４０４．マルチメディア実験

１．目的

フリーソフトウェアのPOV-Rayを用いて3DCG画像を作成します。Pov-Rayではシーンファイルを記述し、レンダリングする事によって画像を生成することができます。Pov-Rayの使用を通して、3DCG画像を生成するための方法を理解します。

２．POV-Rayの基本

3DCG画像作成ソフトウェア「POV-Ray for Windows」を使って実習を行います。これからの実際の作業の流れは以下のようになります。

・CG の内容を専用の言語で記述したシーンファイルを記述する
・Pov-Rayでそのシーンファイルを処理させて画像ファイルを生成する（レンダリング）

シーンファイルの記述にエラーがあり、画像が生成されない場合や、さらに物体などを追加する場合は再度シーンファイルを修正し、レンダリングを再度行い、画像を確認という繰りかえしとなります。

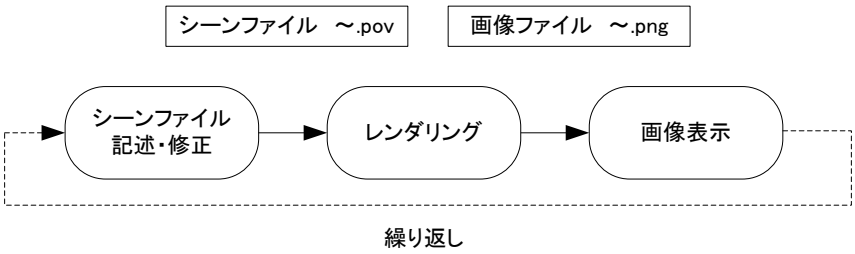


図２．１　製作の流れ

２．１　Pov-Rayを使ってみる

まず、デスクトップ上の画像処理フォルダ内にある「POV-Ray v3.6」のアイコンをダブルクリックし、POV-Rayを起動してください。

図２．２　POV-Rayのアイコン

起動するとウィンドウが立ち上がります。ライセンスに関するダイアログ「License」が出ますので、「OK」を選択します。するとPOV-Rayの使用方法のヒントを示すダイアログ「Tip Of The Day」がでますので、これも「OK」を選択します。

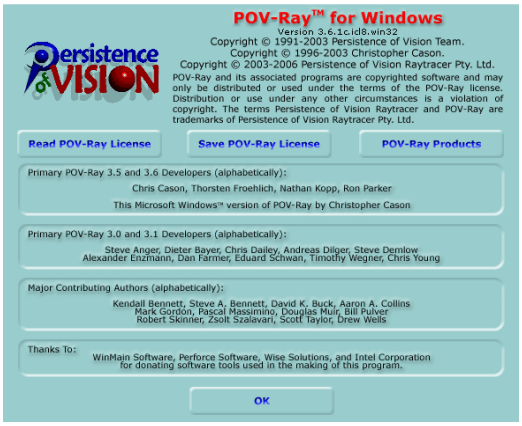


図２．３　License

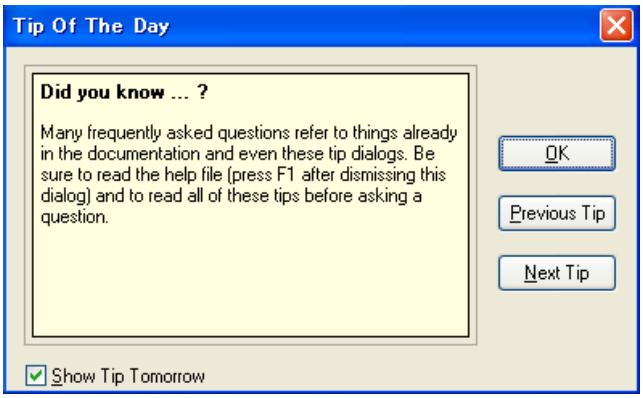


図２．４　Tip Of The Day

そうすると、図２．５の状態になりますので、ウィンドウのタブ選択部分（図２．５の①の部分）から「woodbox.pov」をクリックしてください。「woodbox.pov」のシーンファイルが表示されたら、ウィンドウのメニューアイコン（図２．５の②の部分）から「Run」をクリックしてください。レンダリングが開始され、完了すると図２．８の画像が生成されます。

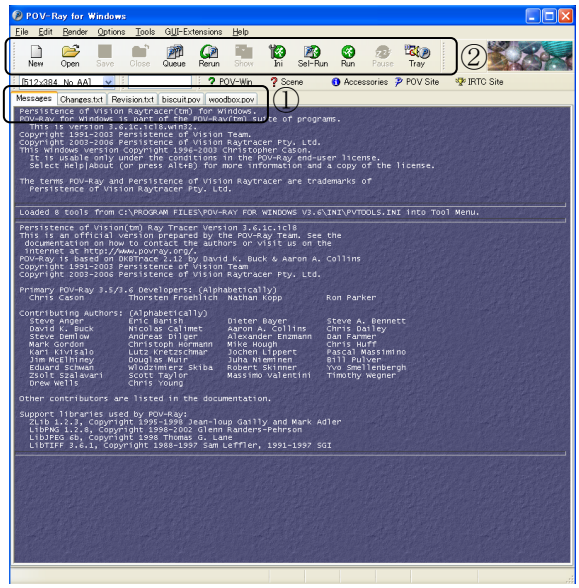


図２．５　POV-Rayのウィンドウ

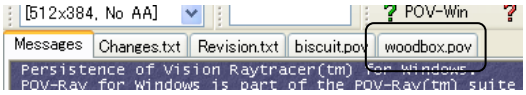


図２．６　タブ

図２．７　アイコン

図２．８　レンダリング結果例

レンダリングが完了すると、完了時の再生する音の設定の変更方法を知らせるダイアログ「Render Complete Sound」、画像の保存先を知らせるダイアログ「Output File Notification」が順に表示されます。「Don’t tell me again」にチェックを入れ、どちらも「OK」を選択してください。その後、レンダリングされたウィンドウを閉じると、描画ウィンドウに関するヘルプの表示方法についてのダイアログ「Render Window」が表示されますので、これも「OK」を選択してください。なお、画像やシーンファイルの保存先はデフォルトで「C:¥Program Files¥POV-Ray for Windows v3.6¥scenes¥advanced」フォルダ内となります。

