授業のおおまかな予定

4/05 ガインダンス
4/12
4/19
4/26
5/10
5/17

2次元グラフィックス
- 四角形，円形，直線の描画
- カラー画像
- 座標変換

5/24 休講
5/27(月) 6限 補講(OD2)
5/31 (3201教室集合) ← 中間試験(小テスト)

6/07
6/14
6/21
6/28 休講
7/01(月) 6限 補講(OD2)
7/05 休講
7/12
7/19

3次元グラフィックス
- 物体形状の表現
- シェーディング
- 隠面消去
- OpenGL
- CG作品

7/31 ← 期末試験

レポート	10%
出席	10%
修了作品	20%
中間試験	20%
期末試験	40%

修了作品提出〆切
8月10日(土) 24時

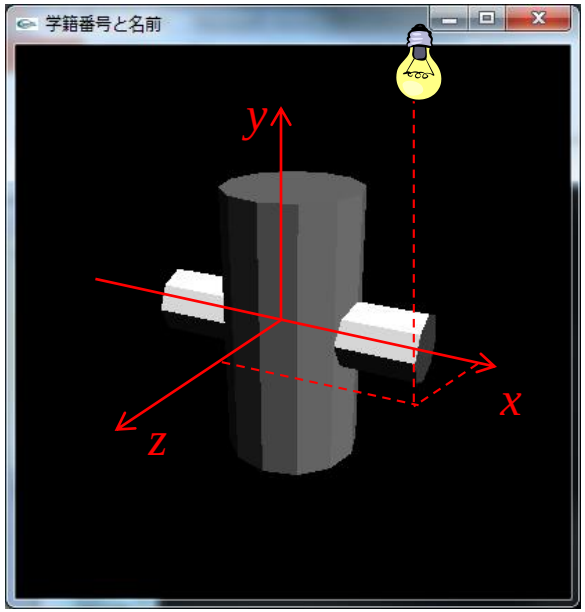
OD教室には8月にオープン利用不可の日がある場合があるので注意！

ライトの位置設定

復習

Example13-1

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("学籍番号と名前");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 1.0, 20.0);
    gluLookAt(4, 4, 0,   0, 0, -8,   0, 1, 0);

    glShadeModel(GL_FLAT);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);

    float position[] = {5.0, 10.0, 2.0, 0.0};
    glLightfv(GL_LIGHT0, GL_POSITION, position);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutMainLoop();
}
```



位置ベクトル position
$= (x, y, z, 0)$

positionをライト0番の位置として設定する

平行光源と点光源

復習

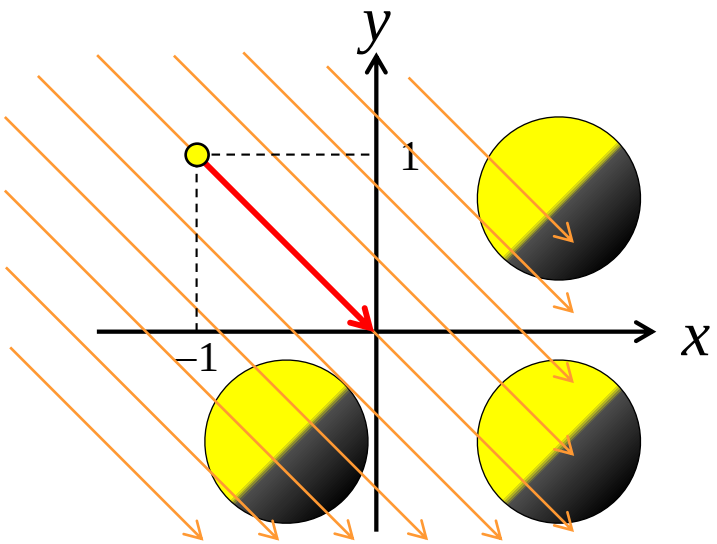
LIGHT0を**平行光源**に設定

```
float position[] = {-1.0, 1.0, 0.0, 0.0};     //平行光源位置
glLightfv(GL_LIGHT0, GL_POSITION, position); //平行光源に設定
```

LIGHT0を**点光源**に設定

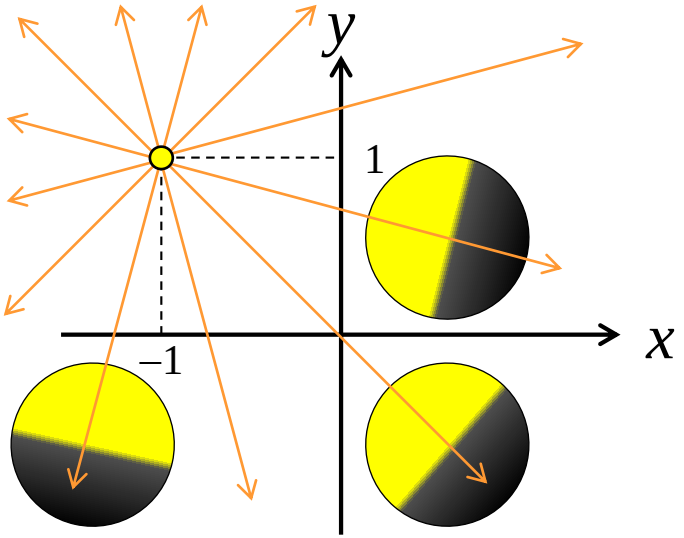
```
float position[] = {-1.0, 1.0, 0.0, 1.0};     //点光源位置
glLightfv(GL_LIGHT0, GL_POSITION, position); //点光源に設定
```

配列の4番目の要素を1にすると点光源になる



平行光源

座標値は光源から原点に向かう**方向ベクトル**を表す



点光源

座標値は光源の**位置**を表す

ライト移動のアニメーション

復習

Example13-2

```
// 問題12-2の改良
float angle = 0;

void display( void )
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    glColor3f(1.0, 1.0, 1.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0, -8.0);

    float position[] = {0.0, 5.0, 2.0, 0.0};
    glPushMatrix();
        glRotatef(angle, 0.0, 1.0, 0.0);
        glLightfv(GL_LIGHT0, GL_POSITION, position);
    glPopMatrix();

    glPushMatrix();
        glScalef(1.0, 2.0, 1.0);
        Prism(12);
    glPopMatrix();

    glPushMatrix();
        glRotatef(90, 0, 0, 1);
        glScalef(0.5, 2.0, 0.5);
        Prism(8);
    glPopMatrix();

    glFlush();
}
```

位置ベクトル position $= (x, y, z, 0)$

y軸周りでangle度回転する幾何変換行列を設定

ここでライトの位置を設定．このとき，幾何変換行列で変換される

この物体の周りをライトが回転する

ライトの位置は幾何変換の対象である

＝ライトの座標に幾何変換行列Mをかけてからシェーディング処理が行われる

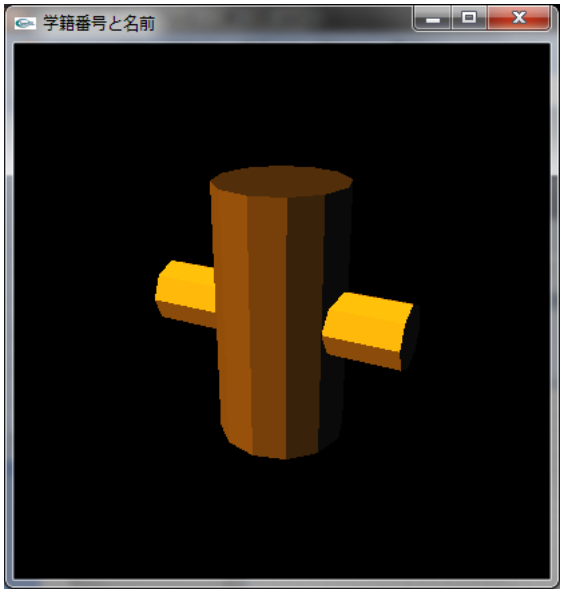
```
void idleFunc(void)
{
    angle += 0.05;
    if (angle >= 360)
        angle = 0;
    glutPostRedisplay();
}

int main(void)
{

    // ・・・
    // main()関数にて

    glutIdleFunc(idleFunc);
}
```

ライト色の設定

復習

Example13-3

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("学籍番号と名前");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 1.0, 20.0);
    gluLookAt(4, 4, 0,   0, 0, -8,   0, 1, 0);

    glShadeModel(GL_FLAT);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);

    float amber[] = {1.0, 0.5, 0.0, 1.0};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, amber);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutMainLoop();
}
```



色 amber
$= (R, G, B, 1.0)$

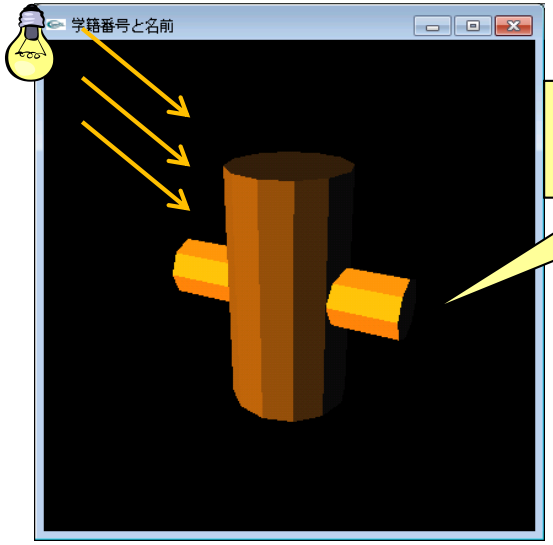
amberをライト0番の色として設定する

色自体をアニメーションで変化することも可能！

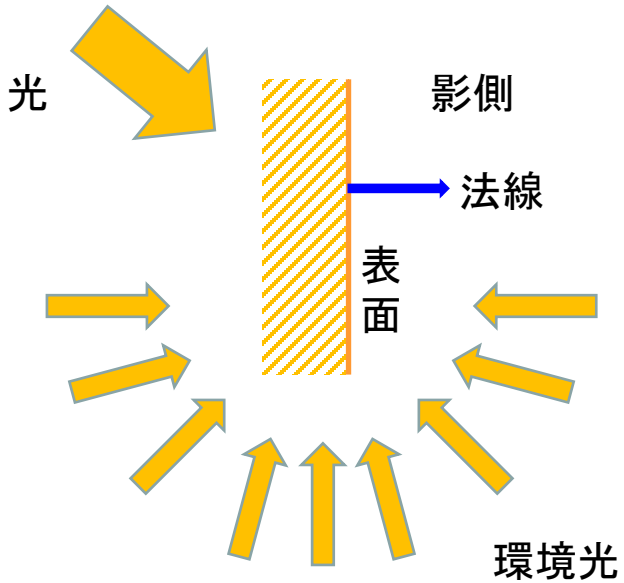
amber[0] 赤色の明度
amber[1] 緑色の明度
amber[2] 青色の明度

