知能情報システム創成 I「T3：CG」

3 次元 CG ソフトを使った 3 次元 CG の作成

情報理工学部知能情報学科　瀬尾昌孝

I.　本課題の内容・目的

コンピュータを使って画像を作る技術や作られた画像をコンピュータグラフィックス（Computer Graphics:CG）とよぶ.

本テーマでは，ＣＧアプリケーションソフトを利用し３次元ＣＧを作成する基本技術を習得した後，このＣＧモデリング法を用いて 3 次元ＣＧ作品を生成し，コンテスト形式で達成度を評価する.

II.　準備

A.　3 次元 CG ソフト

「3 次元 CG ソフト」は 3 次元 CG モデリング（形の定義）やレンダリング（その表示）を行うソフトウェアである.

現在, 様々な「3 次元 CG ソフト」が市販, 配布されており, 工業デザインや映画, ゲーム, 建築など, 数多くの分野で活用されている.

本講義では, 皆さんが自宅でもインストールして利用することができる,「POV-Ray」というフリーソフトを使用する.

B.　POV-Ray のインストール方法

POV-Ray はフリーソフトなので, 自宅 PC にもインストールすることができる.

「POV-Ray の公式サイト」

http://www.povray.org/

C.　各週の内容

1 週目	POV-Ray を用いた形状モデリングと 3 次元 CG 作成
2 ,3 週目	3 次元 CG 作品の作成
第 14 週	製作した CG 作品のコンテスト

D.　POV-Ray の動作認識

D.1.　シーンファイル

POV-Ray はシーンファイル(図 1) というテキスト形式のファイルから画像を作る. これから図 1 のシーンを実際に書いて, それを画像にする.

シーンファイルには, 図 2 に示すように作りたいもの（object）だけでなく，カメラ（camera）とライト（lightsource）も必要である.

POV-Ray では図 3 のような手順で画像ができあがる.

```
#include "colors.inc"    ←色指定に使用
#include "shapes.inc"    ←形状指定に使用

camera{    ←シーンを写すカメラの設定
   location<0, 10, -10>    ←カメラの位置
   look_at<0, 0, 0>    ←カメラの向き
   angle 20    ←カメラの画角
}

light_source{<-10, 10, -10> color White}    ←ライトの
light_source{<10, 10, -10> color White}          位置と色

object{    ←シーンに置く物体の設定
   Sphere    ←球
   pigment{color Red}    ←赤色
}
```

図 1　シーンファイル

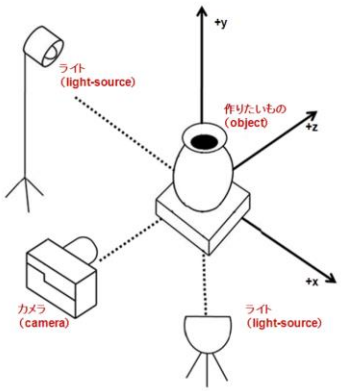


図 2　シーンの説明

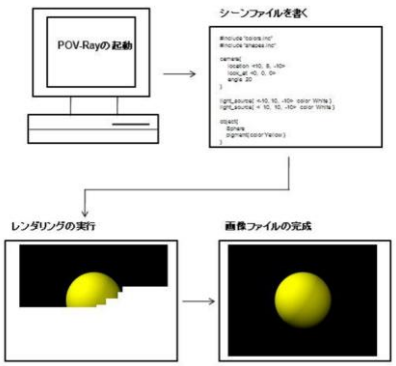


図 3　POV-Ray の画像生成手順

D.2　POV-Ray を使ってみる

1) POV-Ray を起動する

「スタート」→「プログラム」→「POV-Ray for Windows」を選択する. 起動時に表示されるメッセージウィンドウは[OK] ボタンを押して閉じる.

2) 新規ファイルを準備する

左上にある[NEW] ボタンをクリックする(図 4 参照).

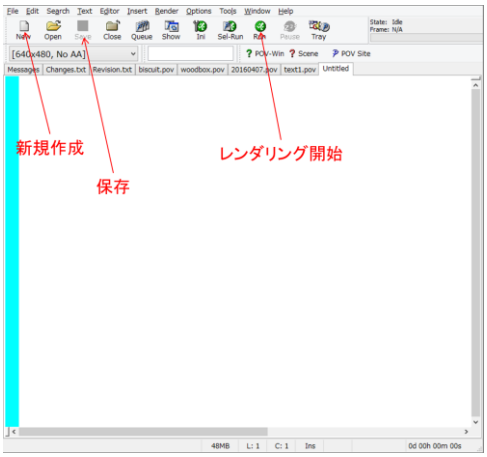


図 4　POV-Ray の処理画面

3) シーンファイルを書く

図 1 のとおりに命令を書きこむ.

※注意

・半角文字で書く

・アルファベットの大文字小文字は区別する

書き終わったら[Save] ボタンをクリックする(図 4 参照).

任意のファイル名をつけて保存する.

4) レンダリングを行う

図 4 の[Run] ボタンをクリックするとレンダリングが開始する.

5) 画像サイズを指定する

図 5 の左上にあるサイズ指定のためのプルダウンをクリックし,選択する.