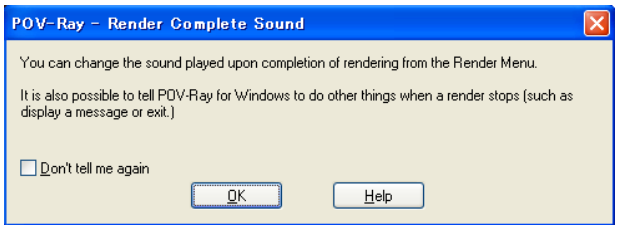


図２．９　Render Complete Sound

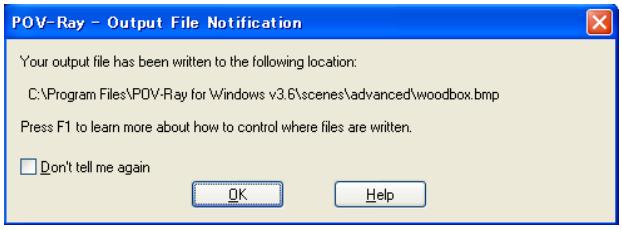


図２．１０　Output File Notification

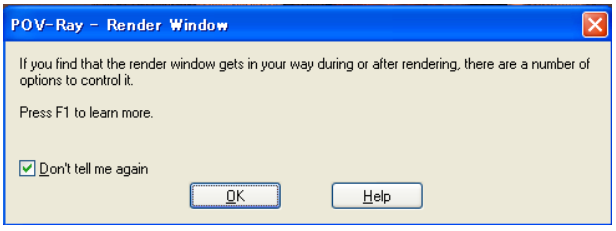


図２．１１　Render Window

それでは、次に新規のシーンファイルを作成する方法について紹介していきます。ウィンドウのメニュー（図２．１２の③の部分）から「File」→「New File」を選択してください。そうするとタブ部分に「Untitled」のタブが表示され、新たにシーンファイルの編集が可能となります。

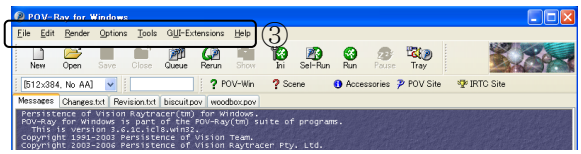


図２．１２　メニュー

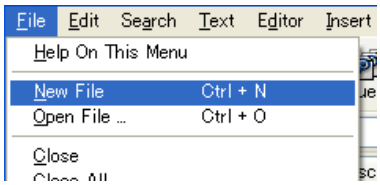


図２．１３　New File

図２．１４　Untitled

シーンファイルを以下のように記述してください。基本的には光、カメラ、物体という3つの要素からシーンファイルは成り立っています。シーンファイルにはコメントを書いてあるので、そのコメントも参考にしてください。ただし、「POV-Ray for Windows」ではテキスト編集に日本語を用いることができません。そのため、日本語を使いたいときには別のテキストエディタを用いるという方法を取ります。本実習では、コメント部分を省略して編集してください。

scene01.pov

```
#include "colors.inc" // 色のインクルードファイル
#include "shapes.inc" // 形のインクルードファイル
#include "textures.inc" // 材質のインクルードファイル
#include "woods.inc" // 木の素材のインクルードファイル
#include "glass.inc" //ガラス素材のインクルードファイル

/*カメラの設定*/
camera {
    location <3, 6, -12>// カメラの場所
    look_at <0, 0, 0> // カメラの方向
    angle 35 // カメラアングル
}
/* 光源の設定*/
light_source {
    <-10, 25, -30> // 光源の位置
    color White // 光源の色，強さ
}
light_source {
    <20, 30, -45> // 光源の位置
    color White*2 // 光源の色，強さ
}

//object{ Disk_X scale <100,.05,.05> pigment{color Red} } // X 軸
//object{ Disk_Y scale <.05,100,.05> pigment{color Green}} // Y 軸
```

```
//object{ Disk_Z scale <.05,.05,100> pigment{color Blue} } // Z 軸

/* 床の設定*/
object{
    Plane_XZ // XZ 面の平面
    texture{T_Wood35} // 物体の素材
    translate <0, -1, 0> // 物体の位置
}

/* 物体の設定*/
object {
    Cube // 立方体
    texture {T_Dark_Green_Glass} // 素材
    scale <1,1,1> // 拡大, 縮小
    rotate <0,0,0> // 回転
    translate <0,0,0> // 移動
}

/* 背景*/
background{color Yellow}
```

編集が終了しましたら、ウィンドウのメニュー（図２．１２の③の部分）から「File」→「Save As」を選択してください。そうすると「名前を付けて保存」ダイアログが立ち上がりますので、ファイル名を「scene01」として、保存して下さい。そして、レンダリングを行う（Runのアイコンをクリック）と、木の床の上に緑色のガラスの直方体がある画像が表示されます（図２．１７参照）。

