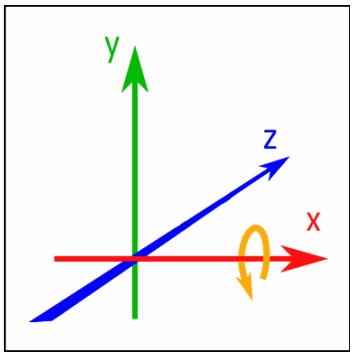
後期課題3

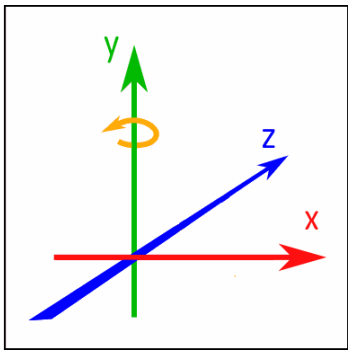
１日で理解する3Dプログラミング 「回転行列編」

■モデルデータの任意軸による回転

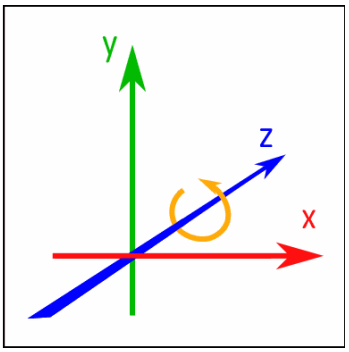
モデルの回転は「X 軸」「Y軸」「Z軸」のそれぞれの基本座標軸で回転させる事ができますが、斜め方向だったり様々な軸による回転が混在すると思うようにできません。それを回避する為に、基本座標軸であっても任意軸での回転を行う事によりすると複数軸での回転が可能になります。



X 軸での回転



Y 軸での回転



Z 軸での回転

任意軸での回転
MATRIX 型 MGetRotAxis(回転軸, 回転量);

回転軸 ： VECTOR　　// 回転させたい軸を VECTOR で表す
回転量 ： float　　// 回転値(ラジアン値)

単位行列(MATRIX)の取得
MATRIX 型 MGetIdent(void);

```
m[0][0]=1.0f m[0][1]=0.0f m[0][2]=0.0f m[0][3]=0.0f
m[1][0]=0.0f m[1][1]=1.0f m[1][2]=0.0f m[1][3]=0.0f
m[2][0]=0.0f m[2][1]=0.0f m[2][2]=1.0f m[2][3]=0.0f
m[3][0]=0.0f m[3][1]=0.0f m[3][2]=0.0f m[3][3]=1.0f
```

「回転なし」「移動なし」「拡大率 1.0」の基本行列になります。

```
２つの行列(MATRIX)の合成（掛け算）
MATRIX型 MMult（行列1, 行列2);
```

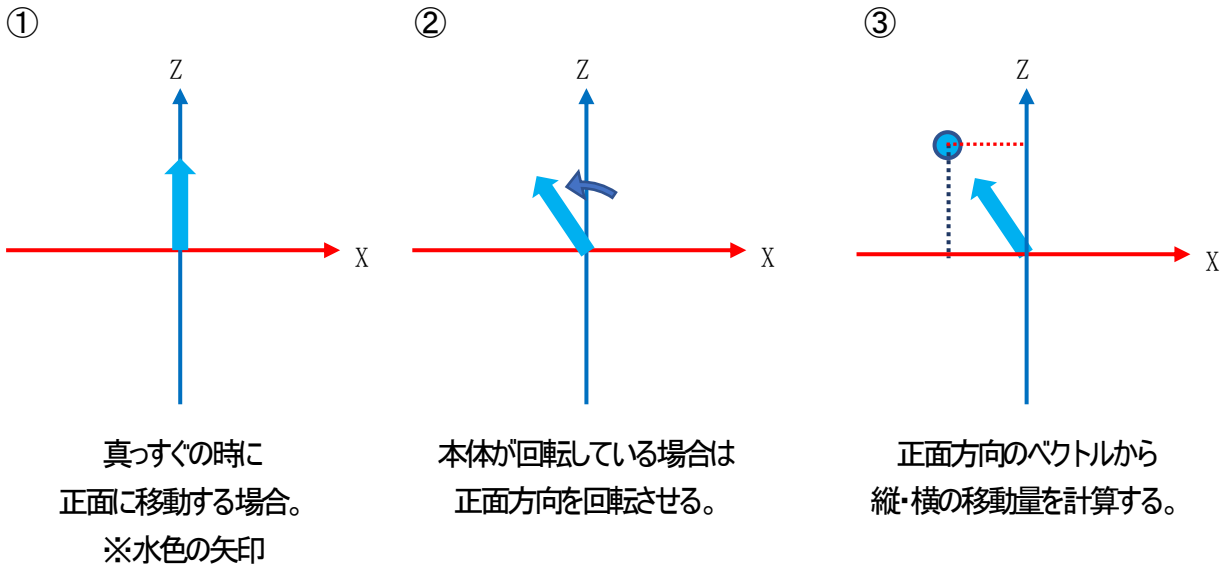
左辺の「行列１」→ 右辺の「行列２」の順番で効果が現れる。
例）MMult(拡大、移動) → 拡大した後に平行移動した状態になる。

