Realtime Computer Graphics on GPUs

Math

Jan Kolomazník

Department of Software and Computer Science Education
Faculty of Mathematics and Physics
Charles University in Prague

Computer Graphics Charles University

Vector Operations

Scalar (Dot) Product

- Definition:

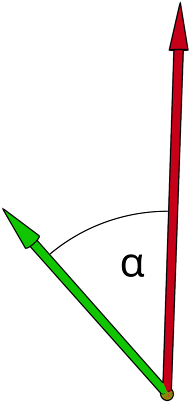
$$\mathbf{p} \cdot \mathbf{q} = \sum_i p_i g_i$$

- Value:

$$\mathbf{p} \cdot \mathbf{q} = ||\mathbf{p}|| \, ||\mathbf{q}|| \cos \alpha$$

- Matrix notation:

$$\mathbf{p} \cdot \mathbf{q} = \mathbf{p}^T \mathbf{q} = [p_0, ... p_{n-1}] \begin{bmatrix} q_0 \\ \vdots \\ q_{n-1} \end{bmatrix}$$



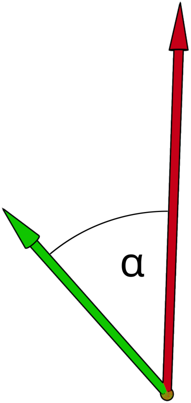
Scalar (Dot) Product

- Definition:

$$\mathbf{p} \cdot \mathbf{q} = \sum_i p_i g_i$$

- Value:

$$\mathbf{p} \cdot \mathbf{q} = \|\mathbf{p}\| \|\mathbf{q}\| \cos \alpha$$



- Matrix notation:

$$\mathbf{p} \cdot \mathbf{q} = \mathbf{p}^T \mathbf{q} = [p_0, \ldots p_{n-1}] \begin{bmatrix} q_0 \\ \vdots \\ q_{n-1} \end{bmatrix}$$

Scalar (Dot) Product

- Definition:

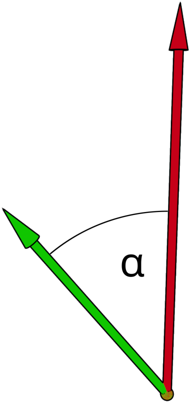
$$\mathbf{p} \cdot \mathbf{q} = \sum_i p_i g_i$$

- Value:

$$\mathbf{p} \cdot \mathbf{q} = \|\mathbf{p}\| \|\mathbf{q}\| \cos \alpha$$

- Matrix notation:

$$\mathbf{p} \cdot \mathbf{q} = \mathbf{p}^T \mathbf{q} = [p_0, \ldots p_{n-1}] \begin{bmatrix} q_0 \\ \vdots \\ q_{n-1} \end{bmatrix}$$



Vector Projection

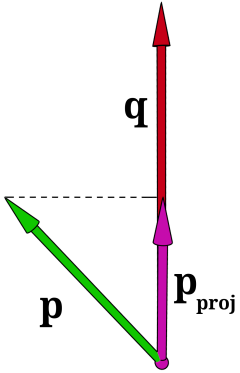
- Projection on another vector:

$$\mathbf{p}_{proj} = \frac{\mathbf{p}\cdot\mathbf{q}}{\|\mathbf{q}\|}\mathbf{q}$$

- Matrix notation ($\mathbf{q}\mathbf{q}^T$):

$$\mathbf{p}_{proj} = \frac{1}{\|\mathbf{q}\|^2}\begin{bmatrix} q_x^2 & q_xq_y & q_xq_z \\ q_xq_y & q_y^2 & q_yq_z \\ q_xq_z & q_yq_z & q_z^2 \end{bmatrix}\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

- Useful for repeated projections, embedding in matrix expressions



VECTOR PROJECTION

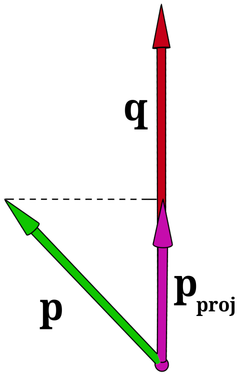
- Projection on another vector:

$$\mathbf{p}_{proj} = \frac{\mathbf{p} \cdot \mathbf{q}}{\|\mathbf{q}\|}\mathbf{q}$$

- Matrix notation ($\mathbf{q}\mathbf{q}^T$):

$$\mathbf{p}_{proj} = \frac{1}{\|\mathbf{q}\|^2} \begin{bmatrix} q_x^2 & q_x q_y & q_x q_z \\ q_x q_y & q_y^2 & q_y q_z \\ q_x q_z & q_y q_z & q_z^2 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

- Useful for repeated projections, embedding in matrix expressions



Cross Product

- Definition:

$$\mathbf{p} \times \mathbf{q} = [p_y q_z - p_z q_y, p_z q_x - p_x q_z, p_x q_y - p_y q_x]$$

- As formal determinant:

$$\mathbf{p} \times \mathbf{q} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p_x & p_y & p_z \\ q_x & q_y & q_z \end{vmatrix}$$

- Matrix formulation:

$$\mathbf{p} \times \mathbf{q} = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}$$

Cross Product

- Definition:

$$\mathbf{p} \times \mathbf{q} = [p_y q_z - p_z q_y, p_z q_x - p_x q_z, p_x q_y - p_y q_x]$$

- As formal determinant:

$$\mathbf{p} \times \mathbf{q} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p_x & p_y & p_z \\ q_x & q_y & q_z \end{vmatrix}$$

- Matrix formulation:

$$\mathbf{p} \times \mathbf{q} = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}$$

