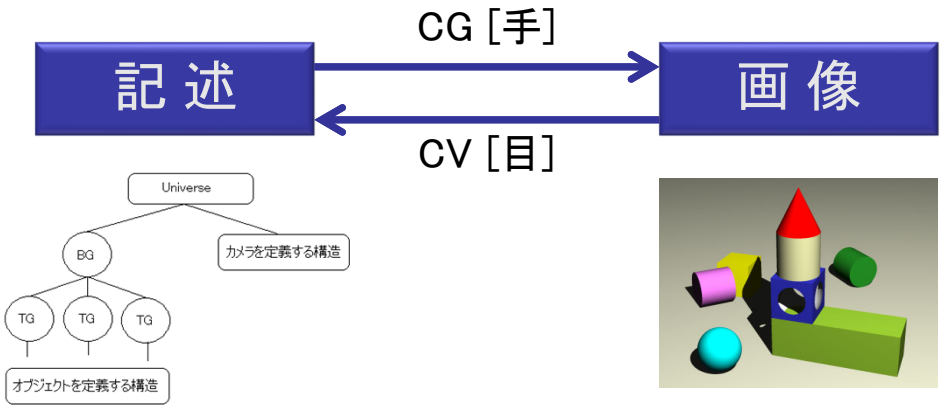
コンピュータグラフィックス
POV-Ray解説

情報工学実験第1A/B
春学期月曜日2～4限
担当教員：藤代 一成
ifujishiro@keio.jp
https://fj.ics.keio.ac.jp/

コンピュータグラフィックス（CG）とは

- コンピュータを利用して，数値データから対応する画像を創り出す技術の総称
- 画像合成（image synthesis）
  cf. コンピュータビジョン（CV），画像解析（image analysis）。
- 詳細は
  - ✓ 藤代&斎藤（英）:「ビジュアルコンピューティングIA/B」（春学期）
  - ✓ 斎藤（英）&藤代:「ビジュアルコンピューティングⅡ」（秋学期）

画像からの
立体形状
キャプチャ



画像合成の手順

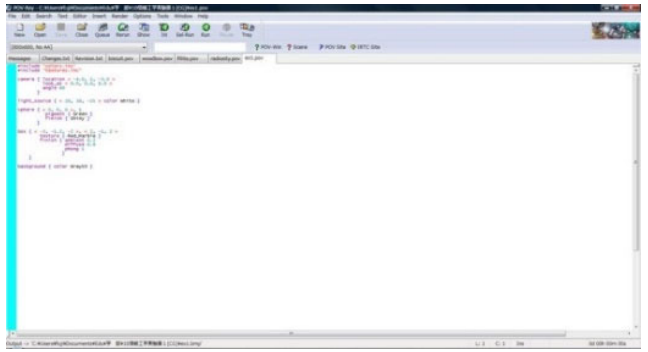
- モデリング：架空の3次元世界（シーン）を定義
  - ✓ 物体
  - ✓ 光源
  - ✓ カメラ
- レンダリング：モデリングの結果に基づいて画像を生成
  - ✓ レイトレーシング法

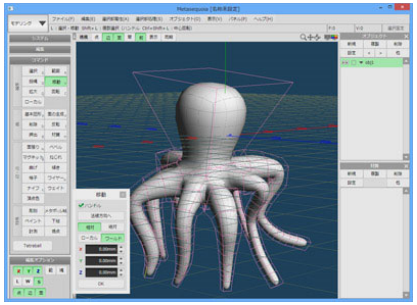
モデリング

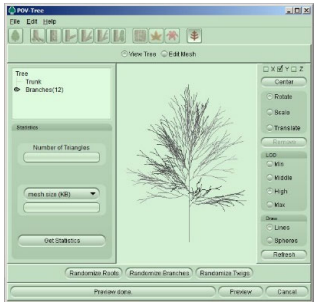
- 物体
  - ✓ 形の種類，位置，大きさ，色，表面の状態，光学的パラメタ（反射率，屈折率，等）
- 光源
  - ✓ 位置，色，大きさ，形，配光特性
- カメラ
  - ✓ 位置，ピントを合わせる位置，視野角

POV-Ray

- レイトレーシング法を実装　　POV：Persistence of Vision（残像）
- フリーソフトウェア
  - ✓ 公式サイト：http://www.povray.org/
  - ✓ 実験ではWindows 3.7.0版を利用（β版の3.8.0もあり）
  - ✓ OS：Windows, Linux
  - ✓ MacOS版：
    http://megapov.inetart.net/povrayunofficial_mac/
    （※includeフォルダをパスに追加すること）
  - ✓ バイナリだけでなくソースコードも公開
- スクリプトベースのモデリング
  - ✓ WYSIWYGタイプのモデラーが主流
    - 例：Metasequoia（http://www.metaseq.net/）
    - POV-Tree（http://f-lohmueller.de/pov_tut/plants/plants_400e.htm）
  - ✓ 内部処理の実際を体験





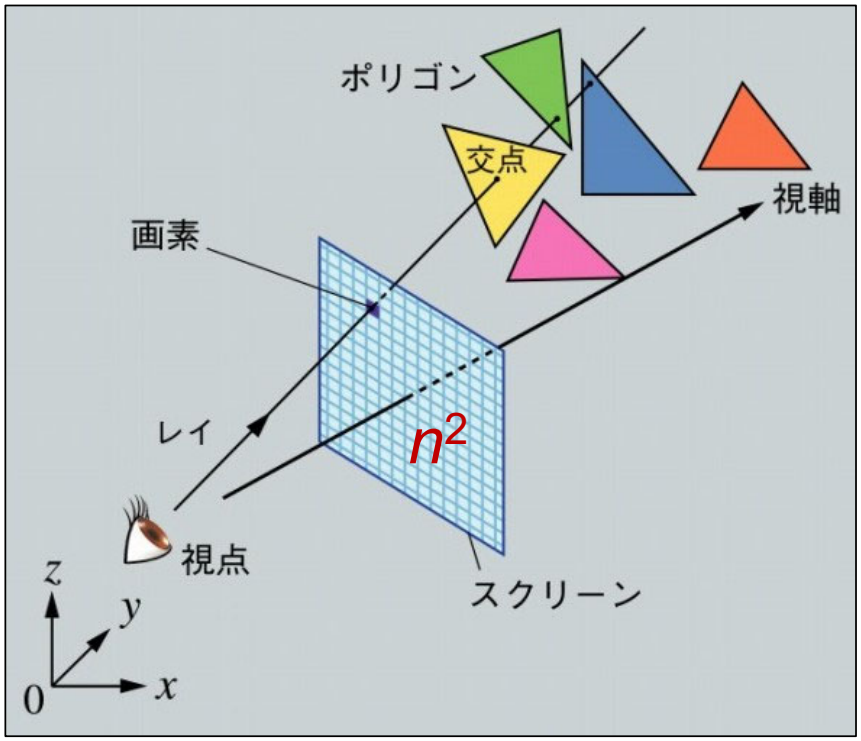


Hall of Fame

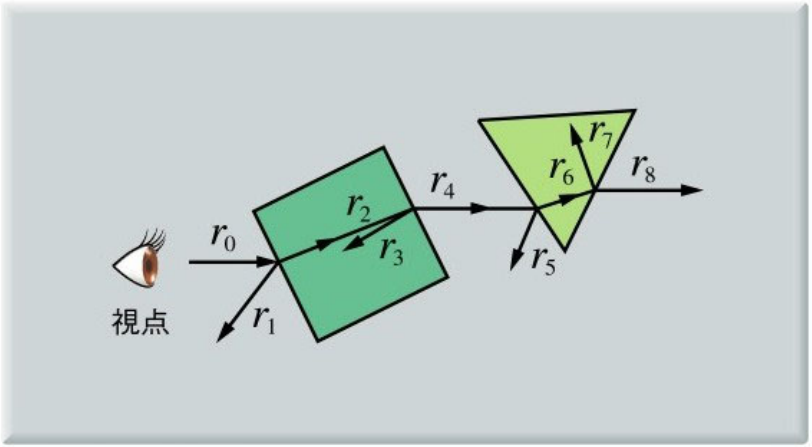
"Boreal" by Norbert Kern (2004)

レイトレーシング法 (pp. 2–3)

