

ゲームグラフィックス特論

第4回 変換 (2)



四元数

回転の道具として



四元数 (クォータニオン)

- 複素数の虚部を三次元に拡張したもの

$$\hat{\mathbf{q}} = (q_x, q_y, q_z, q_w), \mathbf{q}_v = (q_x, q_y, q_z) \Rightarrow \hat{\mathbf{q}} = (\mathbf{q}_v, q_w)$$

虚部

実部

- 定義

$$\hat{\mathbf{q}} = (\mathbf{q}_v, q_w) = iq_x + jq_y + kq_z + q_w = \mathbf{q}_v + q_w$$

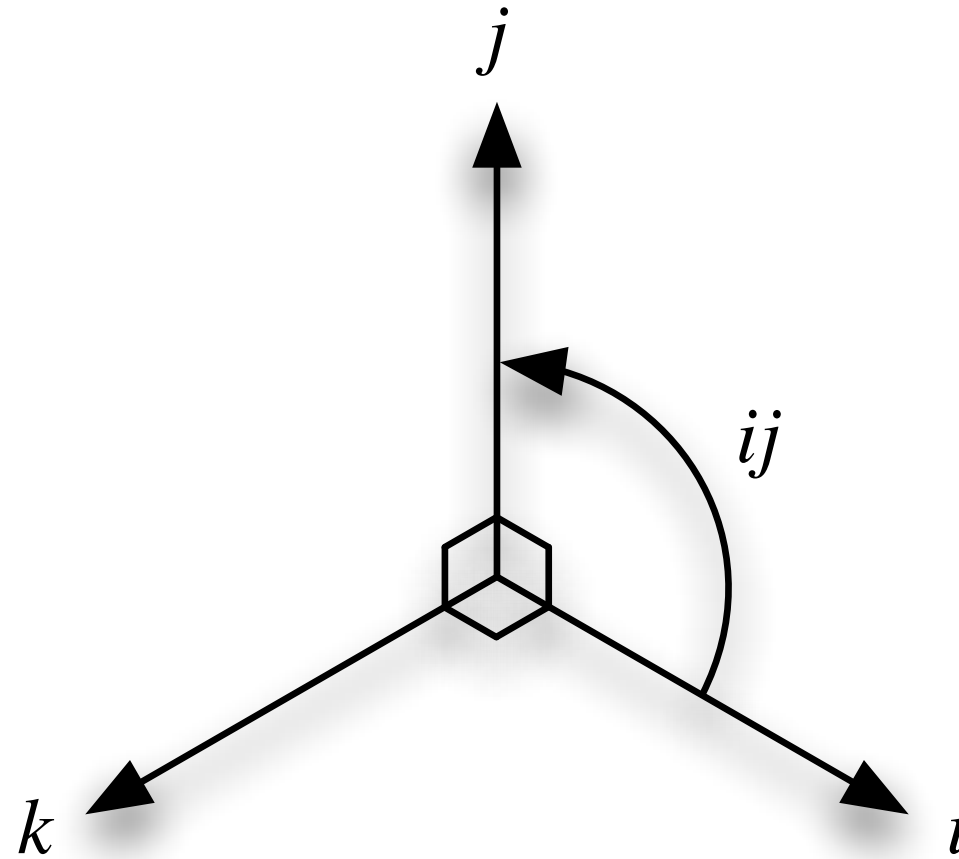
$$\mathbf{q}_v = iq_x + jq_y + kq_z = (q_x, q_y, q_z)$$

$$i^2 = j^2 = k^2 = ijk = -1$$

$$jk = -kj = i, ki = -ik = j, ij = -ji = k$$

 i, j, k : 虚数単位


四元数の虚数単位の関係のイメージ



四元数の演算

・積

$$\begin{aligned}
 \hat{\mathbf{q}}\hat{\mathbf{r}} &= (iq_x + jq_y + kq_z + q_w)(ir_x + jr_y + kr_z + r_w) \\
 &= i(q_y - r_z + r_wq_x + q_wr_x) \\
 &\quad + j(q_z - r_x + r_wq_y + q_wr_y) \\
 &\quad + k(q_x - r_y + r_wq_z + q_wr_z) \\
 &\quad + q_wr_w - q_xr_x - q_yr_y - q_zr_z \\
 &= (\mathbf{q}_v \times \mathbf{r}_v + r_w\mathbf{q}_v + q_w\mathbf{r}_v, q_wr_w - \mathbf{q}_v \cdot \mathbf{r}_v)
 \end{aligned}$$



四元数の積のサンプルコード

```
/*  
** p <- q * r  
*/  
void qmul(float *p, const float *q, const float *r)  
{  
    p[0] = q[1]*r[2] - q[2]*r[1] + r[3]*q[0] + q[3]*r[0];  
    p[1] = q[2]*r[0] - q[0]*r[2] + r[3]*q[1] + q[3]*r[1];  
    p[2] = q[0]*r[1] - q[1]*r[0] + r[3]*q[2] + q[3]*r[2];  
    p[3] = q[3]*r[3] - q[0]*r[0] - r[1]*q[1] - q[2]*r[2];  
}
```



四元数の演算

- 和

- $\hat{\mathbf{q}} + \hat{\mathbf{r}} = (\mathbf{q}_v, q_w) + (\mathbf{r}_v, r_w) = (\mathbf{q}_v + \mathbf{r}_v, q_w + r_w)$

要素ごとに足す

- 共役

- $\hat{\mathbf{q}}^* = (\mathbf{q}_v, q_w)^* = (-\mathbf{q}_v, q_w)$

虚数部の向きを反転

- ノルム

- $n(\hat{\mathbf{q}}) = \sqrt{\hat{\mathbf{q}}\hat{\mathbf{q}}^*} = \sqrt{\hat{\mathbf{q}}^*\hat{\mathbf{q}}} = \sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2}$

$\hat{\mathbf{q}}\hat{\mathbf{q}}^* = \hat{\mathbf{q}}^*\hat{\mathbf{q}}$ は要素ごとの積の和

- 単位元

- $\hat{\mathbf{i}} = (\mathbf{0}, 1)$

$\hat{\mathbf{q}}\hat{\mathbf{i}} = \hat{\mathbf{i}}\hat{\mathbf{q}} = \hat{\mathbf{q}}$



逆元

- ノルム

- $n(\hat{\mathbf{q}}) = \sqrt{\hat{\mathbf{q}}\hat{\mathbf{q}}^*} \Leftrightarrow \frac{\sqrt{\hat{\mathbf{q}}\hat{\mathbf{q}}^*}}{n(\hat{\mathbf{q}})} = 1$

- より, 逆元は

- $\hat{\mathbf{q}}^{-1} = \frac{\hat{\mathbf{q}}^*}{|n(\hat{\mathbf{q}})|^2}$



四元数の公式と法則

- 共役の公式

- $(\hat{\mathbf{q}}^*)^* = \hat{\mathbf{q}}$
- $(\hat{\mathbf{q}} + \hat{\mathbf{r}})^* = \hat{\mathbf{q}}^* + \hat{\mathbf{r}}^*$
- $(\hat{\mathbf{q}}\hat{\mathbf{r}})^* = \hat{\mathbf{r}}^*\hat{\mathbf{q}}^*$

- ノルムの公式

- $n(\hat{\mathbf{q}}^*) = n(\hat{\mathbf{q}})$
- $n(\hat{\mathbf{q}}\hat{\mathbf{r}}) = n(\hat{\mathbf{q}})n(\hat{\mathbf{r}})$