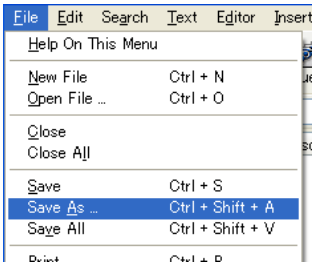


図２．１５　Save As

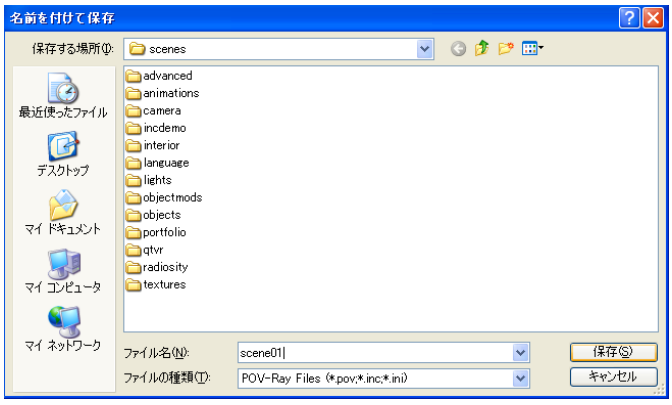


図２．１６　保存用ダイアログ

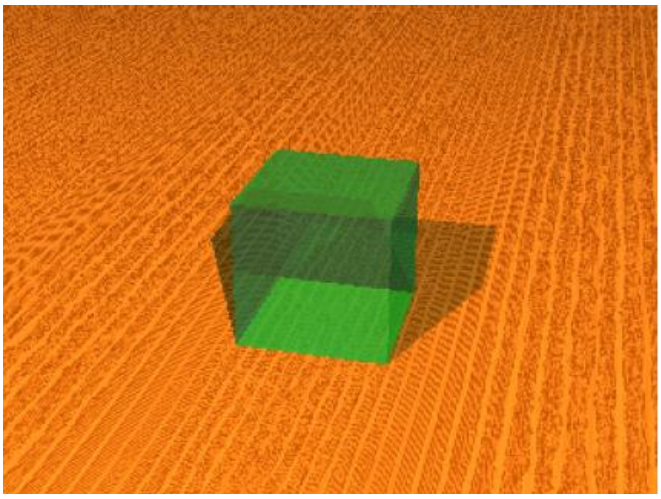


図２．１７　scene01.bmp

それでは、POV-Rayについて、もう少し具体的に見ていきましょう。POV-Rayでは左手座標系と呼ばれる座標系を使います。左手を座標系に対応させると、親指が＋x方向、人さし指が＋y方向、中指が＋z方向に一致します。

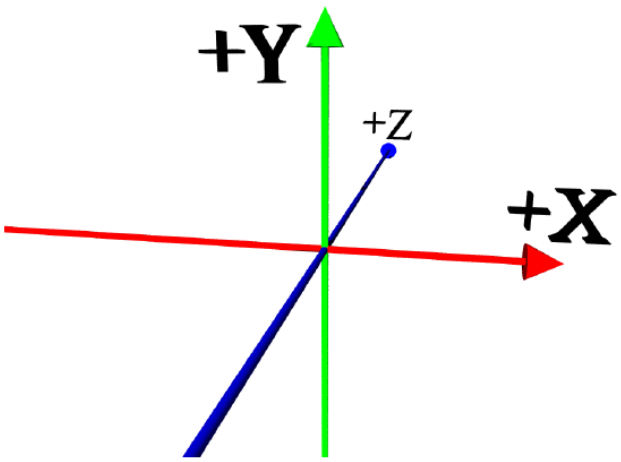


図２．１８　左手座標系モデル

まず、カメラと光について図２．１８を頭に思い浮かべながらいろいろとシーンファイルを改造して、確認していってください。実際に実習を進めていくには座標軸があった方がわかりやすいのでコメントアウトされている座標軸を追加すると良いかもしれません。

```
object{ Disk_X scale <100,.05,.05> pigment{color Red} } // X 軸
object{ Disk_Y scale <.05,100,.05> pigment{color Green}} // Y 軸
object{ Disk_Z scale <.05,.05,100> pigment{color Blue} } // Z 軸
```

ではまずカメラの設定です。

```
/* カメラの設定*/
camera {
    location <3, 6, -12>// カメラの場所
    look_at <0, 0, 0> // カメラの方向
    angle 35 // カメラアングル
}
```

location <*x*, *y*, *z*> でカメラのある位置、look_at <*x*, *y*, *z*> がカメラの見ている座標です。この場合は右上の手前から原点を見ていることがわかります。angle は視野角です。angle を

・大きな角度にすることによって広角レンズ
・小さな角度にすることによって望遠レンズ

で撮影した画像となります。これで物体を好きな所から見ることが出来るようになりました。
次に光の設定です。

```
/* 光源の設定*/
light_source {
    <-10, 25, -30> // 光源の位置
    color White // 光源の色, 強さ
}
light_source {
    <20, 30, -45> // 光源の位置
    color White*2 // 光源の色, 強さ
}
```

<*x*, *y*, *z*> で光源の位置、color Whiteやcolor White*2 で光源の色と強さを表します。光の強さのデフォルトは1です。色の名前のリストを表２．１に示しておきます。 色の名前は最初が大文字であることに注意してください。このシーンファイルでは光源は二つありますが、light source をいくつも書くといくつも光源を置けます。

表２．１　色の名前リスト

Red	Green	Blue
Yellow	Cyan	Magenta
White	Black	Gray05
Gray10	Gray15	Gray20
Gray25	Gray30	Gray35
Gray40	Gray45	Gray50
Gray55	Gray60	Gray65
Gray70	Gray75	Gray80
Gray85	Gray90	Gray95
DimGray	DimGrey	Gray
Grey	LightGray	LightGrey
VLightGray	VLightGrey	Aquamarine
BlueViolet	Brown	CadetBlue

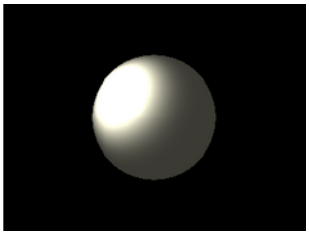
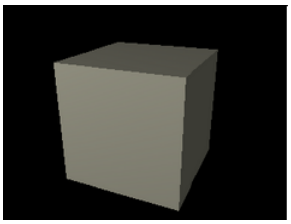
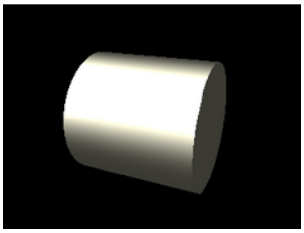
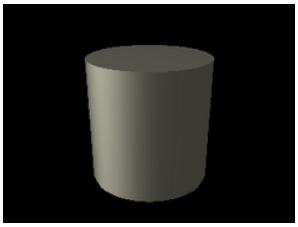
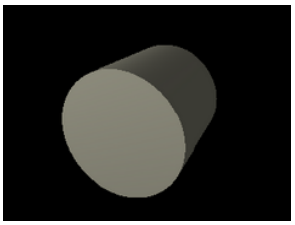
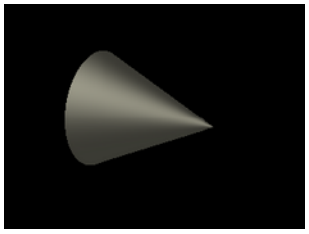
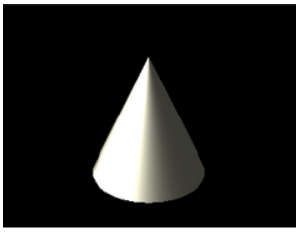
Coral	CornflowerBlue	DarkGreen
DarkOliveGreen	DarkOrchid	DarkSlateBlue
DarkSlateGray	DarkSlateGrey	DarkTurquoise
Firebrick	ForestGreen	Gold
Goldenrod	GreenYellow	IndianRed
Khaki	LightBlue	LightSteelBlue
LimeGreen	Maroon	MediumAquamarine
MediumBlue	MediumForestGreen	MediumGoldenrod
MediumOrchid	MediumSeaGreen	MediumSlateBlue
MediumSpringGreen	MediumTurquoise	MediumVioletRed
MidnightBlue	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	NeonPink	DarkPurple
NeonBlue	CoolCopper	MandarinOrange
LightWood	MediumWood	DarkWood
SpicyPink	SemiSweetChoc	BakersChoc
Flesh	NewTan	NewMidnightBlue
VeryDarkBrown	DarkBrown	DarkTan
GreenCopper	DkGreenCopper	DustyRose
HuntersGreen	Scarlet	Med_Purple
Light_Purple	Very_Light_Purple	

