# 参考文献

田浦健次朗先生の『剛体の物理の基礎』．

# 定義

ある剛体は，質量を持つ*N*個の質点から構成され，各質点の位置と質量をとする．隣接する質点同士は互いに結合しその相対位置関係は変化しないものとする（剛体だから）．この剛体の運動は，以下の4項目により表現できる:

* 重心の位置 :
* 重心の速度 :
* 姿勢（元の状態からの回転）:
* 回転角速度 :

ここで，姿勢（元の状態からの回転）は，3次元ベクトルにより表現する．このベクトルは，方向が回転軸方向，長さが回転角度を表す．同様に回転角速度も3次元ベクトルにて表現する．

# 並進運動（重心の運動）

今，全質点の総質量と重心位置は，以下の通り求められる，

さらに，右式を時間について2階微分すると，

となる．ここで，である．質点*i*にかかる外力をとすると，運動方程式により次式が成り立つ，

これを全ての質点について足し合わせると，

となる．これに式(2),(3)を代入すると，

が得られる．つまり，外力の総和は重心に加速度をもたらす．

重心の並進運動 : 重心の加速度は，外力ベクトルの総和により次式の通り変化する．

# 回転運動（重心周りの運動）

質点*i*の位置をとし（姿勢回転は適用済み），前述の定義を利用すると，この質点*i*の速度は，

と表せる．ただし，と置いた．質点*i*の速度をとするとについて運動方程式より，以下が成り立つ，

これを以下の通り変形する

式(7)~式(8)の変形では，関数の積の微分公式とという関係を利用した．式(8)を全ての質点に対して足し合わせ整理する;

さらに式(\*)を代入すると

この式の右辺第一項はゼロになる()ので

となる．この式の左辺は，重心周りのトルクの総量を表す．右辺は，重心周りの角速度の変化を表す．右辺を整理すると

が得られる．ここで，

を慣性モーメントテンソルと呼ぶ．

重心周りの回転運動 : 重心を通る軸を中心とした回転の角速度は，重心周りのトルクの総量により，以下の通り変化する;

Iを慣性モーメントと呼ぶ．

剛体シミュレーションでは，

として角速度を更新する

また、とも表せる．

これより，

# 剛体シミュレーション.

参考リンク :

<http://eman-physics.net/dynamics/mom_tensor.html>

上記より，剛体の運動を

* 重心の位置 :
* 重心の速度 :
* 姿勢（元の状態からの回転）:
* 回転角速度 :

という4項目により記述し，これを，以下を用いて更新すればよいことが分かった．

* 位置の変化
* 速度の変化
* 姿勢の変化
* 角速度ベクトルの変化

ただし，は，回転ベクトル（）による回転行列．

# 慣性モーメントテンソルの前計算

上記の通り，を保持してシミュレーションステップを進めると，フレームごとに物体が回転するため，慣性モーメントテンソルを再計算する必要がある（ように思える）．今，ある程度シミュレーションステップが進んだ状態を考え，現在の姿勢を表す回転行列を**R**とし，この物体にかかっているトルクを，物体の角加速度をとする．すると，

が成立する．は現在の回転姿勢**R**を考慮した慣性モーメントテンソルである．このは，ワールド座標系に置けるものである．これを物体のローカル座標系（物体が回転する前の座標系）に変換するには，トルクと角加速度ベクトルを以下の通り変換すればよい；

このローカル座標系におけるトルクと角加速度の関係は，

であり，変形を進めると、

となる．これを，ワールド座標系における式と見比べると，

を得る．つまり，慣性モーメントテンソルは，初期状態において一度だけ計算しておき，シミュレーションが進んで物体が行列**R**だけ回転した場合は，と変換すればよい．

ソースコード – レンダリング

//半径rの球を書く

glPushMatrix();

//平行移動

glTranslatef( m\_position[0], m\_position[1], m\_position[2] );

//回転

if (m\_angle.norm() > 0.001){

EMat3f R;

R = Eigen::AngleAxisf(m\_angle.norm(), m\_angle.normalized());

float A[16];

A[0] = R(0, 0); A[4] = R(0, 1); A[8] = R(0, 2); A[12] = 0;

A[1] = R(1, 0); A[5] = R(1, 1); A[9] = R(1, 2); A[13] = 0;

A[2] = R(2, 0); A[6] = R(2, 1); A[10] = R(2, 2); A[14] = 0;

A[3] = 0; A[7] = 0; A[11] = 0; A[15] = 1;

glMultMatrixf(A);

}

//描画

DrawSphere( 20, 20, m\_radius );

glPopMatrix();

ソースコード – シミュレーション

static EVec3f Gravity = EVec3f(0,-100, 0);

m\_force\_onestep += m\_mass \* Gravity;

//step volocify

// F = m x a --> a = F/m

EVec3f accel = (1/m\_mass) \* m\_force\_onestep;

//dv = a x dt, v = v + dv

m\_velocity = m\_velocity + h \* accel;

//dx = v \* dt, x = x + dx

m\_position = m\_position + h \* m\_velocity;

//step rotation

//角加速度 a, トルク N, 慣性モーメントテンソル I,

//半径rの球なら I = diag(r^2,r^2,r^2)

// a = I^-1 x N

EVec3f angleaccel = m\_torque\_onestep / (m\_mass \* m\_radius \* m\_radius);

//角速度 = 角速度 + 角加速度 x dt

m\_anglevelo = m\_anglevelo + angleaccel \* h;

//角度 = 角度 + 角速度 x dt

m\_angle = m\_angle + m\_anglevelo \* h;

//床との交差判定

if ( m\_position[1] - m\_radius < FLOOR\_Y )

{

m\_position[1] = m\_radius; //引っ張り戻して

m\_velocity[1] \*= -0.9f; //跳ね返らせる

}

m\_velocity \*= 0.99999f;//減衰