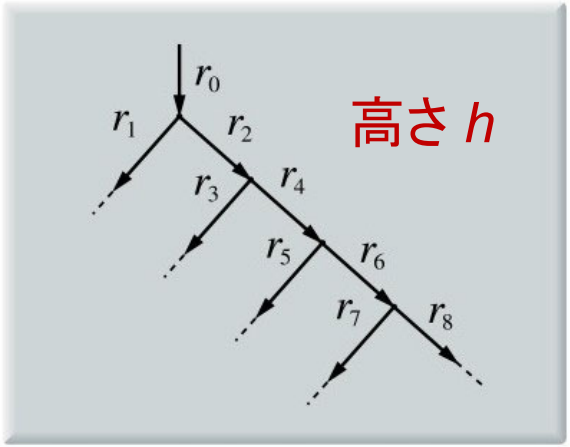
- T. Whitted (1980)
- 和名:「光線追跡法」



レイトレーシング法の原理



[a] 光の反射と屈折

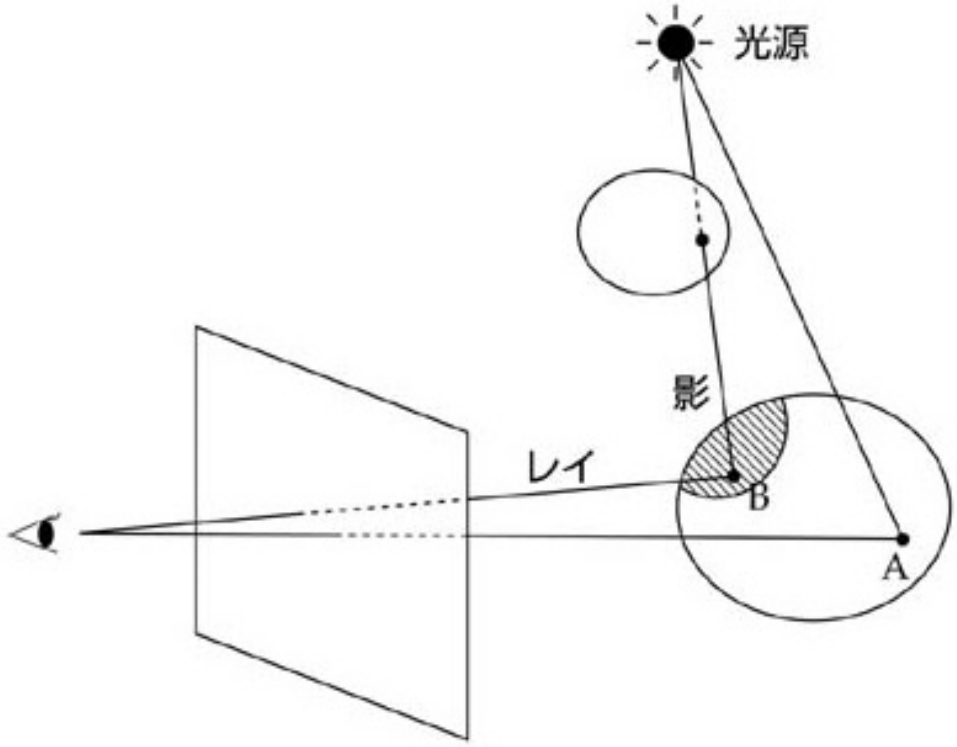


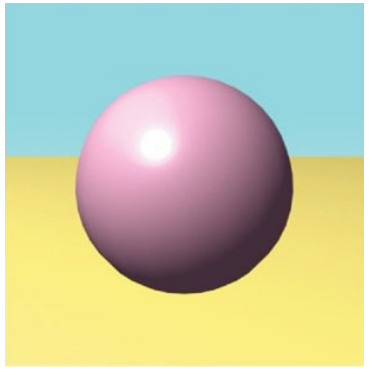
[b] 光線の追跡過程の二分木表現

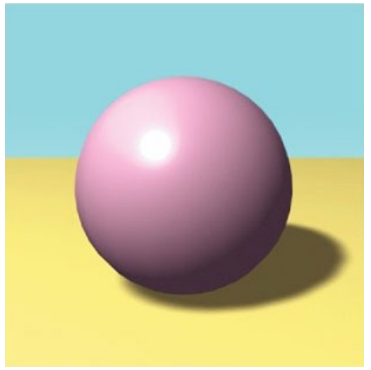
- 逆追跡：目に届く光線に限定して計算量を削減
- 画素ごとに鏡面・透過表面におけるレイの強度（色）を再帰的に評価・合成
- 追跡の停止条件
  - ✓ 完全拡散反射に近い物体に光があたったとき
  - ✓ 光源に達したとき
  - ✓ 定義されている空間の限界に達したとき
  - ✓ 反射・屈折が一定回数に達したとき

$\rightarrow o(n^2h)$

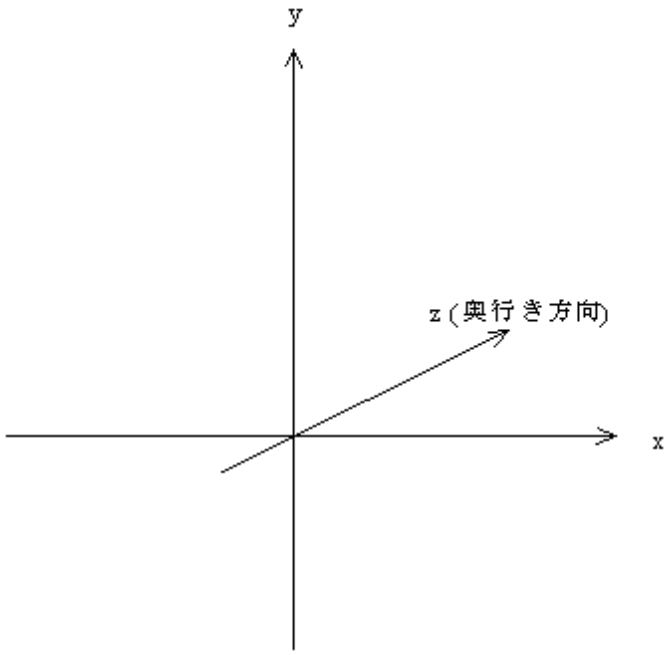
副作用:影領域の自然な算出







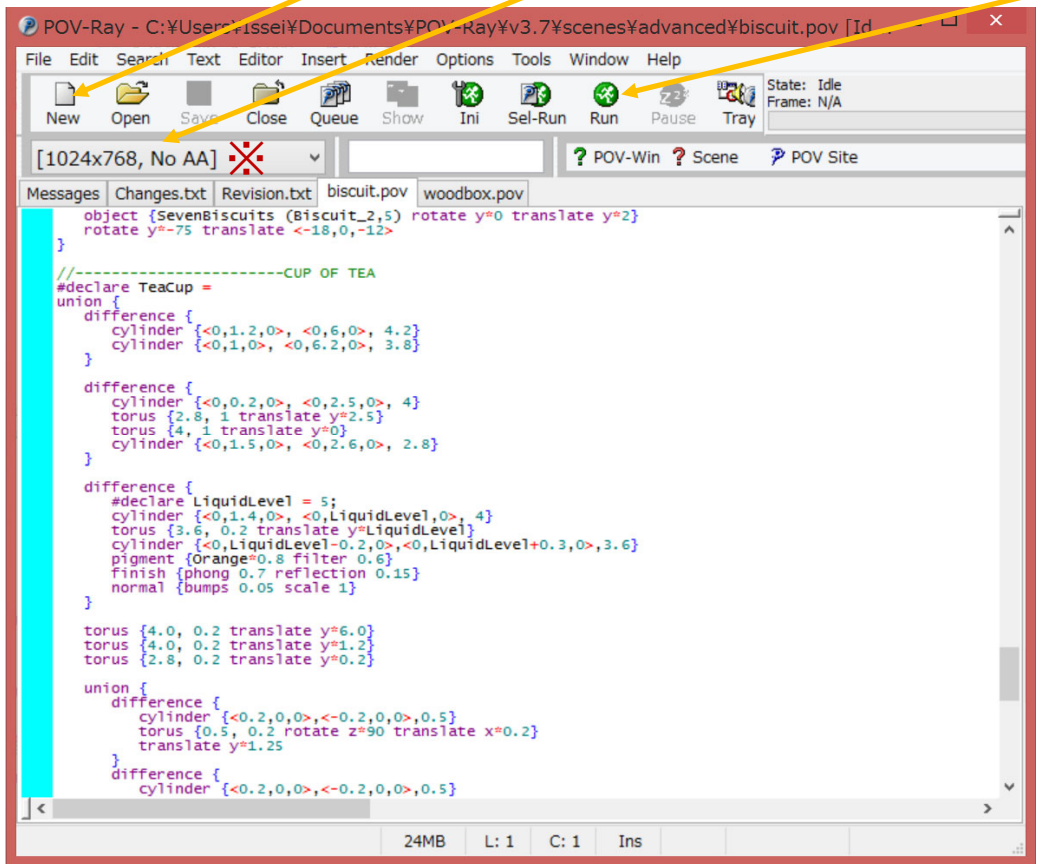
左手座標系 (p. 5)