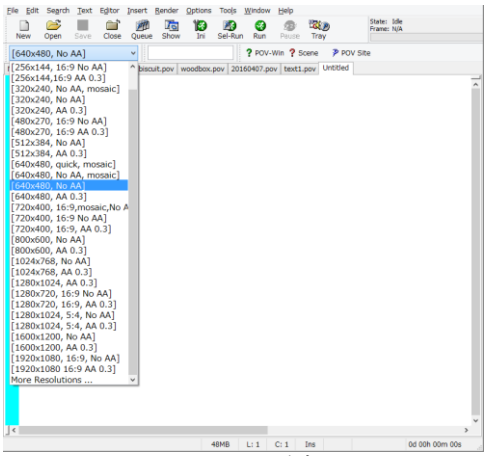


図 5　サイズ変更

III.　基本課題

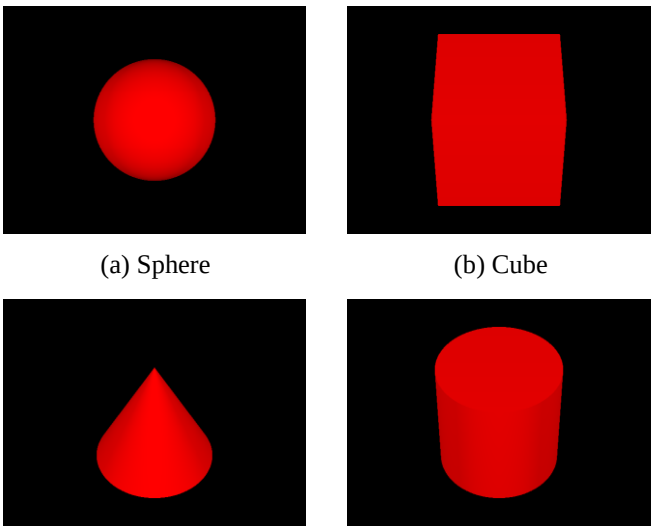
A.　Lesson-1「いろいろな形を作る」

下記のシーンファイル中の Sphere(球) を Cube(立方体) や Cone_Y(円錐) や Disk_Y(円柱) に変えて入力しレンダリング結果を確認する.

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<0, 10, -10>
   look_at<0, 0, 0>
   angle 20
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Sphere
   pigment{color Red}
}
```

ものについての命令は“object”の中に書く

図 6　Lesson-1 シーンファイル



(a) Sphere　(b) Cube

(b) Cone　(d) Disk

図 7　Lesson-1 実行結果

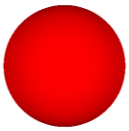
B.　Lesson-2「背景の色を決める」

1) 背景を白にする（下記のシーンファイルの灰色部を書き足す）.

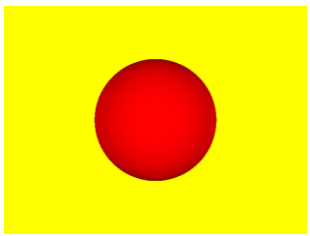
2) 下記のシーンファイル中の White を Blue や Yellow に変えて入力しレンダリング結果を確認する.

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<0, 10, -10>
   look_at<0, 0, 0>
   angle 20
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Sphere
   pigment{color Red}
}

background{color White}    ←背景色を指定
```

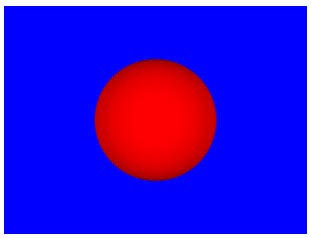
図 8　Lesson-2 シーンファイル



(a) White



(b) Yellow



(c) Blue

図 9　Lesson-2 実行結果

C.　Lesson-3「移動（座標変換 1）」

Sphere で原点に作られた球を移動して, 図 11 のように並べる（下記のシーンファイルの灰色部のように書き換える）.

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<0, 0, -10>
   look_at<0, 0, 0>
   angle 60
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Sphere
   pigment{color Red}
   translate<-3, 0, 0>    ←移動の命令
}                              （X 軸方向に−3 移動）
object{
   Sphere
   pigment{color Red}
   translate<3, 0, 0>
}
object{
   Sphere
   pigment{color Red}
   translate<0, 1, 0>
   translate<0, 2, 0>
}
object{
   Sphere
   pigment{color Red}
   translate<0, -3, 0>
}

background{color White}
```

図 10　Lesson-3 シーンファイル

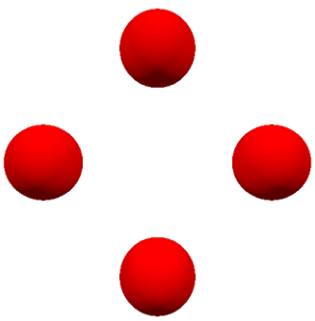


図 11　Lesson-3 実行結果

D.　Lesson-4「拡大・縮小(座標変換 2)」

Cone_Y で原点に作られた円錐を移動したり, 拡大・縮小したりして, 図 13 のように並べてみる（下記のシーンファイルの灰色部のように書き換える）.

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<0, 0, -10>
   look_at<0, 0, 0>
   angle 60
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Cone_Y
   pigment{color Red}
   scale 2   ←拡大縮小の命令（2 倍に拡大）
}
object{
   Cone_Y
   pigment{color Red}
   translate<-3, 0, 0>
}
object{
   Cone_Y
   pigment{color Red}
   translate<3, 0, 0>
}

background{color White}
```

図 12　Lesson-4 シーンファイル

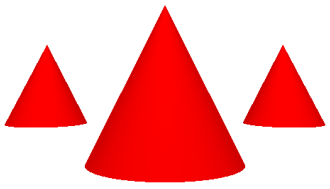


図 13　Lesson-4 実行結果

E.　Lesson-5「回転(座標変換 3)」

Cone_Y で原点に作られた円錐を移動したり, 拡大・縮小したりして, 図 15 のように並べてみる（下記のシーンファイルの灰色部のように書き換える）.