Cross Product

- Definition:

$$\mathbf{p} \times \mathbf{q} = [p_y q_z - p_z q_y, p_z q_x - p_x q_z, p_x q_y - p_y q_x]$$

- As formal determinant:

$$\mathbf{p} \times \mathbf{q} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ p_x & p_y & p_z \\ q_x & q_y & q_z \end{vmatrix}$$

- Matrix formulation:

$$\mathbf{p} \times \mathbf{q} = \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}$$

Cross Product II

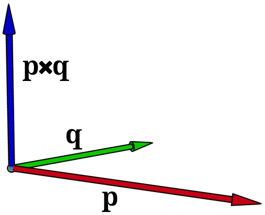
- Perpendicular to $\mathbf{p}, \mathbf{q}$:

$$(\mathbf{p} \times \mathbf{q}) \cdot \mathbf{p} = (\mathbf{p} \times \mathbf{q}) \cdot \mathbf{q} = 0$$

- Size:

$$\|\mathbf{p} \times \mathbf{q}\| = \|\mathbf{p}\| \|\mathbf{q}\| \sin \alpha$$

- Follows *right hand rule*



Rotations

2D ROTATION

- Basic expression:

$$\begin{aligned} x' &= x\cos\alpha - y\sin\alpha \\ y' &= x\sin\alpha + y\cos\alpha \end{aligned}$$

- Matrix notation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Complex exponential:

$$[x, y] \Rightarrow z = x + iy$$

  - Multiply by $e^{i\alpha} = \cos\alpha + i\sin\alpha$
  - Inverse rotation by complex conjugate

2D Rotation

- Basic expression:

$$\begin{aligned} x' &= x\cos\alpha - y\sin\alpha \\ y' &= x\sin\alpha + y\cos\alpha \end{aligned}$$

- Matrix notation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Complex exponential:

$$[x, y] \Rightarrow z = x + iy$$

  - Multiply by $e^{i\alpha} = \cos\alpha + i\sin\alpha$
  - Inverse rotation by complex conjugate

2D Rotation

- Basic expression:

$$\begin{aligned} x' &= x\cos\alpha - y\sin\alpha \\ y' &= x\sin\alpha + y\cos\alpha \end{aligned}$$

- Matrix notation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Complex exponential:

$$[x, y] \Rightarrow z = x + iy$$

  - Multiply by $e^{i\alpha} = \cos\alpha + i\sin\alpha$
  - Inverse rotation by complex conjugate

Elementary Rotations in 3D

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$\mathbf{R_y} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$\mathbf{R_z} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation Around Arbitrary Axis

- Axis $\mathbf{a}$, angle θ, point $\mathbf{p}$, rotated point p':

$$\|\mathbf{a}\| = 1$$

- Project $\mathbf{p}$ onto axis:

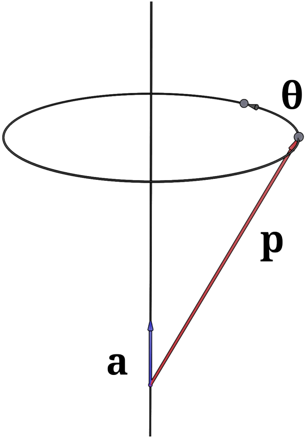
$$\begin{aligned} \mathbf{p}_{proj} &= (\mathbf{a} \cdot \mathbf{p})\mathbf{a} \\ R_{\mathbf{a},\theta}\mathbf{p}_{proj} &= \mathbf{p}_{proj} \end{aligned}$$

- Perpendicular component:

$$\begin{aligned} \mathbf{p}_{perp} &= \mathbf{p} - (\mathbf{a} \cdot \mathbf{p})\mathbf{a} \\ \|\mathbf{p}_{perp}\| &= \|\mathbf{p}\| \sin \alpha \end{aligned}$$

- Cross product with axis:

$$\begin{aligned} (\mathbf{a} \times \mathbf{p}) \cdot \mathbf{p}_{perp} &= 0 \\ \|\mathbf{a} \times \mathbf{p}\| &= \|\mathbf{p}\| \sin \alpha \end{aligned}$$



Rotation Around Arbitrary Axis

- Axis $\mathbf{a}$, angle θ, point $\mathbf{p}$, rotated point p':

$$\|\mathbf{a}\| = 1$$

- Project $\mathbf{p}$ onto axis:

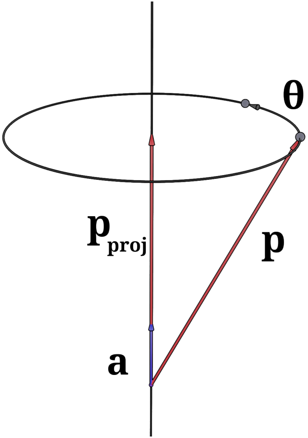
$$\begin{aligned} \mathbf{p}_{proj} &= (\mathbf{a} \cdot \mathbf{p})\mathbf{a} \\ R_{\mathbf{a},\theta}\mathbf{p}_{proj} &= \mathbf{p}_{proj} \end{aligned}$$

- Perpendicular component:

$$\begin{aligned} \mathbf{p}_{perp} &= \mathbf{p} - (\mathbf{a} \cdot \mathbf{p})\mathbf{a} \\ \|\mathbf{p}_{perp}\| &= \|\mathbf{p}\| \sin \alpha \end{aligned}$$

- Cross product with axis:

$$\begin{aligned} (\mathbf{a} \times \mathbf{p}) \cdot \mathbf{p}_{perp} &= 0 \\ \|\mathbf{a} \times \mathbf{p}\| &= \|\mathbf{p}\| \sin \alpha \end{aligned}$$



