Universität Bielefeld

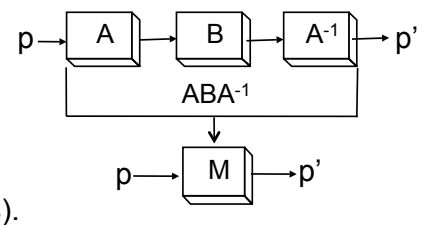
Realtime 3D Computer Graphics
Virtual Reality

Basic mathematics

Composite transforms
Alternative representations
Decomposition
Normal transformation

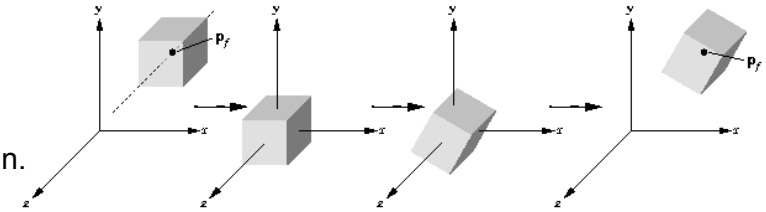
Composite Transformations

- Transforms may be done in sequence, hence they can be **concatenated**.
- This is a basis for
  - pipeline processing.
  - hierarchical CG data structures (scene graphs).



Example:

- Rotate an object about a point along the z axis:
  - Translate Object such that point is on the origin.
  - Rotate Object around the z axis.
  - Translate Object such that the point is back to its original location.



$$p' = (A^{-1}(B(Ap)) \Rightarrow p' = A^{-1}BAp$$

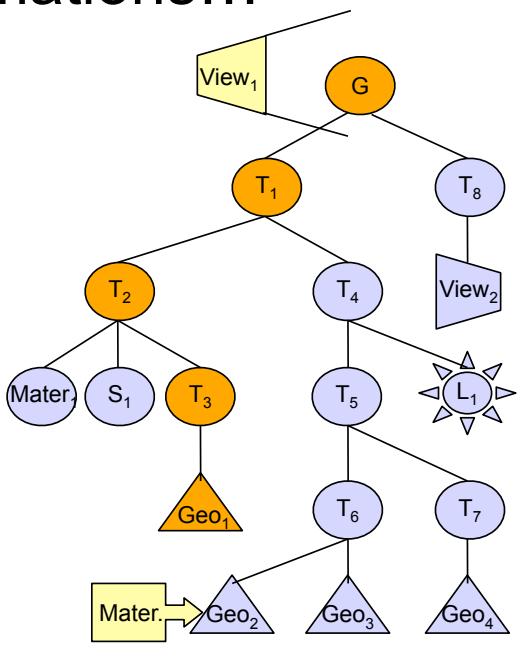
$$M = A^{-1}BA, p' = Mp$$

$$p' = A^{-1}BAp \neq ABA^{-1}p$$

Composite Transformations…

- …are the basis for scene graphs
- main node types:
  - group (G)
  - transform T_x (and group)
- Concatenation is performed during **traversals**
- Transforms represented as 4x4 matrices will be multiplied
- Geometry sent down the pipe will undergo combined transformation
- For Geo_1 to be w.r.t. the CS as defined by T1 (and hence G):