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<0, 0, -10>
   look_at<0, 0, 0>
   angle 60
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Cone_Y
   pigment{color Red}
   scale 2
   rotate -30*x   ←回転の命令（x 軸中心に−30 度）
}
object{
   Cone_Y
   pigment{color Red}
   rotate 30*z
   translate<-3, 0, 0>
}
object{
   Cone_Y
   pigment{color Red}
   rotate -30*z
   translate<3, 0, 0>
}

background{color White}
```

図 14　Lesson-5 シーンファイル

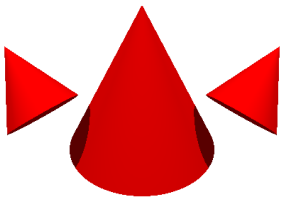


図 15　Lesson-5 実行結果

＜ POV-Ray の座標変換＞

移動は 3 次元座標系の x 方向, y 方向, z 方向に分けて考える. 方向にはプラス方向とマイナス方向がある.POV-Ray では左手系といわれる座標軸を採用している.原点に置いた球を移動するとき, z 軸上のマイナス側から原点に向けられたカメラでそれを見ると図 16 のようになる.

回転は x 軸, y 軸, z 軸を中心にそれぞれ何度回転するか指定する. 回転方向にはプラス方向とマイナス方向があり, 図 16 の軸上の矢印ようになっている.

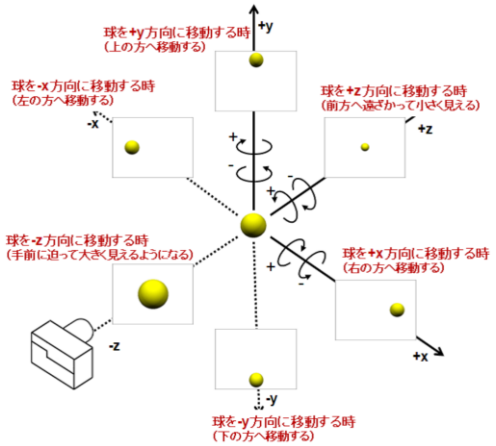


図 16　座標変換

F.　Lesson-6「色を指定する」

Sphere で作った球をいろいろな色を指定して図 17 のように並べてみる（下記のシーンファイルの灰色部のように書き換える）.

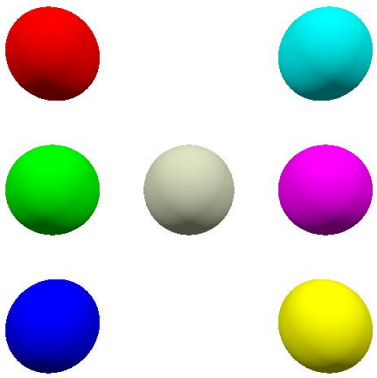


図 17　Lesson-6 実行結果

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<0, 0, -10>
   look_at<0, 0, 0>
   angle 60
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Sphere
   pigment{color red 0.68 green 0.74 blue 0.52}
}              ↑RGB による色指定
object{
   Sphere
   pigment{color Red}    ←色指定
   translate<-3, 3, 0>
}
object{
   Sphere
   pigment{color Green}
   translate<-3, 0, 0>
}
object{
   Sphere
   pigment{color Blue}
   translate<-3, -3, 0>
}
object{
   Sphere
   pigment{color Cyan}
   translate<3, 3, 0>
}
object{
   Sphere
   pigment{color Magenta}
   translate<3, 0, 0>
}
object{
   Sphere
   pigment{color Yellow}
   translate<3, -3, 0>
}

background{color White}
```

図 18　Lesson-6 シーンファイル

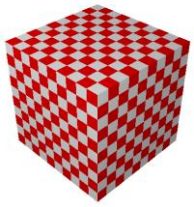
G.　Lesson-7「いろいろなパターンで色をつける」

1) Cube で原点に作られた立方体を 5 倍に拡大し,白と赤のチェッカー模様にする(図 20 参照).

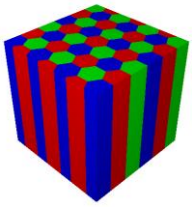
2) pigment の中で模様のパターンを brick と hexagon に変える(以下の命令を参考).

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<20, 20, -20>
   look_at<0, 0, 0>
   angle 40
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Cube    ←立方体
   scale 5    ←5 倍に拡大
   pigment{checker color White, color Red}
}                ↑チェッカー模様

background{color White}
```

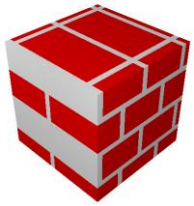
図 19　Lesson-7 シーンファイル



(a) checker



(b) hexagon



(c) brick

図 20　Lesson-7 実行結果

H.　Lesson-8「いろいろな模様で色をつける」

1) Disk_Y で原点に作られた円柱に, agate というパターンで模様をつける(図 22 参照). 下記のシーンファイルの灰色部のように書き換える.

2) pigment の中で agate を granite と radial に変えてレンダリングする.

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<10, 5, -10>
   look_at<0, 0, 0>
   angle 15
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Disk_Y    ←円柱
   pigment{agate}    ←石のような模様の 1 つ
}

background{color White}
```

図 21　Lesson-8 シーンファイル

(a) agate

(b) radial

(c) granite

図 22　Lesson-8 実行結果

I. Lesson-9「用意した画像を貼りつける(テクスチャマッピング)」

1) Disk_Y で原点に作られた円柱の側面に neko.jpg を，テクスチャマッピングする．下記のシーンファイルの灰色部のように書き換える．画像は bmp, png でも貼り付けることができるため，任意に用意して貼り付ける.画像はプログラムを保存したフォルダに置く．

2) object を Disk_Y から Sphere や Cube に変更しテクスチャマッピングする(以下の命令を参考).