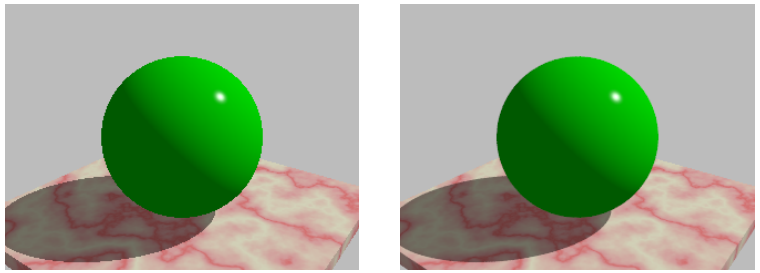
通常，*xz*平面を地面，*y*方向を高さとする

POV-Ray専用エディタ
シーンファイルの生成・コンパイル

シーン作成開始　画像サイズの変更　　画像の作成



シーンファイルの拡張子：*.pov

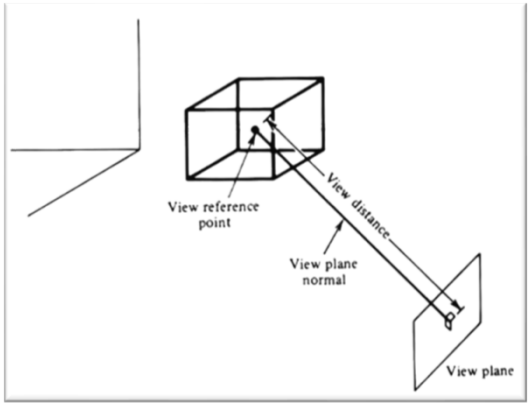


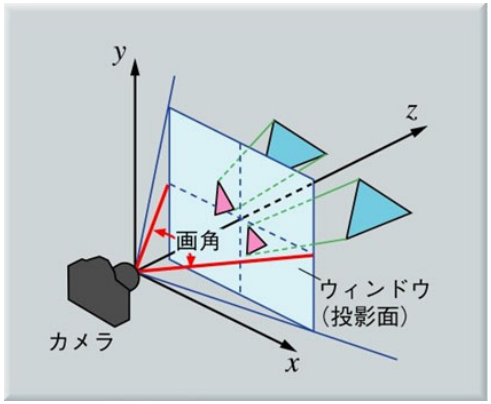
※AA: Anti-Aliasing：AA有りの方がブロッキーなノイズは軽減されるが，その分計算時間が増加

サンプルシーン（p. 6）

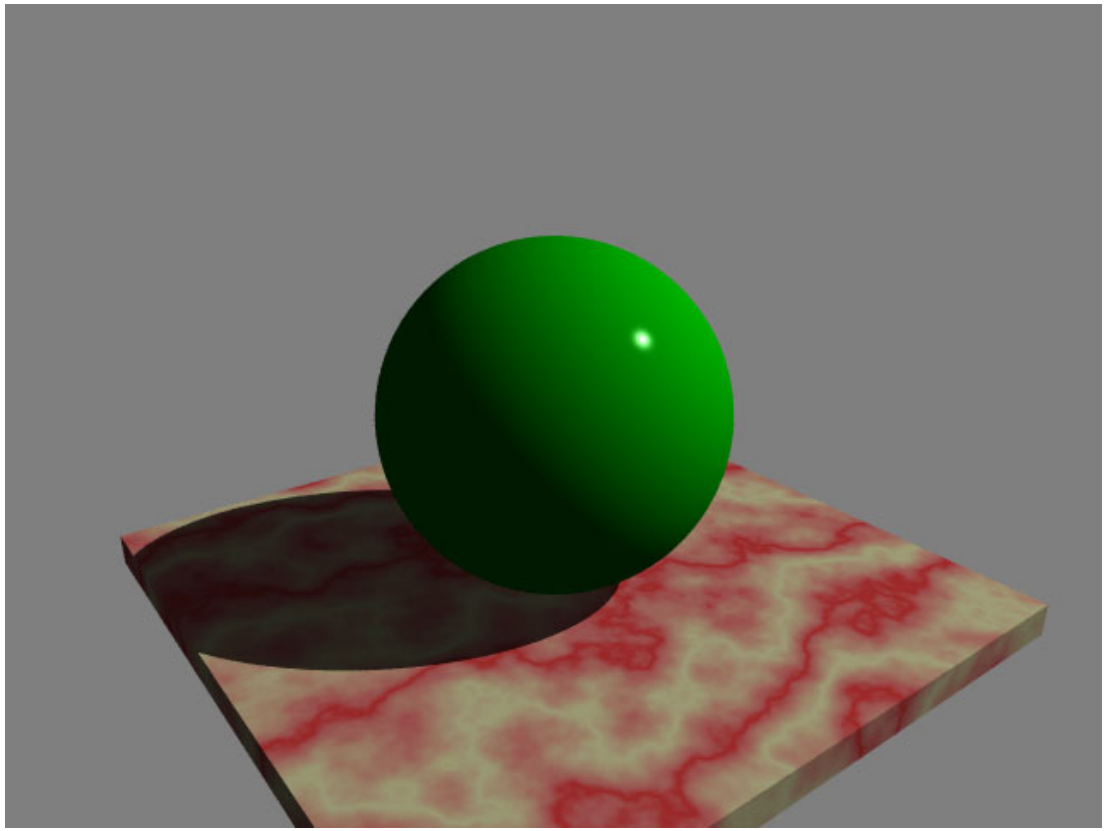
```
 1:  #include “colors.inc”
 2:  #include “textures.inc”
 3:
 4:  camera { location < -4.0, 2.0, -3.0 >
 5:               look_at  < 0.0, 0.0, 0.0 >
 6:               angle 60
 7:             }
 8:
 9:  light_source { < 20, 30, -25 > color White }
10:
11:  sphere { < 0, 0, 0 >, 1
12:              pigment { Green }
13:              finish { Shiny }
14:            }
15:
16:  box { < -2, -1.2, -2 >, < 2, -1, 2 >
17:           texture { Red_Marble }
18:           finish { ambient 0.2
19:                       diffuse 0.8
20:                       phong 1
21:                    }
22:        }
23:
24:  background { color Gray50 }
```

※黒の背景は極力避けること





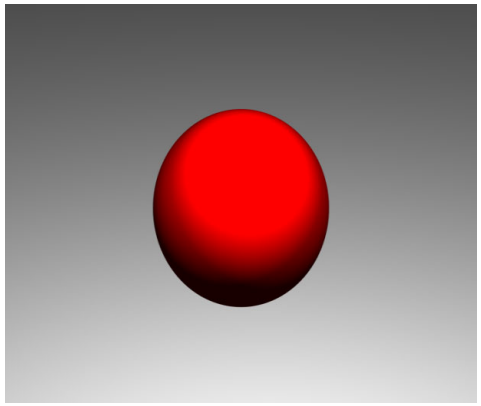
生成画像(p. 6)



コンパイルエラーがなければ，シーンファイルと同一のワーキングフォルダに同名の画像ファイル(*.png)が生成

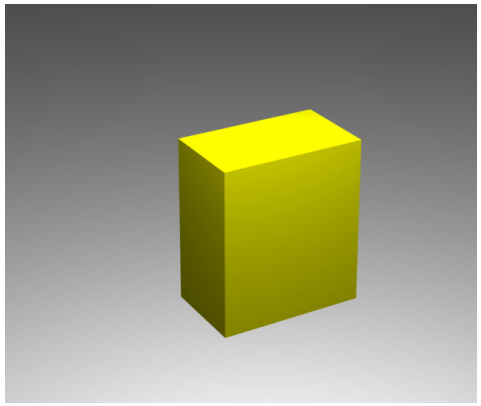
POV-Rayデモ

基本立体（pp. 10-12）



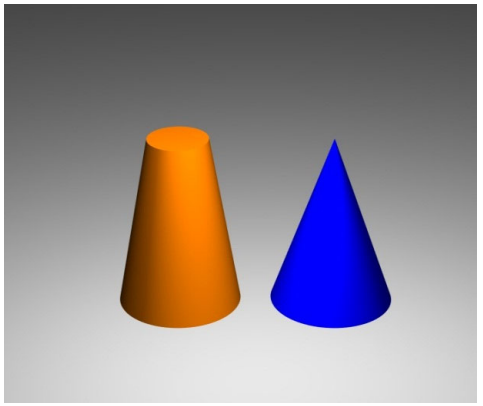
球

```
sphere{
  <0,2,2>, 3
  pigment{color Red}
}
```



直方体

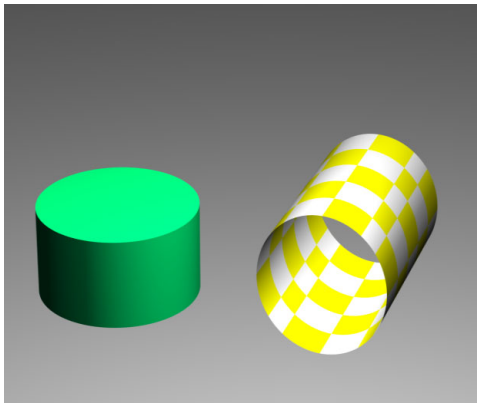
```
box{<0,0,0>,<3,5,5>
  pigment{color Yellow}
}
```



円錐（台）

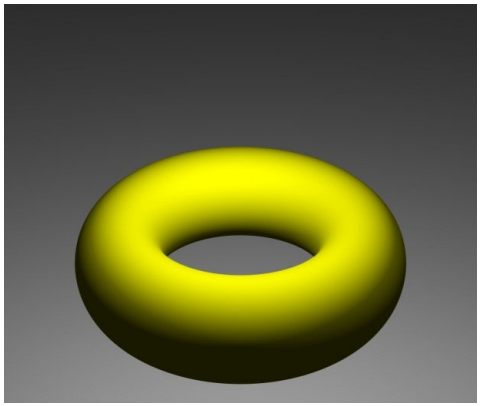
```
cone{<0,0,0> 2,<0,5,0> 1
  pigment{color Orange}
}

cone{<0,0,0> 2,<0,5,0> 0
  pigment{color Blue}
}
```