次に物体の設定です。図２．１７の画像には、床と立方体の２つの物体がありますが、立方体について説明します。

```
/* 物体の設定*/
object {
    Cube // 立方体
    texture {T_Dark_Green_Glass} // 素材
    scale <1,1,1> // 拡大, 縮小
    rotate <0,0,0> // 回転
    translate <0,0,0> // 移動
}
```

物体はobjectで記述します。Cube は物体の形です。物体の形には表２．２のようなものがあります。また床に使っているXZ平面、Plane_XZにも図２．１９の様にPlane_YZやPlane_XYのように、いくつかのパターンがあります。素材の部分は単色で塗りつぶすこともできるのですが、これは後ほど紹介します。

表２．２　物体の形

Sphere	Cube	
		
Disk_X	Disk_Y	Disk_Z
		
Cone_X	Cone_Y	Cone_Z
		

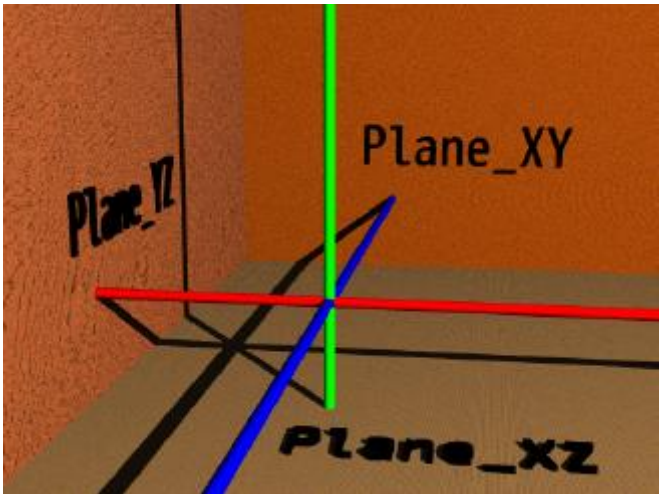


図２．１９　Plane_XZ等

```
scale <x,y,z> // 拡大，縮小
rotate <x,y,z> // 回転
translate <x,y,z> // 移動
```

これはそれぞれ物体を拡大縮小、回転、移動するものです。scale では各軸方向に拡大率を指定できるので、球をラグビーボールの様にしたり、直方体を板のようにしたりすることが出来ます。rotate の単位は度です。各軸でどの程度回転するのか指定できます。回転も左手をつかって考えることが出来ま

す。親指を回転軸の正の方向へ向けると残りの指の巻き込む方向が回転する方向です。translate は座標を示しているのでは無くそれぞれの方向にどれだけ移動するのかといったことを表しています。
　そのため

```
translate <10,10,  0>
```

と

```
translate <10,  0,  0>
translate <  0,10,  0>
```

は同じです。回転と移動は原点を中心として行われるため、rotateしてからtranslateするのとtranslate してからrotate するのでは結果が異なります。

実際にscene01.pov の立方体を

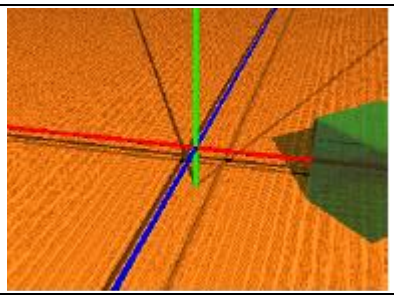
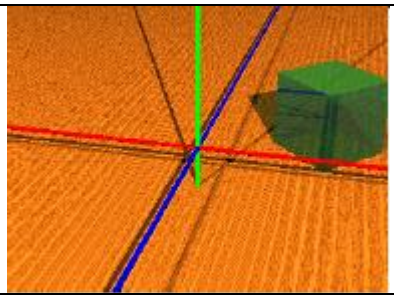
```
rotate <0,-45,0> // 回転
translate <4,0,0> // 移動
```

するのと

```
translate <4,0,0> // 移動
rotate <0,-45,0> // 回転
```

するのはどう違うのでしょうか。表２．３に物体の移動と回転に関する例を示しておきます。

表２．３ 物体の移動と回転

回転して移動	移動して回転
	

　これで、３次元座標の事については終わりました。これで物体やカメラを自由に動かして見ることが出来るようになりました。

２．２ 物体を単色で塗りつぶす

　物体を単色で塗りつぶすにはobject内で

```
pigment { color 色の名前}
```

とします。具体的には以下のようにします。これは「scene02」として、シーンファイルを保存しておいてください。

scene02.pov

```
#include "colors.inc" // 色のインクルードファイル
#include "shapes.inc" // 形のインクルードファイル
#include "textures.inc" // 材質のインクルードファイル
#include "woods.inc" // 木の素材のインクルードファイル
#include "glass.inc" //ガラス素材のインクルードファイル

/* カメラの設定*/
camera {
    location <3, 6, -12>// カメラの場所
    look_at <0, 0, 0> // カメラの方向
    angle 35 // カメラアングル
}

/* 光源の設定*/
light_source {
    <-10, 25, -30> // 光源の位置
    color White // 光源の色, 強さ
}

light_source {
    <20, 30, -45> // 光源の位置
    color White*2 // 光源の色, 強さ
}

//object{ Disk_X scale <100,.05,.05> pigment{color Red} } // X 軸
//object{ Disk_Y scale <.05,100,.05> pigment{color Green}} // Y 軸
//object{ Disk_Z scale <.05,.05,100> pigment{color Blue} } // Z 軸

/* 床の設定*/
object{
Plane_XZ // XZ 面の平面
    // texture{T_Wood35} // 物体の素材
    pigment {color YellowGreen} // 物体を単色で塗る
```

```
    translate <0, -1, 0> // 物体の位置
}

/* 物体の設定*/
object {
    Cube // 立方体
    // texture {T_Dark_Green_Glass} // 素材
    pigment {color Red} // 物体を単色で塗る
    scale <1,1,1> // 拡大, 縮小
    rotate <0,0,0> // 回転
    translate <0,0,0> // 移動
}

/* 背景*/
background{color Yellow}
```