Rotation Around Arbitrary Axis

- Axis $\mathbf{a}$, angle θ, point $\mathbf{p}$, rotated point p':

$$\|\mathbf{a}\| = 1$$

- Project $\mathbf{p}$ onto axis:

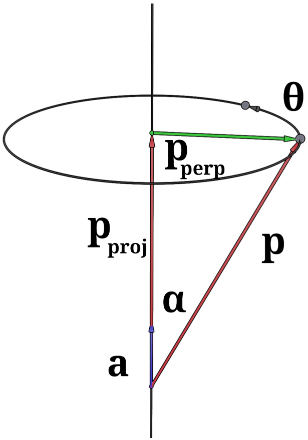
$$\begin{aligned} \mathbf{p}_{proj} &= (\mathbf{a} \cdot \mathbf{p})\mathbf{a} \\ R_{\mathbf{a},\theta}\mathbf{p}_{proj} &= \mathbf{p}_{proj} \end{aligned}$$

- Perpendicular component:

$$\begin{aligned} \mathbf{p}_{perp} &= \mathbf{p} - (\mathbf{a} \cdot \mathbf{p})\mathbf{a} \\ \|\mathbf{p}_{perp}\| &= \|\mathbf{p}\| \sin\alpha \end{aligned}$$

- Cross product with axis:

$$\begin{aligned} (\mathbf{a} \times \mathbf{p}) \cdot \mathbf{p}_{perp} &= 0 \\ \|\mathbf{a} \times \mathbf{p}\| &= \|\mathbf{p}\| \sin\alpha \end{aligned}$$



Rotation Around Arbitrary Axis

- Axis $\mathbf{a}$, angle θ, point $\mathbf{p}$, rotated point p':

$$\|\mathbf{a}\| = 1$$

- Project $\mathbf{p}$ onto axis:

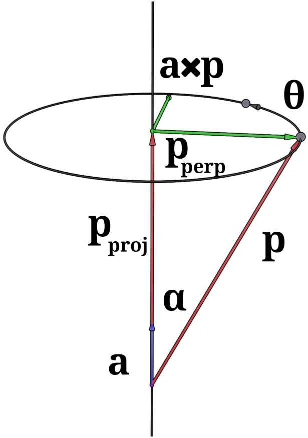
$$\begin{aligned} \mathbf{p}_{proj} &= (\mathbf{a} \cdot \mathbf{p})\mathbf{a} \\ R_{\mathbf{a},\theta}\mathbf{p}_{proj} &= \mathbf{p}_{proj} \end{aligned}$$

- Perpendicular component:

$$\begin{aligned} \mathbf{p}_{perp} &= \mathbf{p} - (\mathbf{a} \cdot \mathbf{p})\mathbf{a} \\ \|\mathbf{p}_{perp}\| &= \|\mathbf{p}\| \sin\alpha \end{aligned}$$

- Cross product with axis:

$$\begin{aligned} (\mathbf{a} \times \mathbf{p}) \cdot \mathbf{p}_{perp} &= 0 \\ \|\mathbf{a} \times \mathbf{p}\| &= \|\mathbf{p}\| \sin\alpha \end{aligned}$$



ROTATION AROUND ARBITRARY AXIS

- Axis $\mathbf{a}$, angle θ, point $\mathbf{p}$, rotated point p':

$$\|\mathbf{a}\| = 1$$

- Project $\mathbf{p}$ onto axis:

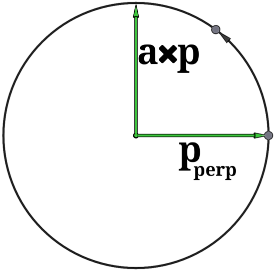
$$\begin{aligned} \mathbf{p}_{proj} &= (\mathbf{a} \cdot \mathbf{p})\mathbf{a} \\ R_{\mathbf{a},\theta}\mathbf{p}_{proj} &= \mathbf{p}_{proj} \end{aligned}$$

- Perpendicular component:

$$\begin{aligned} \mathbf{p}_{perp} &= \mathbf{p} - (\mathbf{a} \cdot \mathbf{p})\mathbf{a} \\ \|\mathbf{p}_{perp}\| &= \|\mathbf{p}\| \sin\alpha \end{aligned}$$

- Cross product with axis:

$$\begin{aligned} (\mathbf{a} \times \mathbf{p}) \cdot \mathbf{p}_{perp} &= 0 \\ \|\mathbf{a} \times \mathbf{p}\| &= \|\mathbf{p}\| \sin\alpha \end{aligned}$$



Rotation Around Arbitrary Axis II

- Final rotated position:

$$\begin{aligned} \mathbf{p}'_{perp} &= \mathbf{p}_{perp}\cos\theta + (\mathbf{a}\times\mathbf{p})\sin\theta \\ \mathbf{p}' &= \mathbf{p}'_{perp} + \mathbf{p}_{proj} \end{aligned} \tag{1}$$

- Matrix representation:

$$\begin{aligned} \mathbf{p}'_{perp} &= [\mathbf{p} - (\mathbf{a}\cdot\mathbf{p})\mathbf{a}]\cos\theta + (\mathbf{a}\times\mathbf{p})\sin\theta \\ &= \mathbf{p}\cos\theta + (\mathbf{a}\times\mathbf{p})\sin\theta + \mathbf{a}(\mathbf{a}\cdot\mathbf{p})(1-\cos\theta) \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\mathbf{p}\cos\theta + \begin{bmatrix} 0 & -A_z & A_y \\ A_z & 0 & -A_x \\ -A_y & A_x & 0 \end{bmatrix}\mathbf{p}\sin\theta \\ &+ \begin{bmatrix} A_x^2 & A_xA_y & A_xA_z \\ A_xA_y & A_y^2 & -A_yA_z \\ A_xA_z & A_yA_z & A_z^2 \end{bmatrix}\mathbf{p}(1-\cos\theta) \end{aligned}$$

