CMSC 427: Chapter 7
3D Rotation, Quaternions and Physics

Reading:

Overview:
- Rotations
- Quaternions
- A bit of physics.

Overview
- Rotations
- Quaternions
- A Bit of Physics
- Physics for the Programming Assignment

Rotations
Planar Rotations: (about the origin)
- Rotation in the plane is easy. Just give the angle of rotation.
- Convention: Positive angles are counter-clockwise.

Rotation in 3-Space: (about the origin)
- Specify both the axis of rotation (a unit vector) and the angle of rotation.
- Convention: Positive angles are “right-handed”.

Applications:
- Rotating objects in animations.
- Specifying the orientation of an object (e.g., a camera) in space. (As opposed to its location.)

Methods:
- Euler Angles: Easy to grasp. Mathematically messy.
- Quaternions: Mathematically clean. Harder to grasp.

Euler Angles
Euler Rotation Theorem: Any rotation in 3-space may be described using three angles. The rotation can be expressed as the composition of three rotation matrices, where each is a rotation about one of the coordinate axes.

Euler Angles: Three angles (φ, θ, ψ) that describe these rotations.

Euler Angles: Gimbal Lock
Gimbal Lock:
- A major problem with Euler-angle rotations.
- Rotation along one axis results in alignment of the other axes, a loss of degrees of freedom.

Example: Track a UFO by a telescope that can point in any direction by:
1. First, specifying the azimuth (compass direction).
2. Next, specifying the altitude (angle above the horizon).

Problem: Suppose that the UFO passes directly overhead.
- Exactly overhead: azimuth is undefined.
- Almost overhead: azimuth is unstable. Tiny wobbles result in massive changes in azimuth.
What are Quaternions?
- A geometric object, which can be encoded as a 4-element vector.
- Quaternions can be used to encode rotations and orientations in 3-dimensional space.
- Multiplying quaternions corresponds to composing rotations. If \( q_i \) and \( q_j \) are two quaternions, then \( q_i q_j \) is a quaternion that encodes rotation of first doing \( q_j \) then \( q_i \).

Why Use Quaternions?
- Non-singular representation of rotation. (No gimbal lock problem, as with Euler angles).
- More compact than rotation matrices (and faster for some ops).
- More natural looking rotations. (No bias due to choice of coordinate axes, as with Euler angles.)
- No matrix drift. (When rotation matrices are multiplied, round off errors result in non-orthogonality of columns.)

Digression: Complex Numbers and Rotation
Complex Numbers:
- Recall that a complex number is of the form \( a + bi \). This is defined in terms of a symbolic value \( i \), which satisfies the equation \( i^2 = -1 \).

Complex Numbers and Vectors:
- A complex number \( a + bi \) can be interpreted as a vector \( (a, b) \) in the plane.
- Adding complex numbers is analogous to adding the two corresponding vectors.

Modulus:
- The modulus of a complex number \( a + bi \) is \( \sqrt{a^2 + b^2} \).
- The modulus corresponds to the vector’s length.
- A complex number of unit modulus can be expressed as \( \cos \theta + i \sin \theta \), for \( 0 \leq \theta \leq 2\pi \).

Digression: Complex Numbers and Rotation
Complex Numbers and Rotation:
- If you view a complex number \( a + bi \) as a vector \( (a, b) \), then complex multiplication is related to rotation in the plane.
- For example, squaring such a unit-modulus complex number effectively doubles the angle:
  \[
  (\cos \theta + i \sin \theta)^2 = (\cos^2 \theta - \sin^2 \theta) + i(2\cos \theta \sin \theta) = \cos(2\theta) + i \sin(2\theta).
  \]

Question:
- Can we extend this concept to rotation in 3-space?
- This was studied extensively in early 1800’s by Hamilton, who was trying to answer the question “How can we multiply vectors?”

Quaternions
History: Hamilton hit upon the idea of using 4-element vectors. He devised a system involving a scalar and three symbolic values \( i, j, \) and \( k \), satisfying the following multiplication rules:
\[
i^2 = j^2 = k^2 = ijk = -1,
\]
and
\[
i = k \quad jk = i \quad kl = j.
\]
Multiplication is associative and anti-commutative. We have:
\[
ji = -k \quad bj = -i \quad ik = -j.
\]
Quaternions: A type of generalized complex number of the form,
\[
q = a + bj + ck + d.
\]
It is common to express this as a scalar part and vector part:
\[
q = (s, \mathbf{v}) = s + u_i + u_j + u_k.
\]

Quadratic Inverse:
- To multiply two quaternions \( q = (s, \mathbf{v}) \) and \( p = (t, \mathbf{w}) \), we expand and simplify using the multiplication rules:
  \[
  qp = (st - \mathbf{v} \cdot \mathbf{w}, s\mathbf{w} + t\mathbf{v})
  \]
- The result is a quaternion. It can be expressed in vector form:
  \[
  qp = (st - \mathbf{v} \cdot \mathbf{w}), s\mathbf{w} + t\mathbf{v}
  \]

More Complex/Quaternion Analogies
Complex:
- Conjugate: of a complex number \( z = (a + bi) \) is defined to be \( \overline{z} = (a - bi) \).
- Modulus: can be defined to be \( |z| = \sqrt{a^2 + b^2} \).
- Multiplicative Inverse: of \( z \) is \( z^{-1} = \frac{\overline{z}}{|z|^2} \).