環境光

復習



こちらの面が真っ黒

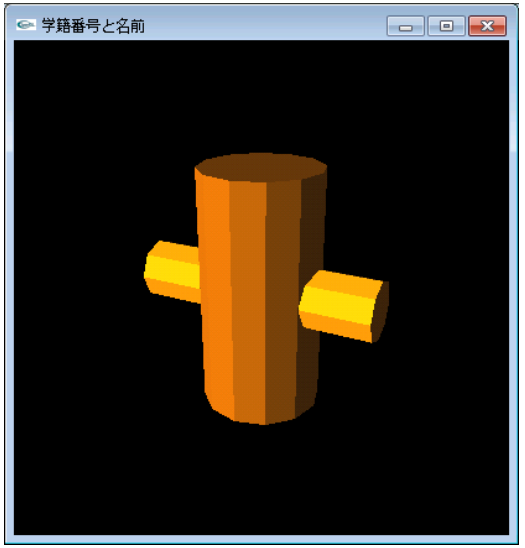


ambient

環境光＝光源の位置や向きに関係なく周囲全体にある光

```
glShadeModel(GL_FLAT);
glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);

float amber[] = {1.0, 0.5, 0.0, 1.0};
glLightfv(GL_LIGHT0, GL_DIFFUSE, amber);
glLightfv(GL_LIGHT0, GL_AMBIENT, amber);
```

環境光もアンバー色に設定

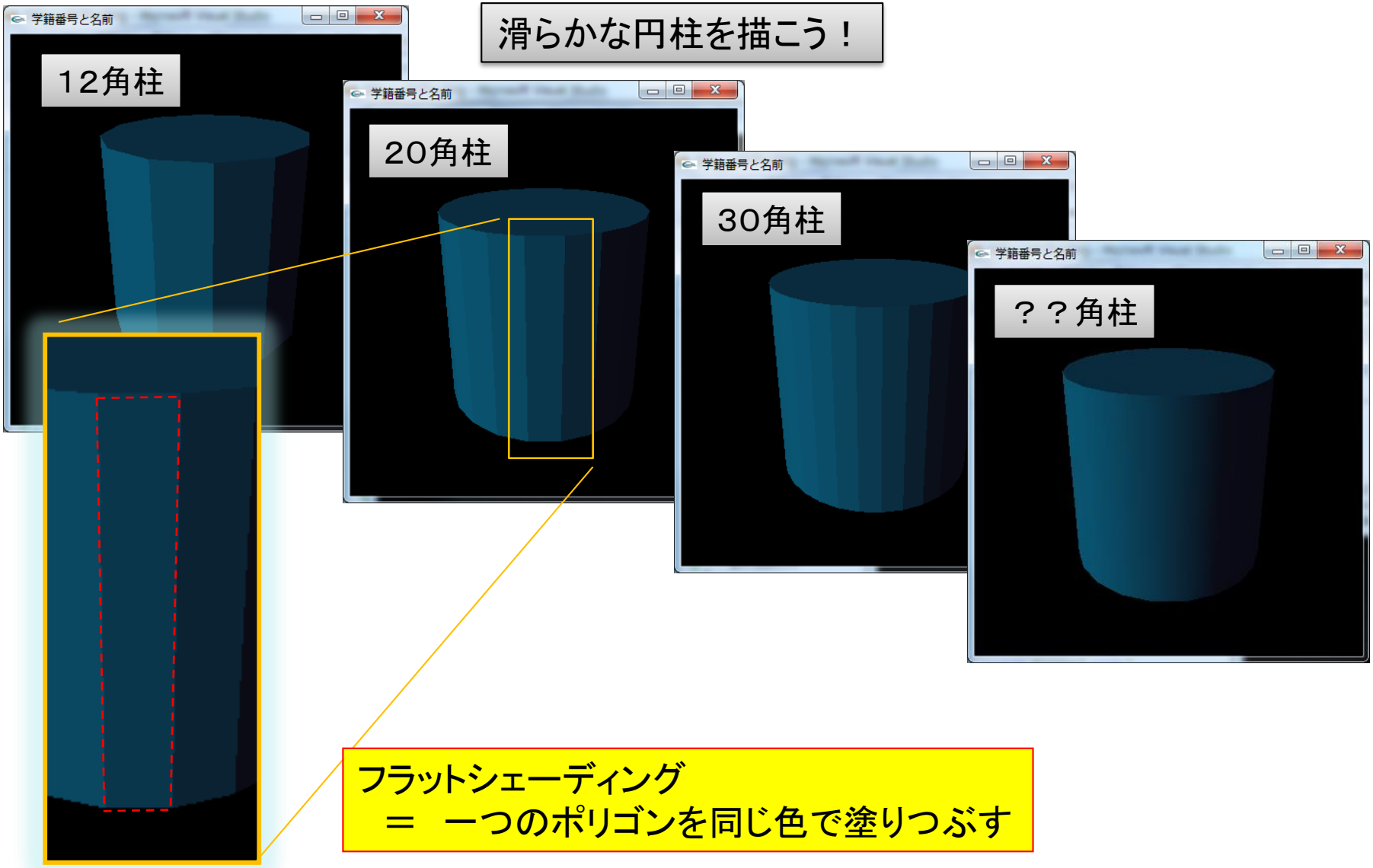


環境光の色をライトの色と違う色に設定することも可能

スムースシェーディング(1)

復習

滑らかな円柱を描こう！

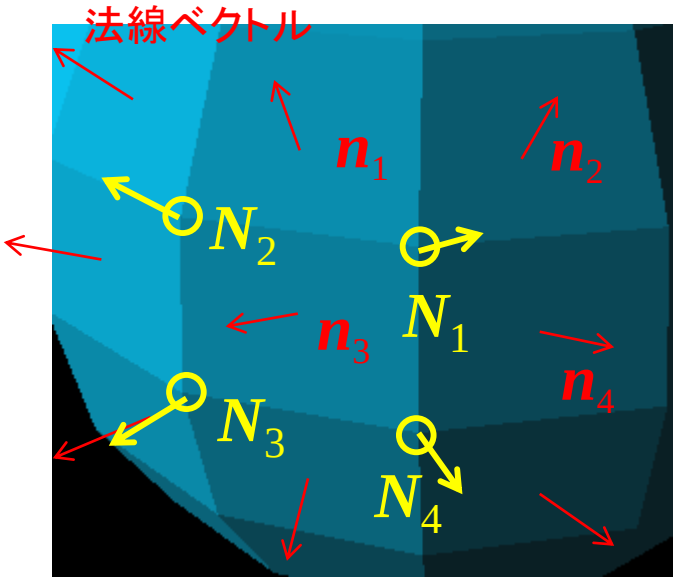


フラットシェーディング
＝　一つのポリゴンを同じ色で塗りつぶす

グーローシェーディング

復習

(Gouraud shading)



(1) 頂点の法線ベクトルを求める

頂点の法線ベクトルは隣接するポリゴンの法線ベクトルの平均値

(例)

$$N_1 = \frac{n_1 + n_2 + n_3 + n_4}{4}$$

(2) 各頂点の明るさを求める

Lambertの余弦則を用いる

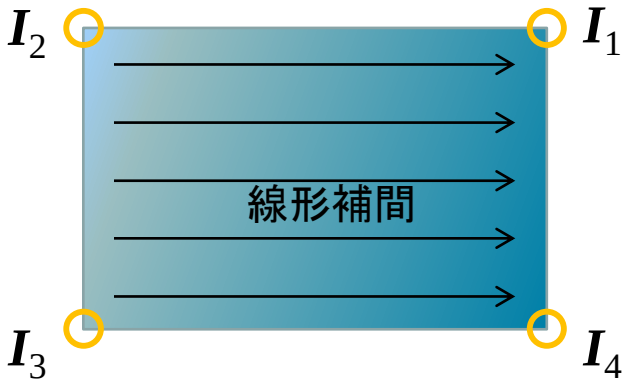
(3) 頂点の明るさを線形補間してポリゴン内を塗りつぶす

Lambertの余弦則 (Lambert反射)

面の明るさ　$I_d = I_0 \cos\theta$

I_0　:元の光の強さ

θ　:光線が面の法線と為す角度

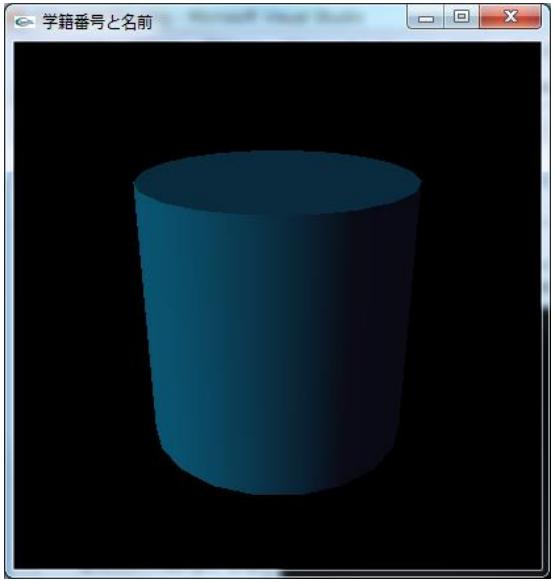


OpenGLでのスムースシェーディングの設定

復習

Example13-4

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("学籍番号と名前");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 1.0, 20.0);
    gluLookAt(4, 4, 0,    0, 0, -8,    0, 1, 0);

    glShadeModel(GL_SMOOTH);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);

    float blue[] = {0.0, 0.8, 1.0, 1.0};
    float tblue[] =  {0.0, 0.0, 0.2, 1.0};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, blue);
    glLightfv(GL_LIGHT0, GL_AMBIENT, tblue);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutMainLoop();
}
```



シェーディングをスムースに切り替える

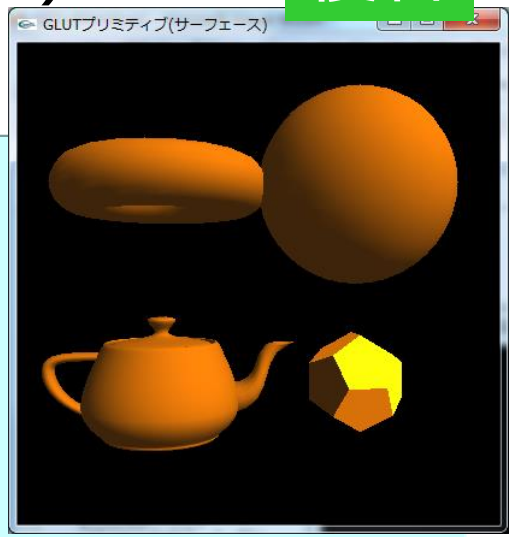
ペイントハンドラでは20角柱を描画

GLUTの幾何オブジェクト(2)

復習

Example13-6