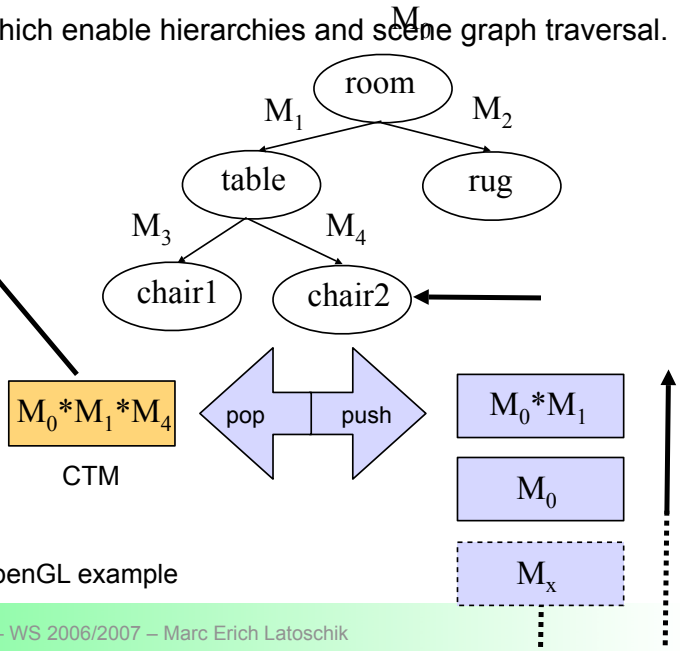
$$\forall p \in Geo_1 : T_1 * T_2 * T_3 * p$$



Composite Transformations

- Composite transformations using matrices is supported
  - by floating point units in the graphics hardware.
  - by low-level matrix stacks which enable hierarchies and scene graph traversal.

```
glLoadMatrixf(M0);
glPushMatrix();
glMultMatrixf(M1);
glPushMatrix();
glMultMatrixf(M4);
render chair2;
glPopMatrix();
glMultMatrixf(M3);
render chair1;
glPopMatrix();
render table;
glPopMatrix();
render room;
glPushMatrix();
glMultMatrixf(M2);
render rug;
```

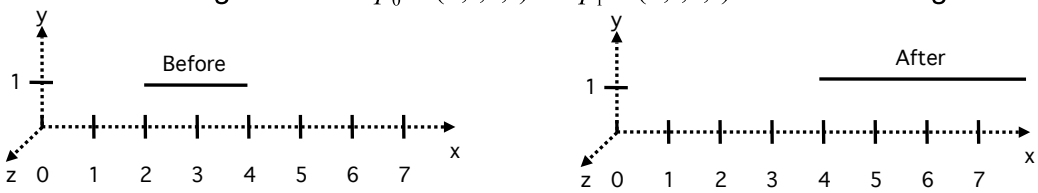


OpenGL example

Composite Transformations - Scaling

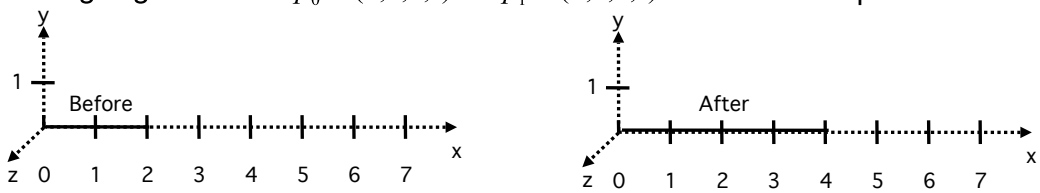
- <u>Observation 1:</u> The scale transformation also moves the object being scaled!

  Scale line segment from $p_0 = (2,1,0,1)^T$ to $p_1 = (4,1,0,1)^T$ to twice its length



- <u>Observation 2:</u>

  Scaling segment from $p_0 = (0,0,0,1)^T$ to $p_1 = (2,0,0,1)^T$ left-hand endpoint does not move.



  $(0,0,0,1)^T$ is known as a **fixed point** for the basic scaling transformation.

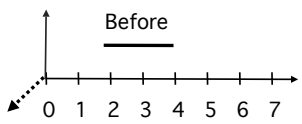
  - ➢ Composite transformations to create desired transformation behavior.

