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

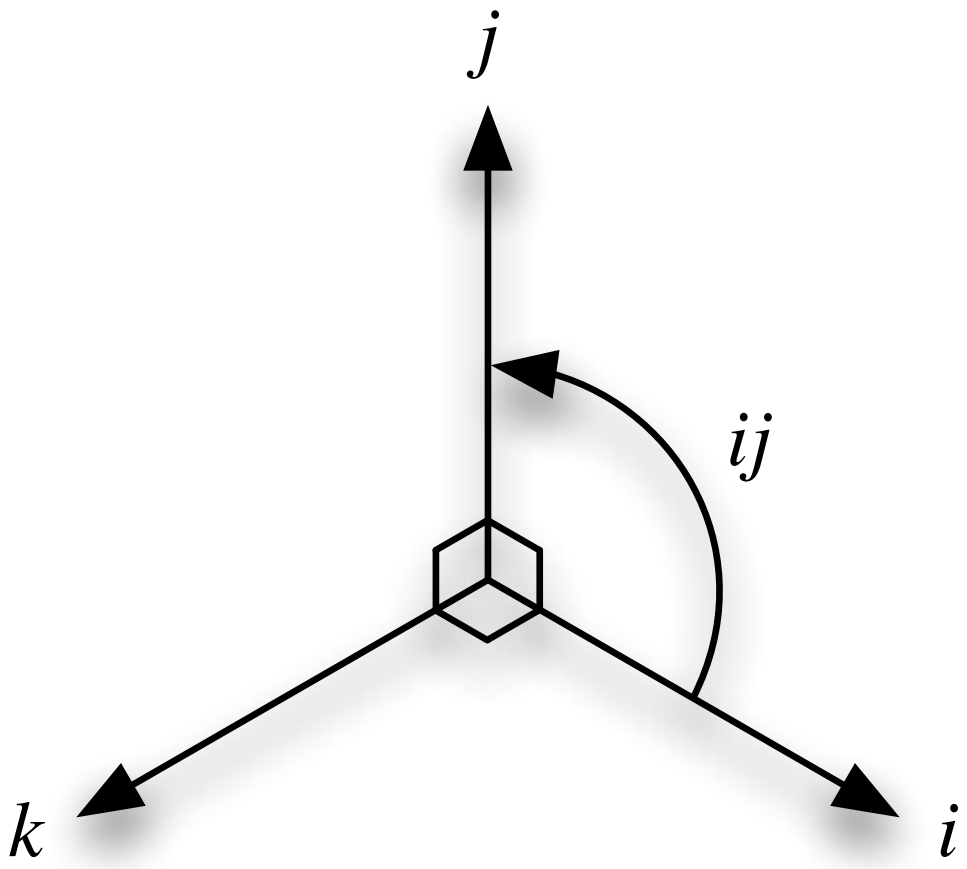
四元数（クォータニオン）

- 複素数の虚部を三次元に拡張したもの
  - $\hat{\mathbf{q}} = (q_x, q_y, q_z, q_w), \mathbf{q}_v = (q_x, q_y, q_z) \Rightarrow \hat{\mathbf{q}} = (\mathbf{q}_v, q_w)$

- 定義
  - $\hat{\mathbf{q}} = (\mathbf{q}_v, q_w) = iq_x + jq_y + kq_z + q_w = \mathbf{q}_v + q_w$
  - $\mathbf{q}_v = iq_x + jq_y + kq_z = (q_x, q_y, q_z)$
  - $i^2 = j^2 = k^2 = ijk = -1$
  - $jk = -kj = i, ki = -ik = j, ij = -ji = k$

q_w: 実部
$\mathbf{q}_v$: 虚部
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

$$\hat{\mathbf{q}}\hat{\mathbf{r}} = \left(iq_x + jq_y + kq_z + q_w\right)\left(ir_x + jr_y + kr_z + r_w\right)$$

$$= i\left(q_y r_z - q_z r_y + r_w q_x + q_w r_x\right)$$

$$+ j\left(q_z r_x - q_x r_z + r_w q_y + q_w r_y\right)$$

$$+ k\left(q_x r_y - q_y r_x + r_w q_z + q_w r_z\right)$$

$$+ q_w r_w - q_x r_x - q_y r_y - q_z r_z$$

$$= \left(\mathbf{q}_v \times \mathbf{r}_v + r_w \mathbf{q}_v + q_w \mathbf{r}_v, q_w r_w - \mathbf{q}_v \cdot \mathbf{r}_v\right)$$