■X 軸で回転させた時の回転行列を取得する。

```
MATRIX masterMat = MGetIdent();        // 基本行列を生成
MATRIX tmp = MGetRotAxis(VGet(1.0f, 0.0f, 0.0f), 回転量); // X軸を回転量で回転させる。
masterMat = MMult(masterMat, tmp);  // masterMatにtmpを合成する。
```

①まっさらな行列を作成しておく。
②X 軸ベクトルとなる(1, 0, 0)を軸に回転させた回転行列を取得する。
③まっさらな行列に合成して、最終的な X 軸に回転させた回転行列を生成する。

■モデルデータの回転方向による移動制御



真っすぐの時に
正面に移動する場合。
※水色の矢印

本体が回転している場合は
正面方向を回転させる。

正面方向のベクトルから
縦・横の移動量を計算する。

```
平行移動行列(MATRIX)の取得
MATRIX型 MGetTranslate( 平行移動値 );

平行移動値 : VECTOR    // 平行移動量を VECTOR で渡す。
```

■Y軸で回転した状態で正面(Z軸方向)に移動した場合

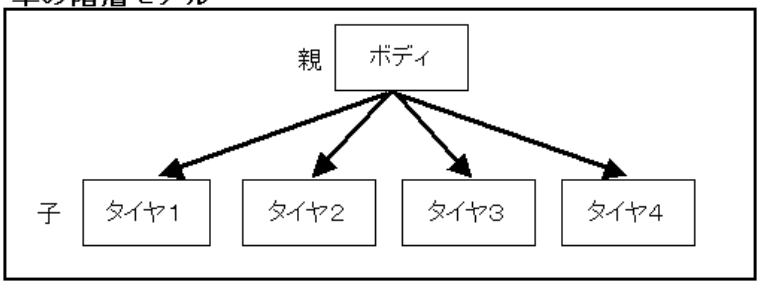
```
VECTOR pos;

VECTOR front = VTransform(VGet(0.0f, 0.0f, 1.0f), 回転Mat); // 回転後のZ軸方向
pos.x += front.x * 移動量; // 移動後のX座標
pos.z += front.z * 移動量; // 移動後のZ座標
masterMat = MMult(masterMat, MGetTranslate(pos));  // master に移動量追加
```

■親子関係（階層モデル）について

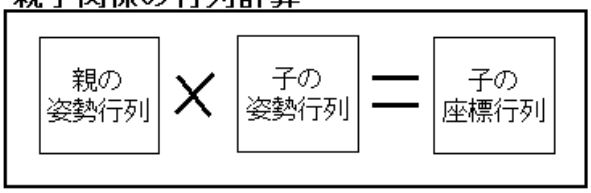
親子関係になっている場合は、それぞれの「子」になっているモデルは「親」の影響を受けます。例えば車の場合は車本体を「親」として、タイヤを「子」とする事で、本体が移動した場合は同時にタイヤもついて行く形になります。
その様に扱う事を「階層モデル」といいます。