- 線形性

- $\hat{\mathbf{p}}(s\hat{\mathbf{q}} + t\hat{\mathbf{r}}) = s\hat{\mathbf{p}}\hat{\mathbf{q}} + t\hat{\mathbf{p}}\hat{\mathbf{r}}$
- $(s\hat{\mathbf{p}} + t\hat{\mathbf{q}})\hat{\mathbf{r}} = s\hat{\mathbf{p}}\hat{\mathbf{r}} + t\hat{\mathbf{q}}\hat{\mathbf{r}}$

- 結合の法則

- $\hat{\mathbf{p}}(\hat{\mathbf{q}}\hat{\mathbf{r}}) = (\hat{\mathbf{p}}\hat{\mathbf{q}})\hat{\mathbf{r}}$



単位四元数

- 四元数 $\hat{\mathbf{q}} = (\mathbf{q}_v, q_w)$ のノルムが 1 のとき
 - すなわち $n(\hat{\mathbf{q}}) = 1$ である
 - ここで \mathbf{u}_q を $\|\mathbf{u}_q\| = 1$ のベクトル (単位ベクトル) とすれば
 - これは $\hat{\mathbf{q}} = (\sin \phi \mathbf{u}_q, \cos \phi) = \sin \phi \mathbf{u}_q + \cos \phi$ と表せる
 - このとき $n(\hat{\mathbf{q}}) = n(\sin \phi \mathbf{u}_q, \cos \phi) = \sqrt{\sin^2 \phi (\mathbf{u}_q \cdot \mathbf{u}_q) + \cos^2 \phi} = 1$
- オイラーの公式より $\cos \phi + i \sin \phi = e^{i\phi}$ であることから
 - $\hat{\mathbf{q}} = \sin \phi \mathbf{u}_q + \cos \phi = e^{\phi \mathbf{u}_q}$



単位四元数の対数と指数

• 対数

$$\bullet \log(\hat{\mathbf{q}}) = \log(e^{\phi \mathbf{u}_q}) = \phi \mathbf{u}_q$$

← 単位ベクトルに角度をかけたもの

• 指数

$$\bullet \hat{\mathbf{q}}^t = (\sin \phi \mathbf{u}_q, \cos \phi)^t = e^{\phi t \mathbf{u}_q} = (\sin \phi t \mathbf{u}_q, \cos \phi t)$$

↑ ↑
角度の指数倍

単位四元数のべき乗は
角度を指数倍したもの



小テスト－四元数

Moodle の小テストに解答してください

四元数による回転

四元数を回転の道具として使う



複素数による回転

- 2次元上の座標 (x, y) を複素数で表す

$$x + iy$$

- 複素数による角度 θ の回転

$$\cos \theta + i \sin \theta$$

- これらを掛ける

$$\begin{aligned}(x' + iy') &= (x + iy)(\cos \theta + i \sin \theta) \\ &= x \cos \theta - y \sin \theta + i(x \sin \theta + y \cos \theta)\end{aligned}$$

- すなわち

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



単位四元数による回転

- 同次座標で表された点または方向 \mathbf{p}

- $\mathbf{p} = (p_x, p_y, p_z, p_w)$

- これを四元数として扱う

- $\hat{\mathbf{p}} = (\mathbf{p}_v, p_w), \mathbf{p}_v = (p_x, p_y, p_z)$

- 単位四元数

- $\hat{\mathbf{q}} = (\sin \phi \mathbf{u}_q, \cos \phi)$

- 回転の変換

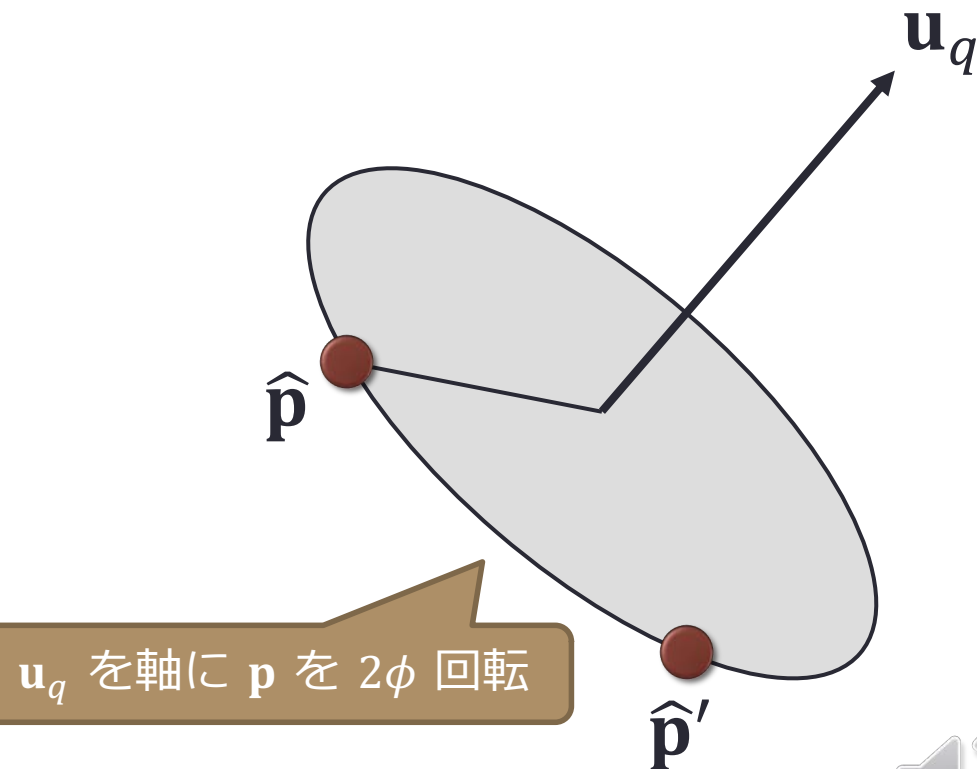
- $\hat{\mathbf{p}}' = \hat{\mathbf{q}}\hat{\mathbf{p}}\hat{\mathbf{q}}^{-1}$

- 単位四元数では $\mathbf{q}^{-1} = \mathbf{q}^*$ なので

- $\hat{\mathbf{p}}' = \hat{\mathbf{q}}\hat{\mathbf{p}}\hat{\mathbf{q}}^*$

$\hat{\mathbf{q}}\hat{\mathbf{p}}$ は4次元空間で ϕ 回転するが虚部を3次元空間の軸に合わせるため反対から $\hat{\mathbf{q}}^{-1}$ をかけてさらに ϕ 回転しつつ4次元目の軸に対しては $-\phi$ 回転するというこらしい

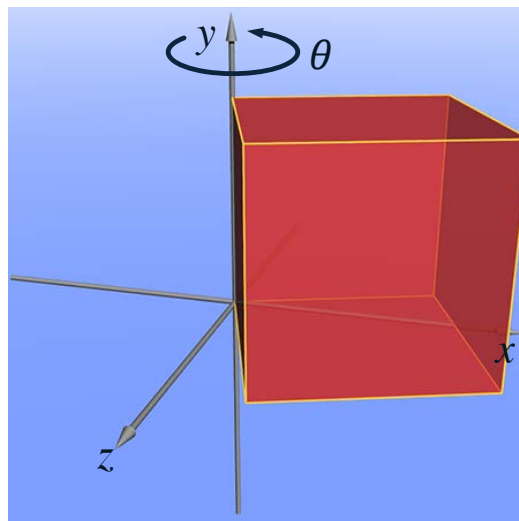
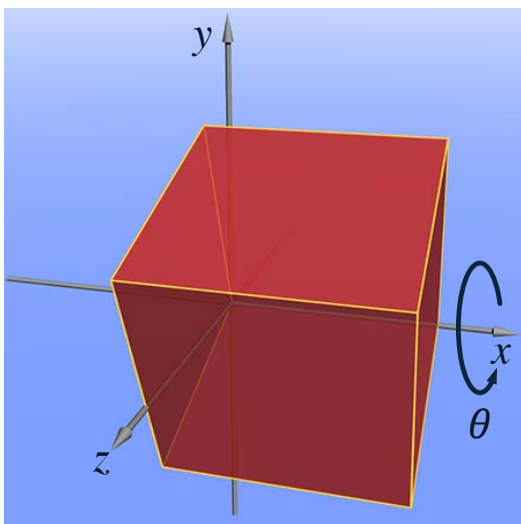
$\hat{\mathbf{q}}$ と $-\hat{\mathbf{q}}$ は同じ回転を表す