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
  - $\hat{\mathbf{q}} + \hat{\mathbf{r}} = (\mathbf{q}_v, q_w) + (\mathbf{r}_v, r_w) = (\mathbf{q}_v + \mathbf{r}_v, q_w + r_w)$
- 共役
  - $\hat{\mathbf{q}}^* = (\mathbf{q}_v, q_w)^* = (-\mathbf{q}_v, q_w)$
- ノルム
  - $n(\hat{\mathbf{q}}) = \sqrt{\hat{\mathbf{q}}\hat{\mathbf{q}}^*} = \sqrt{\hat{\mathbf{q}}^*\hat{\mathbf{q}}} = \sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2}$
- 単位元
  - $\hat{\mathbf{i}} = (\mathbf{0}, 1)$

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

- $\hat{\mathbf{q}} = (\mathbf{q}_v, q_w)$ がノルムが 1 の四元数のとき
  - $n(\hat{\mathbf{q}}) = 1 \Longrightarrow \hat{\mathbf{q}} = \left(\sin\phi\,\mathbf{u}_q, \cos\phi\right) = \sin\phi\,\mathbf{u}_q + \cos\phi$
  - $\mathbf{u}_q$ は $\|\mathbf{u}_q\| = 1$ のベクトル（単位ベクトル）

- ここで
  - $n(\hat{\mathbf{q}}) = n\left(\sin\phi\,\mathbf{u}_q, \cos\phi\right) = \sqrt{\sin^2\phi\left(\mathbf{u}_q \cdot \mathbf{u}_q\right) + \cos^2\phi} = 1$

- $\cos\phi + i\sin\phi = e^{i\phi}$ 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
  - $\log(\hat{\mathbf{q}}) = \log\left(e^{\phi \mathbf{u}_q}\right) = \phi \mathbf{u}_q$

軸に回転角をかけたもの

- 指数
  - $\hat{\mathbf{q}}^t = \left(\sin\phi\, \mathbf{u}_q, \cos\phi\right)^t = e^{\phi t \mathbf{u}_q} = \sin\phi t\, \mathbf{u}_q, \cos\phi t$

回転角の指数倍

単位四元数のべき乗は
回転角を指数倍したもの

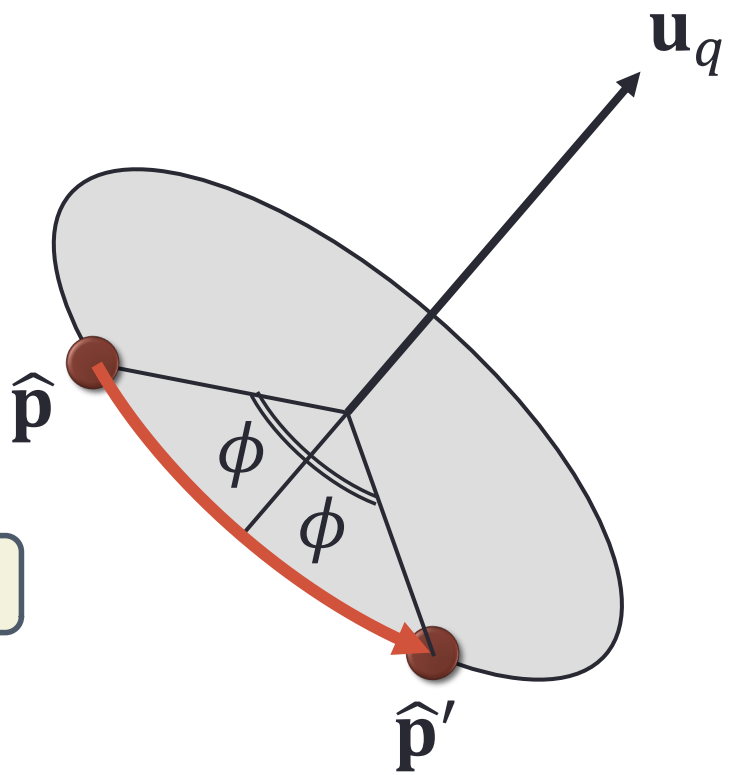
単位四元数による変換

- 点またはベクトル $\mathbf{p}$
  - $\mathbf{p} = (p_x, p_y, p_z, p_w)$
- 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
  - $\hat{\mathbf{p}}' = \hat{\mathbf{q}}\hat{\mathbf{p}}\hat{\mathbf{q}}^{-1}$