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<10, 5, -10>
   look_at<0, 0, 0>
   angle 15
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Disk_Y    ↓テクスチャマッピングの命令
   pigment{image_map{jpeg "neko.jpg" map_type 2}}
}                              ↑形式、画像名、貼り付け方

background{color White}
```

図 23　Lesson-9 シーンファイル

(a) Sphere

(b) Cube

(c) Disk

(d) input image

図 24　Lesson-9 実行結果

J. Lesson-10「表面を凸凹にする」

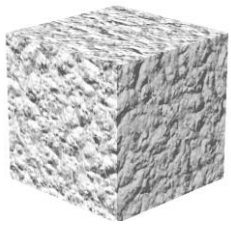
1) Cube で原点に作られた立方体を 2 倍に拡大し,crackle というパターンで表面を凸凹にする．下記のシーンファイルの灰色部のように書き換える．

2) 凸凹のパターンと凸凹の度合いを変化させレンダリングする(以下の命令を参考).

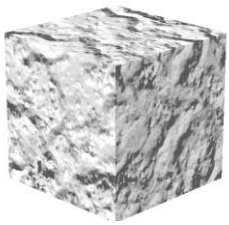
```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<10, 5, -10>
   look_at<0, 0, 0>
   angle 15
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Cube
   pigment{color White}    ←色は白
   normal{crackle 1.0}    ←表面を凸凹にする命令
}            ↑凸凹パターンと凸凹度合

background{color White}
```

図 25　Lesson-10 シーンファイル

(a) crackle



(b) granite



(c) agate

図 26　Lesson-10 実行結果

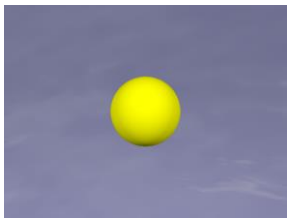
K. Lesson-11「空を作る」

1) skies.inc を使って, リアルな空のある画像を作る.

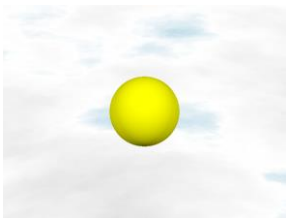
2) sky_sphere{S_Cloud5} の S_Cloud5 を S_Cloud1, S_Cloud2, S_Cloud3, S_Cloud4 と変更しレンダリングする.

```
#include "colors.inc"
#include "shapes.inc"
#include "skies.inc"    ←空指定に使用

camera{
   location<0, 5, -10>
   look_at<0, 0, 0>
   angle 40
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Sphere
   pigment{color Yellow}
}

sky_sphere{S_Cloud1}    ←空を作る命令
```

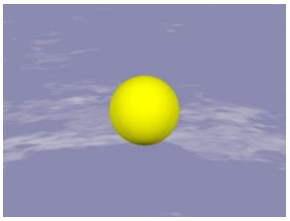
図 27 Lesson-11 シーンファイル



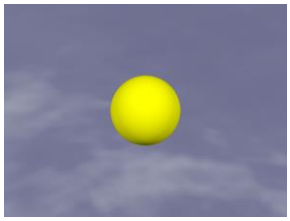
(a) S_Cloud1



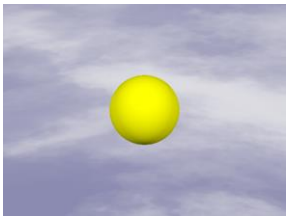
(b) S_Cloud2



(c) S_Cloud3



(d) S_Cloud4



(e) S_Cloud5

図 28 Lesson-11 実行結果

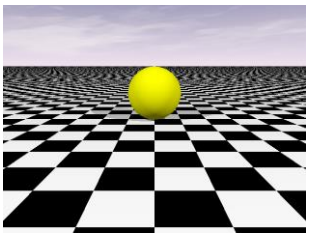
L. Lesson-12「地面を作る」

1) Plane_xz で作られた xy 平面を白と黒のチェッカー模様にし, それを地面とした画像を作る. 下記のシーンファイルの灰色部のように書き換える.

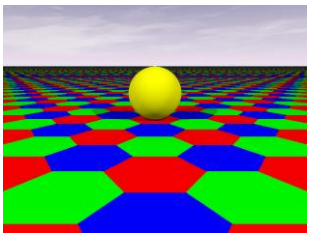
2) 地面を六角形模様(hexagon)や agate 模様に変更しレンダリングする(Lesson-7, Lesson-8 参考).