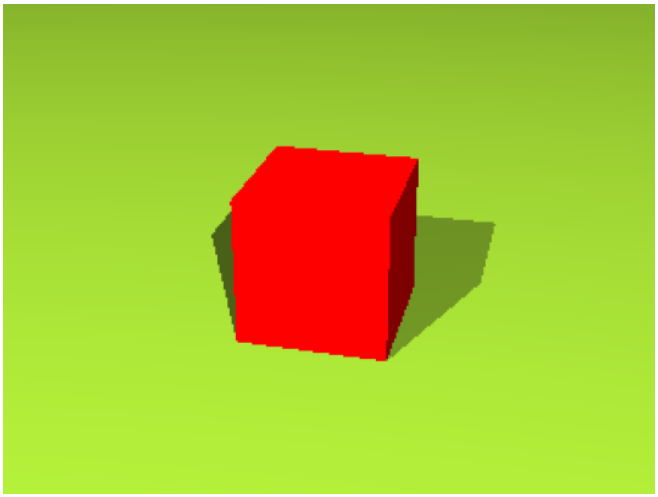


図２．２０ scene02.bmp

物体は基本的には

・texture で素材をはる
・pigment で単色で塗りつぶす

のどちらかしか出来ません.
なお、

```
backgraound { color 色の名前}
```

とすることによっても背景に色を使うことが出来ます。

４．３　素材

textureで使うことが出来る素材を紹介します。

４．３．１ ガラス

#include ”glass.inc”を書き加えるとガラス素材を使えます.

図２．２１　ガラス素材

後列左から

T_Glass1, T_Glass2, T_Glass3, T_Glass4, T_Old_Glass, T_Winebottle_Glass, T_Beerbottle_Glass

T_Ruby_Glass, T_Green_Glass, T_Dark_Green_Glass, T_Yellow_Glass, T_Orange_Glass, T_Vicksbottle_Glass

４．３．２ 木

#include ”woods.inc”を書き加えると木素材を使えます。

図２．２２　木素材

後列左から

T_Wood31, T_Wood32, T_Wood33, T_Wood34, T_Wood35

T_Wood26, T_Wood27, T_Wood28, T_Wood29, T_Wood30

T_Wood21, T_Wood22, T_Wood23, T_Wood24, T_Wood25

T_Wood16, T_Wood17, T_Wood18, T_Wood19, T_Wood20

T_Wood11, T_Wood12, T_Wood13, T_Wood14, T_Wood15

T_Wood6, T_Wood7, T_Wood8, T Wood9, T_Wood10

T_Wood1, T_Wood2, T_Wood3, T_Wood4, T_Wood5

４．３．３ 石

#include ”stones.inc”を書き加えると石素材を使えます。

図２．２３　石素材

後列左から

T_Stone1, T_Stone2, T_Stone3, T_Stone4, T_Stone5, T_Stone6, T_Stone7, T_Stone8

T_Stone9, T_Stone10, T_Stone11, T_Stone12, T_Stone13, T_Stone14, T_Stone15, T_Stone16

T_Stone17, T_Stone18, T_Stone19, T_Stone20, T_Stone21, T_Stone22, T_Stone23, T_Stone24

T_Stone25, T_Stone26, T_Stone27, T_Stone28, T_Stone29, T_Stone30, T_Stone31, T_Stone32

T_Stone33, T_Stone34, T_Stone35, T_Stone36, T_Stone37, T_Stone38, T_Stone39, T_Stone40

