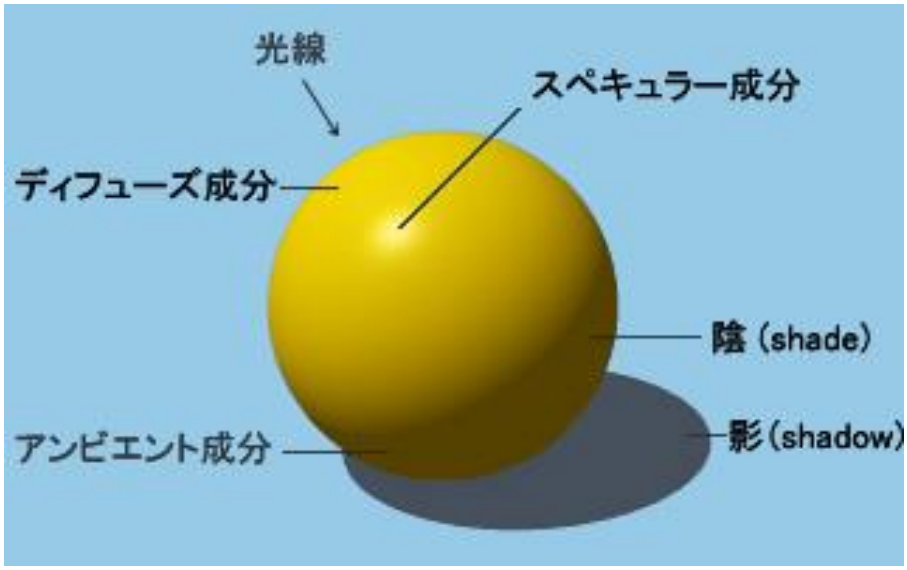
後期課題2

資料「プリミティブモデル」

■そもそも 3DCG の仕組みとは？

1)レンダリングをして画像を得るためには、物体の色やシェーディングと言われる属性を与える必要がある。



2)オブジェクトに、面の向きに応じた陰（shade）をつける事をシェーディングという。※shadow ではない。

・アンビエント（ambient）

周辺光の強さ。光源からの光が全く当たらない陰の部分はこの成分だけで明るさが決まる。
0%で真っ黒になる。立体の置かれた環境に合わせて調整する。

・ディフューズ（diffuse）

物体に当たった光の拡散反射を示します。粗さを大きくすると反射を抑えてツヤ消しとなります。
一般的にはランバート法と呼ばれる入射角の cos に比例する計算を行う（他にトゥーンなどがある）

・スペキュラー（specular）

表面の光沢の反射成分。ハイライトの強さとなり光る部分のサイズも変えられる。

・スペキュラーの強さ(pow)

スペキュラーハイライトの角度範囲を決定する値です。値が小さい程範囲が狭くなる。

これらの値により質感の表現ができる。
・プラスチック：スペキュラーを鈍く弱くする。
・ガラス：アンビエントとディフューズを弱くしスペキュラーを鋭くする。
・光沢のある金属：アンビエントとディフューズを弱くしスペキュラーを強く鋭くする。

3)他にもあります。

・エミッシブ（emissive）

自己発光色の設定で、ライトに影響されない光？となります。物体が光っている様に見えるイメージです。
通常では、ライトを使用しないと真っ暗な物体になりますが、自己発光の設定で色を付ける事ができます。