Rotation Around Arbitrary Axis II

- Final rotated position:

$$\begin{aligned}\mathbf{p}'_{perp} &= \mathbf{p}_{perp}\cos\theta + (\mathbf{a}\times\mathbf{p})\sin\theta \\ \mathbf{p}' &= \mathbf{p}'_{perp} + \mathbf{p}_{proj}\end{aligned} \quad (1)$$

- Matrix representation:

$$\begin{aligned}\mathbf{p}'_{perp} &= [\mathbf{p} - (\mathbf{a}\cdot\mathbf{p})\mathbf{a}]\cos\theta + (\mathbf{a}\times\mathbf{p})\sin\theta \\ &= \mathbf{p}\cos\theta + (\mathbf{a}\times\mathbf{p})\sin\theta + \mathbf{a}(\mathbf{a}\cdot\mathbf{p})(1-\cos\theta) \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\mathbf{p}\cos\theta + \begin{bmatrix} 0 & -A_z & A_y \\ A_z & 0 & -A_x \\ -A_y & A_x & 0 \end{bmatrix}\mathbf{p}\sin\theta \\ &+ \begin{bmatrix} A_x^2 & A_xA_y & A_xA_z \\ A_xA_y & A_y^2 & -A_yA_z \\ A_xA_z & A_yA_z & A_z^2 \end{bmatrix}\mathbf{p}(1-\cos\theta)\end{aligned}$$

Rotation Around Arbitrary Axis III

- Final matrix form:

$$\begin{bmatrix} c+(1-c)A_x^2 & (1-c)A_xA_y - sA_z & (1-c)A_xA_z + sA_y \\ (1-c)A_xA_y + sA_z & c+(1-c)A_y^2 & (1-c)A_yA_z - sA_x \\ (1-c)A_xA_z - sA_y & (1-c)A_yA_z + sA_x & c+(1-c)A_z^2 \end{bmatrix}$$

Euler Angles

- arbitrary rotation decomposed into three components
- Leonard Euler (1707-1783)
- 3 angles – 3 elementary rotations
- order of rotations important (x-y-z, roll-pitch-yaw, z-x-z, ...)
  - intrinsic vs. extrinsics

EULER ANGLES II

Disadvantages:

- Problematic interpolation between two orientations
- Gimbal lock – not as severe in SW as in HW (Apollo)

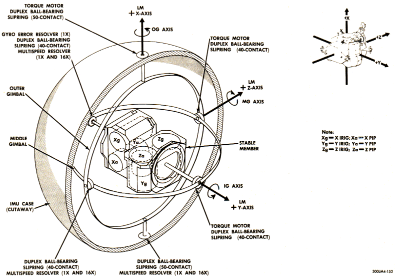


Figure 2.1-24. IMU Gimbal Assembly

Quaternions

- Sir William Rowan Hamilton, 16 Oct 1843 (Dublin)
- generalization of complex numbers in 4D space
- usage in graphics since 1985 (Shoemake)
- $\mathbf{q} = (\mathbf{v}, w) = ix + jy + kz + w = \mathbf{v} + w$
- imaginary part $v = (x, y, z) = ix + jy + kz$
- $i^2 = j^2 = k^2 = -1, jk = -kj = i, ki = -ik = j, ij = -ji = k$

Quaternions

- Sir William Rowan Hamilton, 16 Oct 1843 (Dublin)
- generalization of complex numbers in 4D space
- usage in graphics since 1985 (Shoemake)
- $\mathbf{q} = (\mathbf{v}, w) = ix + jy + kz + w = \mathbf{v} + w$
- imaginary part $v = (x, y, z) = ix + jy + kz$
- $i^2 = j^2 = k^2 = -1, jk = -kj = i, ki = -ik = j, ij = -ji = k$

Quaternions - Why 4D?

- 3D – what is $(ij) =?$

$$(i)(x+iy+jz) = -y+ix+(ij)z$$

- We need to introduce $ij = k$

$$(i)(ix+jy+kz+w) = -x+iw-jz+ky$$

QUATERNIONS - WHY 4D?

- 3D – what is $(ij) = ?$

$$(i)(x + iy + jz) = -y + ix + (ij)z$$

- We need to introduce $ij = k$

$$(i)(ix + jy + kz + w) = -x + iw - jz + ky$$

Quaternion Operations

- addition $(\mathbf{v}_1, w_1) + (\mathbf{v}_2, w_2) = (\mathbf{v}_1 + \mathbf{v}_2, w_1 + w_2)$
- multiplication $\mathbf{qr} = (\mathbf{v}_q \times \mathbf{v}_r + w_r\mathbf{v}_q + w_q\mathbf{v}_r, w_q w_r - \mathbf{v}_q \cdot \mathbf{v}_r)$
- multiplication by a scalar $s\mathbf{q} = (0, s)(\mathbf{v}, w) = (s\mathbf{v}, sw)$
- conjugation $(\mathbf{v}, w)^* = (-\mathbf{v}, w)$
- unit $id = (\mathbf{0}, 1)$
- norm (squared absolute value) $||\mathbf{q}||^2 = n(\mathbf{q}) = \mathbf{qq}^* = x^2 + y^2 + z^2 + w^2$
- reciprocal $\mathbf{q}^{-1} = \mathbf{q}^*/n(\mathbf{q})$