座標軸中心の回転

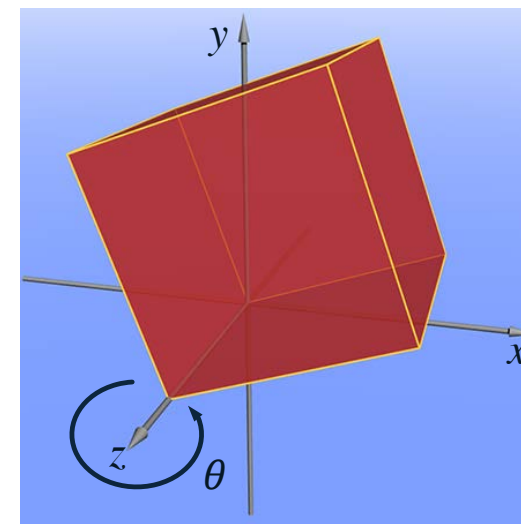
X 軸中心の回転

$$\begin{aligned}\hat{\mathbf{q}}_x(\theta) &= \left(\sin \frac{\theta}{2}, 0, 0, \cos \frac{\theta}{2} \right) \\ &= i \sin \frac{\theta}{2} + \cos \frac{\theta}{2}\end{aligned}$$



Z 軸中心の回転

$$\begin{aligned}\hat{\mathbf{q}}_z(\theta) &= \left(0, 0, \sin \frac{\theta}{2}, \cos \frac{\theta}{2} \right) \\ &= k \sin \frac{\theta}{2} + \cos \frac{\theta}{2}\end{aligned}$$



Y 軸中心の回転

$$\begin{aligned}\hat{\mathbf{q}}_y(\theta) &= \left(0, \sin \frac{\theta}{2}, 0, \cos \frac{\theta}{2} \right) \\ &= j \sin \frac{\theta}{2} + \cos \frac{\theta}{2}\end{aligned}$$



軸と回転角から単位四元数を求める

```
/*  
** q <- 軸(x, y, z) 角度(a)  
*/  
void qmake(float *q, float x, float y, float z, float a)  
{  
    const float l(x * x + y * y + z * z);  
  
    if (l != 0.0f) {  
        const float s(sin(a * 0.5f) / sqrt(l));  
  
        q[0] = x * s;  
        q[1] = y * s;  
        q[2] = z * s;  
        q[3] = cos(a);  
    }  
}
```



単位四元数による回転の変換の合成

- \hat{q} 回転してから更に \hat{r} 回転する

$$\hat{r}(\hat{q}\hat{p}\hat{q}^*)\hat{r}^* = (\hat{r}\hat{q})\hat{p}(\hat{q}^*\hat{r}^*) = (\hat{r}\hat{q})\hat{p}(\hat{r}\hat{q})^*$$

- したがって単位四元数の積 $\hat{r}\hat{q}$ も回転を表す



単位四元数の積は回転の合成になる

合成により誤差が累積しても正規化すれば「回転」であることは保たれる
(正規化するにはノルムで割る → ベクトルの正規化と同じなので簡単)



小テストー四元数による回転

Moodle の小テストに解答してください

四元数と回転の変換

四元数と変換行列との関係



単位四元数から回転変換行列を算出

$$\hat{\mathbf{q}} = (q_x, q_y, q_z, q_w)$$

$$\mathbf{M}^q = \begin{pmatrix} 1 - s(q_y^2 + q_z^2) & s(q_x q_y - q_w q_z) & s(q_z q_x + q_w q_y) & 0 \\ s(q_x q_y + q_w q_z) & 1 - s(q_z^2 + q_x^2) & s(q_y q_z - q_w q_x) & 0 \\ s(q_z q_x - q_w q_y) & s(q_y q_z + q_w q_x) & 1 - s(q_x^2 + q_y^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

ここで $s = 2/n(\hat{\mathbf{q}})$ なので単位四元数では以下のように単純化できる

$$\mathbf{M}^q = \begin{pmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_z q_x + q_w q_y) & 0 \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_z^2 + q_x^2) & 2(q_y q_z - q_w q_x) & 0 \\ 2(q_z q_x - q_w q_y) & 2(q_y q_z + q_w q_x) & 1 - 2(q_x^2 + q_y^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

単位四元数にしてしまえば変換に三角関数は不要



単位四元数から回転変換行列を得る

```
/*  
** 回転変換行列 m <- 単位四元数 q  
*/  
void qrot(float *m, const float *q)  
{  
    float xx = q[0] * q[0] * 2.0f;  
    float yy = q[1] * q[1] * 2.0f;  
    float zz = q[2] * q[2] * 2.0f;  
    float xy = q[0] * q[1] * 2.0f;  
    float yz = q[1] * q[2] * 2.0f;  
    float zx = q[2] * q[0] * 2.0f;  
    float xw = q[0] * q[3] * 2.0f;  
    float yw = q[1] * q[3] * 2.0f;  
    float zw = q[2] * q[3] * 2.0f;
```

```
    m[ 0] = 1.0f - yy - zz;  
    m[ 1] = xy + zw;  
    m[ 2] = zx - yw;  
    m[ 4] = xy - zw;  
    m[ 5] = 1.0f - zz - xx;  
    m[ 6] = yz + xw;  
    m[ 8] = zx + yw;  
    m[ 9] = yz - xw;  
    m[10] = 1.0f - xx - yy;  
    m[ 3] = m[ 7] = m[11] =  
    m[12] = m[13] = m[14] = 0.0f;  
    m[15] = 1.0f;  
}
```



直交行列から四元数への変換

$$\mathbf{M}^q = \begin{pmatrix} m_{00}^q & m_{01}^q & m_{02}^q & m_{03}^q \\ m_{10}^q & m_{11}^q & m_{12}^q & m_{13}^q \\ m_{20}^q & m_{21}^q & m_{22}^q & m_{23}^q \\ m_{30}^q & m_{31}^q & m_{32}^q & m_{33}^q \end{pmatrix} = \begin{pmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_z q_x + q_w q_y) & 0 \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_z^2 + q_x^2) & 2(q_y q_z - q_w q_x) & 0 \\ 2(q_z q_x - q_w q_y) & 2(q_y q_z + q_w q_x) & 1 - 2(q_x^2 + q_y^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

行列のトレース
(対角要素の和)

$$\text{tr}(\mathbf{M}^q) = \sum_{i=0}^3 m_{ii}^q = m_{00}^q + m_{11}^q + m_{22}^q + m_{33}^q = 4 - 4(q_x^2 + q_y^2 + q_z^2) = 4q_w^2$$