```
#include "colors.inc"
#include "shapes.inc"
#include "skies.inc"

camera{
   location<0, 2, -10>
   look_at<0, 0, 0>
   angle 60
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Sphere
   pigment{color Yellow}
   translate<0, 1, 0>
}
object{
   Plane_XZ    ←xz 平面を指定
   pigment{checker color White, color Black}
}

sky_sphere{S_Cloud1}
```

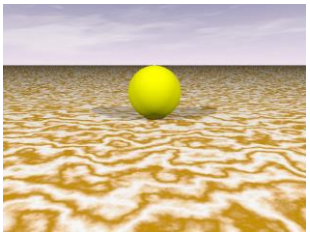
図 29 Lesson-12 シーンファイル



(a) checker



(b) hexagon



(c) agate

図 30 Lesson-12 実行結果

M. Lesson-13「形を組み合わせる（集合演算子）」

1) Disk_Y で作成した円柱と，Cube で作成した立方体を集合演算子の和(union) を作る．

2) Disk_Y で作成した円柱から，Cube で作成した立方体を集合演算子の差(difference) を作る．

3) Disk_Y で作成した円柱と，Cube で作成した立方体の集合演算子の積(intersection) を作る.下記のシーンファイルの灰色部を difference や intersection に書き換える．

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<0, 2, -10>
   look_at<0, 0, 0>
   angle 30
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}
light_source{<0, -10, -10> color White}

union{
   object{
      Disk_Y
      pigment{color Red}
   }
   object{
      Cube
      pigment{color Yellow}
      rotate 45*x
      rotate 45*y
   }
}

background{color White}
```

union で和をとる
（赤い円柱）＋
（黄色い立方体）

図 31　Lesson-13 シーンファイル

(a) union

(b) intersection

(c) difference

図 32　Lesson-13 実行結果

N. Lesson-14「石の質感にする」

1) stone.inc の中にある石の質感 T_Stone12 を使って，リアルな石の球を作る．

2) 下記のシーンファイルの灰色部 T_Stone12 を T_Stone16 や T_Stone33 に書き換えレンダリング結果を確認する.石の質感の名前として T_Stone1～T_Stone44 が使用できる．

```
#include "colors.inc"
#include "shapes.inc"
#include "skies.inc"
#include "stones.inc"    ←石の質感指定に使用

camera{
   location<0, 3, -10>
   look_at<0, 0, 0>
   angle 25
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Sphere
   texture{T_Stone12}    ←質感の指定
}

sky_sphere{S_Cloud2}
```

図 33　Lesson-14 シーンファイル

(a) T_Stone12

(b) T_Stone33

(c) T_Stone16

図 34　Lesson-14 実行結果

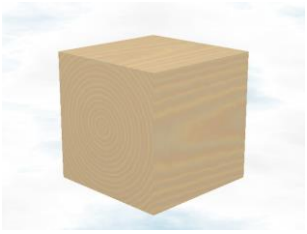
O.　Lesson-15「木の質感にする」

1) woods.inc の中にある木材の質感 T_Woods10 を使って，リアルな木材の立方体を作る．下記のシーンファイルの灰色部のように書き換える．

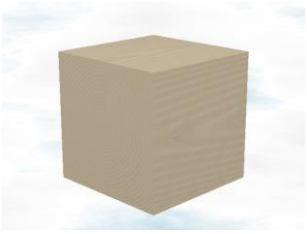
2) 下記のシーンファイルの灰色部 T_Woods10 を T_Woods3 や T_Woods14 に書き換えレンダリング結果を確認する.石の質感の名前として T_Woods1～T_Woods35 が使用できる．

```
#include "colors.inc"
#include "shapes.inc"
#include "skies.inc"
#include "woods.inc"    ←木の質感指定に使用

camera{
   location<0, 3, -10>
   look_at<0, 0, 0>
   angle 25
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Cube
   texture{T_Wood10}    ←質感の指定
   rotate 45*y
}

sky_sphere{S_Cloud2}
```

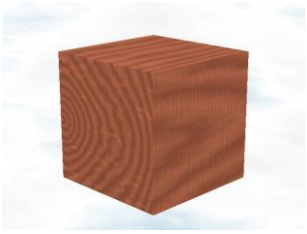
図 35　Lesson-15 シーンファイル



(a) T_Wood10



(b) T_Wood3



(c) T_Wood14

図 36　Lesson-15 実行結果

P.　Lesson-16「金属の質感にする」

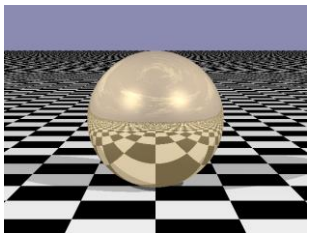
1) metals.inc の中にある金属の質感 T_Gold_4C を使って，リアルな金属の球を作る．下記のシーンファイルの灰色部のように書き換える．

2) 下記のシーンファイルの灰色部 T_Gold_4C を T_Silver_3B や T_Brass_3C や T_Copper_3C や T_Chrome_3B に書き換えレンダリング結果を確認する．

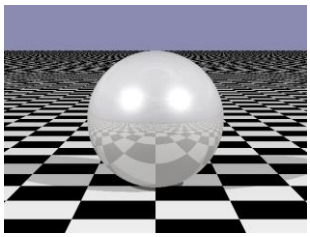
```
#include "colors.inc"
#include "shapes.inc"
#include "skies.inc"
#include "metals.inc"    ←金属の質感指定に使用

camera{
   location<0, 2, -10>
   look_at<0, 1, 0>
   angle 25
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Sphere
   texture{T_Gold_4C}    ←質感の指定
   translate<0, 1, 0>
}
object{
   Plane_XZ
   pigment{checker color White, color Black}
}

sky_sphere{S_Cloud3}
```