Quaternion Operations

- addition $(\mathbf{v}_1, w_1) + (\mathbf{v}_2, w_2) = (\mathbf{v}_1 + \mathbf{v}_2, w_1 + w_2)$
- multiplication $\mathbf{qr} = (\mathbf{v}_q \times \mathbf{v}_r + w_r \mathbf{v}_q + w_q \mathbf{v}_r, w_q w_r - \mathbf{v}_q \cdot \mathbf{v}_r)$
- multiplication by a scalar $s\mathbf{q} = (0, s)(\mathbf{v}, w) = (s\mathbf{v}, sw)$
- conjugation $(\mathbf{v}, w)^* = (-\mathbf{v}, w)$
- unit $id = (\mathbf{0}, 1)$
- norm (squared absolute value) $||\mathbf{q}||^2 = n(\mathbf{q}) = \mathbf{qq}^* = x^2 + y^2 + z^2 + w^2$
- reciprocal $\mathbf{q}^{-1} = \mathbf{q}^*/n(\mathbf{q})$

Quaternion Operations

- addition $(\mathbf{v}_1, w_1) + (\mathbf{v}_2, w_2) = (\mathbf{v}_1 + \mathbf{v}_2, w_1 + w_2)$
- multiplication $\mathbf{qr} = (\mathbf{v}_q \times \mathbf{v}_r + w_r\mathbf{v}_q + w_q\mathbf{v}_r, w_q w_r - \mathbf{v}_q \cdot \mathbf{v}_r)$
- multiplication by a scalar $s\mathbf{q} = (0, s)(\mathbf{v}, w) = (s\mathbf{v}, sw)$
- conjugation $(\mathbf{v}, w)^* = (-\mathbf{v}, w)$
- unit $id = (\mathbf{0}, 1)$
- norm (squared absolute value) $||\mathbf{q}||^2 = n(\mathbf{q}) = \mathbf{qq}^* = x^2 + y^2 + z^2 + w^2$
- reciprocal $\mathbf{q}^{-1} = \mathbf{q}^*/n(\mathbf{q})$

Quaternion Operations

- addition $(\mathbf{v}_1, w_1) + (\mathbf{v}_2, w_2) = (\mathbf{v}_1 + \mathbf{v}_2, w_1 + w_2)$
- multiplication $\mathbf{qr} = (\mathbf{v}_q \times \mathbf{v}_r + w_r\mathbf{v}_q + w_q\mathbf{v}_r, w_q w_r - \mathbf{v}_q \cdot \mathbf{v}_r)$
- multiplication by a scalar $s\mathbf{q} = (0, s)(\mathbf{v}, w) = (s\mathbf{v}, sw)$
- conjugation $(\mathbf{v}, w)^* = (-\mathbf{v}, w)$
- unit $id = (\mathbf{0}, 1)$
- norm (squared absolute value) $\|\mathbf{q}\|^2 = n(\mathbf{q}) = \mathbf{qq}^* = x^2 + y^2 + z^2 + w^2$
- reciprocal $\mathbf{q}^{-1} = \mathbf{q}^*/n(\mathbf{q})$

Quaternion Operations

- addition $(\mathbf{v}_1, w_1) + (\mathbf{v}_2, w_2) = (\mathbf{v}_1 + \mathbf{v}_2, w_1 + w_2)$
- multiplication $\mathbf{qr} = (\mathbf{v}_q \times \mathbf{v}_r + w_r \mathbf{v}_q + w_q \mathbf{v}_r, w_q w_r - \mathbf{v}_q \cdot \mathbf{v}_r)$
- multiplication by a scalar $s\mathbf{q} = (0, s)(\mathbf{v}, w) = (s\mathbf{v}, sw)$
- conjugation $(\mathbf{v}, w)^* = (-\mathbf{v}, w)$
- unit $id = (\mathbf{0}, 1)$
- norm (squared absolute value) $\|\mathbf{q}\|^2 = n(\mathbf{q}) = \mathbf{q}\mathbf{q}^* = x^2 + y^2 + z^2 + w^2$
- reciprocal $\mathbf{q}^{-1} = \mathbf{q}^*/n(\mathbf{q})$

Quaternion Operations

- addition $(\mathbf{v}_1, w_1) + (\mathbf{v}_2, w_2) = (\mathbf{v}_1 + \mathbf{v}_2, w_1 + w_2)$
- multiplication $\mathbf{qr} = (\mathbf{v}_q \times \mathbf{v}_r + w_r\mathbf{v}_q + w_q\mathbf{v}_r, w_q w_r - \mathbf{v}_q \cdot \mathbf{v}_r)$
- multiplication by a scalar $s\mathbf{q} = (0, s)(\mathbf{v}, w) = (s\mathbf{v}, sw)$
- conjugation $(\mathbf{v}, w)^* = (-\mathbf{v}, w)$
- unit $id = (\mathbf{0}, 1)$
- norm (squared absolute value) $\|\mathbf{q}\|^2 = n(\mathbf{q}) = \mathbf{qq}^* = x^2 + y^2 + z^2 + w^2$
- reciprocal $\mathbf{q}^{-1} = \mathbf{q}^*/n(\mathbf{q})$

Quaternion Operations II

- unit quaternion can be expressed by goniometry as $\mathbf{q} = (\mathbf{u}_q \sin\theta, \cos\theta)$
- for some unit 3D vector $\mathbf{u}_q$ it represents a rotation (orientation) in 3D
  - ambiguity: both $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation!
- identity (zero rotation): $(0, 1)$
- power, exponential, logarithm:
  $\mathbf{q} = \mathbf{u}_q \sin\theta + \cos\theta = exp(\theta\mathbf{u}_q), log\mathbf{q} = \theta\mathbf{u}_q$
  $\mathbf{q}^t = (\mathbf{u}_q \sin\theta + cos\theta)^t = exp(t\theta\mathbf{u}_q) = \mathbf{u}_q \sin t\theta + \cos t\theta$