```
#include "glut.h"
#include <GL/gl.h>
#include <math.h>

void KeyboardHandler(unsigned char key, int x, int y);

void display( void )
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f( 1.0, 1.0, 1.0 );

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);

    glPushMatrix();
    glTranslatef(1, 1, 0);
    glRotatef(90, 1, 0, 0);
    glutSolidSphere(1, 20, 20);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-1, 1, 0);
    glRotatef(90, 1, 0, 0);
    glutSolidTorus(0.3, 0.8, 20, 20);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(1, -1, 0);
    glRotatef(5, 0, 1, 0);
    glScalef(0.3, 0.3, 0.3);
    glutSolidDodecahedron();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-1, -1, 0);
    glRotatef(5, 0, 1, 0);
    glutSolidTeapot(0.8);
    glPopMatrix();
    glFlush();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("GLUTプリミティブ(サーフェース)");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45, 1.0, 1.0, 10.0);
    glShadeModel(GL_SMOOTH);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
    float position[] = {1.0, 1.0, 1.0, 0.0};
    glLightfv(GL_LIGHT0, GL_POSITION, position);
    float amber[] = {1.0, 0.5, 0.0, 1.0};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, amber);
    glLightfv(GL_LIGHT0, GL_AMBIENT, amber);
    glEnable(GL_DEPTH_TEST);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutMainLoop();
}
```



詳細は
http://opengl.jp/glut/section11.html

復習

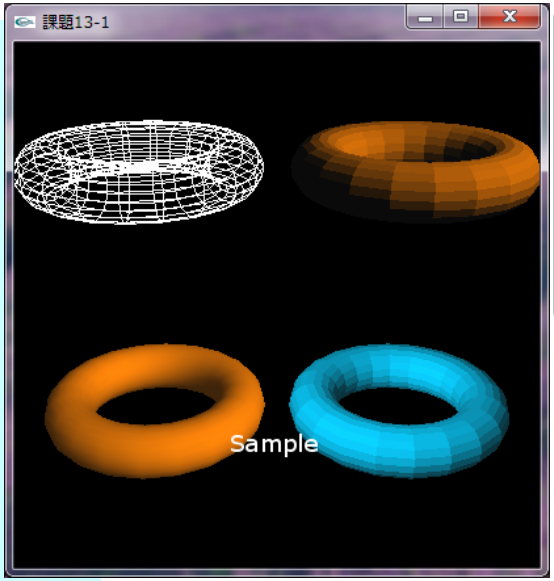
基本課題13

ウィンドウの名前を各自の学籍番号と氏名にすること

次のmain()関数をそのまま用い，実行例のように様々なレンダリングが混在する画像を作成しなさい．なお，物体はすべて内径0.2，外径0.7で20×20に分割したトーラスである．

Report13-1

```
#include "glut.h"
#include <GL/gl.h>
#include <math.h>
void KeyboardHandler(unsigned char key, int x, int y);
void display( void );
float blue[] = {0.0, 0.8, 1.0, 1.0};    //青
float amber[] = {1.0, 0.5, 0.0, 1.0};   //アンバー
float white[] = {1.0, 1.0, 1.0, 1.0};   //白
float black[] = {0.0, 0.0, 0.0, 1.0};   //黒
float positionR[] = {1.0, 1.0, 1.0, 0.0};   //右上位置
float positionL[] = {-1.0, 1.0, 1.0, 0.0};  //左上位置

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("課題13-1");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(35, 1.0, 1.0, 10.0);
    gluLookAt(0, 2.5, 0,   0, 1.2, -3,   0, 1, 0);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
    glEnable(GL_DEPTH_TEST);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutMainLoop();
}
```



以下の二つのファイルをZIPファイルに入れて提出すること．

① Wordのレポート
- ソースプログラム
- glutウィンドウの画面コピー

② **実行プログラム(〇〇〇.exe)**

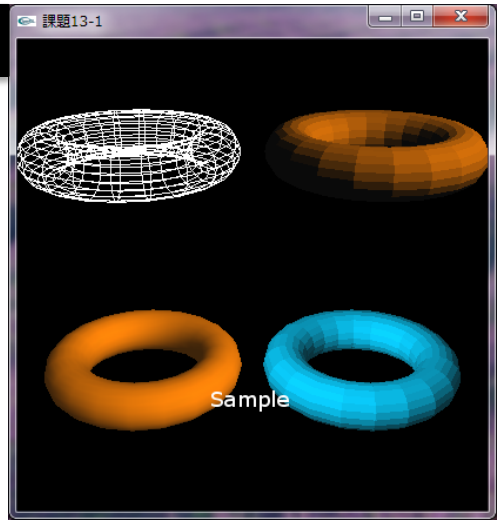
ヒント：シェーディングをオフにするには下記を用いる
glDisable(GL_LIGHTING)

左上 <中心 (−1, +1, −6)> ワイヤフレーム シェーディング無し 白色	右上 <中心 (+1, +1, −6)> サーフェース フラットシェーディング 色：アンバー(右上から照明) 環境光：なし
左下 <中心 (−1, −1, −6)> サーフェース スムースシェーディング 色：アンバー(右上から照明) 環境光：アンバー	右下<中心 (+1, −1, −6)> サーフェース フラットシェーディング 色：青(左上から照明) 環境光：青

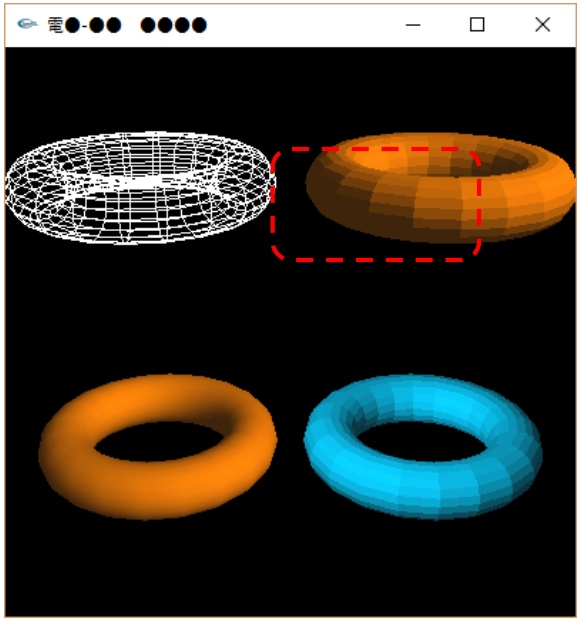
glutウィンドウを変化しても常に同じレンダリング結果になること！

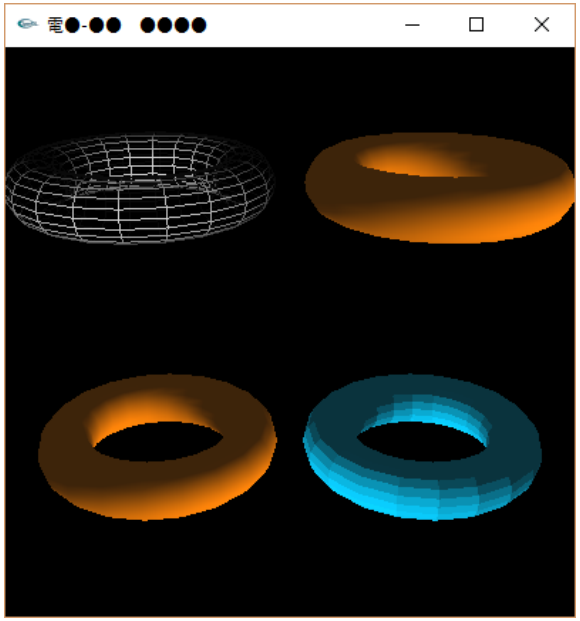
基本課題13の間違い例

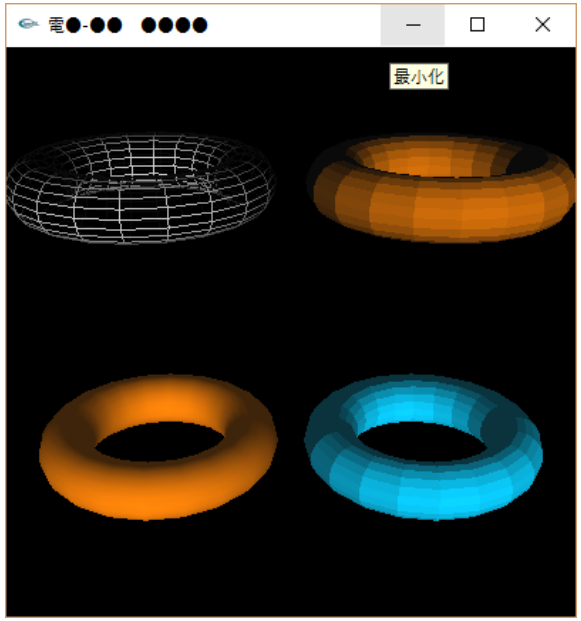
正解



左上 <中心 (−1, +1, −6)> ワイヤフレーム シェーディング無し 白色	右上 <中心 (+1, +1, −6)> サーフェース フラットシェーディング 色：アンバー（右上から照明） 環境光：なし
左下 <中心 (−1, −1, −6)> サーフェース スムースシェーディング 色：アンバー（右上から照明） 環境光：アンバー	右下<中心 (+1, −1, −6)> サーフェース フラットシェーディング 色：青（左上から照明） 環境光：青

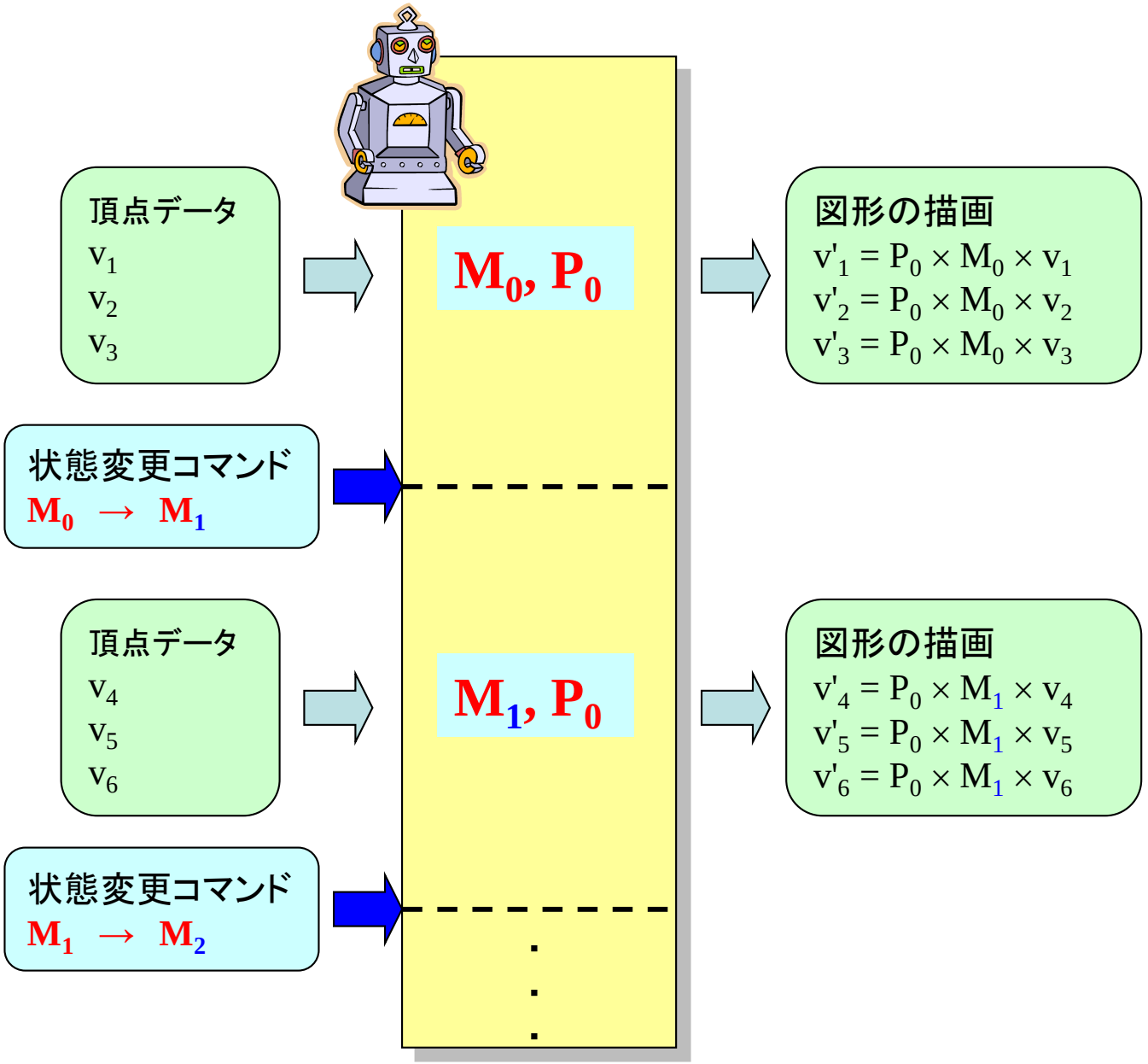






OpenGL状態マシンの動作

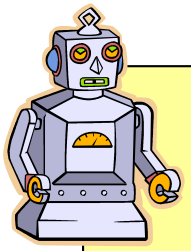
復習



OpenGL状態マシンの動作

OpenGLの状態

- 幾何変換行列
- 投影変換行列
- 光源の位置
- 光源の色
- シェーディング
  フラット or スムース
- シェーディング
  ON or OFF
- 隠面消去
  ON or OFF
- その他
  (全ての設定)



頂点データ
v_1
v_2
v_3

初期状態

初期状態で描画
v'_1
v'_2
v'_3

状態変更コマンド
初期状態→状態1

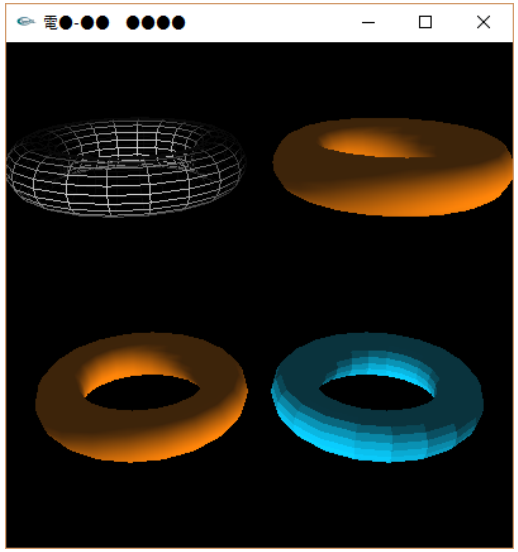
頂点データ
v_4
v_5
v_6

状態1

状態1で描画
v'_4
v'_5
v'_6

状態変更コマンド
状態1 → 状態2

.
.
.



間違いの理由

ペイントハンドラの前回の呼び出しで変更された「状態」が次回の描画に影響する場合がある