■プリミティブ(基本図形)「球型」

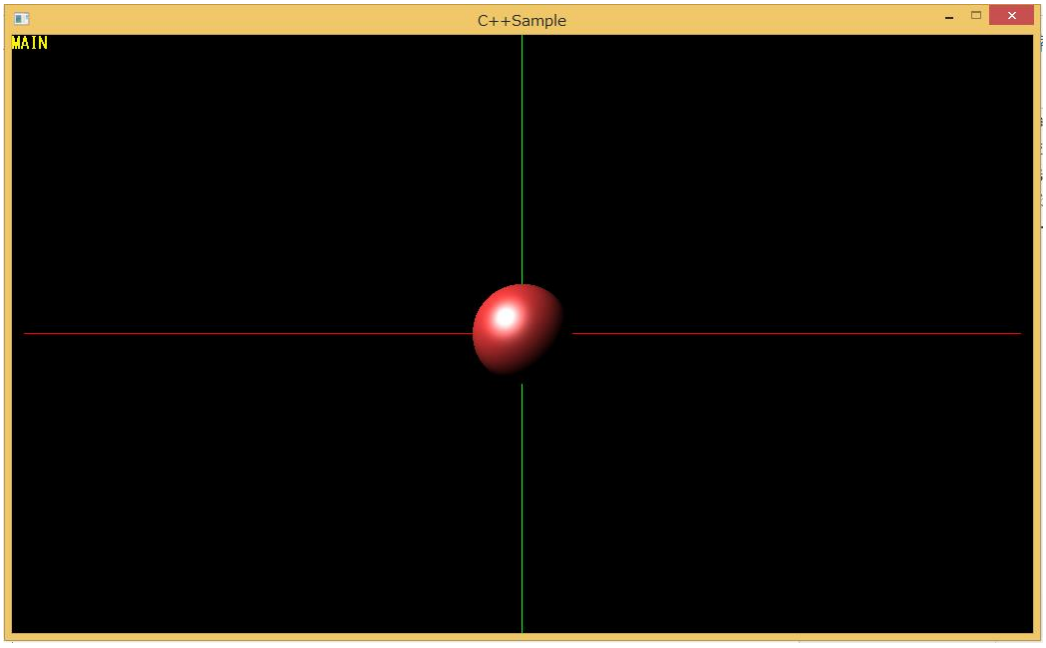
球の描画

int DrawSphere3D(中心座標, 球の半径, ポリゴンの細かさ, 色, 反射の色, 塗りつぶすか);

```
VECTOR pos;             // 中心座標(x, y, z)
float r;                // 半径
int divNum;             // 細かさ
unsigned int difColor;  // 物体色：ﾃﾞｨﾌｭｰｽﾞ(R, G, B)
unsigned int spcColor;  // 反射色：ｽﾍﾟｷｭﾗｰ(R, G, B)
int fillFlag;           // 塗りつぶし(0：線のみ、1:塗り有り)
```

プログラム例

```
pos = VGet(0.0f, 0.0f, 0.0f);
r = 50.0f;
divNum = 32;
difColor = GetColor(255, 64, 64);
spcColor = GetColor(255, 255, 255);
fillFlag = 1;

DrawSphere3D(pos, r, divNum, difColor, spcColor, fillFlag);
```