Quaternion Operations II

- unit quaternion can be expressed by goniometry as $\mathbf{q} = (\mathbf{u}_q \sin\theta, \cos\theta)$
- for some unit 3D vector $\mathbf{u}_q$ it represents a rotation (orientation) in 3D
  - ambiguity: both $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation!
- identity (zero rotation): $(0, 1)$
- power, exponential, logarithm:
  $\mathbf{q} = \mathbf{u}_q \sin\theta + \cos\theta = exp(\theta\mathbf{u}_q), log\mathbf{q} = \theta\mathbf{u}_q$
  $\mathbf{q}^t = (\mathbf{u}_q \sin\theta + cos\theta)^t = exp(t\theta\mathbf{u}_q) = \mathbf{u}_q \sin t\theta + \cos t\theta$

Quaternion Operations II

- unit quaternion can be expressed by goniometry as $\mathbf{q} = (\mathbf{u}_q \sin\theta, \cos\theta)$
- for some unit 3D vector $\mathbf{u}_q$ it represents a rotation (orientation) in 3D
  - ambiguity: both $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation!
- identity (zero rotation): $(0, 1)$
- power, exponential, logarithm:
  $\mathbf{q} = \mathbf{u}_q \sin\theta + \cos\theta = exp(\theta\mathbf{u}_q), log\mathbf{q} = \theta\mathbf{u}_q$
  $\mathbf{q}^t = (\mathbf{u}_q \sin\theta + cos\theta)^t = exp(t\theta\mathbf{u}_q) = \mathbf{u}_q \sin t\theta + \cos t\theta$

Quaternion Operations II

- unit quaternion can be expressed by goniometry as $\mathbf{q} = (\mathbf{u}_q \sin\theta, \cos\theta)$
- for some unit 3D vector $\mathbf{u}_q$ it represents a rotation (orientation) in 3D
  - ambiguity: both $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation!
- identity (zero rotation): $(0, 1)$
- power, exponential, logarithm:
  $\mathbf{q} = \mathbf{u}_q \sin\theta + \cos\theta = exp(\theta\mathbf{u}_q), log\mathbf{q} = \theta\mathbf{u}_q$
  $\mathbf{q}^t = (\mathbf{u}_q \sin\theta + cos\theta)^t = exp(t\theta\mathbf{u}_q) = \mathbf{u}_q \sin t\theta + \cos t\theta$

Quaternion Rotations

- unit quaternion $\mathbf{q} = (\mathbf{u}_q \sin\theta, \cos\theta)$
  - $\mathbf{u}_q$ axis of rotation
  - θ angle
- vector (point) in 3D: $\mathbf{p} = [p_x, p_y, p_z, 0]$
- rotation of vector (point) $\mathbf{p}$ around $\mathbf{u}_q$ by angle 2θ
  $\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1} = \mathbf{q}\mathbf{p}\mathbf{q}^*$

Quaternion Rotations

- unit quaternion $\mathbf{q} = (\mathbf{u}_q \sin\theta, \cos\theta)$
  - $\mathbf{u}_q$ axis of rotation
  - θ angle
- vector (point) in 3D: $\mathbf{p} = [p_x, p_y, p_z, 0]$
- rotation of vector (point) $\mathbf{p}$ around $\mathbf{u}_q$ by angle 2θ
  $\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1} = \mathbf{q}\mathbf{p}\mathbf{q}^*$

Quaternion Rotations - Why 2θ

- Rotate by i from left:

$$(i)(w + ix + jy + kz) = -x + iw - jz + ky$$

- Rotate by i from right:

$$(w + ix + jy + kz)(i) = -x + iw + jz - ky$$

- Rotate by i from both sides:

$$(i)(w + x + jy + kz)(i) = -w - ix + jy + kz$$

- Prevent rotation in w:

$$(i)(w+ix+jy+kz)(i^{-1}) = (i)(ix+jy+kz+w)(-i) = w-ix+jy+kz$$

- To prevent the second 4D rotation we rotated twice around the first axis.

Quaternion Rotations - Why 2θ

- Rotate by i from left:

$$(i)(w + ix + jy + kz) = -x + iw - jz + ky$$

- Rotate by i from right:

$$(w + ix + jy + kz)(i) = -x + iw + jz - ky$$

- Rotate by i from both sides:

$$(i)(w + x + jy + kz)(i) = -w - ix + jy + kz$$

- Prevent rotation in w:

$$(i)(w+ix+jy+kz)(i^{-1}) = (i)(ix+jy+kz+w)(-i) = w-ix+jy+kz$$

- To prevent the second 4D rotation we rotated twice around the first axis.

Quaternion Rotations - Why 2θ

- Rotate by i from left:

$$(i)(w + ix + jy + kz) = -x + iw - jz + ky$$

- Rotate by i from right:

$$(w + ix + jy + kz)(i) = -x + iw + jz - ky$$

- Rotate by i from both sides:

$$(i)(w + x + jy + kz)(i) = -w - ix + jy + kz$$

- Prevent rotation in w:

$$(i)(w+ix+jy+kz)(i^{-1}) = (i)(ix+jy+kz+w)(-i) = w-ix+jy+kz$$

- To prevent the second 4D rotation we rotated twice around the first axis.