```
int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("電●-●●  ●●●●");

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(35, 1.0, 1.0, 10.0);
    gluLookAt(0, 2.5, 0, 0, 1.2, -3, 0, 1, 0);

    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
    glEnable(GL_DEPTH_TEST);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutMainLoop();
}
```

ライト設定

⇨ イベント待ちループ

影響

```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glColor3f(1.0, 1.0, 1.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);

    glPushMatrix();
    glTranslatef(-1, 1, 0);
    glRotatef(90, 1, 0, 0);
    glutWireTorus(0.2, 0.7, 20, 20);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, amber);
    glLightfv(GL_LIGHT0, GL_AMBIENT, amber);
    glLightfv(GL_LIGHT0, GL_POSITION, positionR);
    glEnable(GL_LIGHTING);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(1, 1, 0);
    glRotatef(90, 1, 0, 0);
    glutSolidTorus(0.2, 0.7, 20, 20);
    glShadeModel(GL_SMOOTH);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, amber);
    glLightfv(GL_LIGHT0, GL_AMBIENT, amber);
    glLightfv(GL_LIGHT0, GL_POSITION, positionR);
    glEnable(GL_LIGHTING);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-1, -1, 0);
    glRotatef(90, 1, 0, 0);
    glutSolidTorus(0.2, 0.7, 20, 20);
    glShadeModel(GL_FLAT);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, blue);
    //以下, 略  ・・・・・
```

幾何変換設定

影響

影響

ライト設定

影響

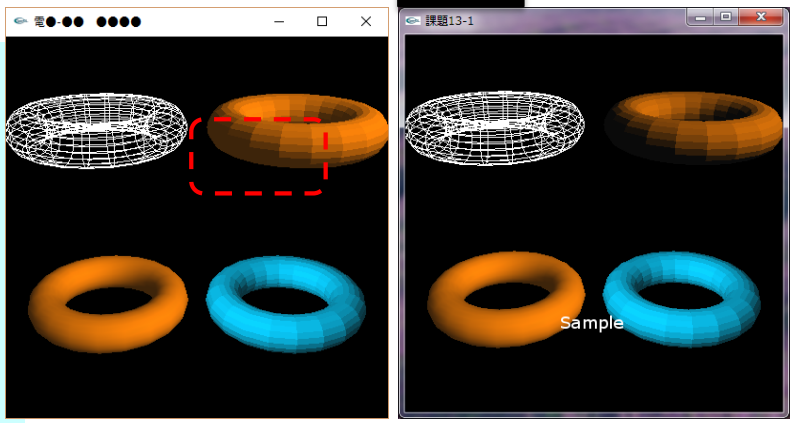
基本課題13解答例

Example14-1

```
void display( void )
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);


    // 左上
    glDisable(GL_LIGHTING);      //シェーディングをオフ
    glPushMatrix();
    glTranslatef(-1, 1, 0);
    glRotatef(90, 1, 0, 0);
    glutWireTorus(0.2, 0.7, 20, 20);
    glPopMatrix();

    // 右上
    glEnable(GL_LIGHTING);       //シェーディングをオン
    glLightfv(GL_LIGHT0, GL_DIFFUSE, amber);
    glLightfv(GL_LIGHT0, GL_AMBIENT, black);
    glLightfv(GL_LIGHT0, GL_POSITION, positionR);
    glShadeModel(GL_FLAT);
    glPushMatrix();
    glTranslatef(1, 1, 0);
    glRotatef(90, 1, 0, 0);
    glutSolidTorus(0.2, 0.7, 20, 20);
    glPopMatrix();
```

環境光
黒(オフ)

正解



環境光
アンバー

```
    // 左下
    glShadeModel(GL_SMOOTH);
    glLightfv(GL_LIGHT0, GL_AMBIENT, amber);
    glPushMatrix();
    glTranslatef(-1, -1, 0);
    glRotatef(90, 1, 0, 0);
    glutSolidTorus(0.2, 0.7, 20, 20);
    glPopMatrix();