■プリミティブ(基本図形)「カプセル型」

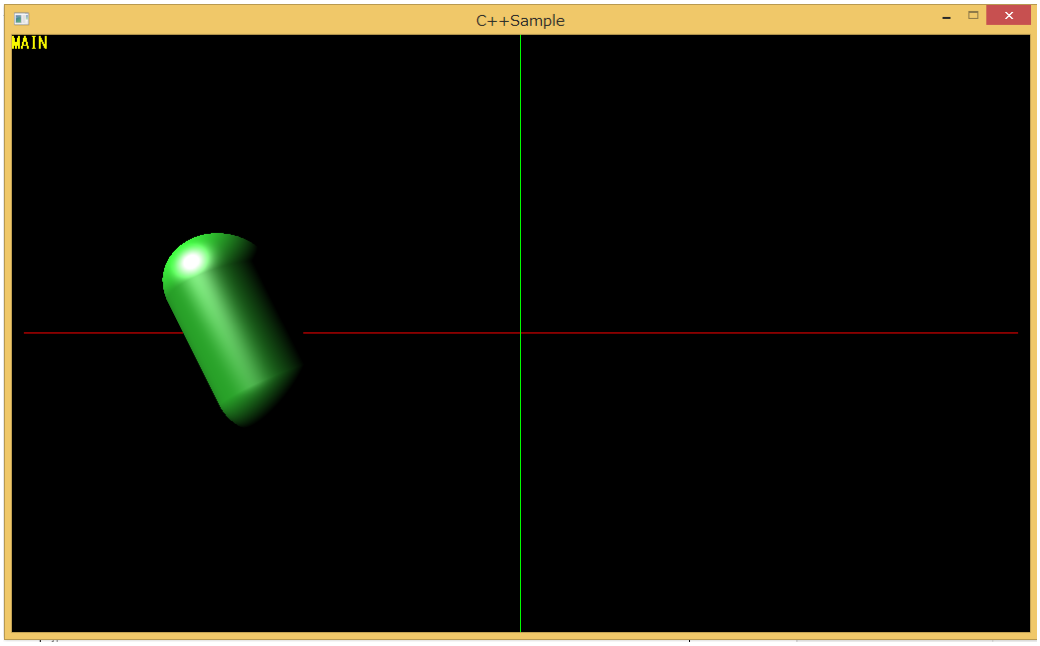
カプセルの描画

int DrawCapsule3D(1点目の座標, 2点目の座標, 幅, 細かさ, 色, 反射の色, 塗りつぶすか);

```
VECTOR pos1;              // 1点目の中心座標(x, y, z)
VECTOR pos2;              // 2点目の中心座標(x, y, z)
float r;                  // ｶﾌﾟｾﾙの幅(半径)、先端の半球の半径
int divNum;               // 細かさ
unsigned int difColor;    // 物体色：ﾃﾞｨﾌｭｰｽﾞ(R, G, B)
unsigned int spcColor;    // 反射色：ｽﾍﾟｷｭﾗｰ(R, G, B)
int fillFlag;             // 塗りつぶし(0：線のみ、1:塗り有り)
```

プログラム例

```
pos1 = VGet(-300.0f, 50.0f, 0.0f);
pos2 = VGet(-250.0f, -50.0f, 0.0f);
r = 50.0f;
divNum = 32;
difColor = GetColor(64, 255, 64);
spcColor = GetColor(255, 255, 255);
fillFlag = 1;

DrawCapsule3D(pos1, pos2, r, divNum, difColor, spcColor, fillFlag);
```