Quaternion Rotations - Why 2θ

- Rotate by i from left:

$$(i)(w + ix + jy + kz) = -x + iw - jz + ky$$

- Rotate by i from right:

$$(w + ix + jy + kz)(i) = -x + iw + jz - ky$$

- Rotate by i from both sides:

$$(i)(w + x + jy + kz)(i) = -w - ix + jy + kz$$

- Prevent rotation in w:

$$(i)(w+ix+jy+kz)(i^{-1}) = (i)(ix+jy+kz+w)(-i) = w-ix+jy+kz$$

- To prevent the second 4D rotation we rotated twice around the first axis.

SPHERICAL LINEAR INTERPOLATION – SLERP

- two quaternions $\mathbf{q}$ and $\mathbf{r}$ ($\mathbf{q} \cdot \mathbf{r} \geq 0$, else take $-\mathbf{q}$)
- real parameter $0 \leq t \leq 1$
- interpolated quaternion $slerp(\mathbf{q}, \mathbf{r}, t) = \mathbf{q}(\mathbf{q}^*\mathbf{r})^t$

$$slerp(q, r, t) = \frac{\sin(\theta(1-t))}{\sin\theta}\mathbf{q} + \frac{\sin\theta t}{\sin\theta}\mathbf{r}$$

- the shortest spherical arc between $\mathbf{q}$ and $\mathbf{r}$

Spherical Linear Interpolation – Slerp

- two quaternions $\mathbf{q}$ and $\mathbf{r}$ ($\mathbf{q} \cdot \mathbf{r} \geq 0$, else take $-\mathbf{q}$)
- real parameter $0 \leq t \leq 1$
- interpolated quaternion $slerp(\mathbf{q}, \mathbf{r}, t) = \mathbf{q}(\mathbf{q}^{\star}\mathbf{r})^{t}$

$$slerp(q, r, t) = \frac{\sin(\theta(1-t))}{\sin\theta}\mathbf{q} + \frac{\sin\theta t}{\sin\theta}\mathbf{r}$$

- the shortest spherical arc between $\mathbf{q}$ and $\mathbf{r}$

Spherical Linear Interpolation – Slerp

- two quaternions $\mathbf{q}$ and $\mathbf{r}$ ($\mathbf{q} \cdot \mathbf{r} \geq 0$, else take $-\mathbf{q}$)
- real parameter $0 \leq t \leq 1$
- interpolated quaternion $slerp(\mathbf{q}, \mathbf{r}, t) = \mathbf{q}(\mathbf{q}^{\star}\mathbf{r})^{t}$

$$slerp(q, r, t) = \frac{\sin(\theta(1-t))}{\sin\theta}\mathbf{q} + \frac{\sin\theta t}{\sin\theta}\mathbf{r}$$

- the shortest spherical arc between $\mathbf{q}$ and $\mathbf{r}$

Quaternion from Two Vectors

- two vectors s and t:
  1. normalization of s, t
  2. unit rotation axis $u = (s \times t)/\|s \times t\|$
  3. angle between s and t: $s \cdot t = \cos\theta$
- Identities to prevent trigonometry:

$$\sin\frac{\theta}{2} = \sqrt{\frac{1-\cos\theta}{2}} \quad (2)$$

$$\cos\frac{\theta}{2} = \sqrt{\frac{1+\cos\theta}{2}} \quad (3)$$

$$\sin\theta = 2\sin\frac{\theta}{2}\cos\frac{\theta}{2} \quad (4)$$

- Final quaternion:

$$q = \left(norm(u)\sin\frac{\theta}{2}, \cos\frac{\theta}{2}\right) = \left(\frac{s \times t}{\sqrt{2(1+s\cdot t)}}, \sqrt{\frac{1+s\cdot t}{2}}\right)$$

Quaternion from Two Vectors

- two vectors s and t:
  1. normalization of s, t
  2. unit rotation axis $u = (s \times t)/\|s \times t\|$
  3. angle between s and t: $s \cdot t = \cos\theta$
- Identities to prevent trigonometry:

$$\sin\frac{\theta}{2} = \sqrt{\frac{1-\cos\theta}{2}} \quad (2)$$

$$\cos\frac{\theta}{2} = \sqrt{\frac{1+\cos\theta}{2}} \quad (3)$$

$$\sin\theta = 2\sin\frac{\theta}{2}\cos\frac{\theta}{2} \quad (4)$$

- Final quaternion:

$$q = \left(norm(u)\sin\frac{\theta}{2}, \cos\frac{\theta}{2}\right) = \left(\frac{s \times t}{\sqrt{2(1+s\cdot t)}}, \sqrt{\frac{1+s\cdot t}{2}}\right)$$

Quaternion from Two Vectors

- two vectors s and t:
  1. normalization of s, t
  2. unit rotation axis $u = (s \times t)/\|s \times t\|$
  3. angle between s and t: $s \cdot t = \cos\theta$
- Identities to prevent trigonometry:

$$\sin\frac{\theta}{2} = \sqrt{\frac{1-\cos\theta}{2}} \quad (2)$$

$$\cos\frac{\theta}{2} = \sqrt{\frac{1+\cos\theta}{2}} \quad (3)$$

$$\sin\theta = 2\sin\frac{\theta}{2}\cos\frac{\theta}{2} \quad (4)$$

- Final quaternion:

$$q = \left(norm(u)\sin\frac{\theta}{2}, \cos\frac{\theta}{2}\right) = \left(\frac{s \times t}{\sqrt{2(1+s\cdot t)}}, \sqrt{\frac{1+s\cdot t}{2}}\right)$$

SUMMARY

rotational matrix

+ HW support, efficient point/vector transformation
– memory (float[9]), other operations are not so efficient

rotational axis and angle

+ memory (float[4] or float[6]), similar to quaternion
– inefficient composition and interpolation

quaternion

+ memory (float[4]), composition, interpolation
– inefficient point/vector transformation