Composite Transformations Fixed Point Scaling

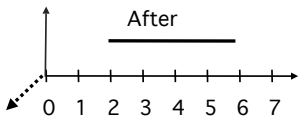
We can use composite transformations to create a scale transformation with **different fixed points**, e.g., scale by 2 with fixed point = $(2,1,0,1)^T$

1. Translate the point $(2,1,0,1)^T$ to the origin
2. Scale by 2
3. Translate origin to point $(2,1,0,1)^T$

$$\underbrace{\begin{bmatrix}1&0&0&2\\0&1&0&1\\0&0&1&0\\0&0&0&1\end{bmatrix}}_{T_{2,1}}\underbrace{\begin{bmatrix}2&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}}_{S_{2,1}}\underbrace{\begin{bmatrix}1&0&0&-2\\0&1&0&-1\\0&0&1&0\\0&0&0&1\end{bmatrix}}_{T_{-2,-1}}=\underbrace{\begin{bmatrix}2&0&0&-2\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}}_{C}$$



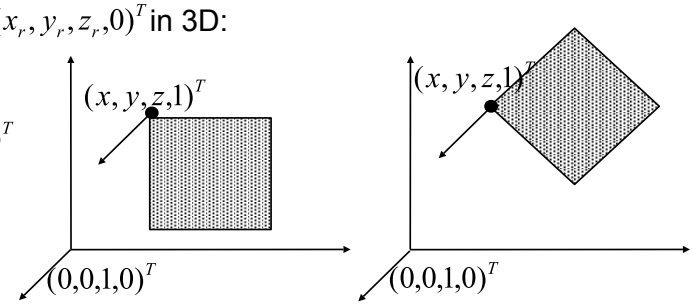
$$\underbrace{\begin{bmatrix}2&0&0&-2\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}}_{C}\begin{bmatrix}2\\1\\0\\1\end{bmatrix}=\begin{bmatrix}2\\1\\0\\1\end{bmatrix}\qquad\underbrace{\begin{bmatrix}2&0&0&-2\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}}_{C}\begin{bmatrix}4\\1\\0\\1\end{bmatrix}=\begin{bmatrix}6\\1\\0\\1\end{bmatrix}$$



Composite Transformations
Fixed Point/Axis Rotation

Rotation of θ degrees about point $(x, y, 1)^T$ in 2D generalizes to a rotation about a point $(x, y, z, 1)^T$ and an axis $(x_r, y_r, z_r, 0)^T$ in 3D:

1. Translate $(x, y, z, 1)^T$ to origin
2. Rotate around axis $(x_r, y_r, z_r, 0)^T$
3. Translate origin to $(x, y, z, 1)^T$

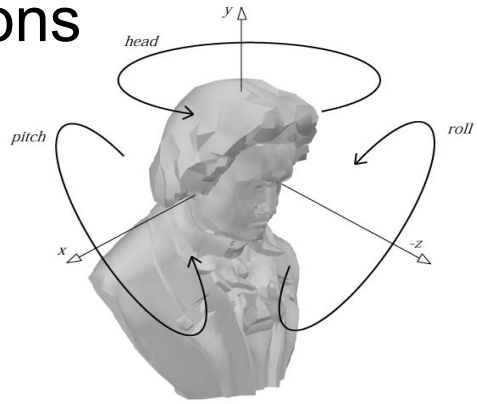


Example for:

$(x, y, z, 1)^T = (1,3,0,1)^T$
$(x_r, y_r, z_r, 0)^T = (0,0,1,0)^T$

$$C = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{T_{x,y,z}} \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{R_{z,\theta}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{T_{-x,-y,-z}}$$

Composite Transformations
The Euler Transform



Assume, that the view looks

1. **down** the negative z-direction with
2. **up** in the positive y-direction and
3. **right** in the positive x-direction

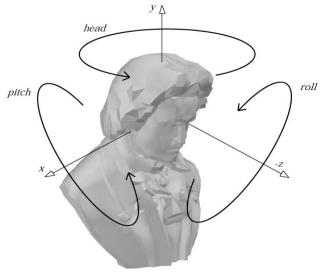
$$\mathbf{E}(h, p, r) = \mathbf{R}_z(r)\mathbf{R}_x(p)\mathbf{R}_y(h)$$

h=head, *p*=pitch, *r*=roll

Example: p=π/2: $\mathbf{E}(h, \frac{\pi}{2}, r) = \mathbf{R}_z(r)\mathbf{R}_x(\frac{\pi}{2})\mathbf{R}_y(h)$