■プリミティブ(基本図形)「円錐型」

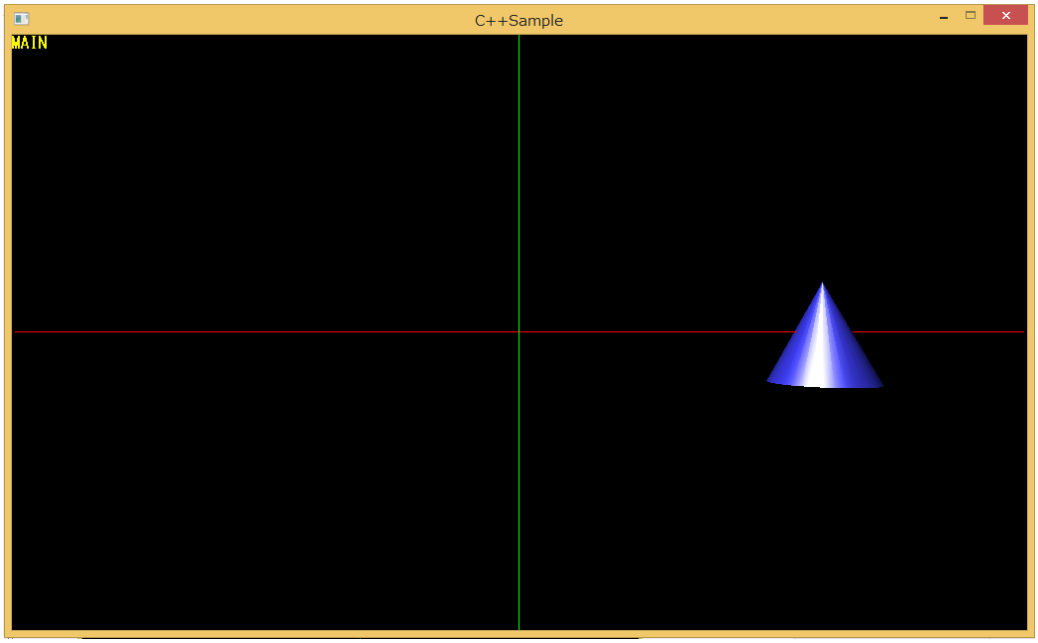
円錐の描画

int DrawCone3D(頂点座標, 底面の中心座標, 底面の半径, 細かさ, 色, 反射の色, 塗りつぶすか);

```
VECTOR top;             // 頂点座標(x, y, z)
VECTOR bottom;          // 底面の中心座標(x, y, z)
float r;                // 底面の半径
int divNum;             // 細かさ
unsigned int difColor;  // 物体色：ﾃﾞｨﾌｭｰｽﾞ(R, G, B)
unsigned int spcColor;  // 反射色：ｽﾍﾟｷｭﾗｰ(R, G, B)
int fillFlag;           // 塗りつぶし(0：線のみ、1:塗り有り)
```

プログラム例

```
top = VGet(300.0f, 50.0f, 0.0f);
bottom = VGet(300.0f, -50.0f, 0.0f);
r = 50.0f;
divNum = 32;
difColor = GetColor(64, 64, 255);
spcColor = GetColor(255, 255, 255);
fillFlag = 1;

DrawCone3D(top, bottom, r, divNum, difColor, spcColor, fillFlag);
```



□マテリアルパラメータ全部設定する方法

MATERIALPARAM 構造体は、各パラメータのメンバー変数がセットになっており、その構造体を使用して「SetMaterialPara()関数」を使用して一気に設定する事ができます。

1)マテリアル関係をひとまとめにした構造体

```
// ―― マテリアル構造体での変数定義
MATERIALPARAM 変数名;
```

2)3D で使用する COLOR_F 型で色指定する場合に使用する。

```
// ―― 浮動小数点型の値を取得する
GetColorF(float red, float green, float blue, float alpha);

float Red   : 赤成分の輝度( 0.0f ～ 1.0f )
float Green : 緑成分の輝度( 0.0f ～ 1.0f )
float Blue  : 青成分の輝度( 0.0f ～ 1.0f )
float Alpha : アルファ成分( 0.0f ～ 1.0f )
```

※0～255 ではなく 0.0f～1.0f なので注意

3)マテリアル構造体でのメンバー変数の役割り。

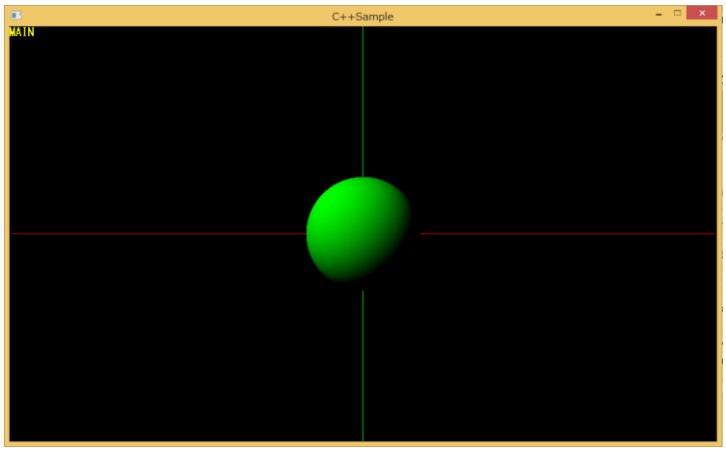
```
// マテリアルの自己発光色を暗い青色にする
GetColorF Diffuse;  // 拡散光色。初期状態では無視される。
                       ※SetMaterialUseVertDefColor()関数で使用するかを設定する。
GetColorF Ambient;  // 環境光色。ライトの色と掛け合わせて使用される。
GetColorF Emissive; // 自己発光色。ライトがなくても光る。
GetColorF Specular; // 反射光色。初期状態では無視される。
                       ※SetMaterialUseVertSpcColor()関数で使用するかを設定する。
float Power;        // スペキュラハイライトの角度範囲を決定する。
```