基本立体(続き)



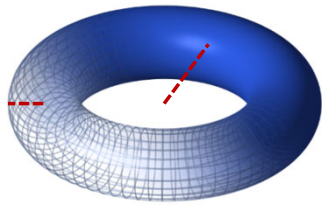
円　柱

```
cylinder{<0,0,0> ,<0,2,0>,2
  pigment{color SpringGreen}
}

cylinder{<0,0,0> ,<0,5,0>, 1.7
  open
  pigment{checker color Yellow color White}
}
```



トーラス

```
torus{5,1.8
  pigment{color Yellow}
}
```

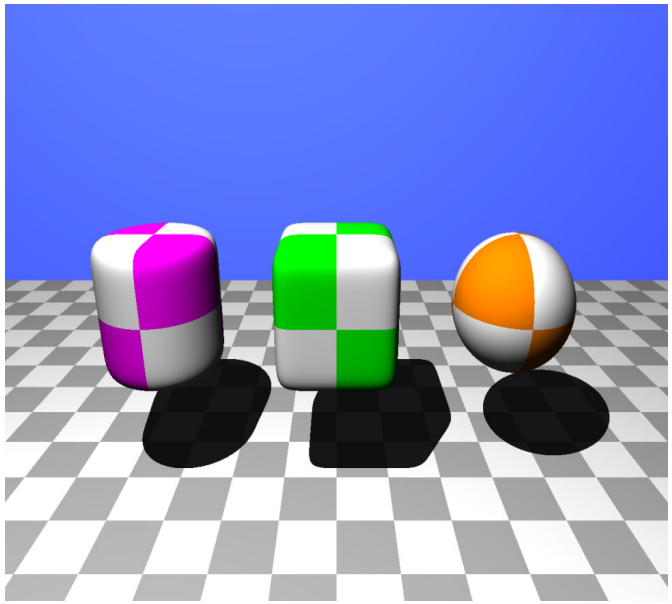


超楕円体（super ellipsoid）

$$\left(\left|x\right|^{\frac{2}{e}}+\left|y\right|^{\frac{2}{e}}\right)^{\frac{e}{n}}+\left|z\right|^{\frac{2}{e}}=1$$

$e=1, n=1$：球
$e=1, 0<n<1$：角丸円柱
$e=\mathrm{n}, 0<e<1, 0<n<1$：角丸直方体

```
superellipsoid{
  <1,0.25>
  pigment{checker Magenta White scale 1.5}
  rotate x*90
  translate <-3,0,0>
}

superellipsoid{
  <0.25,0.25>
  pigment{checker Green White scale 1.5}
}

superellipsoid{
  <1,1>
  pigment{checker Coral White scale 1.5}
  translate <3,0,0>
}
```



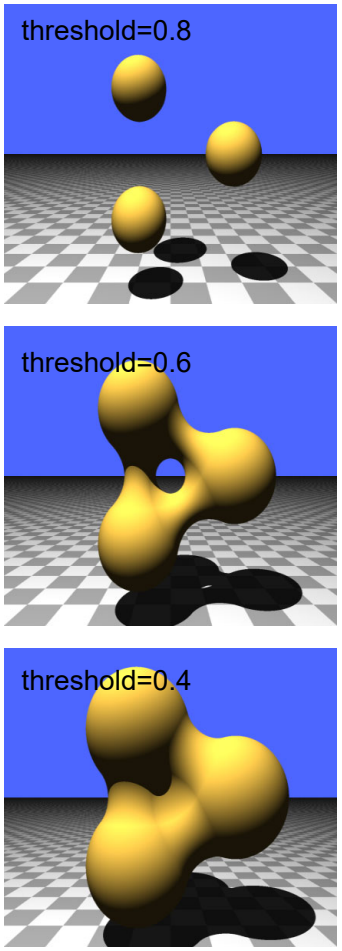
ブロブ（blobs, metaballs）

複数の濃度球の重畳分布の等濃度面

T−1000
T2（1991）

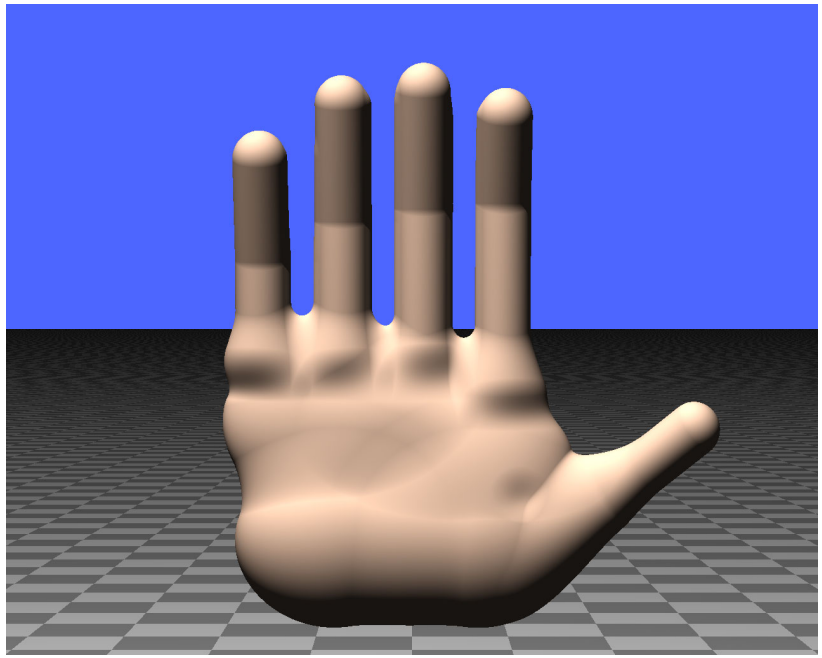
立体の名前はblob →
閾値（しきいち）の設定→
円柱（cylinder）の指定も可→
最後の引数は相対的強度→

```
#include "colors.inc"
#include "textures.inc"

camera{
  location<0,0,-6>
  look_at<0,0,0>
}

light_source{<-3,10,-7> color White*2}

background{color rgb<0.3,0.4,1.2>}

plane{
  <0,1,0>,-3
  pigment{checker Gray90 Gray60}
}

blob{
  threshold 0.6
  sphere{<.75,0,0>,1,1}
  sphere{<-.375, .64952,0>,1,1}
  sphere{<-.375,-.64952,0>,1,1}
  pigment{color rgb<1,0.8,0.3>}
  scale 2
}
```

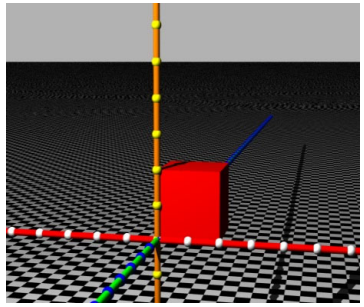
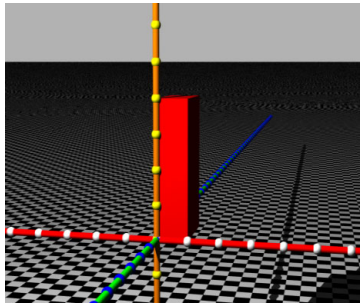
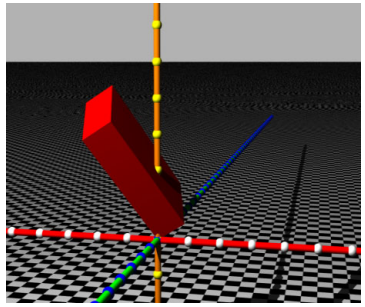
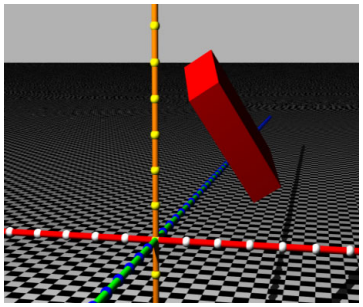