直交行列から四元数への変換

- したがって

$$q_w = \frac{1}{2} \sqrt{\text{tr}(\mathbf{M}^q)}$$

- より

$$q_x = \frac{m_{21}^q - m_{12}^q}{4q_w}$$

$$q_y = \frac{m_{02}^q - m_{20}^q}{4q_w}$$

$$q_z = \frac{m_{10}^q - m_{01}^q}{4q_w}$$

あるいは

$$4q_x^2 = +m_{00}^q - m_{11}^q - m_{22}^q + m_{33}^q$$

$$4q_y^2 = -m_{00}^q + m_{11}^q - m_{22}^q + m_{33}^q$$

$$4q_z^2 = -m_{00}^q - m_{11}^q + m_{22}^q + m_{33}^q$$

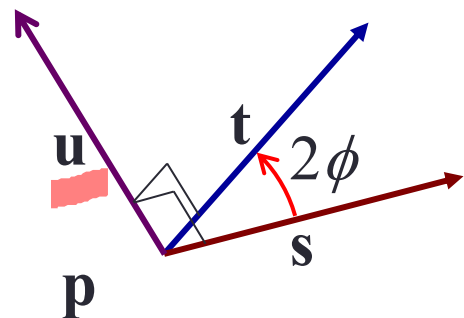
$q_w = 0$ の場合



あるベクトルを別のベクトルの方向に向ける回転

この回転を $\hat{\mathbf{q}} = (\sin \phi \mathbf{u}, \cos \phi)$ とする

回転軸を \mathbf{u} は \mathbf{s} と \mathbf{t} の外積を正規化したもの



$$\mathbf{u} = \frac{\mathbf{s} \times \mathbf{t}}{\|\mathbf{s} \times \mathbf{t}\|}$$

(外積の定義)

(倍角の公式)

この分母 $\|\mathbf{s} \times \mathbf{t}\| = \sin 2\phi = 2 \sin \phi \cos \phi$

したがって

$$\hat{\mathbf{q}} = \left(\frac{\sin \phi}{\sin 2\phi} \mathbf{s} \times \mathbf{t}, \cos \phi \right) = \left(\frac{1}{2 \cos \phi} \mathbf{s} \times \mathbf{t}, \cos \phi \right)$$

(倍角の公式)

$$\mathbf{s} \cdot \mathbf{t} = \cos 2\phi = e \text{ とおけば } \cos \phi = \sqrt{\frac{1+e}{2}}$$

$$\Rightarrow \left(\frac{1}{\sqrt{2(1+e)}} \mathbf{s} \times \mathbf{t}, \frac{\sqrt{2(1+e)}}{2} \right)$$



ベクトルの回転の行列表記

- したがって

$$\hat{\mathbf{q}} = \left(\frac{1}{\sqrt{2(1+e)}} \mathbf{s} \times \mathbf{t}, \frac{\sqrt{2(1+e)}}{2} \right)$$

- より \mathbf{s} を \mathbf{t} に向ける回転の変換 $\mathbf{R}(\mathbf{s}, \mathbf{t})$ は

$$\mathbf{R}(\mathbf{s}, \mathbf{t}) = \begin{pmatrix} e + hu_x^2 & hu_xu_y - u_z & hu_zu_x + u_y & 0 \\ hu_xu_y + u_z & e + hu_y^2 & hu_yu_z - u_x & 0 \\ hu_zu_x - u_y & hu_yu_z + u_x & e + hu_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{u} = (u_x, u_y, u_z) = \frac{\mathbf{s} \times \mathbf{t}}{\|\mathbf{s} \times \mathbf{t}\|}, \quad e = \cos 2\phi = \mathbf{s} \cdot \mathbf{t}, \quad h = 1 - \cos 2\phi = 1 - e$$



剛体アニメーション

時刻を扱う



剛体アニメーション

- 剛体変換によるアニメーション
 - 形状の変形を伴わない
- 剛体変換

$$\mathbf{M} = \begin{pmatrix} \bar{\mathbf{R}} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{R}} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix} \quad \mathbf{I}: \text{単位行列}$$

\mathbf{t} : 位置 $\bar{\mathbf{R}}$: 回転



(オイラー変換を使う場合)

t : 時刻

$$\mathbf{t}(t) = \mathbf{P}(t)$$

$$\mathbf{E}(h, p, r) = \mathbf{R}_y(h) \mathbf{R}_x(p) \mathbf{R}_z(r)$$



$$\bar{\mathbf{R}}(t) = \mathbf{R}_y(h(t)) \mathbf{R}_x(p(t)) \mathbf{R}_z(r(t))$$



現在時刻の取得

- システムの時計から時刻を得る
 - ここでは `glfwSetTime()`, `glfwGetTime()` を利用する

```
const double cycle(5.0);
...

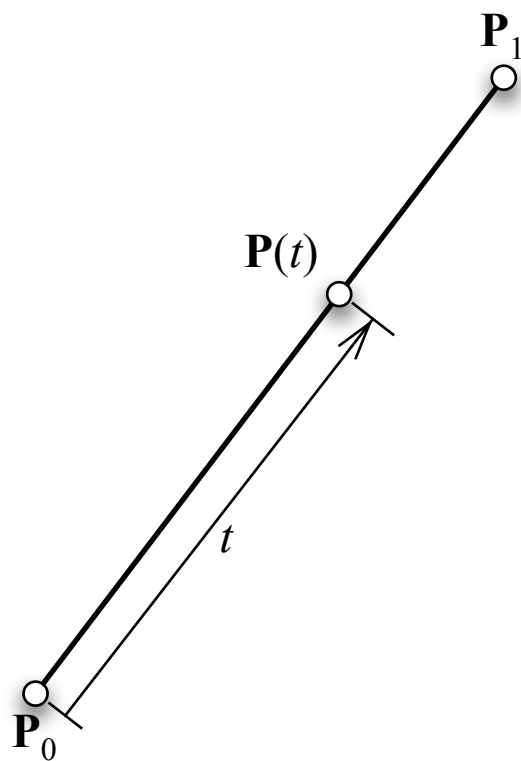
// 経過時間のリセット
glfwSetTime(0.0);
...

// ウィンドウが開いている間くり返し描画する
while (window.shouldClose() == GL_FALSE)
{
    ...

    // 時刻の計測（周期 5 秒）
    const float
        t((float)(fmod(glfwGetTime(), cycle)
        / cycle));
    ...
}
```

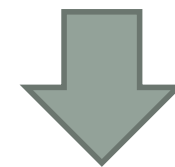


線形補間 (Linear interpolation, Lerp)



- 2点 P_0 と P_1 を結ぶ線分を $t \in [0, 1]$ で内分する点 $P(t)$

$$P(t) = P_0(1 - t) + P_1t$$



この $P(t)$ を使って平行移動する



平行移動の変換行列を求める

```
/*  
** 平行移動変換行列を求める  
*/  
void translate(float *m, float x, float y, float z)  
{  
    m[ 3] = x;  
    m[ 7] = y;  
    m[11] = z;  
    m[ 0] = m[ 5] = m[10] = m[15] = 1.0f;  
    m[ 1] = m[ 2] = m[ 4] =  
    m[ 6] = m[ 8] = m[ 9] =  
    m[12] = m[13] = m[14] = 0.0f;  
}
```