図 37　Lesson-16 シーンファイル

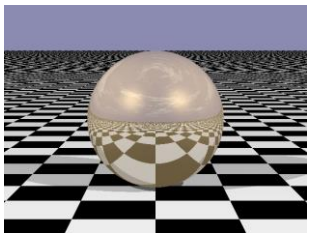
T_Gold_4C が用意された金属の質感の名前である．最後のアルファベットは光沢の度合い，その前の数字が色合いを表わしている．また Gold の他に，Silver(銀), Copper(銅), Brass(真鍮), Chrome(クロム鋼) が使用できる．



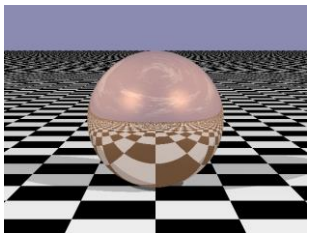
(a) T_Gold_4C



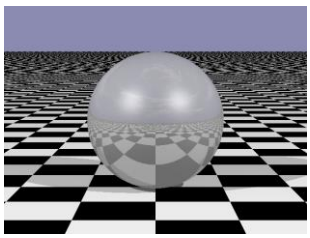
(b) T_Silver_3B



(c) T_Brass_3C



(d) T_Copper_3C



(e) T_Crome_3B

図 38　Lesson-16 実行結果

Q.　Lesson-17「ガラスの質感にする」

1) glass.inc の中にあるガラスの質感 T_Glass_3 を使って，リアルなガラスの球を作る．

2) 地面の模様を変える．下記のシーンファイルの 2 つ目の object の pigment を書き換える．checker を agate や radial に変更する．

3) 屈折率をダイヤモンドの屈折率(2.42)や空気の屈折率(1.00), 水の屈折率(1.33)に変更する．

その他のガラス質感名称

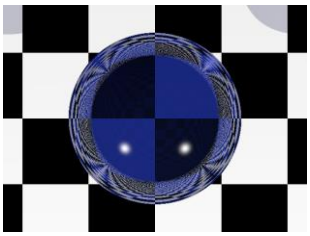
T_Glass1　T_Glass2　T_Glass3　T_Glass4　T_Old_Glass
T_Winebottle_Glass　T_Beerbottle_Glass　T_Ruby_Glass
T_Green_Glass　T_Dark_Green_Glass　T_Yellow_Glass
T_Orange_Glass　T_Vicksbottle_Glass

図 39　ガラス質感の種類

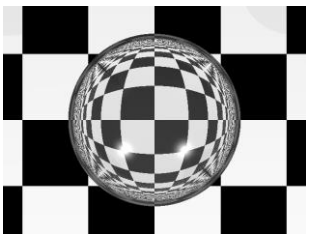
```
#include "colors.inc"
#include "shapes.inc"
#include "glass.inc"    ←ガラスの質感指定に使用

camera{
   location<0, 10, 0>
   look_at<0, 0, 0>
   angle 20
}

light_source{<-10, 10, -10> color White}
light_source{<10, 10, -10> color White}

object{
   Sphere
   texture{T_Glass3}    ←質感の指定
   interior{I_Glass ior 1.5}    ←屈折率の指定
}
object{
   Plane_XZ
   pigment{checker color White, color Black}
   translate<0, -3, 0>
}
```

図 40　Lesson-17 シーンファイル

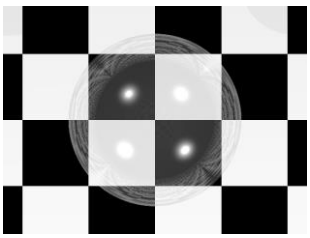
(a) T_Glass_3
屈折率：1.5



(b) T_Winebottle_Glass
屈折率：1.5



(c) T_Glass_3
屈折率：2.42



(d) T_Glass_3
屈折率：1.00

図 41　Lesson-17 実行結果

IV. 応用課題「カプセルのある風景」

次のようなカプセルのある風景を作る. 6 つのステップに分けて制作過程を紹介する.

A. ステップ 1　～カプセルを作る～

黄色い球, 黄色い円柱, 赤い円柱, 赤い球を集合演算子の和でつなげてカプセルとする. よりカプセルらしくするために, 黄色い円柱と球を, 赤いほうより少し小さくする. またカプセルにはハイライトを入れて光沢のある質感を表現する.

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<0, 10, -10>
   look_at<0, 0, 0>
   angle 35
}

light_source{<10, 10, -10> color 2*White}
                              ↑ライトの明るさを 2 倍
union{
   object{
      Disk_X
      pigment{color Yellow}
      finish{phong 1.0}   ←ハイライトを入れる
      scale<1, 0.95, 0.95>   ←x, y, z 方向にそれぞれ
      translate -1*x              1 倍，0.95 倍，0.95 倍
   }
   object{
      Sphere
      pigment{color Yellow}
      finish{phong 1.0}
      scale 0.95
      translate -2*x
   }
   object{
      Disk_X
      pigment{color Red}
      finish{phong 1.0}
      translate 1*x
   }
   object{
      Sphere
      pigment{color Red}
      finish{phong 1.0}
      translate 2*x
   }
}
```

図 41　ステップ 1 シーンファイル

図 42　ステップ 1 実行結果

B. ステップ 2　～カプセルを作る命令を作成する～

カプセルを作る命令を作り, その命令を使ってカプセルを表示するように step1 のシーンファイルを書き換える. カプセルを作るための kapuseru1 という命令を作り,これを使って最後にカプセルを表示する. 出来上がる画像は step1 と変わらない.

このように命令にしておけば, 同じ形のものをいくつでも作ることができる.

