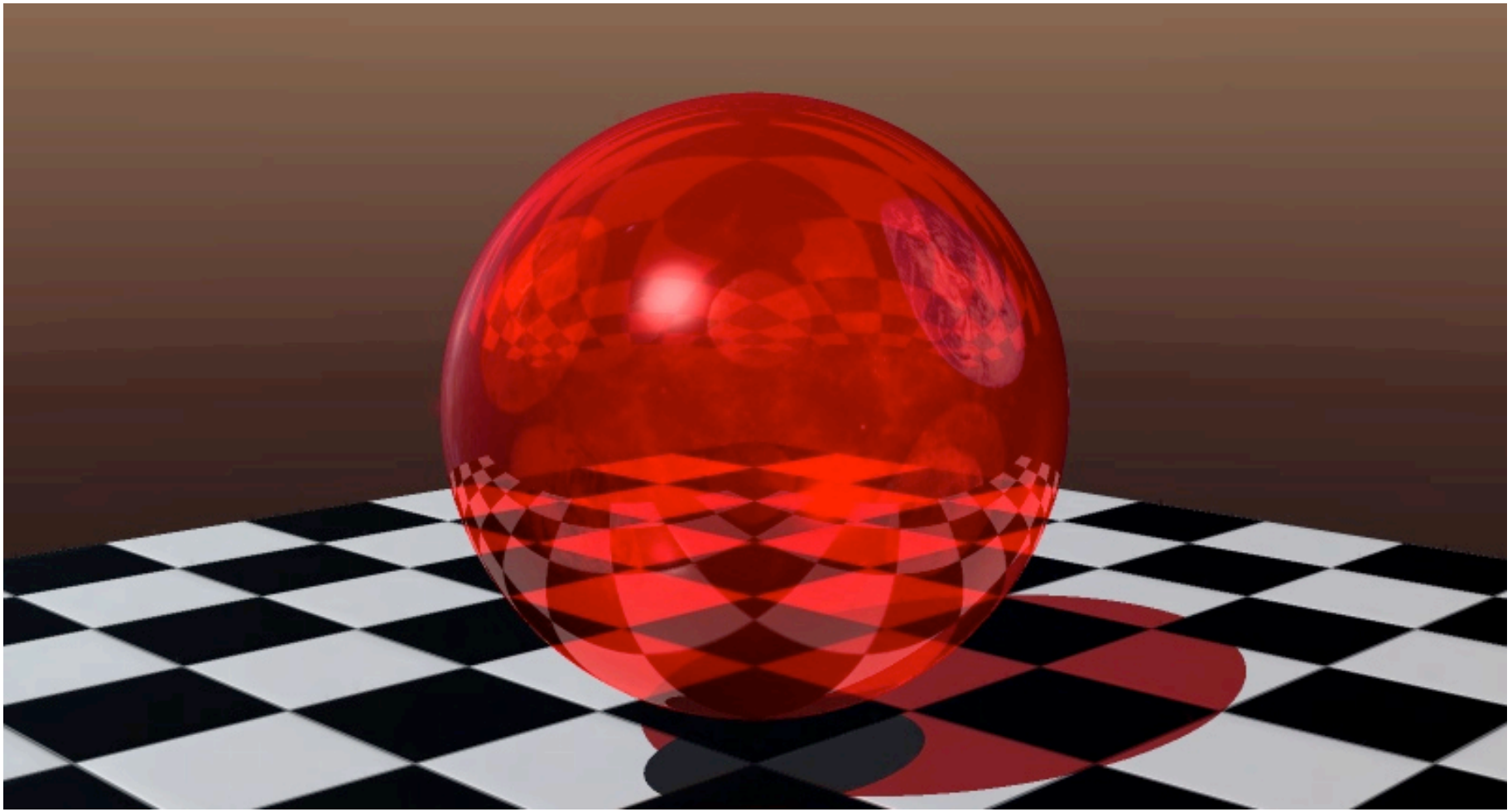


視線の延長線上を追跡する

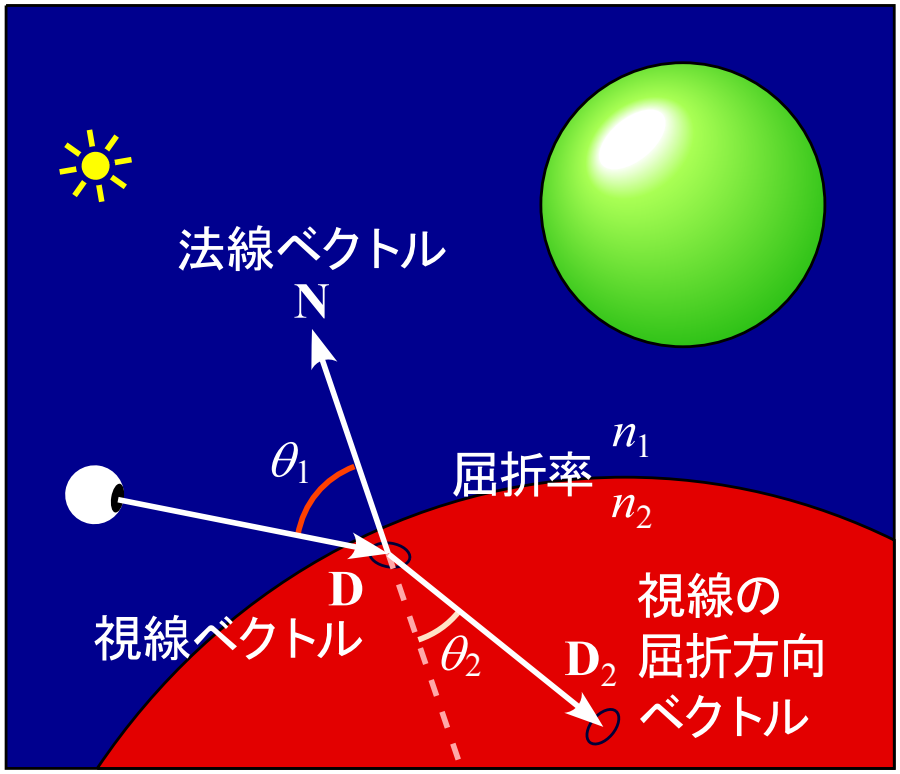


- 視線の延長線上に何が見えるか調べる
  - 延長線と交差する物体との交点を全て求める
- 延長線上に見えた色を可視点の色と合成する
  - 視点から遠い交点から順に陰影を加算合成する

透明



屈折率を考慮する