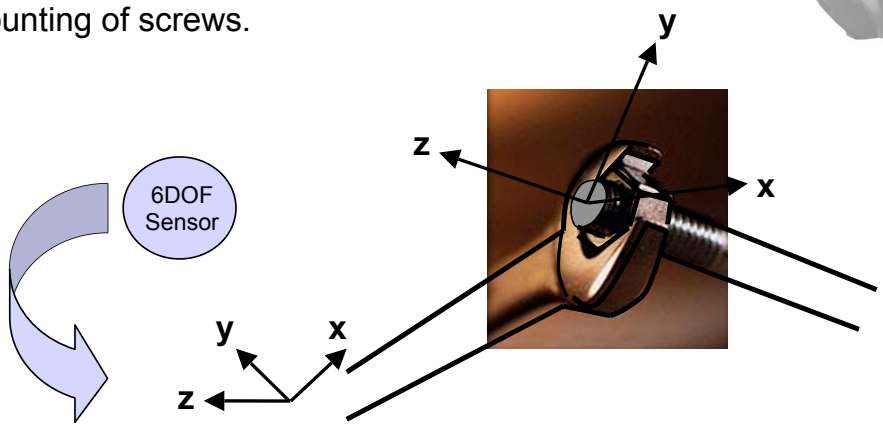
$$\mathbf{E}(h, \frac{\pi}{2}, r) = \begin{bmatrix} \cos r & -\sin r & 0 & 0 \\ \sin r & \cos r & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\frac{\pi}{2} & -\sin\frac{\pi}{2} & 0 \\ 0 & \sin\frac{\pi}{2} & \cos\frac{\pi}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos h & 0 & \sin h & 0 \\ 0 & 1 & 0 & 0 \\ -\sin h & 0 & \cos h & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composite Transformations
The Euler Transform

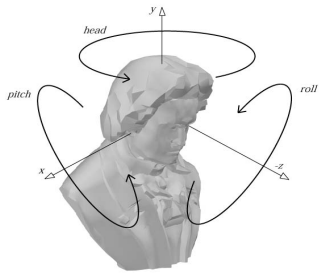


Example usage:

Limit a specific degree of rotational freedom to simulate physical behaviour, e.g., during mounting of screws.



Composite Transformations
The Euler Transform



$$\mathbf{E}(h,\frac{\pi}{2},r) = \mathbf{R}_z(r)\mathbf{R}_x(\frac{\pi}{2})\mathbf{R}_y(h)$$

$$= \begin{bmatrix} \cos r & -\sin r & 0 & 0 \\ \sin r & \cos r & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos h & 0 & \sin h & 0 \\ 0 & 1 & 0 & 0 \\ -\sin h & 0 & \cos h & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos r & -\sin r & 0 & 0 \\ \sin r & \cos r & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos h & 0 & \sin h & 0 \\ \sin h & 0 & -\cos h & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos r\cos h - \sin r\sin h & 0 & \cos r\sin h + \sin r\cos h & 0 \\ \sin r\cos h + \cos r\sin h & 0 & \sin r\sin h - \cos r\cos h & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

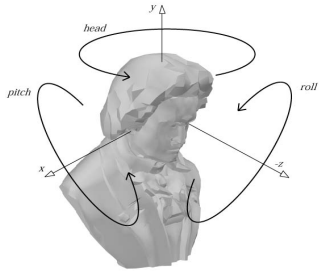
$$= \begin{bmatrix} \cos(r+h) & 0 & \sin(r+h) & 0 \\ \sin(r+h) & 0 & -\cos(r+h) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This matrix is dependent on just one angle!

Gimbal lock: one degree of freedom is lost!

Example: p=π/2, then the *z*-rotation becomes rotation around *y*-axis.