    // 右下
    glLightfv(GL_LIGHT0, GL_DIFFUSE, blue);
    glLightfv(GL_LIGHT0, GL_AMBIENT, blue);
    glLightfv(GL_LIGHT0, GL_POSITION, positionL);
    glShadeModel(GL_FLAT);
    glPushMatrix();
    glTranslatef(1, -1, 0);
    glRotatef(90, 1, 0, 0);
    glutSolidTorus(0.2, 0.7, 20, 20);
    glPopMatrix();

    glFlush();
}
```

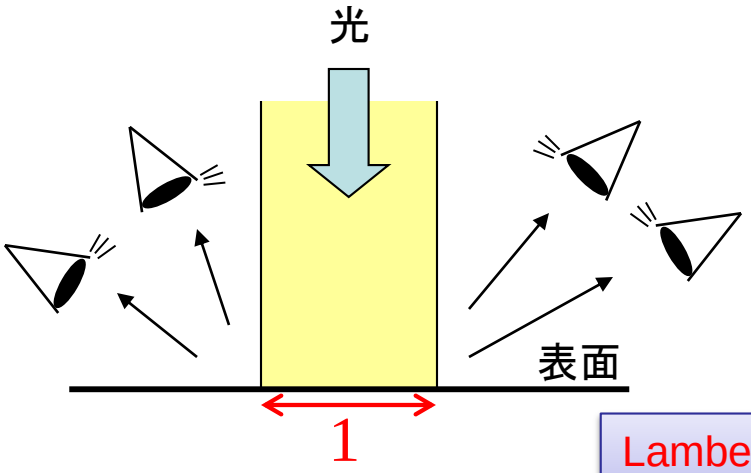
環境光
ブルー

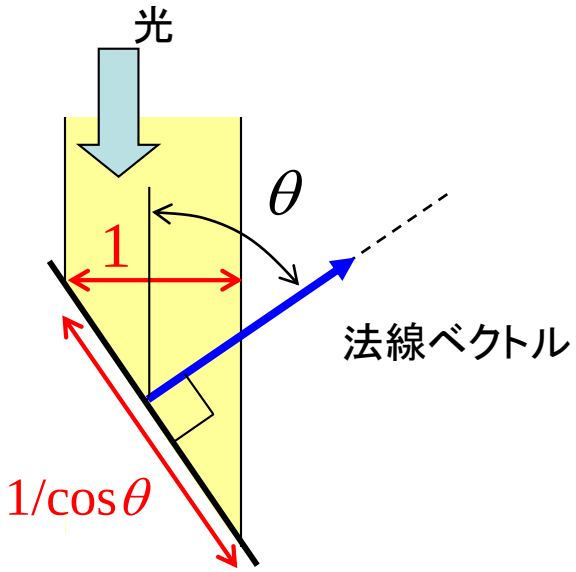
シェーディング

復習

シェーディング　＝　陰影付け
　物体を照明している光に応じてポリゴン面の明るさを変えること

拡散反射モデル
- 見る方向によって面の明るさは変わらない.
- 面に光が照射される角度によって面の明るさは変わる.





Lambertの余弦則
(Lambert反射)

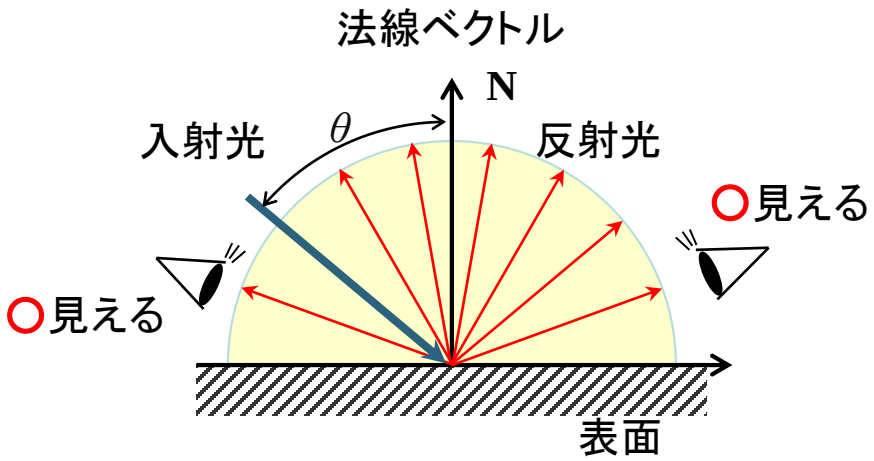
面の明るさ　$I_d = I_0 \cos\theta$

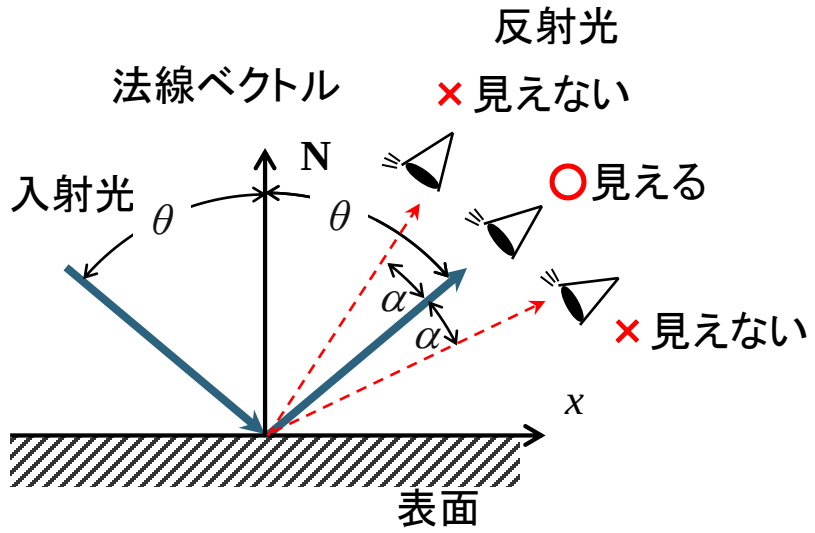
I_0　:元の光の強さ

θ　:光線が面の法線と為す角度

レンダリング
＝　シェーディング等を行い，その物体を実際に見た場合に近い画像にすること

拡散反射と鏡面反射





拡散反射(diffuse reflection)
- どの方向からも一様に見える
- 反射光の強さは次の要素で決まる
  - ✓ 入射光強度　I_{in}
  - ✓ 表面の反射率　R_d
  - ✓ 入射光の傾き　θ

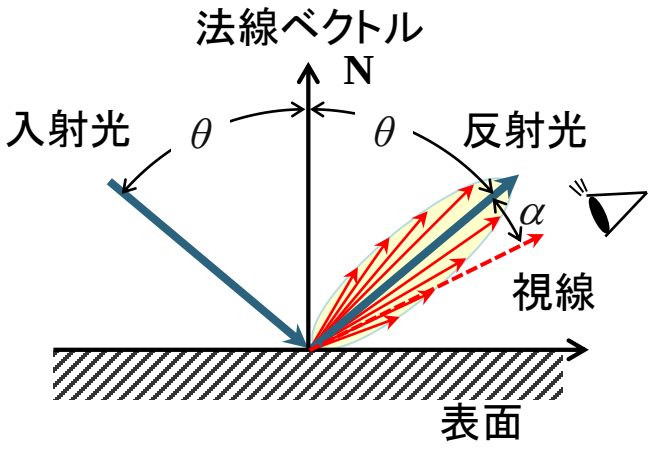
完全な鏡面反射(specular reflection)
- 正反射の方向からしか見えない
- 反射光の強さは次の要素で決まる
  - ✓ 入射光強度　I_{in}
  - ✓ 表面の反射率　R_s
  - ✓ 視線と正反射の角度差 α

拡散光強度 $I_d = I_{in} R_d \cos\theta$

反射光強度 $I_s = \begin{cases} I_{in} R_s & \cdots \ \alpha = 0 \\ 0 & \cdots \ \alpha \neq 0 \end{cases}$

完全な鏡面反射の問題点 → 現実の金属表面等の光沢面では正反射方向以外でも見える

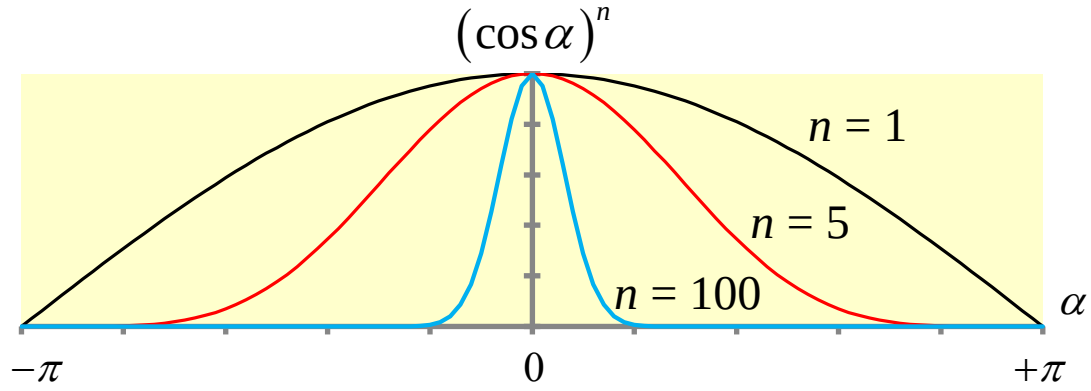
Phongの反射モデル



Phongの鏡面反射(specular reflection)

- 正反射の方向以外も少し見える
- 反射光の強さは次の要素で決まる
  - ✓ 入射光強度　I_{in}
  - ✓ 表面の反射率　R_s
  - ✓ 視線と正反射の角度差 α
  - ✓ 光沢度　n

反射光強度　$I_s = I_{in} R_s (\cos\alpha)^n$

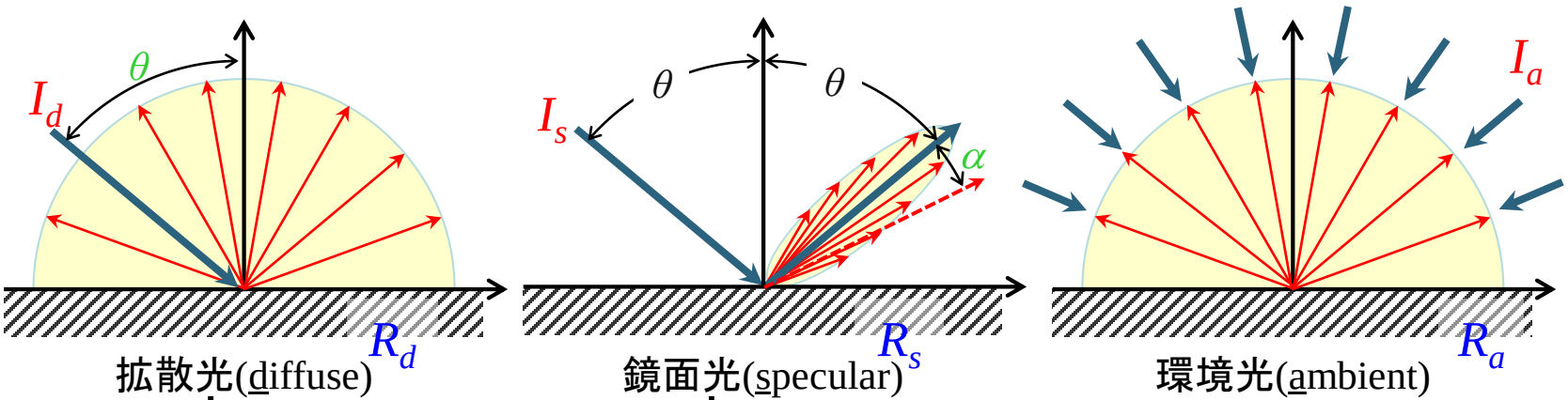


光沢度nによって鏡面反射の程度を表す

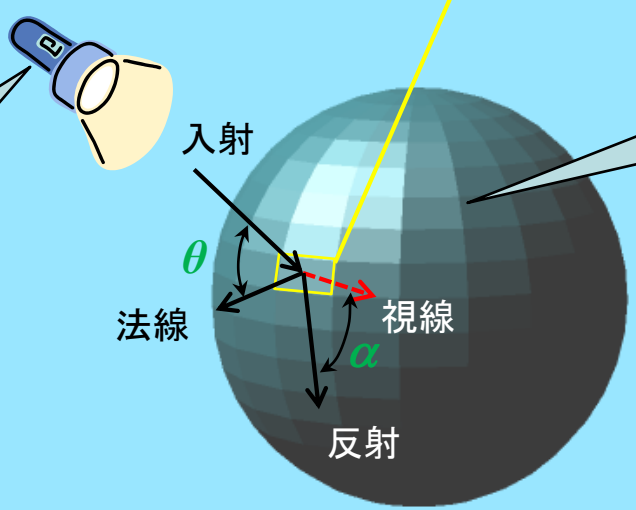
光沢度nが大 → 見える範囲が狭くなる→鏡面反射の性質が強い
(正反射方向周辺のわずかな範囲で見える)

光沢度nが小 → 見える範囲が広くなる→鏡面反射の性質が弱い

OpenGLのシェーディングモデル



明るさ $I = I_d R_d \cos\theta + I_s R_s (\cos\alpha)^n + I_a R_a$



光源の設定

I_d 拡散光の強さ(色)
I_s 鏡面光の強さ(色)
I_a 環境光の強さ(色)

その他の設定
位置,
種類 (平行光源・点光源・スポット光源)

材質の設定

R_d 拡散反射率 (色)
R_s 鏡面反射率 (色)
R_a 環境反射率 (色)
n 光沢度

その他の設定
テクスチャ, 透明度
etc...

光源の設定