Affine and Projective Spaces

AFFINNE AND PROJECTIVE SPACES

Affine space:

- Set V of vectors and set P of points
- Affine transformations can be represented by matrix

Projective space:

- Homogeneous coordinates
- All lines intersect (space contains infinity)
- Affine and projective transformations can be represented by matrix

AFFINNE AND PROJECTIVE SPACES

Affine space:

- Set V of vectors and set P of points
- Affine transformations can be represented by matrix

Projective space:

- Homogeneous coordinates
- All lines intersect (space contains infinity)
- Affine and projective transformations can be represented by matrix

Homogeneous Coordinates

- homogeneous coordinate vector $[x, y, z, w]$
- transformation: multiplying by a 4×4 matrix
- homogeneous matrix is able to translate and to do perspective projections
- from homogeneous coordinates $[x, y, z, w]$ into Cartesian coordinates: by division $(w \neq 0)[\ x/w,\ \ y/w,\ \ z/w]$
- coordinate vector $[x, y, z, 0]$ – point in infinity
- from Cartesian coordinates to homogeneous: trivial extension $[x, y, z] \dots [x, y, z, 1]$

Transformation Matrix

$$\mathbf{Ap} = \left[\begin{array}{ccc|c} & \mathbf{M} & & \mathbf{T} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} p_x \\ p_y \\ p_z \\ p_w \end{bmatrix}$$

- $\mathbf{T}$ defines translation
- $\mathbf{M}$ defines:
  - rotation
  - scaling

$$\mathbf{M}_{scale} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}$$

  - shear

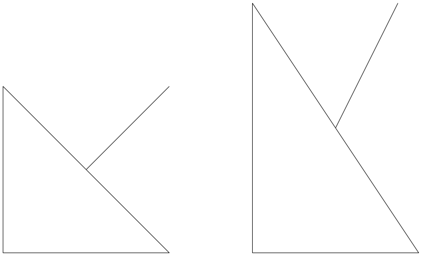
$$\mathbf{M}_{shear} = \begin{bmatrix} 1 & 0 & \lambda \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - and their combinations

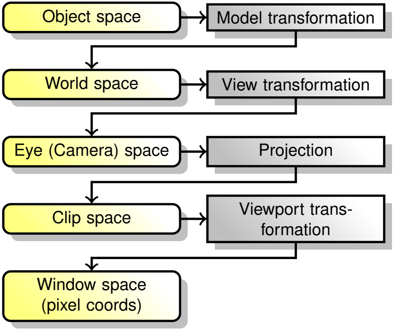
Normal Vector Transformation

- Only orientation change is valid transformation for normals
- Tangents (t) remain valid:

$$\begin{aligned}
\mathbf{n}\cdot\mathbf{t}=0 &\Rightarrow \mathbf{n}'\cdot\mathbf{t}'=(\mathbf{G}\mathbf{n})\cdot(\mathbf{M}\mathbf{t})=0\\
(\mathbf{G}\mathbf{n})\cdot(\mathbf{M}\mathbf{t}) &= (\mathbf{G}\mathbf{n})^T(\mathbf{M}\mathbf{t})\\
&= \mathbf{n}^T\mathbf{G}^T\mathbf{M}\mathbf{t}\\
&\Rightarrow \mathbf{G}=(\mathbf{M}^{-1})^T
\end{aligned}$$



Transformations for Rendering Pipeline



LOOKAT CAMERA MATRIX

- Camera position (eye) $\mathbf{e}$
- Lookat point $\mathbf{p}$
- Up vector $\mathbf{u}$

$$
\begin{aligned}
\mathbf{v} &= norm(\mathbf{e} - \mathbf{p}) \\
\mathbf{n} &= norm(\mathbf{v} \times \mathbf{u})
\end{aligned}
\tag{5}
$$

Matrix which transforms camera into its position:

$$
\mathbf{TR} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & u_x & v_x & 0 \\ n_y & u_y & v_y & 0 \\ n_z & u_z & v_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

LOOKAT CAMERA MATRIX

- Camera position (eye) $\mathbf{e}$
- Lookat point $\mathbf{p}$
- Up vector $\mathbf{u}$

$$\begin{aligned} \mathbf{v} &= norm(\mathbf{e}-\mathbf{p}) \\ \mathbf{n} &= norm(\mathbf{v}\times\mathbf{u}) \end{aligned} \tag{5}$$

Matrix which transforms camera into its position:

$$\mathbf{TR} = \begin{bmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & u_x & v_x & 0 \\ n_y & u_y & v_y & 0 \\ n_z & u_z & v_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

LookAt Camera Matrix II

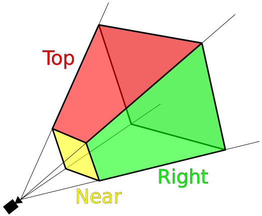
World view needs to be transformed by its inverse:

$$
\begin{aligned}
(\mathbf{TR})^{-1} &= \mathbf{R^{-1}T^{-1}} = \mathbf{R^{T}T^{-1}} = \begin{bmatrix} n_x & n_y & n_z & 0 \\ u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} n_x & n_y & n_z & -(n \cdot e_x) \\ u_x & u_y & u_z & -(u \cdot e_y) \\ v_x & v_y & v_z & -(v \cdot e_z) \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Perspective Projection

Point p projection: $x = -\frac{n}{p_z} p_x$ and $y = -\frac{n}{p_z} p_y$

$$\mathbf{P}_{frustum} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2nf}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



Perspective correct interpolation