```
#include "colors.inc"
#include "shapes.inc"

camera{
   location<0, 10, -10>
   look_at<0, 0, 0>
   angle 35
}

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

#declare Kapuseru1=union{   ←Kapuseru1 という命令を作る
   object{
      Disk_X
      pigment{color Yellow}
      finish{phong 1.0}
      scale<1, 0.95, 0.95>
      translate -1*x
   }
   object{
      Sphere
      pigment{color Yellow}
      finish{phong 1.0}
      scale 0.95
      translate -2*x
   }
   object{
      Disk_X
      pigment{color Red}
      finish{phong 1.0}
      translate 1*x
   }
   object{
      Sphere
      pigment{color Red}
      finish{phong 1.0}
      translate 2*x
   }
}

object{Kapuseru1}   ←Kapuseru1 を使ってカプセルを作る
```

図 43　ステップ 2 シーンファイル

図 44　ステップ 2 実行結果

C. ステップ 3　～カメラの位置を調節する/台を作る～

カメラの設定を変えて，より広く見渡せるようにする（画角を広くする）．またカプセルの置かれている台を木材の質感で表わす．

```
#include "colors.inc"
#include "shapes.inc"
#include "woods.inc"

camera{
   location<1, 10, -10>
   look_at<0, 0, 0>
   angle 60
}

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

#declare Kapuseru1=union{
   object{
      Disk_X
      pigment{color Yellow}
      finish{phong 1.0}
      scale<1, 0.95, 0.95>
      translate -1*x
   }
   object{
      Sphere
      pigment{color Yellow}
      finish{phong 1.0}
      scale 0.95
      translate -2*x
   }
   object{
      Disk_X
      pigment{color Red}
      finish{phong 1.0}
      translate 1*x
   }
   object{
      Sphere
      pigment{color Red}
      finish{phong 1.0}
      translate 2*x
   }
}

object{Kapuseru1}

object{
   Plane_XZ
   texture{T_Wood7}
   translate -0.5*y
}
```

図 45　ステップ 3 シーンファイル

図 46　ステップ 3 実行結果

D. ステップ 4　～色の違うカプセルを作る～

kapuseru2 という名前でもう 1 つ色違いのカプセルを作成する命令を作る．そしてこの kapuseru2 を使って最初に作った黄と赤のカプセルと違う位置に，色違いのカプセルを表示させる．

```
…省略，step3 と同一プログラム

#declare Kapuseru2=union{   ←新しく Kapuseru2 という命令を作る
   object{
      Disk_X
      pigment{color White}
      finish{phong 1.0}
      scale<1, 0.95, 0.95>
      translate -1*x
   }
   object{
      Sphere
      pigment{color White}
      finish{phong 1.0}
      scale 0.95
      translate -2*x
   }
   object{
      Disk_X
      pigment{color PaleGreen}
      finish{phong 1.0}
      translate 1*x
   }
   object{
      Sphere
      pigment{color PaleGreen}
      finish{phong 1.0}
      translate 2*x
   }
}

object{Kapuseru1}   ←1 つ目のカプセル

object{   ←2 つ目のカプセルは向きを変え、位置を変更
   Kapuseru2
   rotate -120*y
   translate<-5, 0, 0>
}

object{
   Plane_XZ
   texture{T_Wood7}
   translate -0.5*y
}
```

図 47　ステップ 4 シーンファイル

図 48　ステップ 4 実行結果

E. ステップ 5　～沢山のカプセルを作る～

kapuseru2 を使って, さらにカプセルを追加する.

```
…省略, step4 と同一プログラム

object{Kapuseru1}

object{
   Kapuseru2
   rotate -120*y
   translate<-5, 0, 0>
}
object{
   Kapuseru2
   rotate 30*y
   translate<1, 0, 4>
}
object{
   Kapuseru2
   rotate -200*y
   translate<-2, 0, -4>
}
object{
   Kapuseru2
   rotate 120*y
   translate<5, 0, -4>
}

object{
   Plane_XZ
   texture{T_Wood7}
   translate -0.5*y
}
```

これらを追加

図 49　ステップ 5 シーンファイル

図 50　ステップ 5 実行結果

F. ステップ 6　～照明をスポットライトにする～

ライトをスポットライトにする. そのためライトの設定にスポットライト用の命令を追加する.

