Quaternions in 3D graphics, SLERP

Reporter:

Mykola Byelytskyy

Supervisor:

Dr. Alexis Heloir

Contents

- The Problem
- Matrices and Euler Angles
- Quaternions
- SLERP
- Trackball interface using quaternions
- Conclusions

The Problem: Interpolation.

- Interpolation of camera animation

  Without smooth interpolation, camera motion looks very unnatural and makes a bad experience if you try to follow the action.

  DEMO

The Problem: Interpolation.

- Could we interpolate with traditional rotational matrices?

Yes, but it is expensive, since we have to extract axis/angle from the matrix, interpolate it and convert to a new matrix. This requires reverse trigonometry and square root operations.

The Problem: Gimbal Lock

- Example of gimbal lock:

Source of image: https://de.wikipedia.org/wiki/Gimbal_Lock

The Problem: Rotation Description

- Intuitive way to describe rotations

$$\alpha - angle, \quad \vec{v} = (v_1, v_2, v_3) - axis$$

$$\begin{pmatrix} v_1^2(1-\cos(\alpha))+\cos(\alpha) & v_1 v_2(1-\cos(\alpha)-v_3\sin(\alpha)) & v_1 v_3(1-\cos(\alpha)+v_2\sin(\alpha)) \\ v_1 v_2(1-\cos(\alpha))+v_3\sin(\alpha) & v_2^2(1-\cos(\alpha))+\cos(\alpha) & v_2 v_3(1-\cos(\alpha)-v_1\sin(\alpha)) \\ v_1 v_3(1-\cos(\alpha)-v_2\sin(\alpha)) & v_2 v_3(1-\cos(\alpha)+v_1\sin(\alpha)) & v_3^2(1-\cos(\alpha))+\cos(\alpha) \end{pmatrix}$$

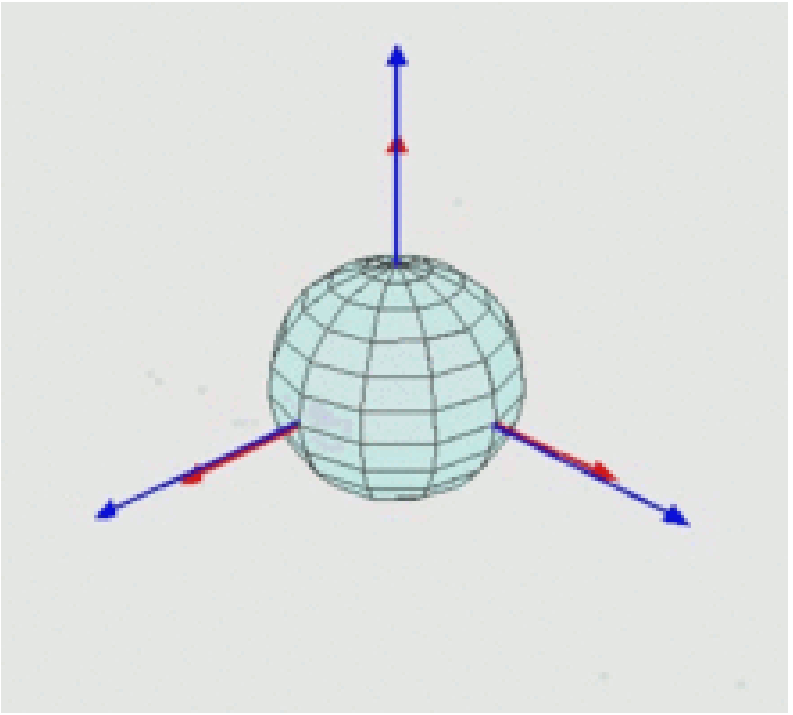
$$q = \left(\cos\left(\frac{\alpha}{2}\right), v_1 \cdot \sin\left(\frac{\alpha}{2}\right), v_2 \cdot \sin\left(\frac{\alpha}{2}\right), v_3 \cdot \sin\left(\frac{\alpha}{2}\right)\right)$$