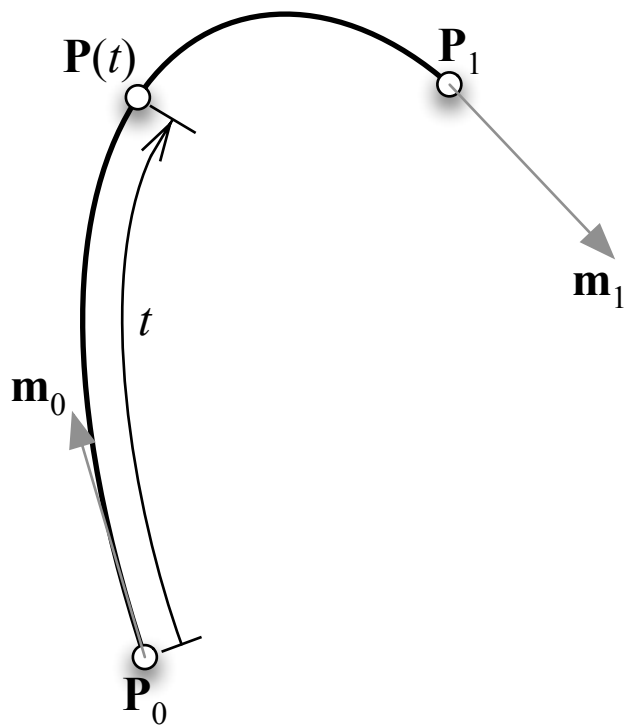


2点間を直線移動する変換行列を得る

```
/*  
** 平行移動アニメーションの変換行列を求める  
*/  
void lerp(float *m, const float *p0, const float *p1, float t)  
{  
    const float x((p1[0] - p0[0]) * t + p0[0]);  
    const float y((p1[1] - p0[1]) * t + p0[1]);  
    const float z((p1[2] - p0[2]) * t + p0[2]);  
  
    translate(m, x, y, z);  
}
```



Cubic Hermite Spline



- P_0 における速度が \mathbf{m}_0 である点が、 P_1 において速度が \mathbf{m}_1 となるように滑らかに移動する場合の、時刻 $t \in [0, 1]$ における点の位置 $P(t)$

$$\begin{aligned} P(t) = & P_0(2t^3 - 3t^2 + 1) \\ & + \mathbf{m}_0(t^3 - 2t^2 + t) \\ & + P_1(-2t^3 + 3t^2) \\ & + \mathbf{m}_1(t^3 - t^2) \end{aligned}$$



Cubic Hermite Spline の求め方

- 二点 \mathbf{p}_0 , \mathbf{p}_1 を通り, 点 \mathbf{p}_0 における接線が \mathbf{m}_0 , \mathbf{p}_1 における接線が \mathbf{m}_1 となる三次曲線

$$\mathbf{p}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

- これを一回微分

$$\mathbf{p}'(t) = 3\mathbf{a}t^2 + 2\mathbf{b}t + \mathbf{c}$$

- $t = 0$ において

$$\mathbf{p}(0) = \mathbf{p}_0 = \mathbf{d}$$

$$\mathbf{p}'(0) = \mathbf{m}_0 = \mathbf{c}$$

- $t = 1$ において

$$\mathbf{p}(1) = \mathbf{p}_1 = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$$

$$\mathbf{p}'(1) = \mathbf{m}_1 = 3\mathbf{a} + 2\mathbf{b} + \mathbf{c}$$

- したがって

$$\mathbf{c} = \mathbf{m}_0$$

$$\mathbf{d} = \mathbf{p}_0$$

- \mathbf{a} を消去して

$$3\mathbf{p}_1 - \mathbf{m}_1 = \mathbf{b} + 2\mathbf{c} + 3\mathbf{d}$$

$$\mathbf{b} = 3\mathbf{p}_1 - \mathbf{m}_1 - 3\mathbf{p}_0 - 2\mathbf{m}_0$$

$$= 3(\mathbf{p}_1 - \mathbf{p}_0) - \mathbf{m}_1 - 2\mathbf{m}_0$$

- \mathbf{b} を消去して

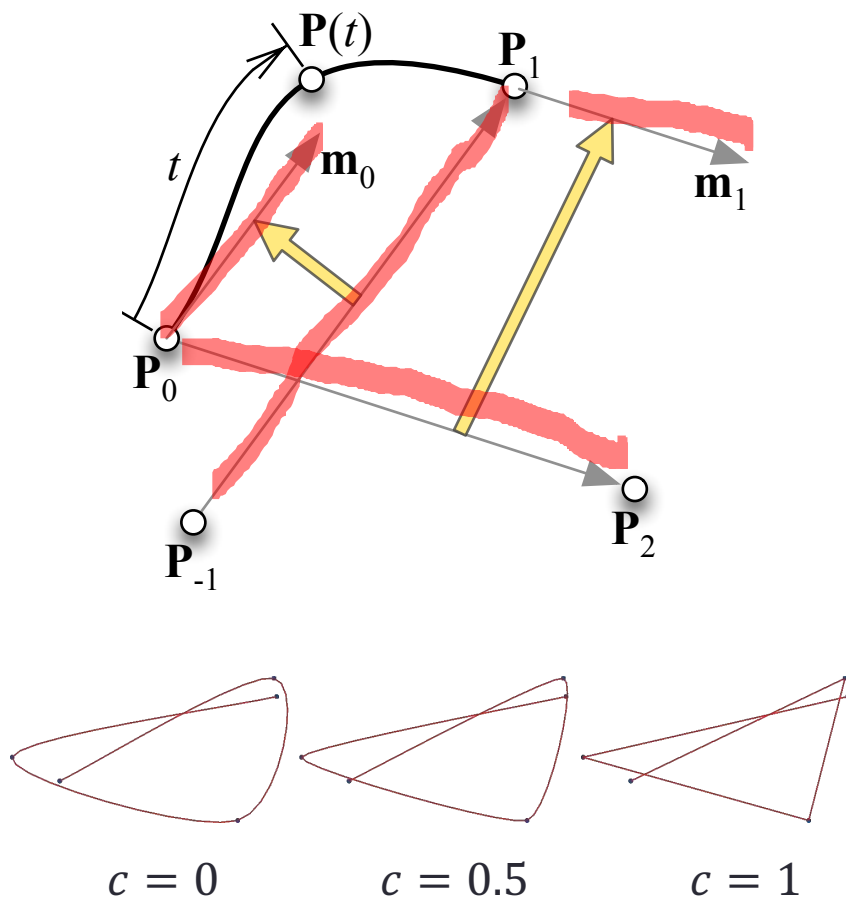
$$\mathbf{m}_1 - 2\mathbf{p}_1 = \mathbf{a} - \mathbf{c} - 2\mathbf{d}$$