$\mathbf{u}_q$ を軸に $\mathbf{p}$ を 2ϕ 回転

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

$\hat{\mathbf{q}}$ 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);
  }
}
```

課題

- ベクトル $(u_x, u_y, u_z) = (0, 1, 0)$ を軸に $\pi/2$ 回転する回転を表す四元数 $\hat{\mathbf{q}}$ を求めなさい．$\cos\frac{\pi}{4} = \sin\frac{\pi}{4} = \frac{1}{\sqrt{2}}$とする．

ヒント

$$\hat{\mathbf{q}} = iq_x + jq_y + kq_z + q_w = \sin\theta\left(iu_x + ju_y + ku_z\right) + \cos\theta$$

- ベクトル $\hat{\mathbf{p}} = (1, 0, 0, 0)$ を $\hat{\mathbf{q}}$ によって回転した結果 $\hat{\mathbf{r}}$ を求めなさい．

$$\hat{\mathbf{r}} = \hat{\mathbf{q}}\hat{\mathbf{p}}\hat{\mathbf{q}}^*$$

単位四元数による変換の合成

- $\hat{\mathbf{q}}$ 回転してから更に $\hat{\mathbf{r}}$ 回転する

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

- したがって単位四元数の積 $\hat{\mathbf{r}}\hat{\mathbf{q}}$ 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）

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

$$\mathbf{M}^q = \begin{pmatrix} m_{00}^q & m_{01}^q & m_{02}^q & m_{03}^q \\ m_{10}^q & m_{11}^q & m_{12}^q & m_{13}^q \\ m_{20}^q & m_{21}^q & m_{22}^q & m_{23}^q \\ m_{30}^q & m_{31}^q & m_{32}^q & m_{33}^q \end{pmatrix}$$

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

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

直交行列から四元数への変換

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

$$q_x = \frac{m^q_{21} - m^q_{12}}{4q_w}$$

$$q_y = \frac{m^q_{02} - m^q_{20}}{4q_w}$$

$$q_z = \frac{m^q_{10} - m^q_{01}}{4q_w}$$

あるいは

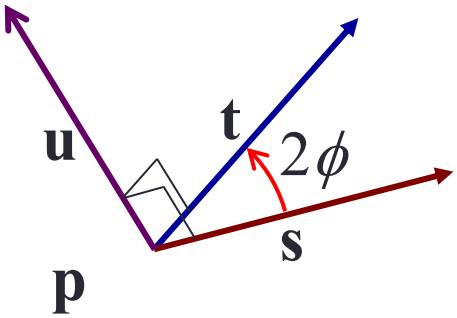
$$4q_x^2 = +m^q_{00} - m^q_{11} - m^q_{22} + m^q_{33}$$

$$4q_y^2 = -m^q_{00} + m^q_{11} - m^q_{22} + m^q_{33}$$

$$4q_z^2 = -m^q_{00} - m^q_{11} + m^q_{22} + m^q_{33}$$

四元数を用いたベクトルの回転

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



$$\mathbf{u} = \frac{\mathbf{s}\times\mathbf{t}}{\|\mathbf{s}\times\mathbf{t}\|} \Rightarrow \|\mathbf{s}\times\mathbf{t}\| = \sin 2\phi = 2\sin\phi\cos\phi$$

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

$$\mathbf{s}\cdot\mathbf{t} = \cos 2\phi = e$$

$$\cos\phi = \sqrt{\frac{1+e}{2}}$$

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

ベクトルの回転の行列表記

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

$$\mathbf{R}(\mathbf{s},\mathbf{t}) = \begin{pmatrix} e + h u_x^2 & h u_x u_y - u_z & h u_z u_x + u_y & 0 \\ h u_x u_y + u_z & e + h u_y^2 & h u_y u_z - u_x & 0 \\ h u_z u_x - u_y & h u_y u_z + u_x & e + h u_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{u} = \begin{pmatrix} u_x & u_y & u_z \end{pmatrix} = \frac{\mathbf{s} \times \mathbf{t}}{\|\mathbf{s} \times \mathbf{t}\|}$$

$$e = \cos 2\phi = \mathbf{s} \cdot \mathbf{t}$$

$$h = \frac{1 - \cos 2\phi}{\sin^2 2\phi} = \frac{1 - e}{\mathbf{u} \cdot \mathbf{u}}$$

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

$$\mathbf{M} = \begin{pmatrix} \bar{\mathbf{R}} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{R}} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix}$$

$\mathbf{I}$: 単位行列

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

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

(オイラー変換を使う場合)

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

$$\Downarrow$$

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

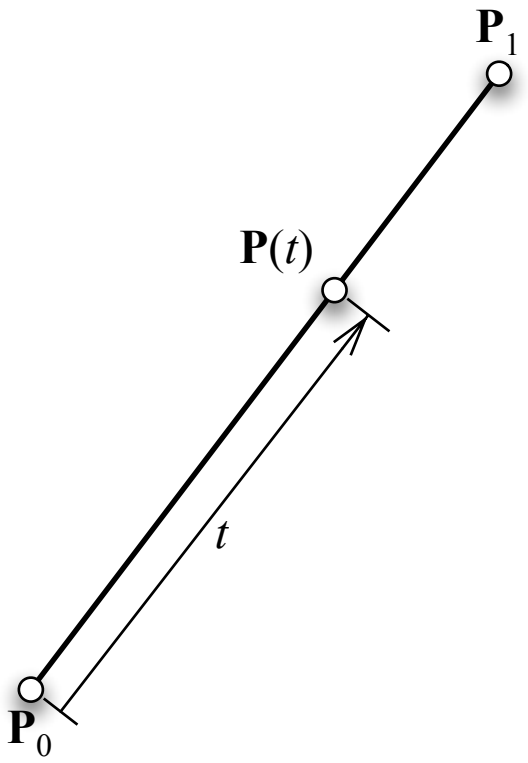
t: 時刻

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



$$\mathbf{P}(t) = \mathbf{P}_0(1-t) + \mathbf{P}_1 t$$

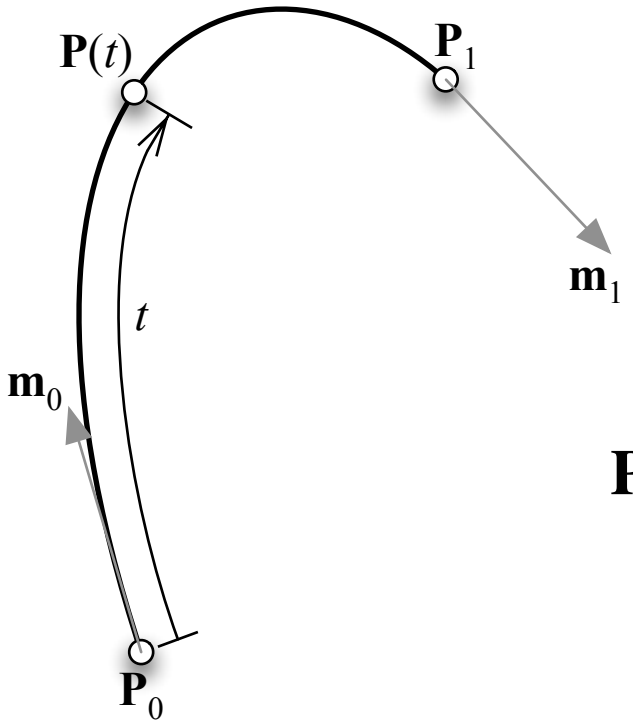
平行移動の変換行列を求める