ブロブの利用例

```
blob{
  threshold 0.65

  sphere{<-0.65,0.28,-0.05>,0.26,-1}  //counteract pinky knuckle bulge
  sphere{<-0.65,-0.28, 0>,  0.26,-1}  //counteract pinky palm bulge
  sphere{<-0.3,0.44,-0.05>, 0.26,-1}  //counteract ring knuckle bulge
  sphere{<-0.3,-0.28,0>,    0.26,-1}  //counteract ring palm bulge
  sphere{<0.05,0.49,-0.05>, 0.26,-1}  //counteract middle knuckle bulge
  sphere{<0.05,-0.28,0>,    0.26,-1}  //counteract middle palm bulge
  sphere{<0.4,0.512,-0.05>, 0.26,-1}  //counteract index knuckle bulge
  sphere{<0.4,-0.4,0>,      0.26,-1}  //counteract index palm bulge
  sphere{<0.85,-0.68,-0.05>,0.25,-1}  //counteract thumb knuckle bulge
  sphere{<0.41,-0.7,0>,  0.25,-0.89}  //counteract thumb heel bulge

  sphere{<-0.23,-0.32,0>,0.43,1    scale<1.95,1.05,0.8>}   //palm
  sphere{< 0.12,-0.41,0>,0.43,1    scale<1.95,1.075,0.8>}  //palm
  sphere{<-0.23,-0.63,0>,0.45,0.75 scale<1.78,1.3,1>}      //midhand
  sphere{< 0.19,-0.63,0>,0.45,0.75 scale<1.78,1.3,1>}      //midhand
  sphere{<-0.22,-0.73,0>,0.45,0.85 scale<1.4,1.25,1>}      //heel
  sphere{< 0.19,-0.73,0>,0.45,0.85 scale<1.4,1.25,1>}      //heel

  cylinder{<-0.65,-0.28, 0>,   <-0.65,0.28,-0.05>, 0.26,1}  //lower pinky
  cylinder{<-0.65, 0.28,-0.05>,<-0.65,0.68,-0.2>,  0.26,1}  //upper pinky
  cylinder{<-0.3, -0.28, 0>,   <-0.3,0.44,-0.05>,  0.26,1}  //lower ring
  cylinder{<-0.3,  0.44,-0.05>,<-0.3,0.9, -0.2>,   0.26,1}  //upper ring
  cylinder{< 0.05,-0.28, 0>,   < 0.05,0.49,-0.05>, 0.26,1}  //lower middle
  cylinder{< 0.05, 0.49,-0.05>,< 0.05,0.95,-0.2>,  0.26,1}  //upper middle
  cylinder{< 0.4, -0.4,  0>,   < 0.4,0.512,-0.05>, 0.26,1}  //lower index
  cylinder{< 0.4, 0.512,-0.05>,< 0.4,0.85,-0.2>,   0.26,1}  //upper index
  cylinder{< 0.41,-0.95, 0>,   < 0.85,-0.68,-0.05>,0.25,1}  //lower thumb
  cylinder{< 0.85,-0.68,-0.05>,< 1.2,-0.4,-0.2>,   0.25,1}  //upper thumb
    :
```



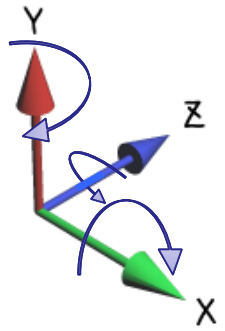
立体の幾何学的変換

①スケーリング　②回転　③平行移動

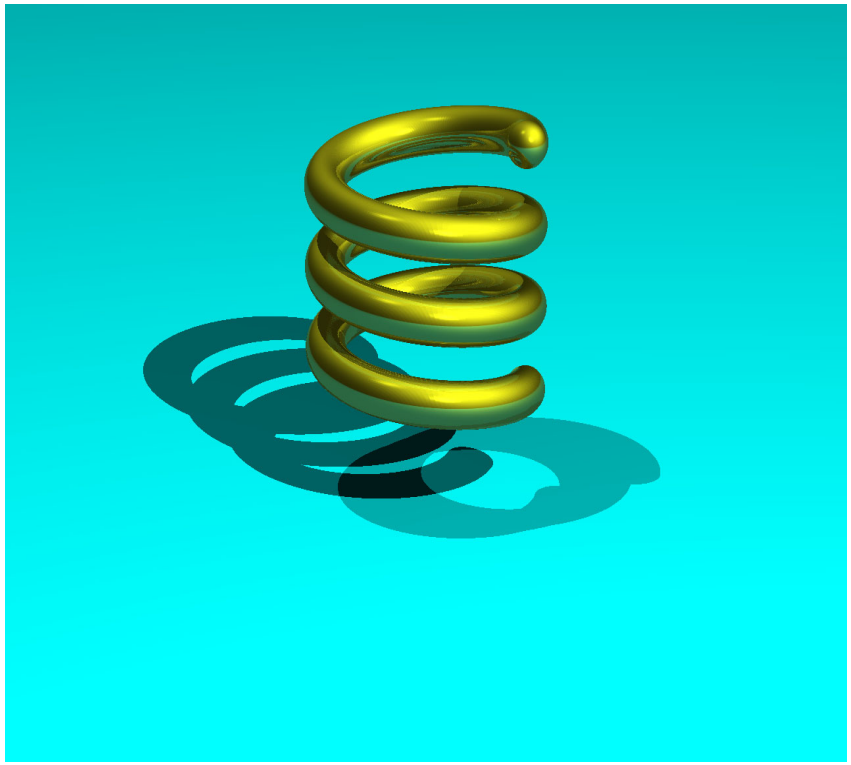
```
box{ <0,0,0>,<2,2,2>
 scale<0.5,2,1> ・・・①
 rotate<-30,0,30> ・・・②
 translate<3,1,1> ・・・③
 pigment{color Red}
}
```

※変換の順序に注意



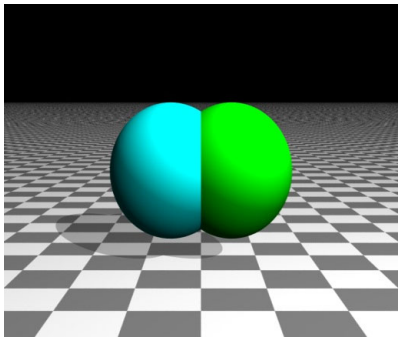
例：反復によるスィープ効果

```
#declare N=0;
#while (N<3)
  object{
    Sphere
    scale 0.2
    translate <1, N*0.7, 0>
    rotate <0, 360*N, 0>
    texture{Gold_Metal}
  }
  #declare N = N + 0.001;
#end
```

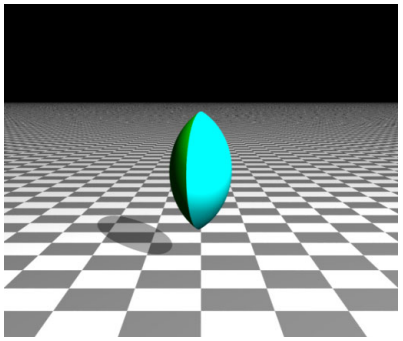


CSG：物体同士の集合演算

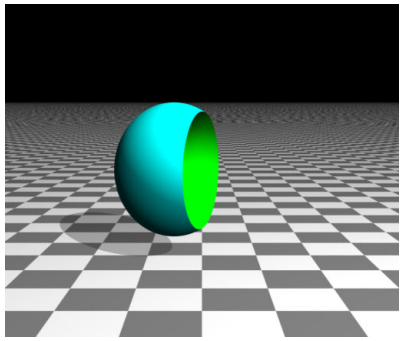
（例2: p. 8, pp. 13–14）



```
union{
  sphere { <0, 0, 0>, 1
    pigment { color Cyan}
    translate <-0.5,0,0>
  }
  sphere { <0, 0, 0>, 1
    pigment { color Green}
    translate <0.5,0,0>
  }
}
```



```
intersection{
  sphere{<0, 0, 0>, 1
    pigment{color Cyan}
    translate <-0.5,0,0>
  }
  sphere{<0, 0, 0>, 1
    pigment{color Green}
    translate <0.5,0,0>
  }
}
```



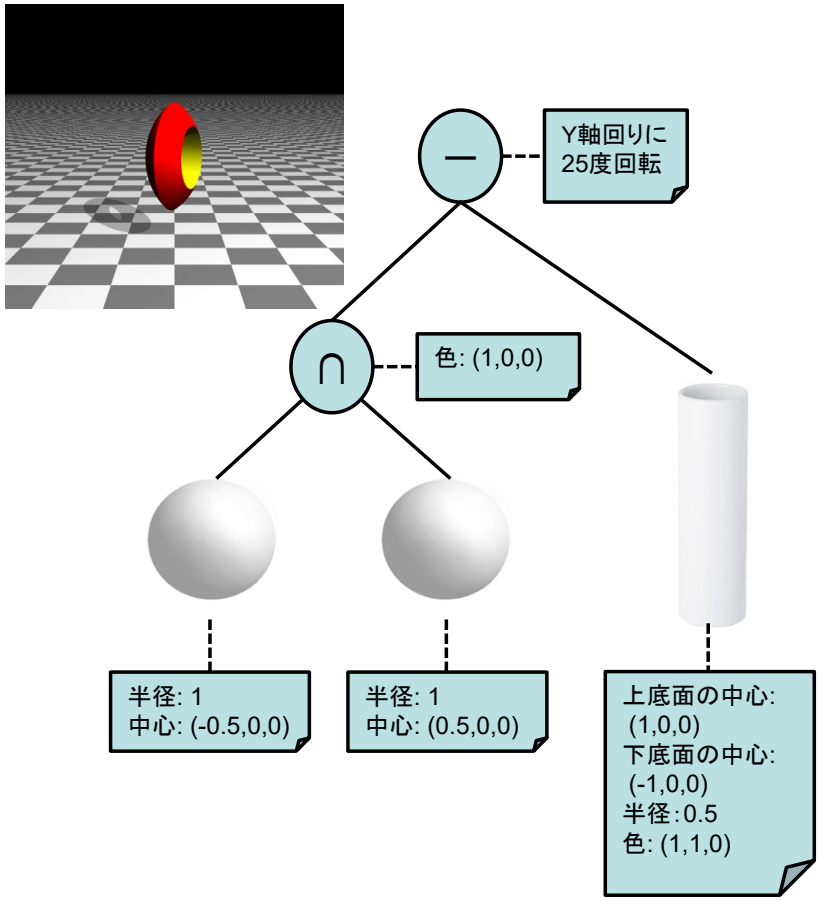
```
difference{
  sphere{<0, 0, 0>, 1
    pigment{color Cyan}
    translate <-0.5,0,0>
  }
  sphere{<0, 0, 0>, 1
    pigment{color Green}
    translate <0.5,0,0>
  }
}
```