４．３．４ 金

#include ”golds.inc”を書き加えると金素材を使えます。

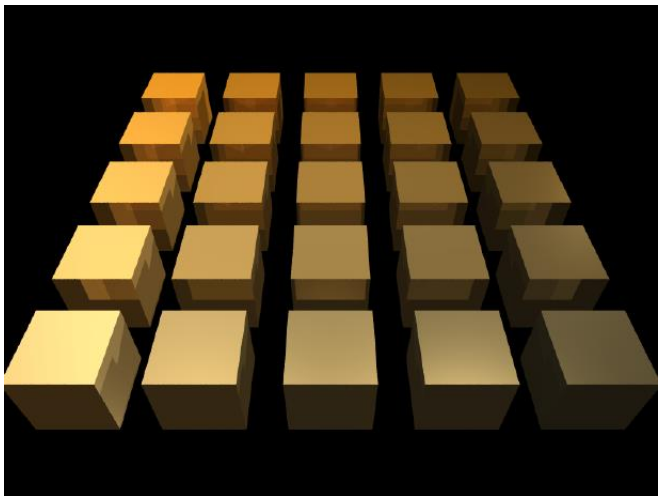


図２．２４　金素材

後列左から

T_Gold_1A, T_Gold_1B, T_Gold_1C, T_Gold_1D, T_Gold_1E

T_Gold_2A, T_Gold_2B, T_Gold_2C, T_Gold_2D, T_Gold_2E

T_Gold_3A, T_Gold_3B, T_Gold_3C, T_Gold_3D, T_Gold_3E

T_Gold_4A, T_Gold_4B, T_Gold_4C, T_Gold_4D, T_Gold_4E

T_Gold_5A, T_Gold_5B, T_Gold_5C, T_Gold_5D, T_Gold_5E

４．３．５ 銀

#include ”metals.inc”を書き加えると銀素材を使えます。

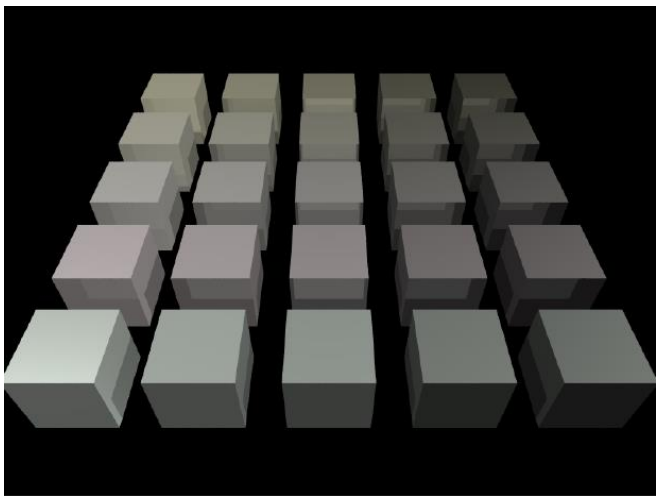


図２．２５　銀素材

後列左から

T_Silver_1A, T_Silver_1B, T_Silver_1C, T_Silver_1D, T_Silver_1E

T_Silver_2A, T_Silver_2B, T_Silver_2C, T_Silver_2D, T_Silver_2E

T_Silver_3A, T_Silver_3B, T_Silver_3C, T_Silver_3D, T_Silver_3E
T_Silver_4A, T_Silver_4B, T_Silver_4C, T_Silver_4D, T_Silver_4E
T_Silver_5A, T_Silver_5B, T_Silver_5C, T_Silver_5D, T_Silver_5E

４．３．６ 銅

#include ”metals.inc”を書き加えると銅素材を使えます。

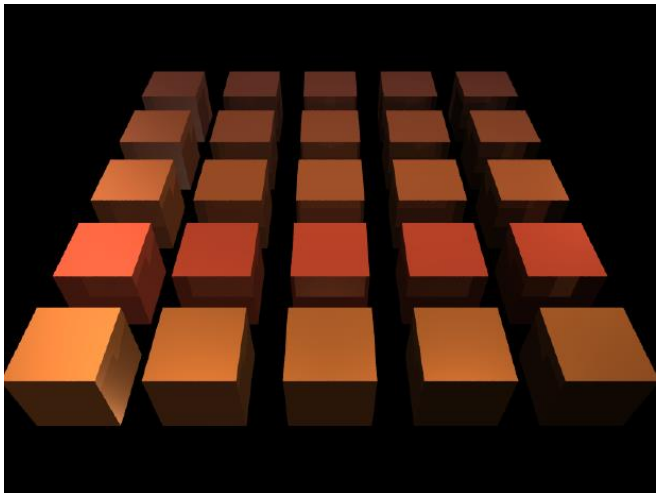


図２．２６　銅素材

後列左から

T_Copper_1A, T_Copper_1B, T_Copper_1C, T_Copper_1D, T_Copper_1E
T_Copper_2A, T_Copper_2B, T_Copper_2C, T_Copper_2D, T_Copper_2E
T_Copper_3A, T_Copper_3B, T_Copper_3C, T_Copper_3D, T_Copper_3E
T_Copper_4A, T_Copper_4B, T_Copper_4C, T_Copper_4D, T_Copper_4E
T_Copper_5A, T_Copper_5B, T_Copper_5C, T_Copper_5D, T_Copper_5E

４．３．７ 空

#include ”skies.inc”を書き加える事によって、空や雲の様子を描くことができ、屋外のようなイメージとなります。

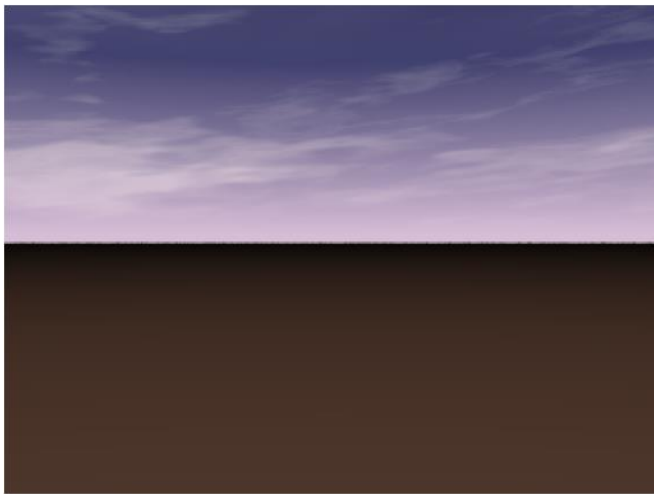


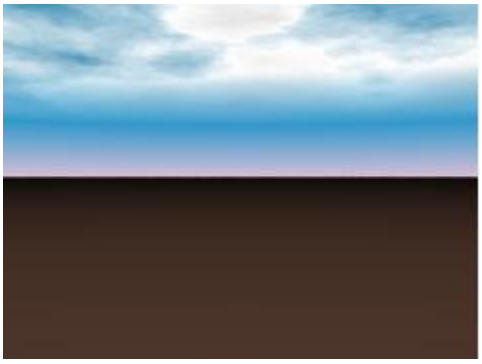
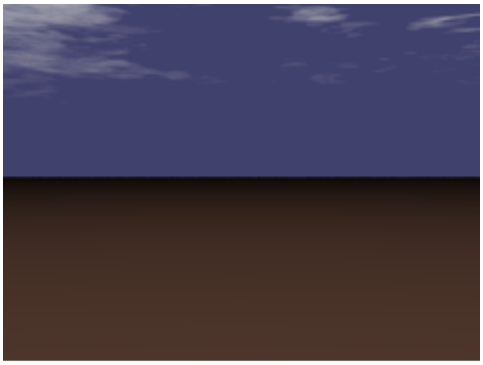
図２．２７ 空