- 二つの媒質の屈折率 n_1, n_2 から屈折方向を求める
  - 屈折方向ベクトル

$$\mathbf{D}_2 = r\mathbf{D} + (w - k)\mathbf{N}$$

$$r = n_1 / n_2$$

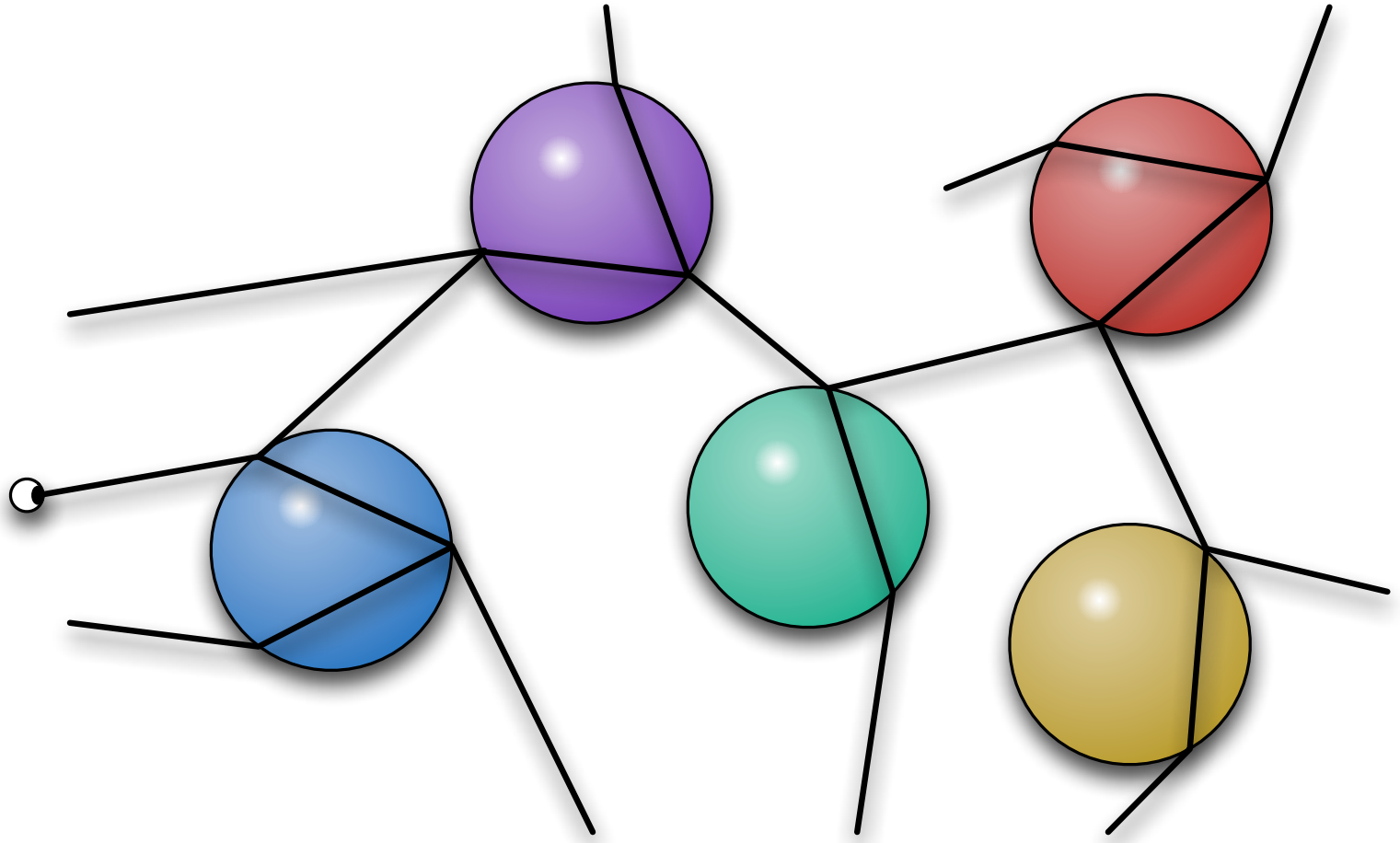
$$w = -(\mathbf{D} \cdot \mathbf{N})\mathbf{N}$$