$$\mathbf{a} = \mathbf{m}_1 - 2\mathbf{p}_1 + \mathbf{m}_0 + 2\mathbf{p}_0$$

$$= -2(\mathbf{p}_1 - \mathbf{p}_0) + \mathbf{m}_1 + \mathbf{m}_0$$



Catmull-Rom Spline



- Cubic Hermite Spline において

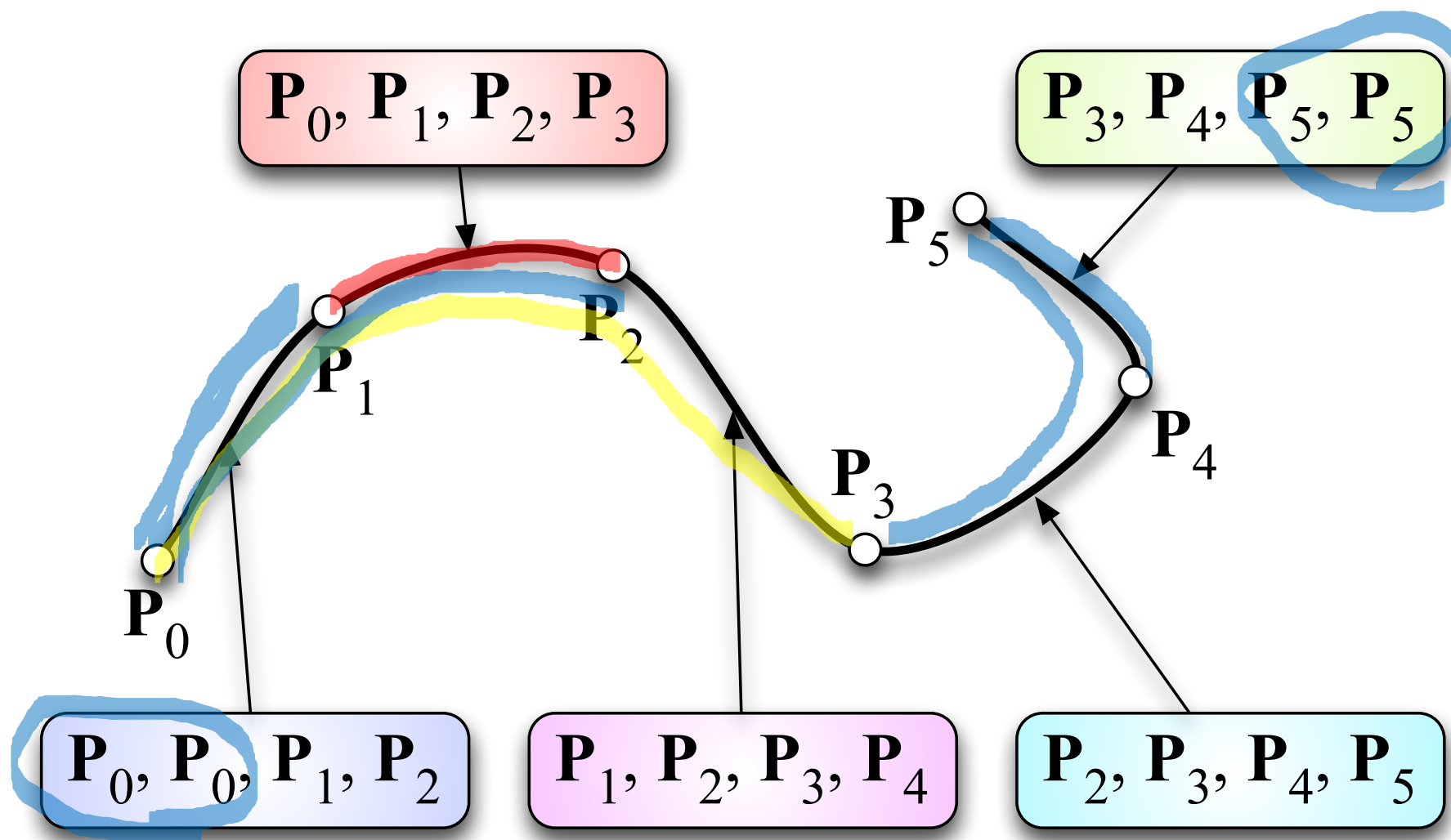
- $m_0 = \frac{1}{2}(P_1 - P_{-1})$
- $m_1 = \frac{1}{2}(P_2 - P_0)$

- Cardinal Spline

- $m_0 = \frac{1-c}{2}(P_1 - P_{-1})$
- $m_1 = \frac{1-c}{2}(P_2 - P_0)$
- c はテンション（張り）
 - $c = 1$: 折れ線
 - $c = 0$: Catmull-Rom Spline



Catmull-Rom Spline 曲線の接続



Catmull-Rom Spline のサンプルコード

```
/*  
** Catmull-Rom Spline  
*/  
static float catmull_rom(float x0, float x1, float x2, float x3, float t)  
{  
    const float m0((x2 - x0) * 0.5f);  
    const float m1((x3 - x1) * 0.5f);  
  
    const float d(x1 - x2);  
    const float a(2.0f * d + m0 + m1);  
    const float b(-3.0f * d - 2.0f * m0 - m1);  
  
    return ((a * t + b) * t + m0) * t + x1;  
}
```



Catmull-Rom Spline による補間

```
/*  
** Catmull-Rom Spline による点列の補間  
*/  
void interpolate(float *p,  
    const float *p0, const float *p1,  
    const float *p2, const float *p3,  
    float t)  
{  
    p[0] = catmull_rom(p0[0], p1[0], p2[0], p3[0], t);  
    p[1] = catmull_rom(p0[1], p1[1], p2[1], p3[1], t);  
    p[2] = catmull_rom(p0[2], p1[2], p2[2], p3[2], t);  
}
```

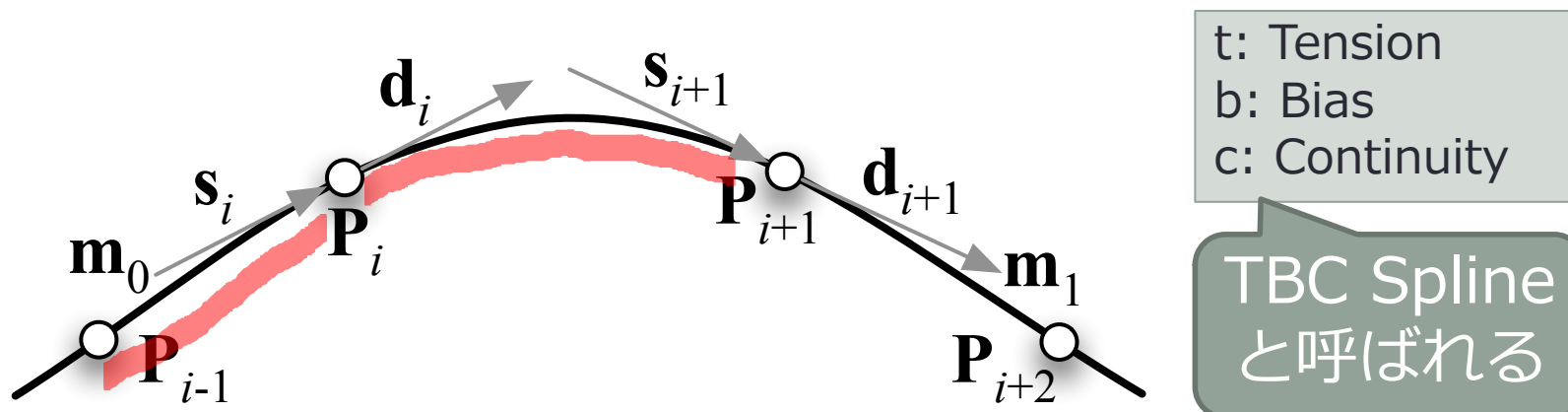


Kochanek-Bartels Spline

- Cubic Hermite Spline において点 i の進入側と退出側の速度 $\mathbf{s}_i, \mathbf{d}_i$ を別々に求める

$$\mathbf{s}_i = \frac{(1-t)(1+b)(1-c)}{2}(\mathbf{P}_i - \mathbf{P}_{i-1}) + \frac{(1-t)(1-b)(1+c)}{2}(\mathbf{P}_{i+1} - \mathbf{P}_i)$$

$$\mathbf{d}_i = \frac{(1-t)(1+b)(1+c)}{2}(\mathbf{P}_i - \mathbf{P}_{i-1}) + \frac{(1-t)(1-b)(1-c)}{2}(\mathbf{P}_{i+1} - \mathbf{P}_i)$$



回転の補間

球面線形補間



四元数を線形補間すると途中でスケールが変わる

$\hat{\mathbf{q}}_t = (1 - t)\hat{\mathbf{q}}_0 + t\hat{\mathbf{q}}_1 \quad (t \in [0, 1])$ という補間をする