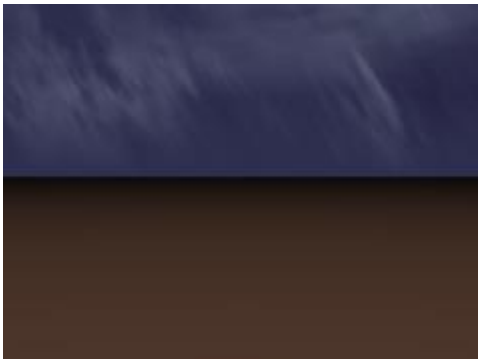
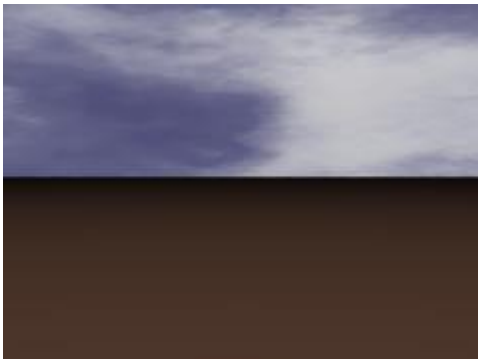
scene03.pov

```
#include "colors.inc"
#include "shapes.inc"
#include "textures.inc"
#include "skies.inc" // 空の状態を扱う

// カメラ　人間の目線から100m 先を見ているようなイメージ
camera{
    location <0, 1, 0>
    look_at <0, 0, 100>
    angle 50
}

// ライト
light_source{
    <0, 10, -10>
    color White*2
}

// 地面
object{
    Plane_XZ
    pigment {color DarkBrown}
}

// 空
sky_sphere{ S_Cloud1 }
```

S_Cloud1の部分に入るものには、その他にS_Cloud2, S_Cloud3, S_Cloud4, S_Cloud5 があります。

表２．４　空素材

S_Cloud2	S_Cloud3
	

S_Cloud4	S_Cloud5
	

この他、skies.incに定義されているわけではありませんが(textures.incに定義されている物です) Waterという材質があります（材質なのでtexture Water として記述します)。表２．４に記述したS_ Cloud2の床の材質にこれを使うと日中の海の様に見えます。

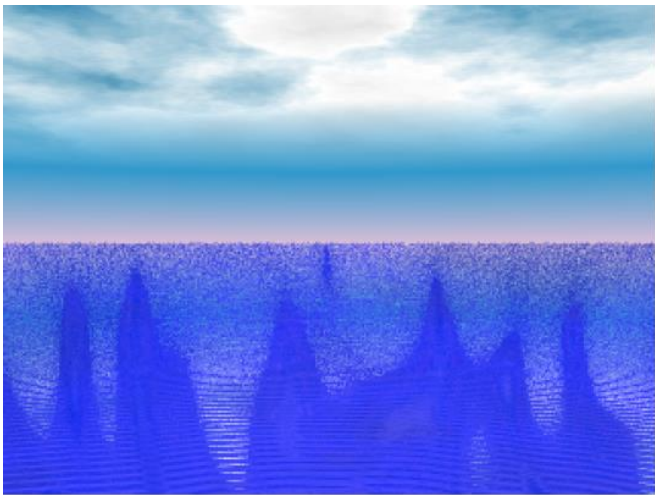


図２．２８　日中の海

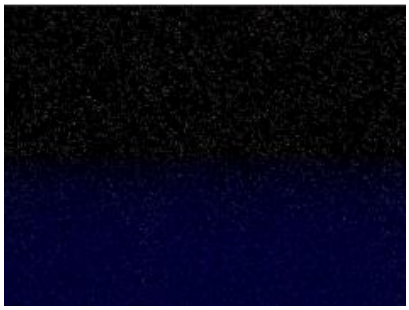
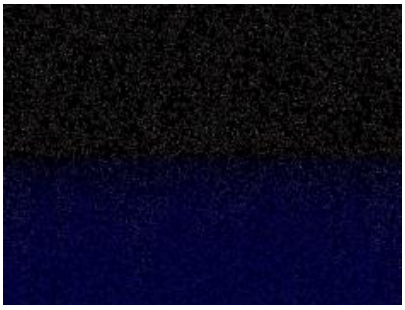
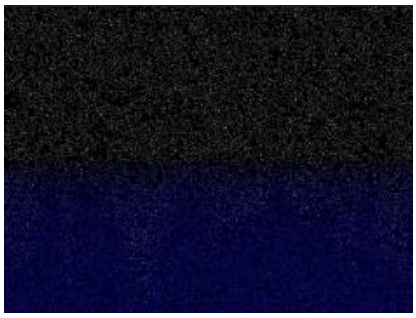
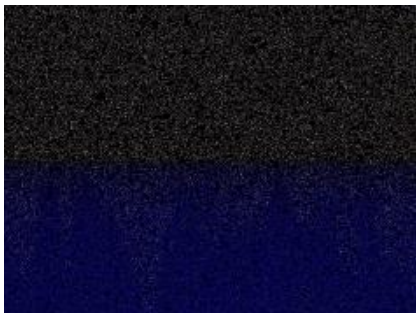
４．３．８ 星空

星空は#include ”stars.inc”を記述する事によって使用できます。textureで使用する素材として用意されているため表２．４に書いた空の様に使うのではなく物体の表面に張り付けることが必要です。

この例では前方にPlane_XYの物体を用意し、それに張り付ける事によって星空を実現しています。

Starfield1から6 までありますが，数字が大きくなる程星が多くなります。

表２．５　星空

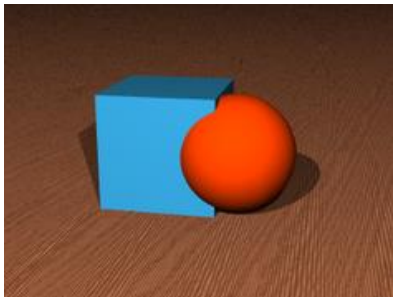
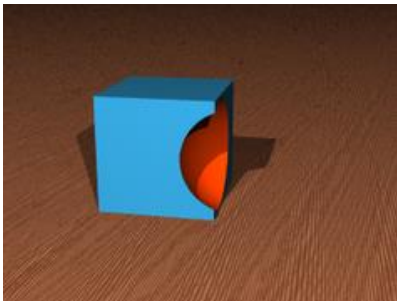
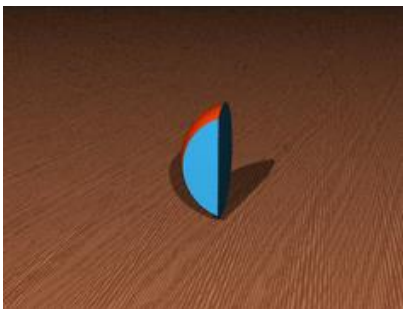
Starfield1	Starfield1	Starfield1
		