Composite Transformations
The Euler Transform



Extracting parameters h,p,r from the orthogonal Euler transform (just using the upper 3x3):

$$\mathbf{F} = \begin{bmatrix} f_{00} & f_{01} & f_{02} \\ f_{10} & f_{11} & f_{12} \\ f_{20} & f_{21} & f_{22} \end{bmatrix} = \mathbf{R}_z(r)\mathbf{R}_x(p)\mathbf{R}_y(h) = \mathbf{E}(h,p,r)$$

$$\mathbf{F} = \begin{bmatrix} \cos r\cos h - \sin r\sin p\sin h & -\sin r\cos p & \cos r\sin h + \sin r\sin p\cosh \\ \sin r\cos h + \cos r\sin p\sin h & \cos r\cos p & \sin r\sin h - \cos r\sin p\cos h \\ -\cos p\sin h & \sin p & \cos p\cos h \end{bmatrix}$$

$$\Rightarrow f_{21} = \sin p \qquad \frac{f_{01}}{f_{11}} = \frac{-\sin r}{\cos r} = -\tan r \qquad \frac{f_{20}}{f_{22}} = \frac{-\sin h}{\cos h} = -\tan h$$

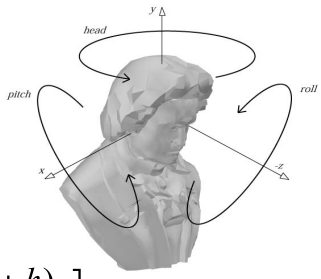
$$\boxed{\begin{aligned} h &= \operatorname{atan}2(f_{20}, f_{22}) \\ p &= \arcsin(f_{21}) \\ r &= \operatorname{atan}2(-f_{01}, f_{11}) \end{aligned}}$$

where atan2 is the C-math function which avoids divide by 0

$$-\frac{\pi}{2} < \arctan(x) < \frac{\pi}{2}$$

$$-\pi < \texttt{atan2(y,x)} < \pi; x = \frac{y}{x}, \text{iff } x \neq 0;$$

Composite Transformations
The Euler Transform



Extracting parameters h,p,r from the orthogonal Euler transform special case:

$\cos p = 0 \Rightarrow f_{01} = f_{11} = 0 \Rightarrow$ `atan2` can not be used!

$$\cos p = 0 \Rightarrow \sin p = \pm 1 \Rightarrow \mathbf{F} = \begin{bmatrix} \cos(r \pm h) & 0 & \sin(r \pm h) \\ \sin(r \pm h) & 0 & -\cos(r \pm h) \\ 0 & \pm 1 & 0 \end{bmatrix}$$

Extracting remaining parameters by arbitrarily setting:

$$h = 0 \Rightarrow \frac{\sin r}{\cos r} = \tan r = \frac{f_{10}}{f_{00}} \Rightarrow r = \texttt{atan2}\,(f_{10}, f_{00})$$

Remember:

$$y = \sin p \Leftrightarrow p = \arcsin(y),$$

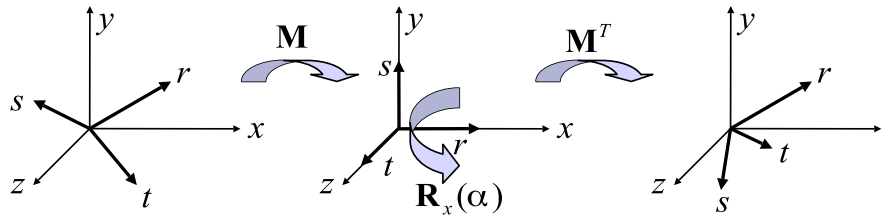
$$-1 \le y \le 1, -\frac{\pi}{2} \le p \le \frac{\pi}{2}$$

$\mathbf{F}$ with $p < -\frac{\pi}{2} \vee p > \frac{\pi}{2} \Rightarrow$

1. **Original parameter can not be extracted!**
2. **h,p,r are not unique!**

Transformations
Rotation about an arbitrary axis

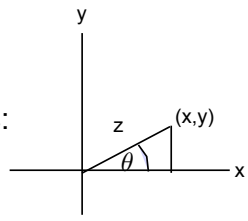
1. Assume rotation axis r which is normalized.
2. Rotate α around r:
3. Idea: Change bases, create base from r:



how to compute s,t?
s=?
t=?