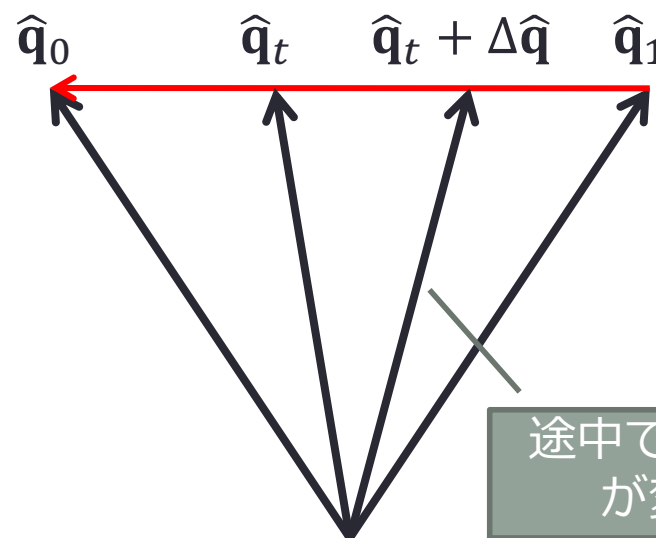
$$\hat{\mathbf{q}}_{t=0} = \hat{\mathbf{q}}_0$$

$$\hat{\mathbf{q}}_{t+\Delta t} = \{1 - (t + \Delta t)\}\hat{\mathbf{q}}_0 + (t + \Delta t)\hat{\mathbf{q}}_1$$

$$= (1 - t)\hat{\mathbf{q}}_0 + t\hat{\mathbf{q}}_1 + \Delta t(\hat{\mathbf{q}}_1 - \hat{\mathbf{q}}_0)$$

$$= \hat{\mathbf{q}}_t + \Delta\hat{\mathbf{q}}$$

ここで $\Delta\hat{\mathbf{q}} = \Delta t(\hat{\mathbf{q}}_1 - \hat{\mathbf{q}}_0)$



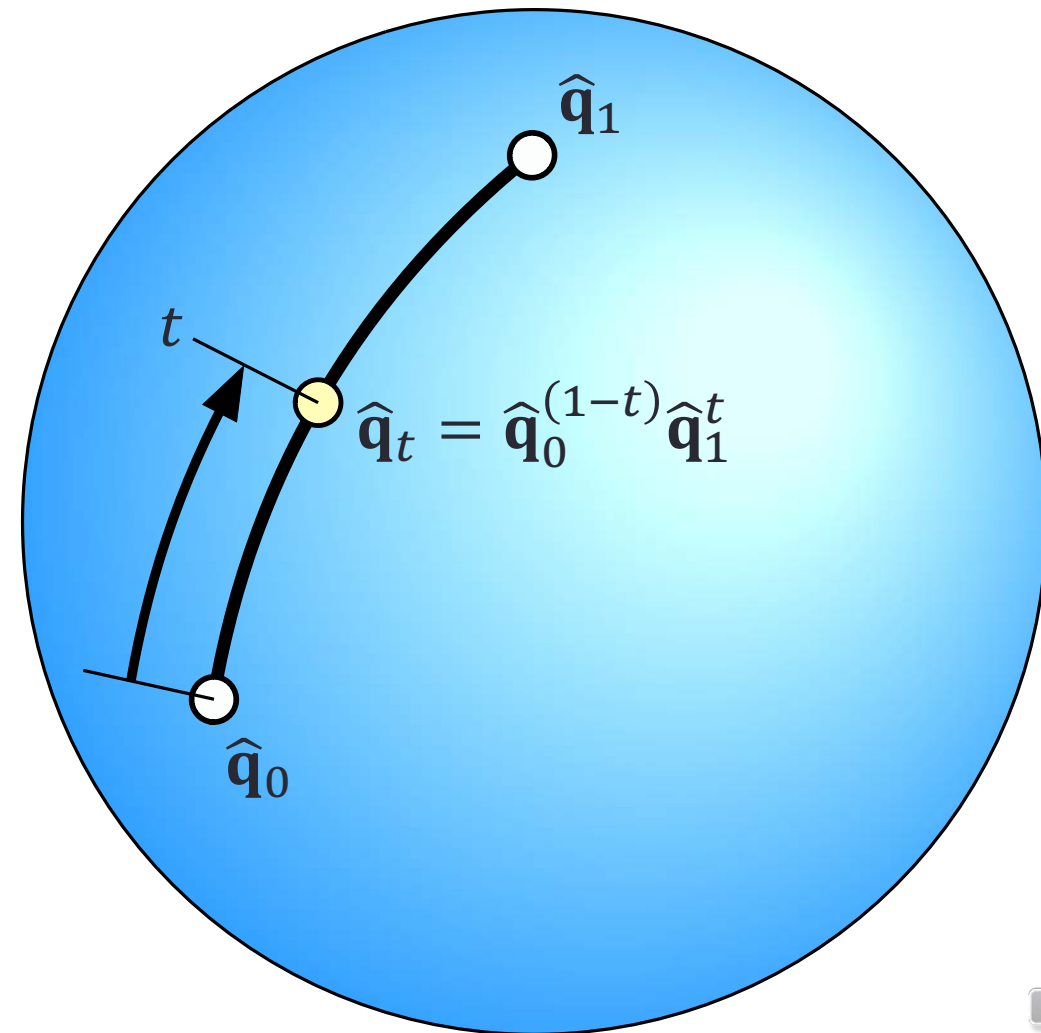
途中でスケール
が変わる



指数を補間すれば回転のみの補間ができる

$$\hat{\mathbf{q}}_t = \hat{\mathbf{q}}_0^{(1-t)} \hat{\mathbf{q}}_1^t \quad (t \in [0, 1])$$

- 単位四元数のべき乗は回転角を指数倍したもの
- 単位四元数の積は（回転）変換の合成
 - $t \rightarrow 0 \Rightarrow \hat{\mathbf{q}}_{t=0} = \hat{\mathbf{q}}_0$
 - $t \rightarrow 1 \Rightarrow \hat{\mathbf{q}}_{t=1} = \hat{\mathbf{q}}_1$



球面線形補間 (Spherical Linear interpolation, Slerp)

- $\hat{\mathbf{q}}_0^{(1-t)} \hat{\mathbf{q}}_1^t = \hat{\mathbf{q}}_0 \hat{\mathbf{q}}_0^{-t} \hat{\mathbf{q}}_1^t = \hat{\mathbf{q}}_0 (\hat{\mathbf{q}}_0^{-1} \hat{\mathbf{q}}_1)^t = \text{slerp}(\hat{\mathbf{q}}_0, \hat{\mathbf{q}}_1, t)$ とすると

$$\text{slerp}(\hat{\mathbf{q}}_0, \hat{\mathbf{q}}_1, t) = \frac{\sin\{\phi(1-t)\}}{\sin \phi} \hat{\mathbf{q}}_0 + \frac{\sin(\phi t)}{\sin \phi} \hat{\mathbf{q}}_1$$