cf. merge{}

CSG: Constructive Solid Geometry

N. Okino (1973)

CSG木



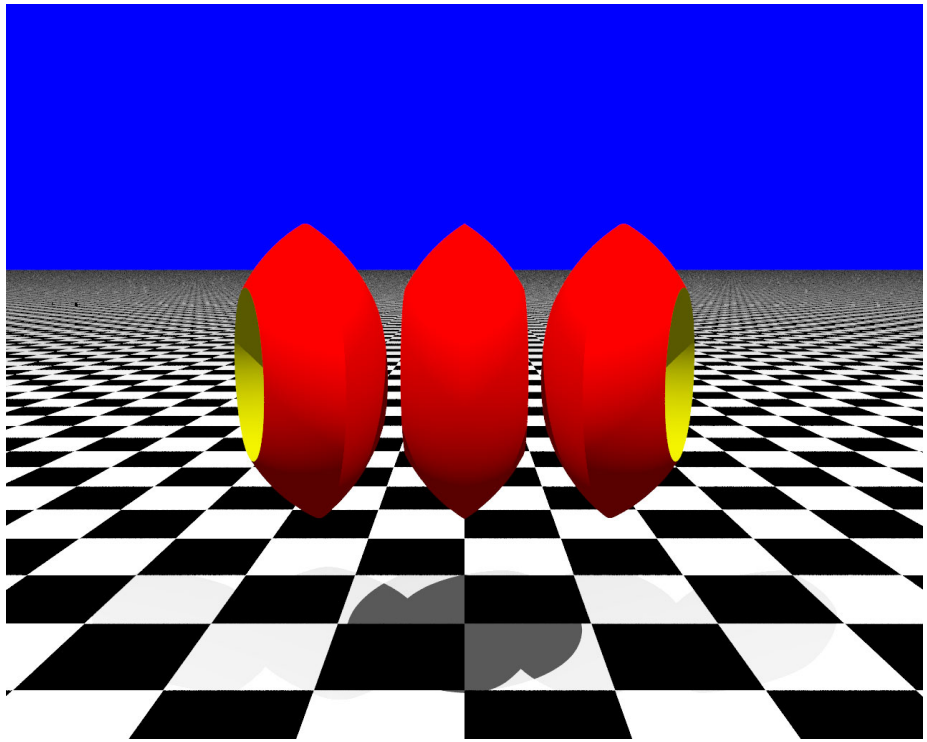

```
difference{
  intersection{
    sphere{<0, 0, 0>, 1
      translate <-0.5,0,0>
    }
    sphere{<0, 0, 0>, 1
      translate <0.5,0,0>
    }
    pigment{color Red}
  }

  cylinder{<-1, 0, 0> <1, 0, 0>, 0.5
    pigment{Yellow}
  }
  rotate <0,25,0>
}
```

※属性や変換の継承に注意

マクロの利用

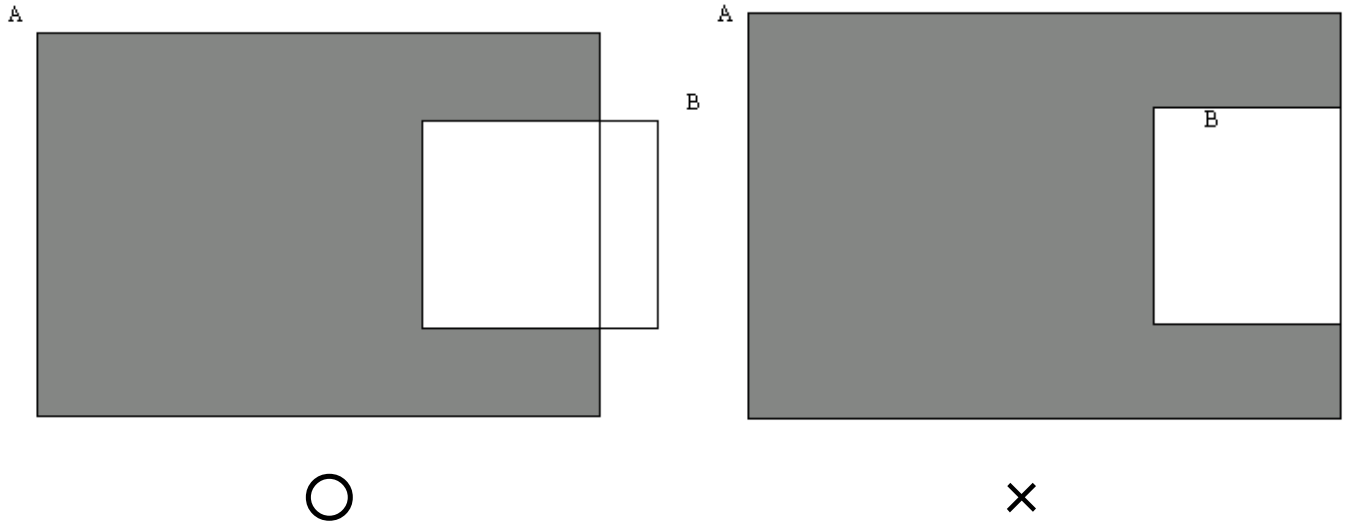
```
#macro Tama(theta)
  difference{
    intersection{
      sphere{<0, 0, 0>, 1
      translate <-0.5,0,0>
      }
      sphere{<0, 0, 0>, 1
      translate <0.5,0,0>
      }
      pigment{color Red}
    }
    cylinder{<-1, 0, 0> <1, 0, 0>, 0.5
    pigment{Yellow}
    }
   rotate <0,theta,0>
  }
#end
```

```
object {Tama(-25) translate <-1,0,0>}
object {Tama(  0) translate <  0,0,0>}
object {Tama( 25)  translate < 1,0,0>}
```



CSGモデリング上の注意

厚さゼロの膜（非多様体）をつくらないようにすること

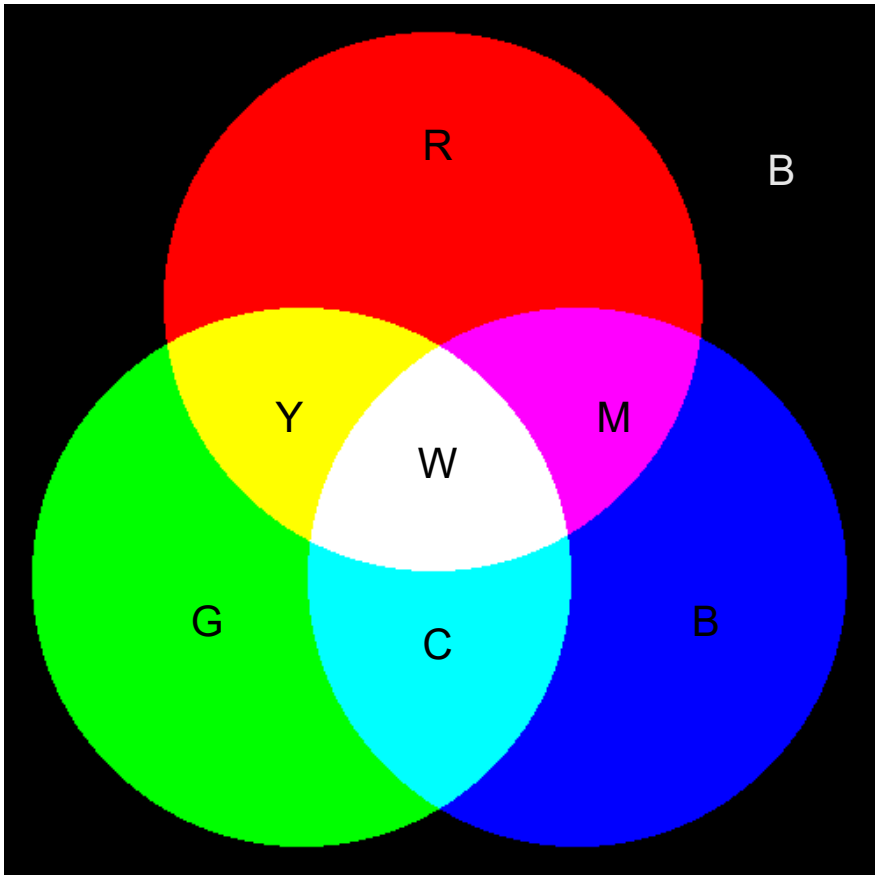


○　　×

サンプルシーン：色指定

```
 1:  #include “colors.inc”
 2:  #include “textures.inc”
 3:
 4:  camera { location < -4.0, 2.0, -3.0 >
 5:               look_at  < 0.0, 0.0, 0.0 >
 6:               angle 60
 7:             }
 8:
 9:  light_source { < 20, 30, -25 > color White }
10:
11:  sphere { < 0, 0, 0 >, 1
12:             pigment { Green }
13:             finish { Shiny }
14:           }
15:
16:  box { < -2, -1.2, -2 >, < 2, -1, 2 >
17:          texture { Red_Marble }
18:          finish { ambient 0.2
19:                       diffuse 0.8
20:                       phong 1
21:                     }
22:       }
23:
24:  background { color Gray50 }
```