4)マテリアルパラメータを設定する。

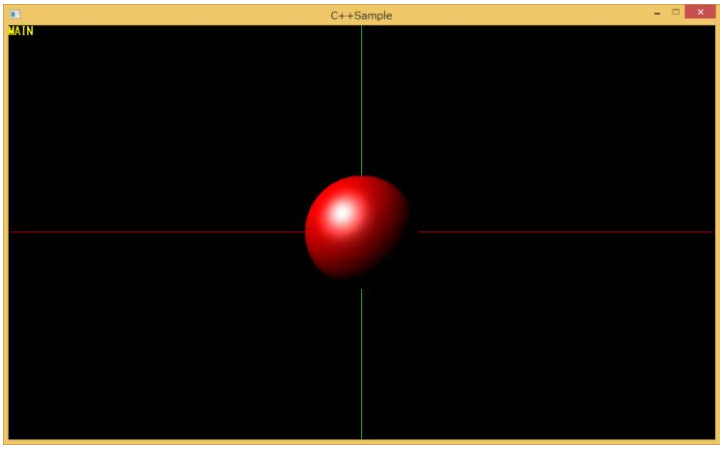
```
// ―― マテリアルパラメータによる設定
int SetMaterialParam(MATERIALPARAM Material);
```

5)設定の一例。

```
// ----- ﾏﾃﾘｱﾙの個別設定
MATERIALPARAM Material; // 変数定義

Material.Diffuse = GetColorF(0.0f, 1.0f, 0.0f, 1.0f);
Material.Specular = GetColorF(0.0f, 0.0f, 0.0f, 0.0f);
Material.Ambient = GetColorF(0.0f, 0.0f, 0.0f, 0.0f);
Material.Emissive = GetColorF(0.0f, 0.0f, 0.0f, 0.0f);
Material.Power = 20.0f;

SetMaterialUseVertDifColor(false);  // 頂点ﾃﾞｰﾀのﾃﾞｨﾌｭｰｽﾞでなくﾏﾃﾘｱﾙ設定の方を使用する
SetMaterialUseVertSpcColor(false);  // 頂点ﾃﾞｰﾀのｽﾍﾟｷｭﾗｰでなくﾏﾃﾘｱﾙ設定の方を使用する
SetMaterialParam(Material);         // ﾏﾃﾘｱﾙﾃﾞｰﾀのｾｯﾄ

// ----- 3D空間上に球を描画する
VECTOR pos = VGet(0.0f, 0.0f, 0.0f);                    // ﾓﾃﾞﾙﾃﾞｰﾀの中心座標
unsigned int difColor = GetColor(255, 0, 0);            // 頂点ﾃﾞｰﾀのﾃﾞｨﾌｭｰｽﾞをｾｯﾄ
unsigned int spcColor = GetColor(255, 255, 255);        // 頂点ﾃﾞｰﾀのｽﾍﾟｷｭﾗｰをｾｯﾄ

DrawSphere3D(pos, 80.0f, 32, difColor, spcColor, true); // ﾓﾃﾞﾙﾃﾞｰﾀの描画
```



頂点データの色設定を使用した場合　　SetMaterialUseVertDifColor(False);



マテリアルの色設定を使用した場合　　SetMaterialUseVertDifColor(true);