```
#include "colors.inc"
#include "shapes.inc"
#include "woods.inc"

camera{
   location<1, 10, -10>
   look_at<0, 0, 0>
   angle 60
}

light_source{<10, 10, -10> color 2*White
                  spotlight
                  point_at<0, 0, 0>
                  radius 15
                  falloff 25
}

…省略, step5 と同一プログラム
```

図 51　ステップ 6 シーンファイル

図 52　ステップ 6 実行結果

V.　最後に

基本編で学んだこともとにして, 応用編では, 少し複雑な作品を生成した. POV-Ray にはたくさんの命令があり, 本資料に登場するのは, そのほんの一部である. 使う人のアイデアと勉強次第で, 創造的な素晴らしい作品を自由に作れるようになる.

VI. 作例と参考資料

A. 作例「置時計のある風景」

```
#include "colors.inc"
#include "shapes.inc"
#include "metals.inc"
#include "stones.inc"
#include "glass.inc"
#include "woods.inc"

camera{
   location<15, 20, -50>
   look_at<0, 0, 0>
   angle 35
}

light_source{
   <10, 10, -10> color 0.5*White
   area_light
   <8, 0, 0><0, 0, 8>
   4, 4
   jitter
   translate<-10, 10, -10>
}
light_source{
   <10, 10, -10> color White
   area_light
   <8, 0, 0><0, 0, 8>
   4, 4
   jitter
   translate<100, 100, -100>
}

background{color White}

#declare Hontai = union{
   object{
      Disk_Z
      scale<10, 10, 4>
   }
   object{
      torus{9, 1}
      rotate 90*x
      translate -4*z
   }
}

#declare Dai = object{
   Cube
   texture{T_Stone12}
   scale<8, 1, 4>
   translate -10*y
}

#declare Mojiban = union{
   object{
      Disk_Z
      scale<8, 8, 0.01>
      translate -4*z
      pigment{color White}
   }

//memori 5minutes
#declare Cnt=1;
#while(Cnt<=12)
   object{
      Disk_Y
      texture{T_Chrome_3B}
      scale<0.3, 0.5, 0.3>
      translate<0, 7, -4>
      rotate 30*Cnt*z
   }
   #declare Cnt = Cnt+1;
#end

//memori 1minute
#declare Cnt=1;
#while(Cnt<=60)
   object{
      Disk_Y
      texture{T_Chrome_3B}
      scale<0.1, 0.2, 0.1>
      translate<0, 7, -4>
      rotate 6*Cnt*z
   }
   #declare Cnt = Cnt+1;
#end

//jiku
object{
   Disk_Z
   texture{T_Chrome_3B}
   scale<0.5, 0.5, 0.8>
   translate -4*z
}

//jikan settei
#declare Hour=4;
#declare Min=38;
#declare Sec=3;

//tanshin
object{
   Disk_Y
   pigment{color Black}
   scale<0.2, 3, 0.2>
   translate<0, 1.5, -4.2>
   rotate -(30*Hour+30/60*Min+30/3600*Sec)*z
}

//chousin
object{
   Disk_Y
   pigment{color Black}
   scale<0.15, 4, 0.15>
   translate<0, 2.5, -4.4>
   rotate -(6*Min+6/60*Sec)*z
}

//byoshin
object{
   Disk_Y
   pigment{color Red}
   scale<0.1, 4, 0.1>
   translate<0, 2.5, -4.6>
   rotate -(6*Sec)*z
}

//moji
object{
   text{ttf "crystal.ttf", "CG-CLOCK", 0.2, 0}
   texture{T_Chrome_3B}
   translate<-2, 3.5, -4.1>
}
}

#declare Garasu = object{
   Disk_Z
   scale<9, 9, 0.2>
   translate -4.8*z
   texture{T_Glass3}
```

```
    interior{I_Glass}
}
#declare Tokei = union{
    object{Hontai texture{T_Brass_3C}}
    object{Dai}
    object{Mojiban}
    object{Garasu}
}

object{Tokei}

//yuka
object{
    Plane_XZ
    texture{T_Wood34 scale 4}
    translate -11*y
}

//kabe
object{
    Plane_XY
    pigment{color White}
    normal{crackle 2.0 scale 3}
    translate 25*z
}
```

図 51　ステップ 6 シーンファイル

図 52　ステップ 6 実行結果

B. 参考資料

- 教科書「はじめてのCG」, 小室日出樹, CG-ARTS協会.
  本資料の内容は, この教科書に沿って進めている.
- http://izumi-math.jp/sanae/Pov_Ray/Pov_Ray.htm
  サイト名「POV-Ray First」.
  本資料をより詳しくした内容が掲載されている.
  金属, ガラス, 空などの詳しい種類も.
- http://nishimulabo.edhs.ynu.ac.jp/ povray/
  サイト名「POV-Ray station」.

VII. 付録（POV-RAY のインストール方法）

Windows の場合：

1. POV-Ray の公式サイト（http://www.povray.org/）へ行く.
2. 公式サイトの左上メニューにある"Download"という文字をクリックする.
3. 出てきたページの中段左側に”Windows Binary”という枠がある. その枠内の下部にある”Download Windows Installer”というボタンをクリックし, ファイルをダウンロードする.
4. ダウンロードが終わると指定した保存先に povwin-3.7-agpl3-setup.exe というアイコンがあるので, これを開くことで自動的にインストールを促す画面が出る.以下, ウインドウの指示に従いインストールを実行する.

Mac の場合：

1. オフィシャルの POV-Ray は Mac に対応していない. そこで非公式のものを

”http://news.povray.org/povray.general/thread/%3C1kyr400.1yw0xyuy8n5s8N%25yvo.s%40cancel_This_gmx.net%3E/?ttop=384552&toff=50”

からダウンロードする.

2. 起動できたらインクルードファイルのディレクトリにパスを通す. メニューの PovRay3.7Unofficial>Preferences... と進んで System Includes タブの Change をクリックする. 開いたウインドウで先ほどアプリケーションに置いた POV-Ray 3.7Unofficial 内の include を選択して Open をクリックすると使用できる状態になる.