$$\vec{s} = \begin{cases} (0,-r_z,r_y), \text{if } |r_x| < |r_y| \wedge |r_x| < |r_z| \\ (-r_z,0,r_x), \text{if } |r_y| < |r_x| \wedge |r_y| < |r_z| \\ (-r_y,r_x,0), \text{if } |r_z| < |r_x| \wedge |r_z| < |r_y| \end{cases} \qquad s = \frac{\vec{s}}{\|\vec{s}\|} \qquad t = r \times s \qquad \mathbf{M} = \begin{bmatrix} r^T \\ s^T \\ t^T \end{bmatrix} \qquad \mathbf{X} = \mathbf{M}^T \mathbf{R}_x(\alpha) \mathbf{M}$$

Transformations
Rotation - quaternions



- Based on usage of complex numbers in 2D to express rotations:

$$z = x + yi = r(cos\theta + sin\theta)$$

- Let there be the pure imaginary number: $i, i^2 = -1$
- Recall Euler's identity: $e^{i\theta} = \cos\theta + i\sin\theta$
- then the polar representation of a complex number c is: $c = a + ib = re^{i\theta}$
- where $r = \sqrt{a^2 + b^2}, \theta = \tan^{-1} b/a$
- Rotating c about ϕ degrees about the origin using the polar representation:

$$c' = re^{i(\theta+\phi)} = re^{i\theta}e^{i\phi}$$

➡ $e^{i\phi}$ is a rotation operator in the complex plane (and an alternative to transformations)

- But for 3D rotations we need a direction (vector) and degree (scalar).
- A suitable mathematical object is a **quaternion**:

$$\boxed{a = (q_0, q_1, q_2, q_3) = (q_0, \mathbf{q}), \text{with } \mathbf{q} = (q_1, q_2, q_3)}$$

Transformations
Rotation - quaternions

- Quaternion operations are based on three “complex” numbers with

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

- which are analogous to unit vectors in 3D, hence for $a = (q_0, \mathbf{q})$

$$\mathbf{q} = (q_1 i + q_2 j + q_3 k)$$

- Then for two quaternions $a = (q_0, \mathbf{q}), b = (p_0, \mathbf{p})$ addition is given as

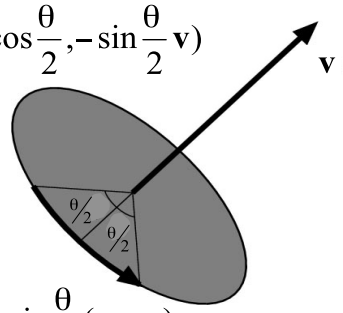
$$a + b = (p_0 + q_0, \mathbf{p} + \mathbf{q})$$

- and multiplication as $ab = (p_0 q_0 - \mathbf{q} \cdot \mathbf{p}, q_0 \mathbf{p} + p_0 \mathbf{q} + \mathbf{p} \times \mathbf{q})$
- with the multiplicative identity being $(1, \mathbf{0})$
- Quaternion magnitude is $|a|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2 = q_0^2 + \mathbf{q} \cdot \mathbf{q}$
- and the inverse as

$$a^{-1} = \frac{1}{|a|^2}(q_0, -\mathbf{q})$$

Transformations
Rotation - quaternions

- Relating the mathematical object to required geometric objects:
- Using the quaternion’s vector part as point in space: $p = (0, \mathbf{p}), \mathbf{p} = (x, y, z)$
- Consider (unit) quaternion: $r = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{v})$
- with $\mathbf{v}$ having unit length $|r| = 1$ and hence: $r^{-1} = (\cos\frac{\theta}{2}, -\sin\frac{\theta}{2}\mathbf{v})$
- Then the product of p and r results in the quaternion $p' = rpr^{-1}$ having the form $(0, \mathbf{p}')$ which **rotates the point p by θ degrees around $\mathbf{v}$!**