```
glLightfv(light, pname, params)
```

明るさ $I = I_d R_d \cos\theta + I_s R_s (\cos\alpha)^n + I_a R_a$

light　設定するライトの番号の指定
GL_LIGHT0 ～ GL_LIGTH7 のいずれか
OpenGLでは8個の光源が設定できる

pname　設定する内容を示す
GL_POSITION　光源位置をparamsで示す (注: 幾何変換行列で変換される)
GL_DUFFUSE　拡散光の色成分(強さI_d)をparamsで示す
GL_SPECULAR　鏡面光の色成分(強さI_s)をparamsで示す
GL_AMBIENT　環境光の色成分(強さI_a)をparamsで示す

params　lightとpnameで示された各種パラメータの値
要素数4個のfloat型配列

色成分と光の強さ(明るさ)

R　G　B
例) アンバー色　(0.0, 0.8, 1.0)

色　強さ
(0.0, 0.8, 1.0) × 0.5 → (0.0, 0.4, 0.5) やや暗いアンバー
(0.0, 0.8, 1.0) × 0.1 → (0.0, 0.08, 0.1) かなり暗いアンバー

等倍すると色成分が変わらずに光の強さだけが変わる

光源の設定例

Example14-2 (主要部のみ)

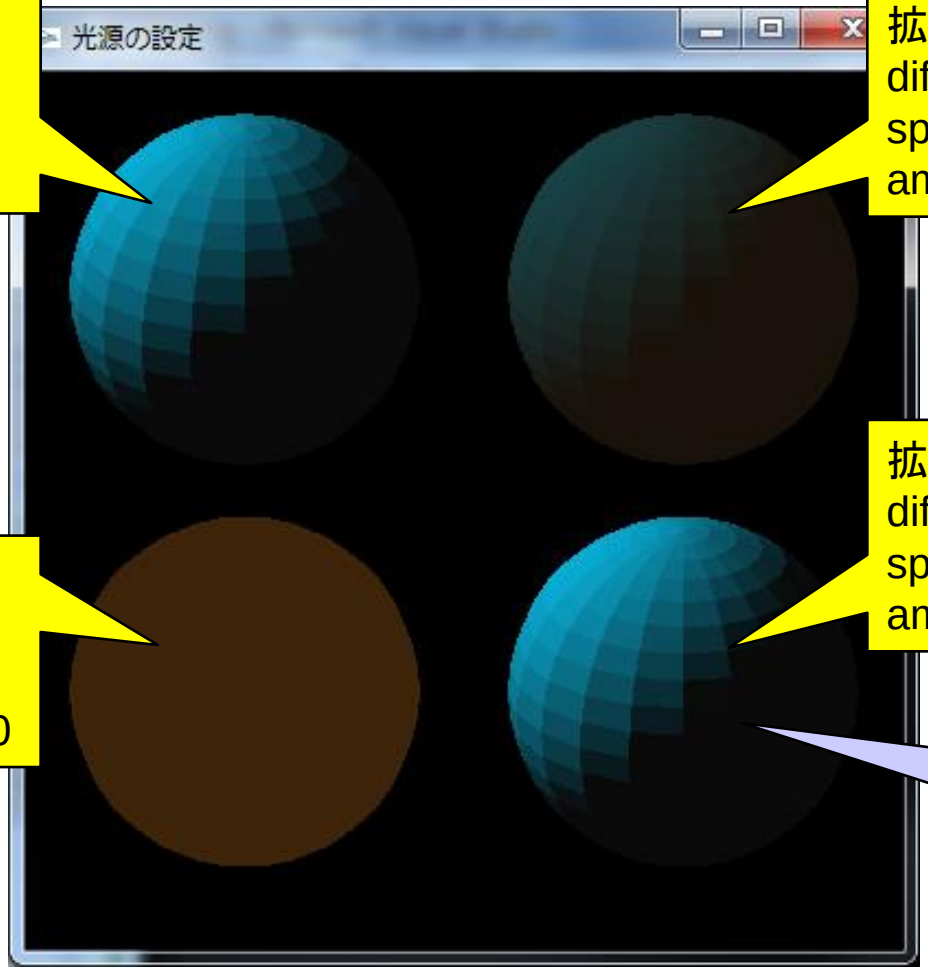
```
float blue[] = {0.0, 0.8, 1.0, 1.0};    //青
float amber[] = {1.0, 0.5, 0.0, 1.0};   //アンバー
float dir_pos[] = {-1.0, 1.0, 0.0, 0.0};    //平行光源位置

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(400, 400);
    glutInitDisplayMode(GLUT_RGBA);
    glutCreateWindow("光源の設定");
    glClearColor ( 0.0, 0.0, 0.0, 1.0 );
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-2.0, 2.0, -2.0, 2.0, 0.0, 10.0);
    //gluPerspective(35, 1.0, 1.0, 10.0);
    gluLookAt(0, 2.5, 0,   0, 1.2, -3,   0, 1, 0);

    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
    glEnable(GL_DEPTH_TEST);

    glutDisplayFunc(display);
    glutKeyboardFunc(KeyboardHandler);
    glutMainLoop();
}

void Sphere20(float x, float y, float z); //球を描く関数(略)

// パラメータの値を等倍する関数
float new_params[4];    //等倍された結果を入れる配列
float* Multiply(float params[], float val)
{
    int i;
    for(i = 0; i < 3; i++)
        new_params[i] = params[i] * val;
    return new_params;
}
```

```
void display( void )
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glEnable(GL_LIGHTING);      //シェーディングをオン
    glShadeModel(GL_FLAT);
    glLightfv(GL_LIGHT0, GL_POSITION, dir_pos); //平行光源

    // 左上
    glLightfv(GL_LIGHT0, GL_DIFFUSE, Multiply(blue, 1.0));
    glLightfv(GL_LIGHT0, GL_SPECULAR, Multiply(blue, 0.0));
    glLightfv(GL_LIGHT0, GL_AMBIENT, Multiply(blue, 0.0));
    Sphere20(-1, 1, 0);

    // 右上
    glLightfv(GL_LIGHT0, GL_DIFFUSE, Multiply(blue, 0.3));
    glLightfv(GL_LIGHT0, GL_SPECULAR, Multiply(blue, 0.0));
    glLightfv(GL_LIGHT0, GL_AMBIENT, Multiply(amber, 0.3));
    Sphere20(1, 1, 0);

    // 左下
    glLightfv(GL_LIGHT0, GL_DIFFUSE, Multiply(blue, 0.0));
    glLightfv(GL_LIGHT0, GL_SPECULAR, Multiply(blue, 0.0));
    glLightfv(GL_LIGHT0, GL_AMBIENT, Multiply(amber, 1.0));
    Sphere20(-1, -1, 0);

    // 右下
    glLightfv(GL_LIGHT0, GL_DIFFUSE, Multiply(blue, 1.0));
    glLightfv(GL_LIGHT0, GL_SPECULAR, Multiply(blue, 1.0));
    glLightfv(GL_LIGHT0, GL_AMBIENT, Multiply(blue, 0.0));
    Sphere20(1, -1, 0);