車の階層モデル



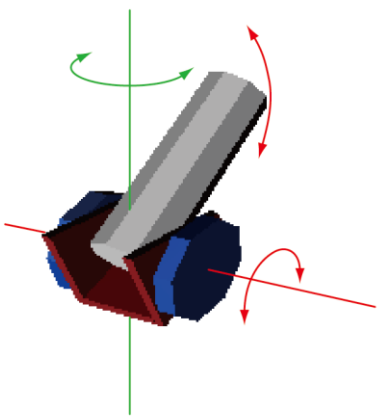
階層モデルの特性は、親子関係といい、親の「回転」「拡大」「移動」の影響を子が受けます。更に子の下に孫がいる場合は、孫は子の影響を受ける事になります。つまり、親は子へ、子は孫へ・・という具合に、子は「座標」「回転」「拡大」を親から伝えて貰って、更に自分の「座標」「回転」「拡大」を掛け合わせていく必要があります。

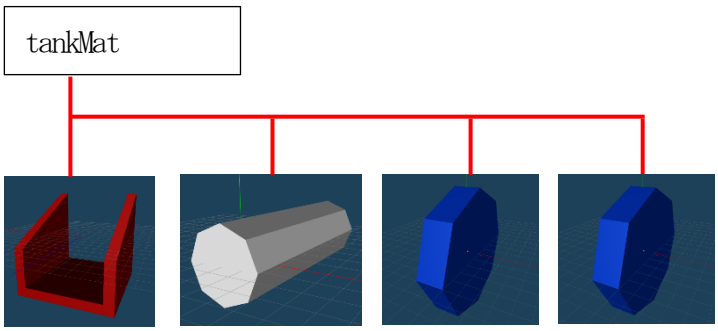
親子関係の行列計算



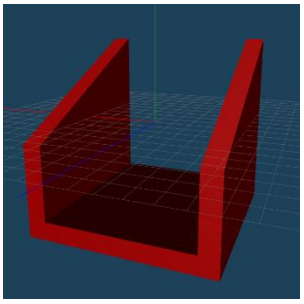
□タンクの親子関係の回転と移動（オフセット）の制御

タンク全体を親として、各パーツを子としてそれぞれを制御する。





「本体」 frame.mv1



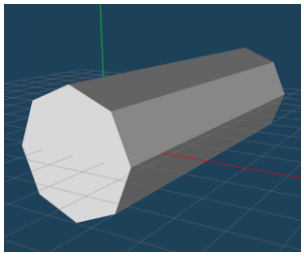
回転

```
bodyMat = MGetIdent();
bodyMat = MMuit(bodyMat, tankMat); // 親から貰う
```

移動

```
offsetY = 上移動量;
bodyMat = MMult(bodyMat, MGetTanslate(VGet(tankPos.x,
          tankPos.y + offsetY, tankPos.z);
```

「大砲」 barrel.mv1



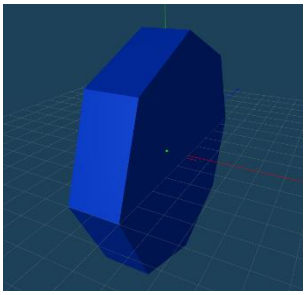
回転

```
barrelMat = MGetIdent();
barrelMat = MMult(barrelMat, tankMat); // 親から貰う
axis = VTransform(VGet(1, 0 0), barrelMat);
mat = MGetRotAxis(axis, barrelRotX); // X軸回転
barrelMat = MMuit(barrelMat, mat); // 親×子で合成
```

移動

```
offsetY = 上移動量;
barrelMat
    = MMult(bodyMat, MGetTanslate(VGet(tankPos.x,
            tankPos.y + offsetY, tankPos.z);
```