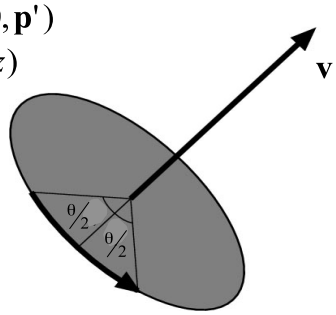
$$p' = \cos^2\frac{\theta}{2}\mathbf{p} + \sin^2\frac{\theta}{2}(\mathbf{p} \cdot \mathbf{v})\mathbf{v} + 2\sin\frac{\theta}{2}\cos\frac{\theta}{2}(\mathbf{v} \times \mathbf{p}) - \sin\frac{\theta}{2}(\mathbf{v} \times \mathbf{p}) \times \mathbf{v}$$

Transformations
Rotation - quaternions

- Example: Rotating around the z axis with fixed point at origin and angle θ:

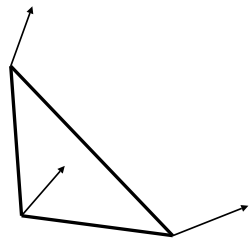
$$\mathbf{v} = (0,0,1) \Rightarrow r = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}(0,0,1)$$

- with $\mathbf{p} = (x, y, z) \Rightarrow p' = rpr^{-1} = r(0,\mathbf{p})r^{-1} = (0,\mathbf{p}')$
- where $\mathbf{p}' = (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta, z)$
- which is the same as $\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$



- ➢ Expected result with fewer operations!
- ➢ Avoids gimbal lock!
- ➢ Quaternions allow smooth interpolation between orientations.

Transformations
Normal transforms

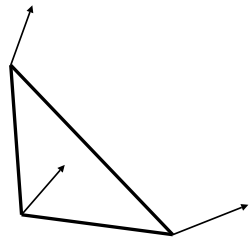


- Object descriptions involve normals for lighting and shading calculation.
- What happens if we apply (composite) transformation $\mathbf{M}$ to normals as we do it to vertices? Consider that $\mathbf{S}_x(0.5)$ is a part of $\mathbf{M}$:

1. Given a planar surface, a tangent vector is given by: $\vec{t}_1 = p_1 - p_0$
2. And hence a normal vector as: $\vec{n} = \vec{t}_1 \times \vec{t}_2$
3. Normal is orthogonal to tangent vectors by construction: $\vec{n} \cdot \vec{t} = \vec{n}\vec{t}^T = 0$
4. Transforming a surface using a matrix: $p' = \mathbf{M}p$
5. results in transforming the tangent vectors by the upper 3x3 matrix: $t' = \mathbf{M}_1 t$

$$\vec{n} \cdot \vec{t} = \vec{n}\mathbf{M}_1^{-1}\mathbf{M}_1\vec{t}^T = (\mathbf{M}_1^{-1T}\vec{n}^T) \cdot (\mathbf{M}_1\vec{t}^T) = \vec{n}' \cdot \vec{t}' = 0 \Rightarrow \vec{n}'^T = \mathbf{N}\vec{n}^T = \mathbf{M}_1^{-1T}\vec{n}^T$$

Transformations
Normal transforms



- Normal vectors are transformed by the transpose of the inverse of tangents transformation!
- Well known from tensor algebra where
  - $\vec{t}$ is called contravariant tensor of rank 1 and
  - $\vec{n}$ is the covariant tensor of rank 1.

- Normal transforms require $\mathbf{N} = (\mathbf{M}^{-1})^T$ instead of $\mathbf{M}$
- Know the matrix, Neo!
- Try to avoid calculation of invert!
  - For an orthogonal matrix: $\mathbf{R} : (\mathbf{R}^{-1})^T = \mathbf{R}$
  - Translations do not affect normal directions.
  - <u>Uniform</u> scaling only alters normals' lengths: can be renormalized if scaling is known.
  - Even in worse case only adjoint of upper 3x3 is needed (with later renormalization).