Matrices and Euler angles

- Matrices

$$\vec{p}\,' = R \cdot \vec{p}$$

- Euler angles



Source of image: https://de.wikipedia.org/wiki/Eulersche_Winkel/

Quaternions

- Complex numbers:

$$z = r + iv_1; \quad i^2 = -1$$

- Quaternions:

$$q = r + iv_1 + jv_2 + kv_3; \quad i^2 = j^2 = k^2 = ijk = -1$$

$$q = (r, \vec{v})$$

Quaternions

- Quaternion math:

$$Norm(q) = r^2 + v_1^2 + v_2^2 + v_3^2$$

$$q \cdot w = \left(r_q r_w - \vec{v}_q \cdot \vec{v}_w , r_q \vec{v}_w + \vec{v}_q r_w + \vec{v}_q \times \vec{v}_w \right)$$

$$\bar{q} = r - iv_1 - jv_2 - kv_3$$

$$q^{-1} = \frac{\bar{q}}{Norm(q)}$$

Quaternions

- Pure quaternions

$$q = (0, \vec{v})$$

- Unit quaternions

$$Norm(q) = 1$$

$$q^{-1} = \bar{q}$$

Quaternions

- Unit quaternions and rotations:

$$q = (0, \vec{v})$$

$$\vec{p}\,' = q\,\vec{p}\,\bar{q}$$

On can show directly, that this is a rotation in 3D space. Or that unit quaternion represents the group of 3D rotations.

$$q = \left(\cos\left(\frac{\alpha}{2}\right), \vec{v}\sin\left(\frac{\alpha}{2}\right)\right)$$

Quaternions

- Simple example:

$$\vec{p} = (0, i, j, k) \qquad q = (0, i, 0, 0) \qquad \vec{p}\,' = q\,\vec{p}\,\bar{q}$$

$$q \cdot w = (r_q r_w - \vec{v}_q \cdot \vec{v}_w,\ r_q \vec{v}_w + \vec{v}_q r_w + \vec{v}_q \times \vec{v}_w)$$

$$\vec{p}\,\bar{q} = (0+1, (1,1,1) \times (-1,0,0)) = (1, (0,1,-1))$$

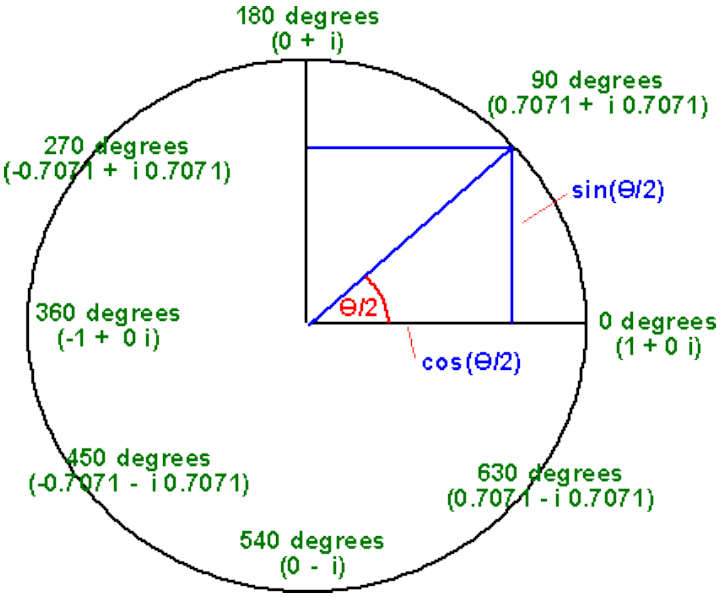
$$q(\vec{p}\,\bar{q}) = (0, (1,0,0) + (1,0,0) \times (0,1,-1)) = (0, (1,-1,-1))$$

Quaternions and Complex Numbers

- Space of rotations 2D. Both directions.

$$Norm(z)=1$$

$$\vec{p}\,'=z\,\vec{p}\,z$$

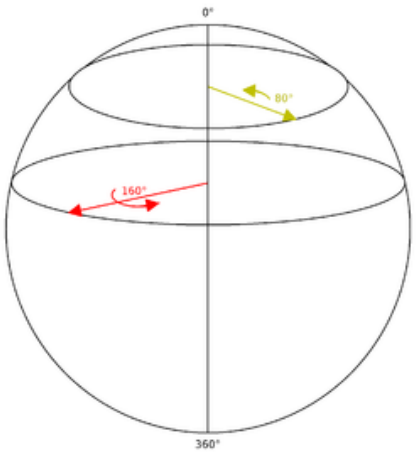


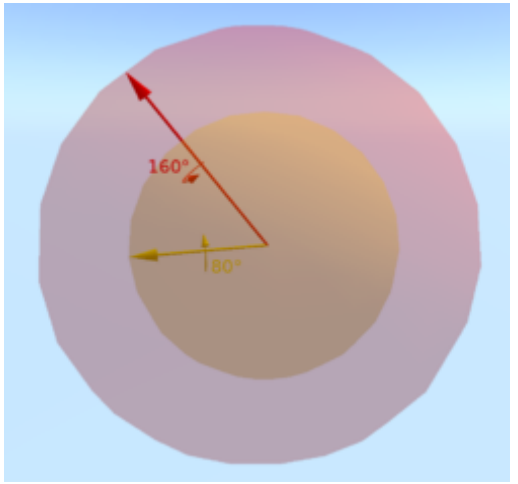
$$\vec{p}\,'=e^{i\frac{\alpha}{2}}R_p e^{ib}e^{i\frac{\alpha}{2}}=R_p e^{i(b+\alpha)}$$