    glFlush();
}
```

色成分を変えずに明るさ(強さ)だけを変えるために用いる関数

Example14-2の実行結果

拡散光のみ
diffuse : blue ×1.0
specular: blue ×0.0
ambient: blue ×0.0



拡散光と環境光
diffuse : blue ×0.3
specular: blue ×0.0
ambient: amber ×0.3

環境光のみ
diffuse : blue ×0.0
specular: blue ×0.0
ambient: amber ×1.0

拡散光と鏡面光
diffuse : blue ×1.0
specular: blue ×1.0
ambient: blue ×0.0

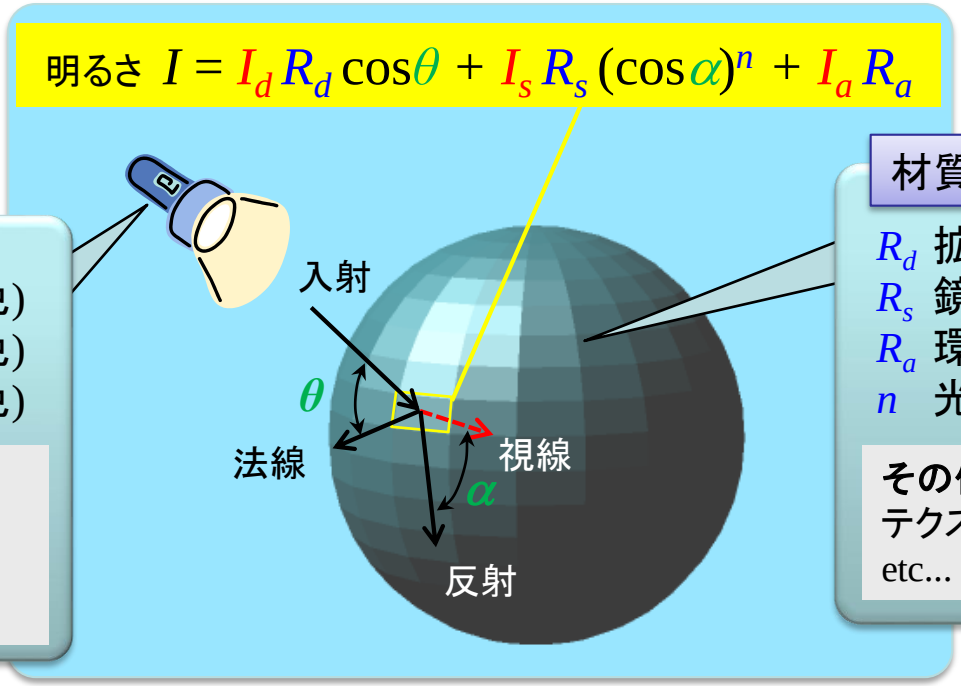
左上と同じ結果??

材質(表面属性)を設定していないため，鏡面反射の効果が出ていない

材質と表面属性

- レンダリングされる物体の材質が何であるかによってレンダリング結果が大きく変わる
  - 例) 金属，プラスチック，木材
- 材質表面の性質を「表面属性」と呼ぶ
- OpenGLで用いる表面属性
  - 拡散光の反射係数(R_d)と色成分
  - 鏡面光の反射係数(R_s)と色成分
  - 環境光の反射係数(R_a)と色成分
  - 光沢度(n) ← 鏡面の程度を表す
  - テクスチャマッピング，透明度，屈折率，・・・

明るさ $I = I_d R_d \cos\theta + I_s R_s (\cos\alpha)^n + I_a R_a$



光源の設定

I_d 拡散光の強さ(色)
I_s 鏡面光の強さ(色)
I_a 環境光の強さ(色)

その他の設定
位置，
種類 (平行光源・点光源・スポット光源)

材質の設定

R_d 拡散反射率 (色)
R_s 鏡面反射率 (色)
R_a 環境反射率 (色)
n 光沢度

その他の設定
テクスチャ，透明度
etc...

表面属性の設定

`glMaterialfv (face, pname, params)`

明るさ $I = I_d R_d \cos\theta + I_s R_s (\cos\alpha)^n + I_a R_a$

face 属性を設定する表面の指定
- GL_FRONT ポリゴンの表面にのみ属性を設定する
- GL_BACK ポリゴンの裏面にのみ属性を設定する
- GL_FRONT_AND_BACK ポリゴンの両面に属性を設定する

pname 設定する内容を示す
- GL_DUFFUSE 拡散反射の色成分(反射係数R_d)をparamsで示す
- GL_SPECULAR 鏡面反射の色成分(反射係数R_s)をparamsで示す
- GL_AMBIENT 環境反射の色成分(反射係数R_a)をparamsで示す
- GL_EMISSION 自発光の色成分(強さ)をparamsで示す

params pnameで示された各種パラメータの値
要素数4個のfloat型配列

光源に関係なく,
自ら発光する成分

`glMaterialf (face, pname, param)`

face 同上
pname 設定する内容を示す(以下のみ設定可能)
- GL_SHININESS 光沢度(n)を設定

param 光沢度の値(n)
float型の数値(配列ではない)

光沢度の設定にはこの関数を用いる
関数名の末尾にvが無いことに注意
(使用例)
`glMaterialf(GL_FRONT, GL_SCHININESS, n)`

これらの関数を呼び出した後に
`glBegin(...)` ～ `glVertex3f(...)` ～ `glEnd()`
等を用いて作成したポリゴンにその材質が設定される

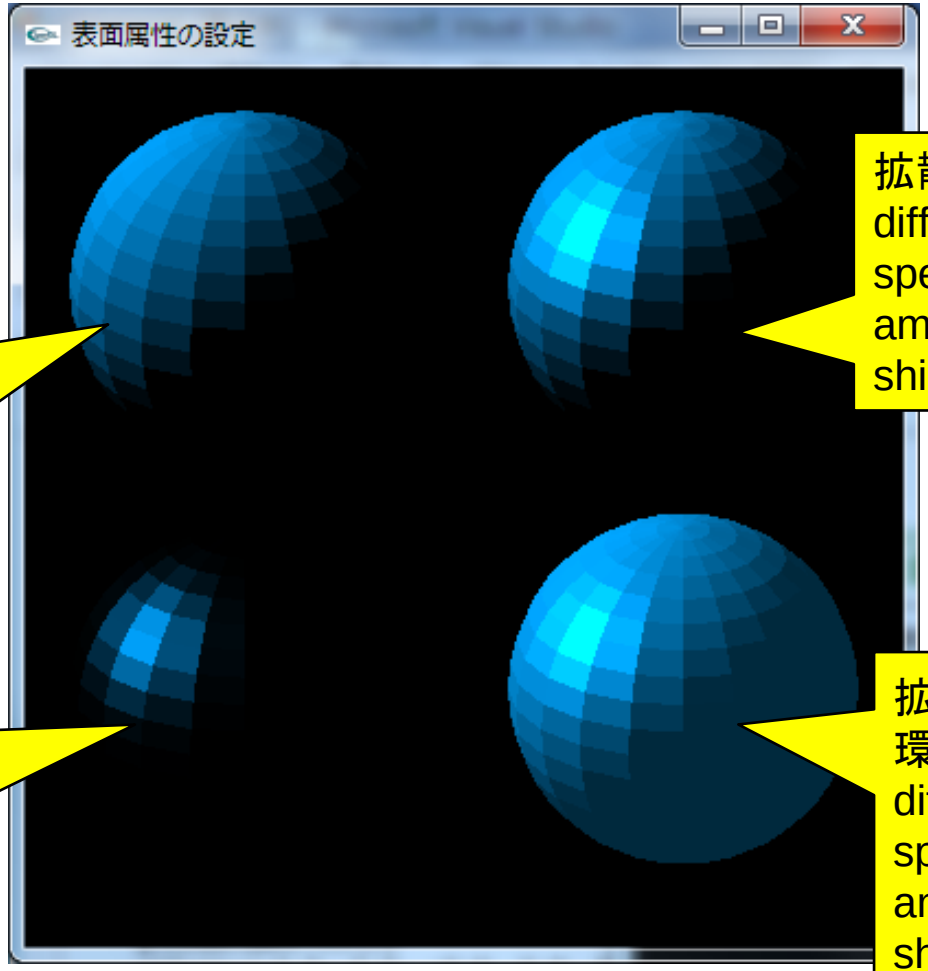
表面属性の設定例

Example14-3 (主要部のみ)

```
float blue[] = {0.0, 0.8, 1.0, 1.0};        //青
float dir_pos[] = {-1.0, 1.0, 0.0, 0.0};    //平行光源位置

// パラメータの値を等倍する関数
float new_params[4];    //等倍された結果を入れる配列
float* Multiply(float params[], float val)
{
    int i;
    for(i = 0; i < 3; i++)
    {
        new_params[i] = params[i] * val;
    }
    return new_params;
}

void display( void )
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glEnable(GL_LIGHTING);      //シェーディングをオン
    glShadeModel(GL_FLAT);

    // 光源の設定
    glLightfv(GL_LIGHT0, GL_POSITION, dir_pos); //平行光源
    glLightfv(GL_LIGHT0, GL_DIFFUSE, Multiply(blue, 1.0));
    glLightfv(GL_LIGHT0, GL_SPECULAR, Multiply(blue, 1.0));
    glLightfv(GL_LIGHT0, GL_AMBIENT, Multiply(blue, 1.0));

    // 左上
    glMaterialfv(GL_FRONT, GL_DIFFUSE, Multiply(blue, 1.0));
    glMaterialfv(GL_FRONT, GL_SPECULAR, Multiply(blue, 0.0));
    glMaterialfv(GL_FRONT, GL_AMBIENT, Multiply(blue, 0.0));
    glMaterialf(GL_FRONT, GL_SHININESS, 0.0);
    Sphere20(-1, 1, 0);

    // 右上
    glMaterialfv(GL_FRONT, GL_DIFFUSE, Multiply(blue, 1.0));
    glMaterialfv(GL_FRONT, GL_SPECULAR, Multiply(blue, 1.0));
    glMaterialfv(GL_FRONT, GL_AMBIENT, Multiply(blue, 0.0));
    glMaterialf(GL_FRONT, GL_SHININESS, 20.0);
    Sphere20(1, 1, 0);

    // 左下
    glMaterialfv(GL_FRONT, GL_DIFFUSE, Multiply(blue, 0.0));
    glMaterialfv(GL_FRONT, GL_SPECULAR, Multiply(blue, 1.0));
    glMaterialfv(GL_FRONT, GL_AMBIENT, Multiply(blue, 0.0));
    glMaterialf(GL_FRONT, GL_SHININESS, 20.0);
    Sphere20(-1, -1, 0);

    // 右下
    glMaterialfv(GL_FRONT, GL_DIFFUSE, Multiply(blue, 0.8));
    glMaterialfv(GL_FRONT, GL_SPECULAR, Multiply(blue, 0.8));
    glMaterialfv(GL_FRONT, GL_AMBIENT, Multiply(blue, 0.2));
    glMaterialf(GL_FRONT, GL_SHININESS, 20.0);
    Sphere20(1, -1, 0);

    glFlush();
}
```

Example14-3の実行結果

光源の設定
diffuse : blue ×1.0
specular: blue ×1.0
ambient: blue ×1.0



拡散反射のみ
diffuse : blue ×1.0
specular: blue ×0.0
ambient: blue ×0.0
shininess:0.0

拡散反射と鏡面反射
diffuse : blue ×1.0
specular: blue ×1.0
ambient: blue ×0.0
shininess:20.0

鏡面反射のみ
diffuse : blue ×0.0
specular: blue ×1.0
ambient: blue ×0.0
shininess:20.0

拡散反射+鏡面反射+環境光
diffuse : blue ×0.8
specular: blue ×0.8
ambient: blue ×0.2
shininess:20.0

複数光源と表面属性の設定例

Example14-4 (主要部のみ)

```
float blue[] = {0.0, 0.8, 1.0, 1.0};     //青
float amber[] = {1.0, 0.5, 0.0, 1.0};    //アンバー
float white[] = {1.0, 1.0, 1.0, 1.0};    //白
float dir_posL[] = {-1.0, 1.0, 0.0, 0.0};    //平行光源左位置
float dir_posR[] = {+1.0, 1.0, 0.0, 0.0};    //平行光源右位置

