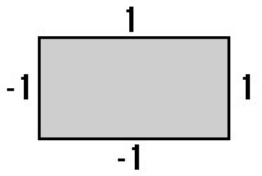
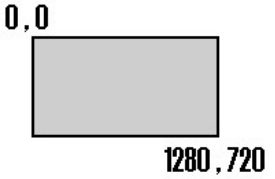
# 02 – 行列スタック

## 投影変換

画面上に表示されるオブジェクトは「　　」の範囲に表示される。しかし、２Ｄゲームや UI の表示は画面左上を原点とした、ウィンドウの大きさそのままの座標の方が考えやすい。そこで 2D の表示に関しては、配置されたオブジェクトを画面に表示(投影)する際、-1～1 に収まるよう縮小を行う。



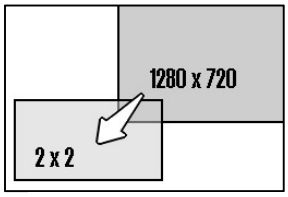
本来のウィンドウの座標



よくある 2D 座標

$$\text{投影座標}\left(-1\sim1\right)=\frac{2D座標}{画面サイズ}*2.0-1.0$$

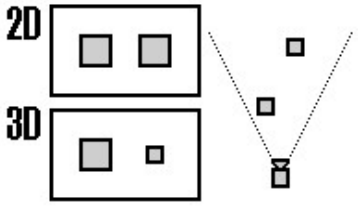
［例］座標(320,360)
= (320 / 1280, 360 / 720) * 2 – 1
= (0.25, 0.5) * 2 – 1
= (0.5, 1.0) – 1　= (-0.5, 0.0)



大きいサイズから小さいサイズへ変換

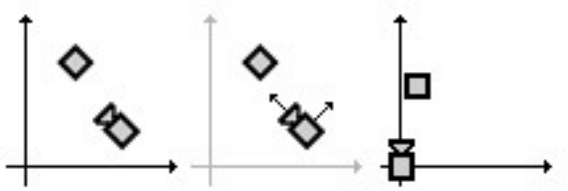
これは 3D も同様に考える必要がある。

ただし、同じやり方では正しく表示されない。例えば同じ大きさのオブジェクトを２つ用意し、それぞれ別の位置へ配置する。その後、2D のように画面サイズで割って小さくするだけでは、カメラからの距離が異なっていたとしても、同じ大きさのオブジェクトとして表示されてしまう。3D なら遠くのオブジェクトは小さく表示されないとおかしいため、位置によって小さくする割合を変えることが必要になる。




これらの変換を 2D なら「　　」、3D なら「　　」という。

なお、これらの投影は映し出すスクリーンが原点にある前提で行われる。２Ｄは特に問題ないが、３Ｄでは投影するスクリーンの位置が 3D 空間上のカメラと同じ位置になるため、正しく表示されない。そのため、投影変換を行う前に、カメラを原点の位置に移動させ、その時の移動と同じ移動をすべてのオブジェクトに適用させる必要がある。このときの変換を「　ビュー座標変換　」という。



```
gluOrtho(
    left   : ウィンドウ左端,
    right  : ウィンドウ右端,
    bottom : ウィンドウ下端,
    top    : ウィンドウ上端,
    zNear  : ニアクリップ,
    zFar   : ファークリップ
);
```

```
gluPerspective(
    fovy   : 縦のビュー角度(°),
    aspect : 画面縦横比,
    zNear  : ニアクリップ,
    zFar   : ファークリップ
);
```

```
gluLookAt(
    eye[X,Y,Z]    : カメラ位置
    center[X,Y,Z] : 注視点
    up[X,Y,Z]     : カメラ上方向
);
```

## 行列スタック

3D オブジェクトが移動したり方向転換したり徐々に小さくなったりする際、オブジェクトの頂点座標を変換する必要がある。これらの変換には**行列(マトリックス)**を用いる。DirectX では DirectXMath を利用して行列計算を行っていた。OpenGL では行列スタックという仕組みを採用しており、計算のために行列のデータを用意する必要がなく、グローバルな領域に変換用のデータを保存するも必要がない。