```
/*
** 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 linearMotion(
  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



$$\mathbf{P}(t) = \mathbf{P}_0\left(2t^3 - 3t^2 + 1\right) + \mathbf{m}_0\left(t^3 - 2t^2 + t\right)$$
$$+\mathbf{P}_1\left(-2t^3 + 3t^2\right) + \mathbf{m}_1\left(t^3 - t^2\right)$$

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

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

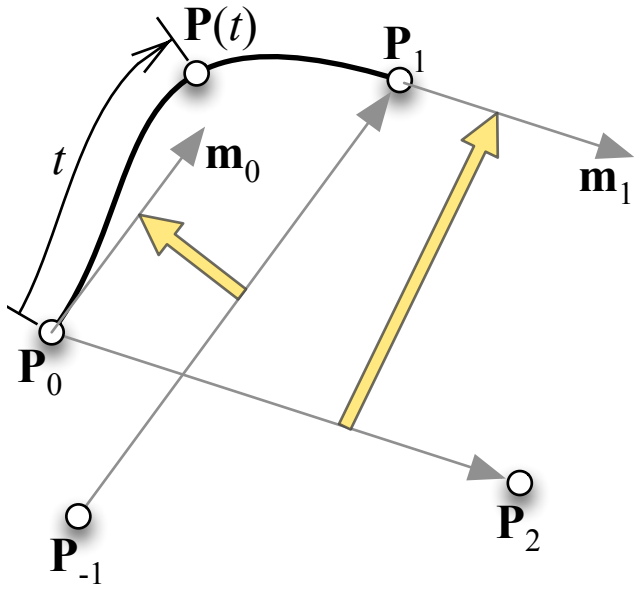
- $\mathbf{a}$を消去して

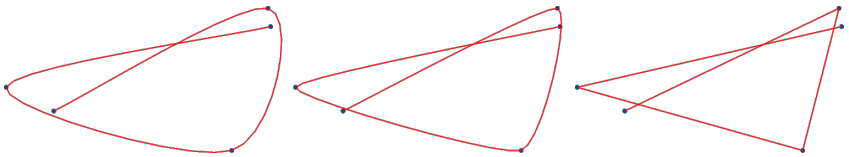
$$3\mathbf{p}_1 - \mathbf{m}_1 = \mathbf{b} + 2\mathbf{c} + 3\mathbf{d}$$
$$\mathbf{b} = 3\mathbf{p}_1 - \mathbf{m}_1 - 3\mathbf{p}_0 - 2\mathbf{m}_0$$
$$= 3(\mathbf{p}_1 - \mathbf{p}_0) - \mathbf{m}_1 - 2\mathbf{m}_0$$

- $\mathbf{b}$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





$c = 0$　　$c = 0.5$　　$c = 1$

- Cubic Hermite Spline において
  - $\mathbf{m}_0 = \frac{1}{2}(\mathbf{P}_1 - \mathbf{P}_{-1})$
  - $\mathbf{m}_1 = \frac{1}{2}(\mathbf{P}_2 - \mathbf{P}_0)$
- Cardinal Spline
  - $\mathbf{m}_0 = \frac{1-c}{2}(\mathbf{P}_1 - \mathbf{P}_{-1})$
  - $\mathbf{m}_1 = \frac{1-c}{2}(\mathbf{P}_2 - \mathbf{P}_0)$
  - c はテンション（張り）
    - $c = 1$：折れ線
    - $c = 0$：Catmull-Rom Spline