Starfield4	Starfield5	Starfield6
		

４．４　物体の演算

ここからは物体を任意の形状にするために、和、差、積を使う方法を学びます。

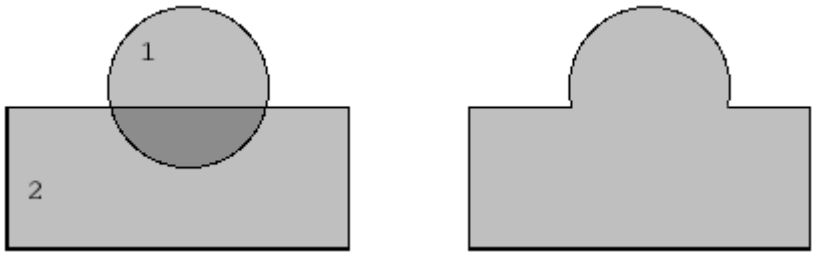
・和 merge
・差 difference
・積 intersection

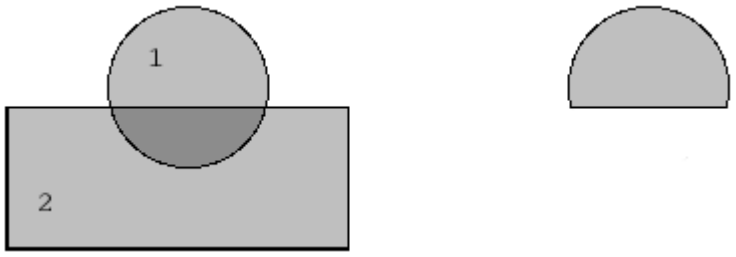
表２．６ 図形の演算

Merge	difference	intersection
		

二次元的に表現すると表２．７のようになります。図形の演算を用いた作品例を表２．８に示します。

表２．７　図形の演算のイメージ

merge {
　object {
　　一つ目の物体
　}
　object {
　　二つ目の物体
　}
}
	

difference {
　object {
　　一つ目の物体
　}
　object {
　　二つ目の物体
　}
}
	

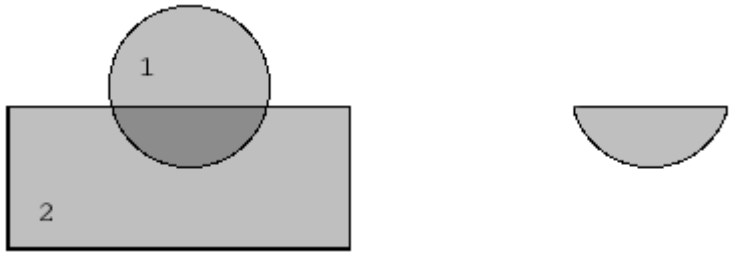
intersection {
　object {
　　一つ目の物体
　}
　object {
　　二つ目の物体
　}
}
	

表２．８　作品例

intersection {
　merge {
　　object { Disk_Y }
　　object { Cone_Y translate <0,2,0>}
　}
　object { Cube scale <0.75,3,1> }
　object { Cube scale <0.75,3,1> rotate <0,60,0> }
　object { Cube scale <0.75,3,1> rotate <0,120,0>}
　pigment {color Green}
}
	

２．５　３次元文字

文字を３次元テキストとして作成することが出来ます。

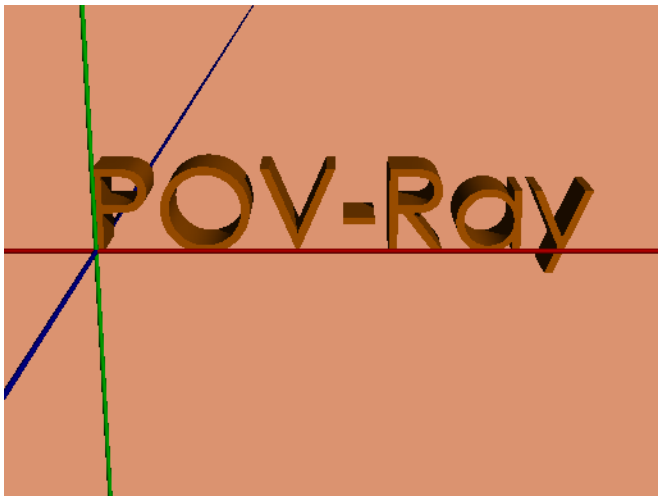


図２．２９　作品例

scene04.pov

```
#include "colors.inc" // 色のインクルードファイル
#include "shapes.inc" // 形のインクルードファイル

/* カメラ*/
camera {
        location <2,3, -10>// カメラの場所
        look_at <2, 0, 0>  // カメラの方向
        angle 30 // カメラアングル
}

/* 光源の設定*/
light_source {
        <10, 10, -18>// 光源の位置
        color White// 光源の色, 強さ
}

object{ Disk_X scale <100,.02,.02> pigment{color Red} } // X 軸
object{ Disk_Y scale <.02,100,.02> pigment{color Green}} // Y 軸
object{ Disk_Z scale <.02,.02,100> pigment{color Blue} } // Z 軸

/* 文字*/
text {
        ttf "gothic.ttf" "POV-Ray" 0.5 , 0
```

```
        pigment {color Orange}
        scale <1,1,1>
        translate <0,0,0>
}

/* 背景*/
background{color Tan}
```

文字を使う場合は

```
ttf "フォント名" "文字列" 文字の厚さ, 文字間隔
```

というように使います。

文字の厚さは0 にしてしまうとまったく無くなってしまうのである程度の数値にしなくてはいけないのですが、0.25から1程度が一般的であるためこのシーンファイルでは0.5にしています。文字間隔についても本当はいろいろと細かく設定できるのですが0としてください。

そのほかの物体の色(pigment)、素材(texture)、拡大縮小(scale)、回転(rotate)、移動(translate)そのほか物体の演算等はobject と同様に使えます。

○演習課題

オリジナルのCG画像を作成してください。シーンファイルと画像を提出してもらいます。

○レポートの構成

① 目的
② 演習課題
　シーンファイルと画像
③ 参考文献

【参考・引用】

釧路工業高等専門学校技術室報告集第９号、二谷聡志、「オープンソースソフトウェアによるCG 作成および動画作成」