■プリミティブ(基本図形)「三角形の板」

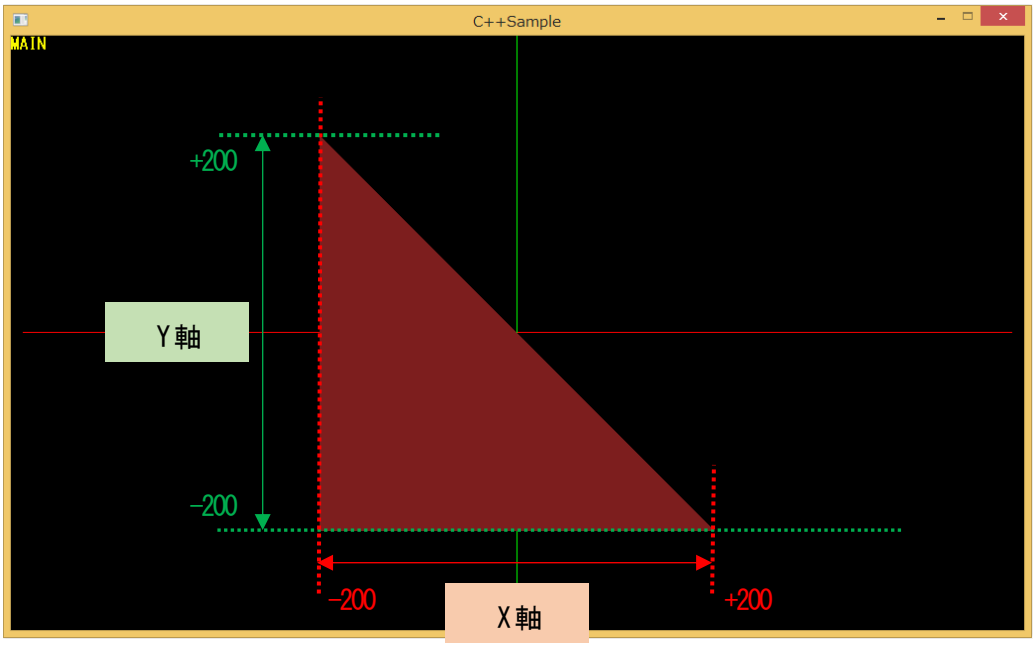
三角形の描画

int DrawTriangle3D(頂点１の座標, 頂点２の座標, 頂点３の座標, 色, 塗りつぶすか);

```
VECTOR pos1;         // 三角形の頂点1の座標(x, y, z)
VECTOR pos2;         // 三角形の頂点2の座標(x, y, z)
VECTOR pos3;         // 三角形の頂点3の座標(x, y, z)
unsigned int color; // 三角形の色(R, G, B)
int fillFlag;        // 塗り潰すかどうか(0:線のみ、1:塗る)
```