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

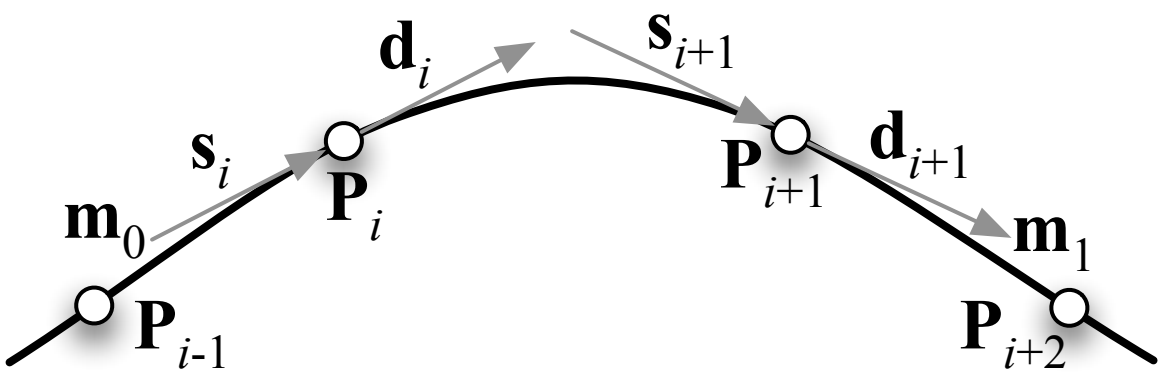
```
/*
** Catmull-Rom Spline による点列の補間
*/
void interpolate(
  float *p,
  const float *p0, const float *p1,
  const float *p2, const float *p3,
  float t)
{
  p[0] = catmull_rom(p0[0], p1[0], p2[0], p3[0], t);
  p[1] = catmull_rom(p0[1], p1[1], p2[1], p3[1], t);
  p[2] = catmull_rom(p0[2], p1[2], p2[2],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

Kochanek-Bartels Spline

- Cubic Hermite Spline において

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

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



t: Tension
b: Bias
c: Continuity

TBC Spline

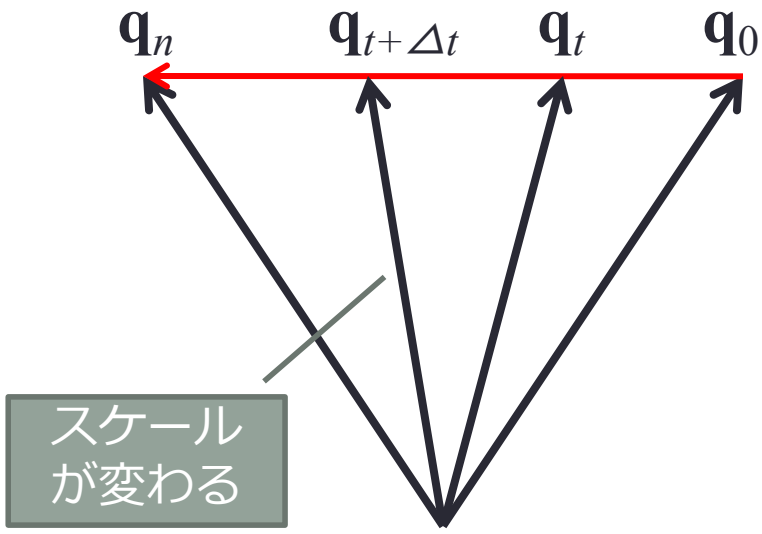
四元数の線形補間

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

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

$$\begin{aligned}\hat{\mathbf{q}}_{t+\Delta t} &= \left(1-(t+\Delta t)\right)\hat{\mathbf{q}}_0 + (t+\Delta t)\hat{\mathbf{q}}_n \\ &= (1-t)\hat{\mathbf{q}}_0 + t\hat{\mathbf{q}}_n + \Delta t\left(\hat{\mathbf{q}}_n - \hat{\mathbf{q}}_0\right) \\ &= \hat{\mathbf{q}}_t + \Delta\hat{\mathbf{q}}\end{aligned}$$