$$k = \sqrt{1 + (w - r)(w + r)}$$

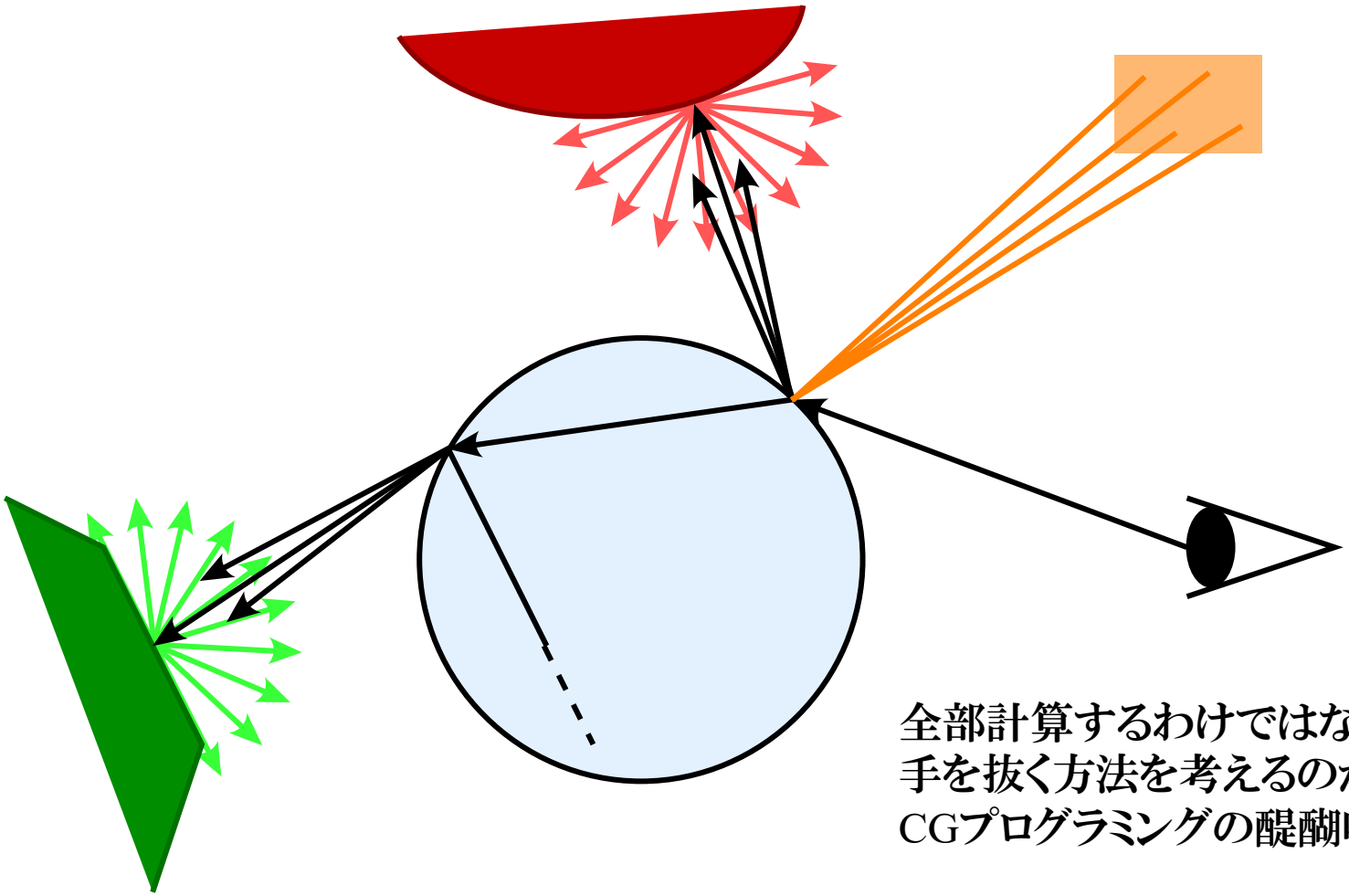
- 屈折方向に見えた色を可視点の色と加算合成する

屈折

視線は二分木になる



分散レイトレーシング



全部計算するわけではない
手を抜く方法を考えるのが
CGプログラミングの醍醐味

おわり大明神