- ここで

$$\cos \phi = \hat{\mathbf{q}}_0 \cdot \hat{\mathbf{q}}_1 = q_{0_x} q_{1_x} + q_{0_y} q_{1_y} + q_{0_z} q_{1_z} + q_{0_w} q_{1_w}$$

$$\phi = \cos^{-1}(\hat{\mathbf{q}}_0 \cdot \hat{\mathbf{q}}_1)$$

$$\sin \phi = \sqrt{1 - (\hat{\mathbf{q}}_0 \cdot \hat{\mathbf{q}}_1)^2}$$



球面線形補間のサンプルコード

```
#include <cmath>

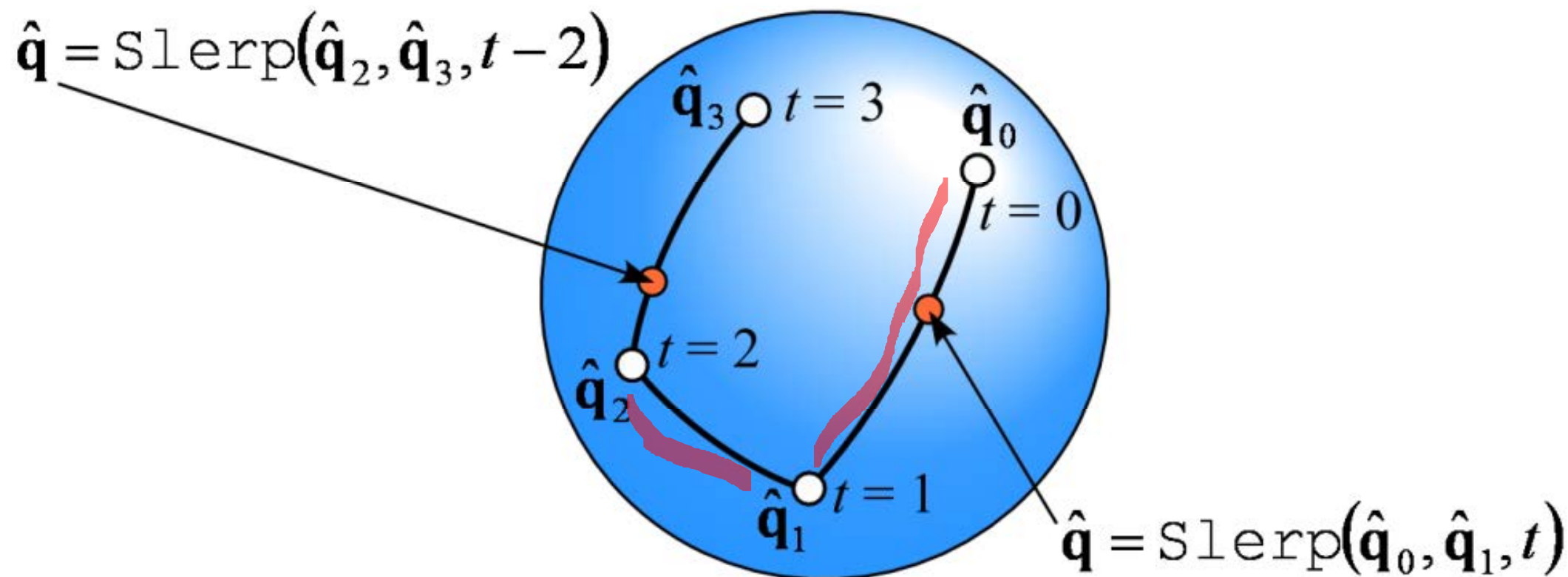
/*
** p ← q と r を t で補間
*/
void slerp(float *p,
           const float *q,
           const float *r,
           const float t)
{
    const float qr(q[0] * r[0]
                  + q[1] * r[1]
                  + q[2] * r[2]
                  + q[3] * r[3]);
    const float ss(1.0f - qr * qr);
```

```
    if (ss == 0.0) {
        p[0] = q[0];
        p[1] = q[1];
        p[2] = q[2];
        p[3] = q[3];
    }
    else {
        const float sp(sqrt(ss));
        const float ph(acos(qr));
        const float pt(ph * t);
        const float t1(sin(pt) / sp);
        const float t0(sin(ph - pt) / sp);

        p[0] = q[0] * t0 + r[0] * t1;
        p[1] = q[1] * t0 + r[1] * t1;
        p[2] = q[2] * t0 + r[2] * t1;
        p[3] = q[3] * t0 + r[3] * t1;
    }
}
```



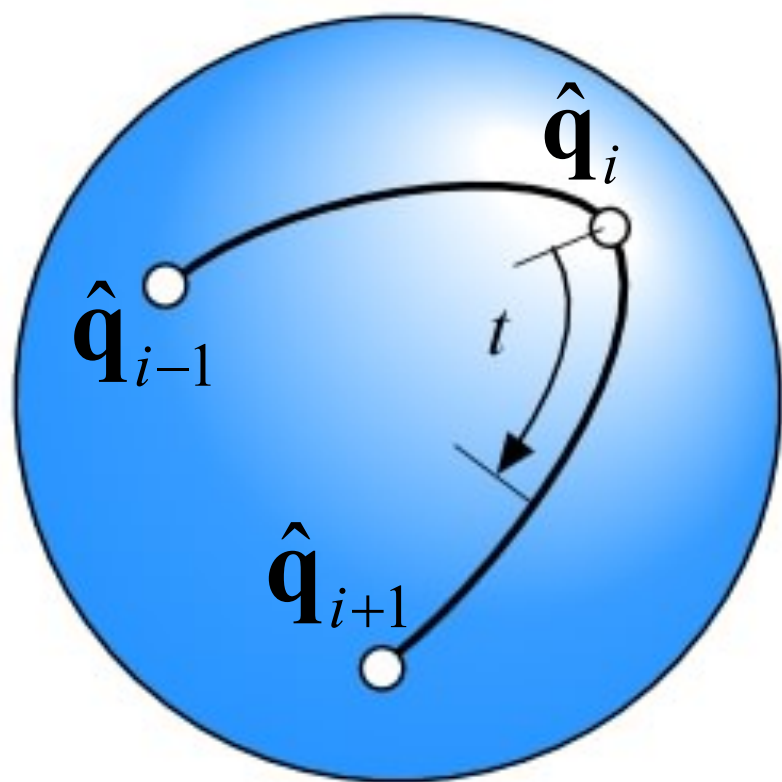
3 個以上の四元数の補間



区間 $[\hat{\mathbf{q}}_i, \hat{\mathbf{q}}_{i+1}]$ において $\hat{\mathbf{q}} = \text{Slerp}(\hat{\mathbf{q}}_i, \hat{\mathbf{q}}_{i+1}, t-i)$



四元数のスプライン補間



$$\hat{\mathbf{a}}_i = \hat{\mathbf{q}}_i \exp \left\{ - \frac{\log(\hat{\mathbf{q}}_i^{-1} \hat{\mathbf{q}}_{i-1}) + \log(\hat{\mathbf{q}}_i^{-1} \hat{\mathbf{q}}_{i+1})}{4} \right\}$$

$$\text{squad}(\hat{\mathbf{q}}_i, \hat{\mathbf{q}}_{i+1}, \hat{\mathbf{a}}_i, \hat{\mathbf{a}}_{i+1}, t) = \\ \text{slerp}(\text{slerp}(\hat{\mathbf{q}}_i, \hat{\mathbf{q}}_{i+1}, t), \text{slerp}(\hat{\mathbf{a}}_i, \hat{\mathbf{a}}_{i+1}, t), 2t(1-t))$$



宿題

- 球面線形補間を使ってアニメーションに回転のアニメーションを加えてください
 - 次のプログラムは線画の立方体を平行移動するアニメーションを表示します
 - <https://github.com/tokoik/ggsample04>
 - この起点で立方体を $(1, 0, 0)$ を軸に 1 ラジアン回転し, そこから終点において $(0, 0, 1)$ を軸に 2 ラジアン回転した状態に至る回転のアニメーションを加えてください
 - 軸と回転角から単位四元数を求める関数を作成してください
 - 単位四元数を球面線形補間する関数を作成してください
 - 単位四元数から回転変換行列を求める関数を作成してください
- ggsample04.cpp を**アップロード**してください

結果

このような画像が表示されれば、多分、正解です。