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

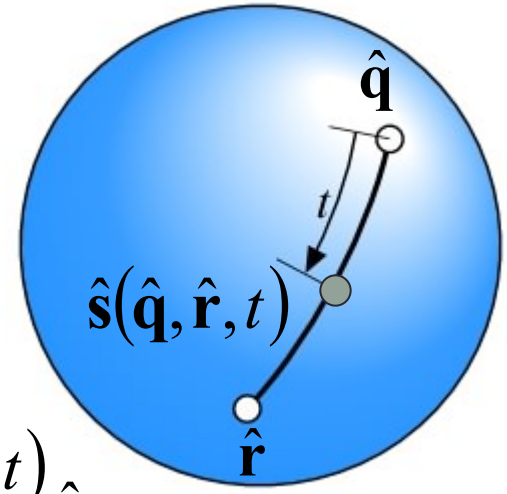


球面線形補間 (Slerp)

$$\hat{\mathbf{s}}(\hat{\mathbf{q}},\hat{\mathbf{r}},t)=\left(\hat{\mathbf{r}}\hat{\mathbf{q}}^{-1}\right)^{t}\hat{\mathbf{q}},\ t\in[0,1]$$

$$t\to 0\Rightarrow \hat{\mathbf{s}}(\hat{\mathbf{q}},\hat{\mathbf{r}},t)\to\hat{\mathbf{q}}$$

$$t\to 1\Rightarrow \hat{\mathbf{s}}(\hat{\mathbf{q}},\hat{\mathbf{r}},t)\to\hat{\mathbf{r}}$$



$$\hat{\mathbf{s}}(\hat{\mathbf{q}},\hat{\mathbf{r}},t)=\texttt{slerp}(\hat{\mathbf{q}},\hat{\mathbf{r}},t)=\frac{\sin(\phi(1-t))}{\sin\phi}\hat{\mathbf{q}}+\frac{\sin(\phi t)}{\sin\phi}\hat{\mathbf{r}}$$

$$\cos\phi=\hat{\mathbf{q}}\cdot\hat{\mathbf{r}}=q_x r_x+q_y r_y+q_z r_z+q_w r_w$$

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

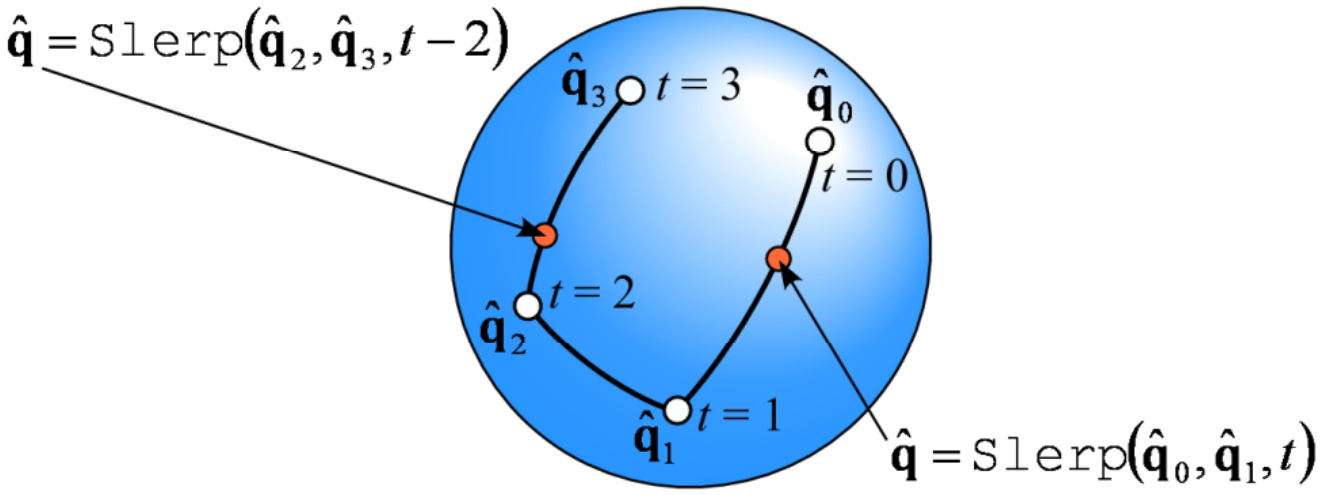
$$\sin\phi=\sqrt{1-(\hat{\mathbf{q}}\cdot\hat{\mathbf{r}})^2}$$

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);
```