光の三原色 (p. 1)



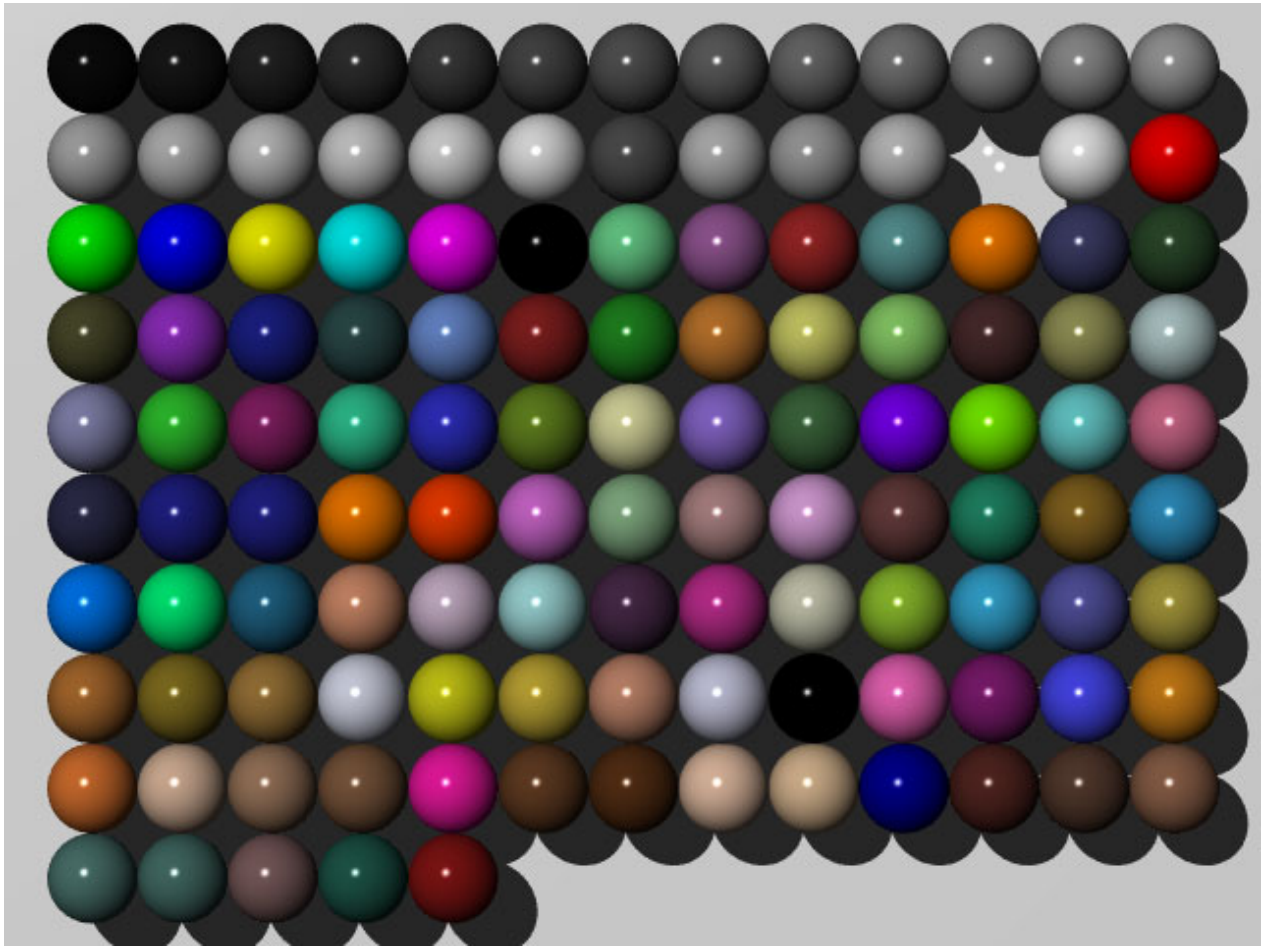
色のRGB表現 (pp. 14-16)

	red	green	blue
White	1.0	1.0	1.0
Red	1.0	0.0	0.0
Yellow	1.0	1.0	0.0
Black	0.0	0.0	0.0
Brown	0.647059	0.164706	0.164706
Maroon	0.556863	0.137255	0.419608

color.incで定義済の色名

Gray05	Gray10	Gray15	Gray20	Gray25	Gray30	Gray35	Gray40	Gray45	Gray50	Gray55	Gray60	Gray65
Gray70	Gray75	Gray80	Gray85	Gray90	Gray95	Dim Gray	Grey	Light Grey	VLight Grey	Clear	White	Red
Green	Blue	Yellow	Cyan	Magenta	Black	Aquamarine	BlueViolet	Brown	CadetBlue	Coral	Cornflower Blue	DarkGreen
Dark OliveGreen	DarkOrchid	Dark SlateBlue	Dark SlateGrey	Dark Turquoise	Firebrick	ForestGreen	Gold	Goldenrod	GreenYellow	IndianRed	Kharki	LightBlue
Light SteelBlue	LimeGreen	Maroon	Medium Aquamarine	MediumBlue	Medium ForestGreen	Medium Goldenrod	Medium Orchid	Medium SeaGreen	Medium SlateBlue	Medium Spring Green	Medium Turquoise	Medium VioletRod
Midnight Blue	Navy	NavyBlue	Orange	OrangeRed	Orchid	PaleGreen	Pink	Plum	Salmon	SeaGreen	Sienna	SkyBlue
SlateBlue	SpringGreen	SteelBlue	Tan	Thistle	Turquoise	Violet	VioletRed	Wheat	YellowGreen	SummerSky	RichBlue	Brass
Copper	Bronze	Bronze2	Silver	BrightGold	OldGold	Feldspar	Quartz	Mica	NeonPink	DarkPurple	NeonBlue	CoolCopper
Mandarin Orange	LightWood	Medium Wood	DarkWood	SpicyPink	SemiSweet Choc	BakersChoc	Flesh	NewTan	New Midnight Blue	VeryDark Brown	DarkBrown	DarkTan
Green Copper	DkGreen Copper	DustyRose	Hunters Green	Scarlet								

color.incの色の実際



サンプルシーン：テクスチャ指定

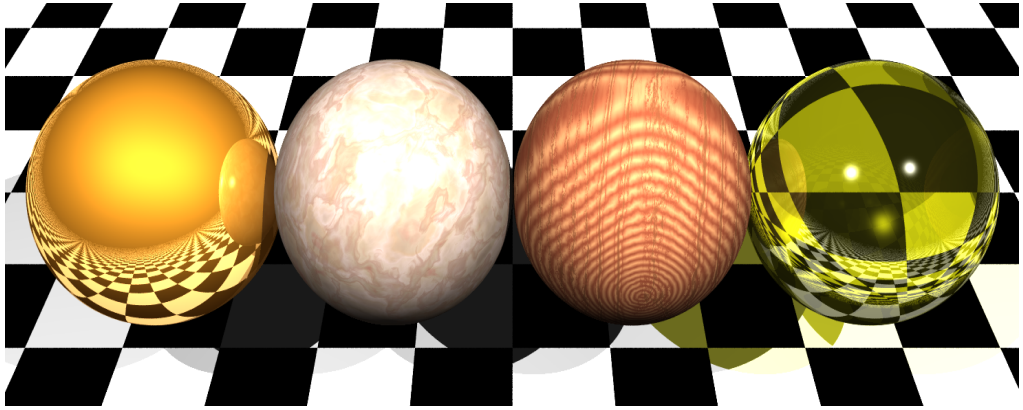
```
 1:  #include “colors.inc”
 2:  #include “textures.inc”
 3:
 4:  camera { location < -4.0, 2.0, -3.0 >
 5:               look_at  < 0.0, 0.0, 0.0 >
 6:               angle 60
 7:             }
 8:
 9:  light_source { < 20, 30, -25 > color White }
10:
11:  sphere { < 0, 0, 0 >, 1
12:              pigment { Green }
13:              finish { Shiny }
14:             }
15:
16:  box { < -2, -1.2, -2 >, < 2, -1, 2 >
17:           texture { Red_Marble }
18:           finish { ambient 0.2
19:                        diffuse 0.8
20:                        phong 1
21:                    }
22:        }
23:
24:  background { color Gray50 }
```

テクスチャ(肌理)の例 (pp. 17–18)

Jade*	Red_Marble*	White_Marble*	Blood_Marble*	Blue_Agate*	Sapphire_Agate*	Brown_Agate*	Pink_Granite*	PinkAlabaster
Blue_Sky*	Bright_Blue_Sky*	Blue_Sky2*	Blue_Sky3*	Blood_Sky*	Apocalypse*	Clouds*	FBM_Clouds*	Shadow_Clouds
Cherry_Wood*	Pine_Wood*	Dark_Wood*	Tan_Wood*	White_Wood*	Tom_Wood*	DMFWood1*	DMFWood2*	DMFWood3*