Source of image: http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/complex/transforms/

Quaternions

- Space of rotations 3D





Source of image: https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

Quaternions

- Rotation chaining

$$q_{res} = q_1 \cdot q_2 \cdot q_3 \cdots$$

- A nice property

  after a 360 degree rotation, quaternion inverts the rotation axis. But the “inverted” quaternion still represents the same 360 degree rotation. One can imagine it, as a 360 degree rotation in opposite direction.

Quaternions

- Efficiency:

  Storage: **quaternion 4**, **matrix 9**

  Rotation chaining: **quaternion 16M + 12A**, **matrix 27M + 18A**

  Rotating: **quaternion 15A + 15M, matrix 6A + 9M**

- Rounding errors:

  By chaining rotations, rounding errors accumulate Matrix may not be orthogonal anymore, harder to convert back. Quaternion can be renormalized.

Source: https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

SLERP

- Interpolation between quaternions q and p

$$s(t\,;\,p\,,q)=p\,(\bar{p}\,q)^{t},\quad t\in[0,1]$$

$$e^{\vec{v}\,\alpha}=q\,;\quad q^{t}=(\cos(t\,\alpha)\,,\vec{v}\sin(t\,\alpha))$$

$$s(t\,;\,p\,,q)=\frac{\sin((1-t)\,\theta)\,p+\sin(t\,\theta)\,q}{\sin\theta},\quad t\in[0,1]$$

$$\theta-\textit{angle between } q\,,\,p$$

Operations: 16M 8A 1D 4T

Source of operations estimate : http://www.geometrictools.com/Documentation/RotationIssues.pdf

SLERP

- Interpolation with matrices.

  Extract axis-angle from the matrix.

  Calculate the next step of interpolation.

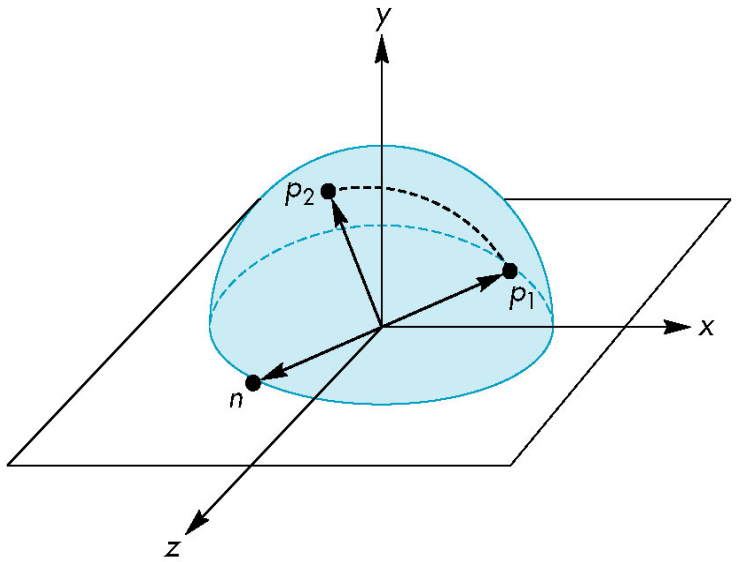
  Calculate the new matrix and rotate.

  Operations: 58A 77M 1D 4T

Source of operations estimate : http://www.geometrictools.com/Documentation/RotationIssues.pdf

Trackball interface

- Description



Projection: $x^2+y^2+z^2=1\,;\quad z=\sqrt{1-x^2-y^2}$

Source of image: http://vip.cs.utsa.edu/classes/cs5113s2007/lectures/materials/TrackBallRotation.html

Trackball interface

- Rotation

$$q = (p_1 \cdot p_2,\ p_1 \times p_2)$$

$$p_2 \cdot p_1 = \cos(\alpha);\quad p_1 \times p_2 = \vec{n} \cdot \sin(\alpha)$$

$$\alpha = \arccos(p_2 \cdot p_1)$$

than rotate the coordinate system using this quaternion (q).

Conclusion

- Quaternions are more efficient if you need SLERP.
- They are more robust to accumulation of rounding errors.
- They are more intuitive.
- There is no gimbal lock with quaternions.