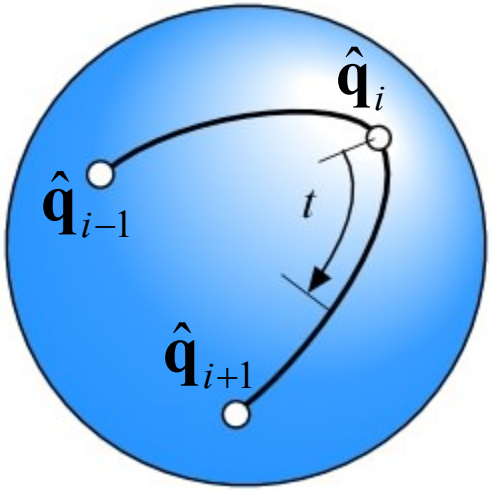
```
  if (ss == 0.0) {
    p[0] = q[0];
    p[1] = q[1];
    p[2] = q[2];
    p[3] = q[3];
  }
  else {
    const float sp(sqrt(ss));
    const float ph(acos(qr));
    const float pt(ph * t);
    const float t1(sin(pt) / sp);
    const float t0(sin(ph - pt) / sp);

    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;
  }
}
```

2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 $\hat{\mathbf{q}} = \mathtt{Slerp}(\hat{\mathbf{q}}_i, \hat{\mathbf{q}}_{i+1}, t-i)$

四元数のスプライン補間



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

$$\mathtt{squad}\left(\hat{\mathbf{q}}_i,\hat{\mathbf{q}}_{i+1},\hat{\mathbf{a}}_i,\hat{\mathbf{a}}_{i+1},t\right)=$$

$$\mathtt{slerp}\left(\mathtt{slerp}\left(\hat{\mathbf{q}}_i,\hat{\mathbf{q}}_{i+1},t\right),\mathtt{slerp}\left(\hat{\mathbf{a}}_i,\hat{\mathbf{a}}_{i+1},t\right),2t\left(1-t\right)\right)$$

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．