「車輪」 wheel.mv1



※右車輪の場合

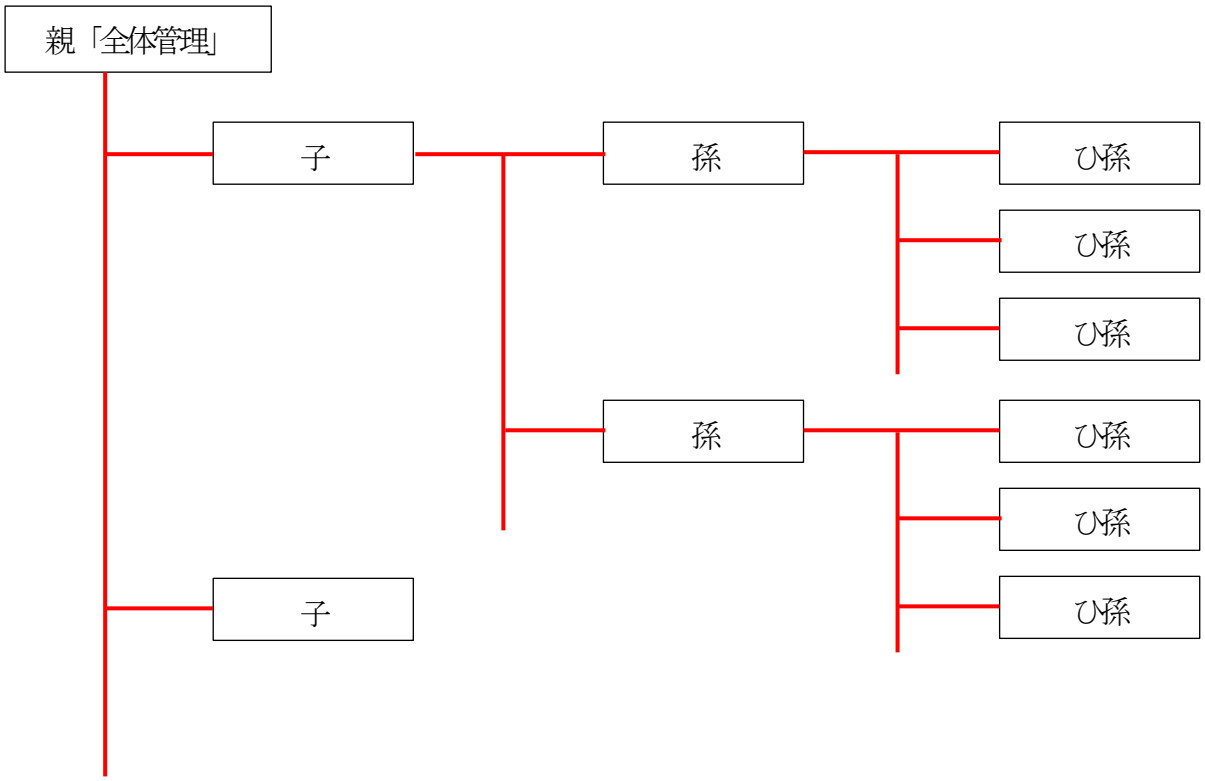
回転

```
wheelRMat = MGetIdent();
wheelRMat = MMult(wheelRMat, tankMat); // 親から貰う
axis = VTransform(VGet(1, 0, 0), wheelRMat);
mat = MGetRotAxis(axis, wheelRRotX); // X軸回転
wheelRMat = MMuit(wheelRMat, mat); // 親×子で合成
```

移動

```
offset = VTransform(VGet(1, 0, 0), wheelRMat);
offset.x = offset.x * 横向きのオフセット量;
offset.z = offset.z * 横向きのオフセット量;
wheelRMat
    = MMult(wheelRMat, MGetTanslate(VGet(tankPos.x
     +offset.x, tankPos.y, tankPos.z+offset.z);
```

■親・子・孫・ひ孫の管理



■親の回転Mat ＝ ゲームの進行に合わせて任意に設定
□子の移動Mat ＝ ゲームの進行に合わせて任意に設定

■子の回転Mat ＝ 親の回転Mat × 子のローカル回転Mat
□子の移動Mat ＝ 親の移動Mat ＋ 子のローカル移動Mat

■孫の回転Mat ＝ 子の回転Mat × 孫のローカル回転Mat
□孫の移動Mat ＝ 子の移動Mat ＋ 孫のローカル移動Mat

■ひ孫の回転Mat ＝ 孫の回転Mat × ひ孫のローカル回転Mat
□ひ孫の移動Mat ＝ 孫の移動Mat ＋ ひ孫のローカル移動Mat