その他の定義済テクスチャ

```
#include "woods.inc"
#include "metals.inc"
#include "stones.inc“
#include “glass.inc"
```



T_Gold_1A　　T_Stone2　　T_Wood1　　M_Yellow_Glass

ガラスのときはmaterialと書く→

```
object{
    Sphere
    texture{T_Wood1}
}

object{
    Sphere
    translate <-2,0,0>
    texture{T_Stone2}
}

object{
    Sphere
    translate <-4,0,0>
    texture{T_Gold_1A}
}

object{
    Sphere
    translate <2,0,0>
     material{M_Yellow_Glass}
}
```

イメージマップ：好みの画像を物体に貼付け

```
box{<0,0,0>,<1,1,0>
  pigment{
    image_map{
    png "WorldMap.png"
    map_type 0
    }
  }
  scale <4,2>
  translate <-5, 0>
}

sphere{<0,0,0>,1
   pigment{
     image_map{
     png "WorldMap.png"
     map_type 1
     }
     rotate -23.4*z
   }
   translate <2,1,0>
}
```

←看板を立てる

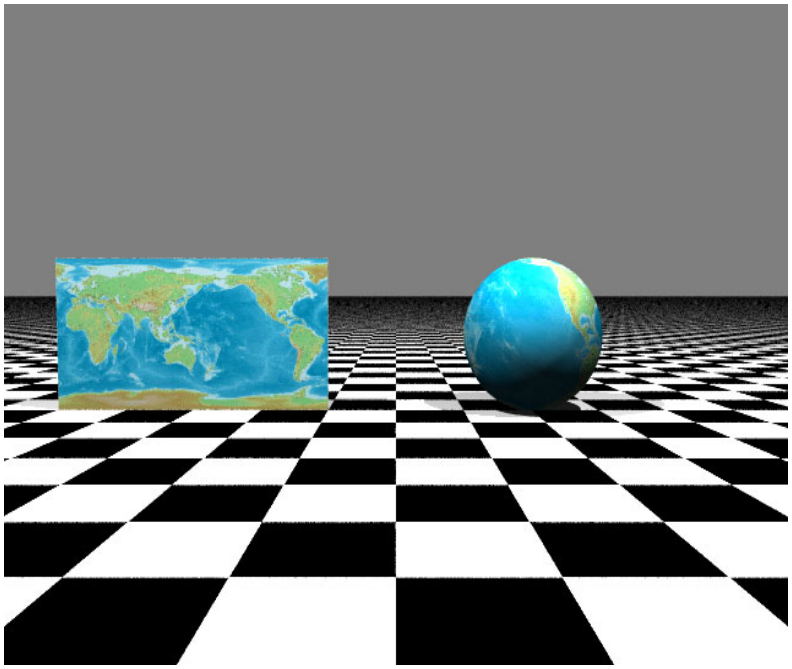
←平面へ貼付け

←球を指定

←球に貼付け

←地軸を傾ける

←横に移動



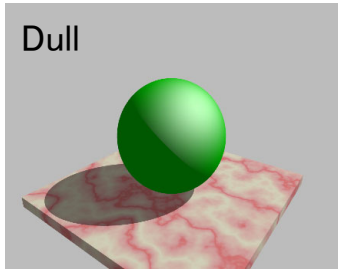
- 画像フォーマット：gif, png, pgm, 等 [jpgは不可]
- *xy* 平面上の0から1までの範囲にマッピング→要調整
- 円柱（トーラス）の場合はmaptypeを2（5）に変更

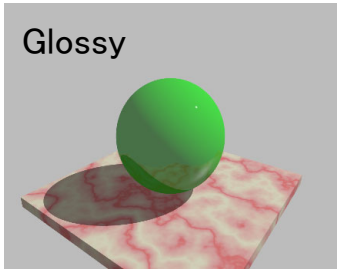
サンプルシーン：シェーディング指定

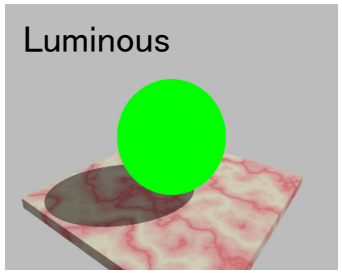
```
 1:  #include “colors.inc”
 2:  #include “textures.inc”
 3:
 4:  camera { location < -4.0, 2.0, -3.0 >
 5:               look_at  < 0.0, 0.0, 0.0 >
 6:               angle 60
 7:             }
 8:
 9:  light_source { < 20, 30, -25 > color White }
10:
11:  sphere { < 0, 0, 0 >, 1
12:              pigment { Green }
13:              finish { Shiny }
14:             }
15:
16:  box { < -2, -1.2, -2 >, < 2, -1, 2 >
17:           texture { Red_Marble }
18:           finish { ambient 0.2
19:                        diffuse 0.8
20:                        phong 1
21:                    }
22:        }
23:
24:  background { color Gray50 }
```

定義済の表面仕上げ (p. 17)

- Dull (くすんだ)
- Shiny (つやのある)
- Glossy (つやつやした)
- Luminous (輝く)
- Mirror (完全鏡面), ... など





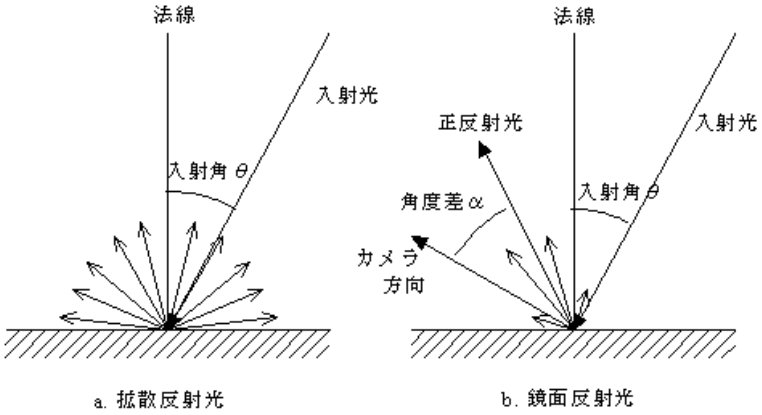


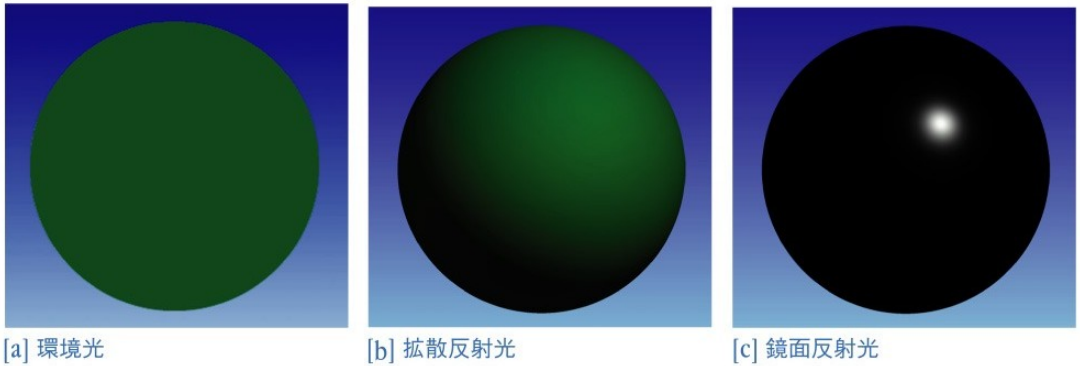
表面の光学的属性の詳細定義

```
 1:  #include “colors.inc”
 2:  #include “textures.inc”
 3:
 4:  camera { location < -4.0, 2.0, -3.0 >
 5:               look_at  < 0.0, 0.0, 0.0 >
 6:               angle 60
 7:             }
 8:
 9:  light_source { < 20, 30, -25 > color White }
10:
11:  sphere { < 0, 0, 0 >, 1
12:              pigment { Green }
13:              finish { Shiny }
14:             }
15:
16:  box { < -2, -1.2, -2 >, < 2, -1, 2 >
17:           texture { Red_Marble }
18:           finish { ambient 0.2
19:                    diffuse 0.8
20:                    phong 1
21:                   }
22:        }
23:
24:  background { color Gray50 }
```

シェーディング（p. 3）

- 環境光 (ambient)
- 拡散反射光 (diffuse)
- 鏡面反射光 (phong, specular) ※ 両者は少し挙動が異なる





参考サイト/書籍

- 参考サイト多数
  - ✓ Ver.3.5日本語マニュアル http://mana.starfree.jp/povjp/
- 参考書（2冊ずつ配備）
  - ✓ 鈴木，倉田，佐藤：POV-Rayによる3次元CG制作―モデリングからアニメーションまで―，画像情報教育振興協会，2008
  - ✓ 齊藤，年森，田代：3D-CGをはじめよう，POV-Ray入門，オーム社，2009
  - ✓ 松下，山本，柳川，鈴木，星，羽入：POV-Rayで学ぶはじめての3DCG制作，講談社，2017