// 光源の設定を簡単化する関数
void Lighting(GLenum light, float color[], float diffuse,
                                        float specular, float ambient)
{
    glLightfv(light, GL_DIFFUSE, Multiply(color, diffuse));
    glLightfv(light, GL_SPECULAR, Multiply(color, specular));
    glLightfv(light, GL_AMBIENT, Multiply(color, ambient));
}

// 表面属性の設定を簡単化する関数
void Material(float color[], float diffuse, float specular,
                                       float ambient, float shininess)
{
    glMaterialfv(GL_FRONT, GL_DIFFUSE, Multiply(color, diffuse));
    glMaterialfv(GL_FRONT, GL_SPECULAR, Multiply(color, specular));
    glMaterialfv(GL_FRONT, GL_AMBIENT, Multiply(color, ambient));
    glMaterialf(GL_FRONT, GL_SHININESS, shininess);
}
```

color: 光源の色
diffuse : 拡散光の強さ
specular: 鏡面光の強さ
ambient: 環境光の強さ

color: 表面の色
diffuse : 拡散反射率
specular: 鏡面反射率
ambient: 環境反射率
shininess: 0.0

複数光源と表面属性の設定例

Example14-4 (続き)

```
void display( void )
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -6.0);
    glEnable(GL_LIGHTING);
    glShadeModel(GL_FLAT);
    //glShadeModel(GL_SMOOTH);

    glLightfv(GL_LIGHT0, GL_POSITION, dir_posL);     //平行光源左位置
    glLightfv(GL_LIGHT1, GL_POSITION, dir_posR);     //平行光源右位置
    Lighting(GL_LIGHT0, blue, 1.0, 1.0, 1.0);        //青いライト
    Lighting(GL_LIGHT1, amber, 1.0, 1.0, 1.0);       //アンバーのライト
    glDisable(GL_LIGHT0);
    glDisable(GL_LIGHT1);

    // 左上
    glEnable(GL_LIGHT0);
    Material(white, 1.0, 0.0, 0.1, 0.0);
    Sphere20(-1, 1, 0);
    glDisable(GL_LIGHT0);

    // 右上
    glEnable(GL_LIGHT1);
    Material(white, 1.0, 0.0, 0.1, 0.0);
    Sphere20(1, 1, 0);
    glDisable(GL_LIGHT1);

    // 左下
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHT1);
    Material(amber, 1.0, 0.0, 0.1, 0.0);
    Sphere20(-1, -1, 0);
    glDisable(GL_LIGHT0);
    glDisable(GL_LIGHT1);

    // 右下
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHT1);
    Material(blue, 0.5, 0.5, 0.1, 20.0);
    Sphere20(1, -1, 0);
    glDisable(GL_LIGHT0);
    glDisable(GL_LIGHT1);

    glFlush();
}
```

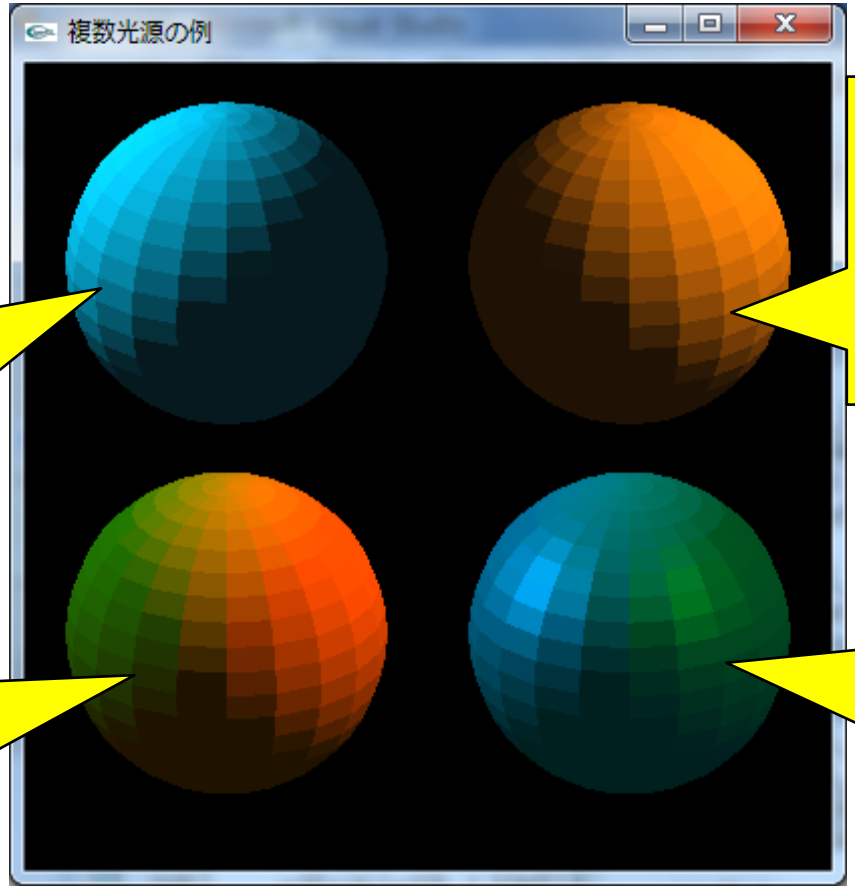
Example14-4の実行結果

青

LIGHT0の設定
位置:　　左上
diffuse :　blue ×1.0
specular: blue ×1.0
ambient:　blue ×1.0

アンバー

LIGHT1の設定
位置:　　右上
diffuse :　amber ×1.0
specular: amber ×1.0
ambient:　amber ×1.0



光源: LIGHT0のみ
表面属性の設定
diffuse :　white ×1.0
specular: white ×0.0
ambient:　white ×0.1
shininess:0.0

光源: LIGHT1のみ
表面属性の設定
diffuse :　white ×1.0
specular: white ×0.0
ambient:　white ×0.1
shininess:0.0

光源: LIGHT0+LIGHT1
表面属性の設定
diffuse :　amber ×1.0
specular: amber ×0.0
ambient:　amber ×0.1
shininess:0.0

光源: LIGHT0+LIGHT1
表面属性の設定
diffuse :　blue ×0.5
specular: blue ×0.5
ambient:　blue ×0.1
shininess:20.0

修了作品

ウィンドウの名前を各自の学籍番号と氏名にすること

OpenGLを用いて自由にコンピュータグラフィックス映像を制作しなさい

- 静止画でもアニメーションでもOK
- ワイヤーフレームモデルでもサーフェースモデルでもOK
- ゲームでもOK

入選作品

- 入選作品をLMSを通じて発表します．また，入選作品は次年度の授業で紹介します．
- 入選作品発表時に氏名等を公表します(公表して欲しくない場合は説明書にそれを記入)
- 入選作品をLMS等で配布する場合があります(配布して欲しくない場合は説明書にそれを記入)

提出物と提出方法

種類	提出ファイル	提出物(1)	提出物(2)
静止画	WordまたはPDF	**修了作品説明書** **(用紙ファイルと記入見本あり)** ➢ 学籍番号・氏名 ➢ 作品のタイトル・解説・使用方法 ➢ 作品の実行結果（画面コピー） ➢ 制作にあたって参考にしたサイトや書籍 ➢ **ソースプログラム**	なし
アニメーション ゲーム	ZIP		**実行プログラム** (必ずリリースモードでビルド)

作品説明書自体も採点対象です！

リリースモードでない場合は0点となります！

締め切り

- 8月10日(土曜)深夜24時

評価

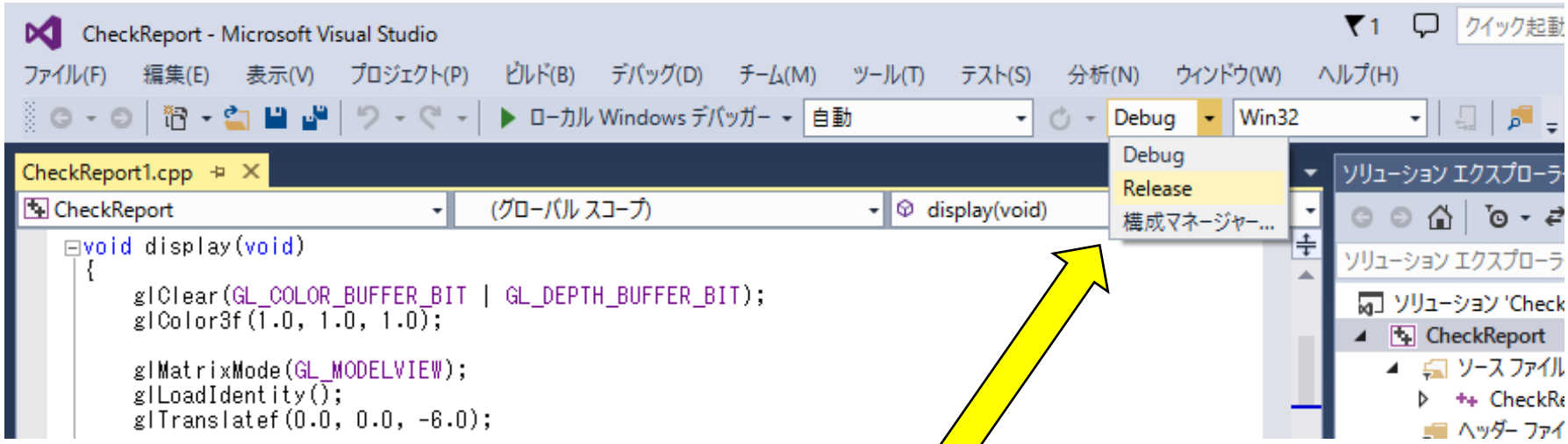
- 本授業の評価点100点のうち20点をこの課題で評価します．
- **独創的な作品**や**技術的に凝った作品**，**完成度が高い作品**を高く評価します．
- 美しいものや楽しいものも高く評価します．

してはいけないこと

- 先輩の作品や，書籍やネットで見つけたソースをそのまま使って提出
  ⇒ 試験の得点や出席・レポートの評価に関わらず不合格とします
- 同じ作品や似たような作品が提出された場合　⇒ その作品数で割り算して評価点にします．

他人が作らないものを作ってください！

Visual Studioの ReleaseモードとDebugモード



Debug
デバッグ時に用いる．デバッグ用のデータが実行ファイルに入るので，速度が遅くなりサイズも大きくなる．
実行マシンによっては実行不能になる場合もある．

Release
デバッグが終わり完成したプログラムを実行したり，実行プログラムを人に渡す際に用いる．スピードが速くなり実行プログラムのサイズも小さくなる．

Debugから Releaseに切り替えたら，再コンパイルが必要！

こちらで提出！