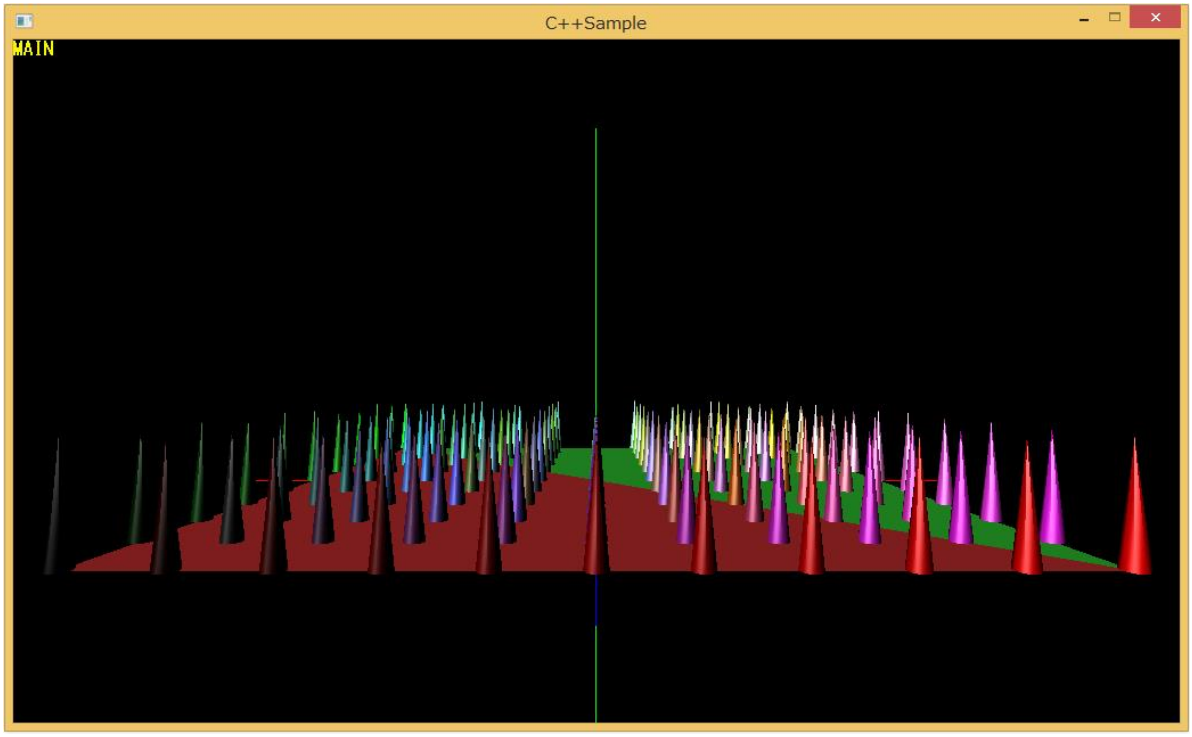
プログラム例

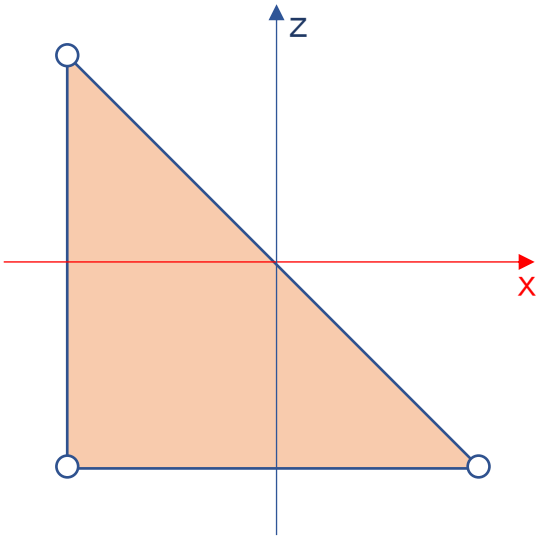
```
pos1 = VGet( 200.0f, -200.0f, 0.0f);
pos2 = VGet(-200.0f, -200.0f, 0.0f);
pos3 = VGet(-200.0f,  200.0f, 0.0f);
color = GetColor(125, 30, 30);
fillFlag = 1;
DrawTriangle3D(pos1, pos2, pos3, color, fillFlag);
```



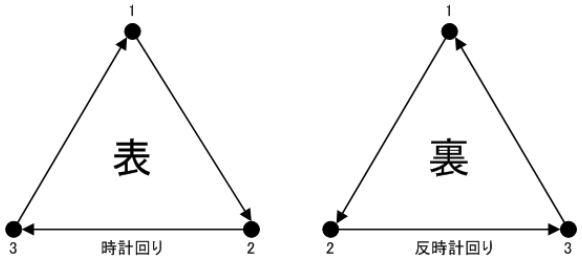
1)演習①次の様なフィールドを作成しなさい。

・フィールド・・・・三角形の板を 2 枚使用する。(高さ Y=0.0f、一辺 800.0f の正方形)
・オブジェクト・・・円錐を使用してある程度の数で設置する(色や形は自由)。





1 辺の長さ 800.0f の平面



三角形の座標は時計回りの順番で指定します！
※DirectX は左手座標系なので時計回りです。
OpenGL や Unity は右手座標系なので、逆に半時計回りになります。