Quaternion:
- Conjugate: of a quaternion \( q = (s, \mathbf{v}) \) is defined to be \( \overline{q} = (s, -\mathbf{v}) \).
- Modulus: of \( q \) is defined to be \( |q| = \sqrt{s^2 + \mathbf{v} \cdot \mathbf{v}} \).
- Multiplicative Inverse: of \( q \) is \( q^{-1} = \frac{\overline{q}}{|q|^2} \).
Rotation with Quaternions

**General 3-Dimensional Rotation:**
- Suppose that we want to rotate vector \( v \) about unit vector \( u \) by an angle \( \theta \).
- This can be implemented by two quaternion multiplications.
  - First, we convert \( u, v, \) and \( \theta \) into quaternion form. Define:
    - \( p = (0, v) \) and \( q = \left( \cos \frac{\theta}{2}, \sin \frac{\theta}{2} u \right) \).
  - Define the rotation operator:
    \[ R(p) = pq^{-1}. \]

**Fact:** \( R(p) \) is a pure quaternion, whose vector part is the rotation of \( v \) about axis \( u \) by angle \( \theta \).

---

**Example:**
- Consider the 3-d rotation shown in the figure right.
- This rotation can be expressed as a rotation about the \( y \)-axis by 90 degrees. Thus \( \theta = \pi/2 \), and \( u = (0, 1, 0) \).
- The quaternion that encodes this rotation is:
  \[ q = \left( \cos \frac{\pi}{4}, \sin \frac{\pi}{4} \right)(0,1,0) = \left( 1, 0, -\frac{\sqrt{2}}{2} \right). \]
- To rotate a vertex \( v \), we encode it as a pure quaternion \( p = (0, v) \), and then compute \( p' = pq \). We output the vector part \( v' \) of the result.
- As an exercise, verify that \( v' = (1, 0, 0) \) is mapped to \( (0, 0, 1) \).

---

**What do I need to Know about Quaternions**

What you need to know to use Quaternions:
- Quaternions can be used to represent the rotation (orientation) of an object in space.
  - It is easy to convert from the representation as angle \( \theta \) about a unit vector \( u \) to a quaternion:
    \[ q = \left( \cos \frac{\theta}{2}, \sin \frac{\theta}{2} u \right). \]
  - You can rotate a vector \( v \) by a quaternion \( q \), by first constructing the quaternion \( p = (0, v) \) and performing \( pq \).
  - To compose to rotations together, multiply their quaternions. Thus, \( pq \) is the rotation \( q \) followed by \( p \).
  - To undo a rotation, multiply by the inverse quaternion.

Where do I get Quaternion code?
- Examples are all over the web.
A Bit of Physics

Simulating Physics:
- Let \( t \) be the current time, and let \( \Delta t \) be a small time step. The task is to update the physical state to the next time step \( t + \Delta t \).

Chain of influence:
- **Forces:** (Gravity, friction, propulsion) Act on bodies and modify acceleration in accordance with Newton’s law \( F = ma \).
- **Torques:** Twisting forces, which affect angular acceleration.
- **Linear (Angular) Acceleration:** Rate of change of linear and angular velocity over time. Denoted \( \alpha(t) \) and \( \mathbf{\omega}(t) \). When accelerations are known, we can update \( \mathbf{v}(t) \) and \( \mathbf{R}(t) \).
- **Velocity:** When velocities are known, we can update position \( \mathbf{p}(t) \) and orientation \( \mathbf{R}(t) \).

Overview

- Rotations
- Quaternions
- A Bit of Physics
- Physics for the Programming Assignment

Physics for Programming Assignment?

**Katamari** can be implemented with very little physics:
- Apply and compose rotations (given as quaternions).
- Perform simple collision detection (sphere-to-sphere, sphere-to-cube).

**Katamari State:** (What you need to store)
- **Position:** \( p = (p_x, p_y, p_z) \). In fact, only \( (p_x, p_z) \) are needed, since height \( z \) is determined by the fact that it sits on the ground.
- **Orientation:** Suggest using a quaternion \( q = (s, \mathbf{u}) \).
- **Linear velocity:** \( v = (v_x, v_y, v_z) \). Only \( (v_x, v_z) \) are needed, since \( v_y \) is zero for motion on the ground.

A Bit of Physics

Numerical Integration:
- Simulating physics by updating state over successive time steps.
- Doing this in a manner so that errors do not accumulate is tricky.

Simple Integration:
- Compute all the forces and torques at time \( t \).
- Compute accelerations from forces (given object properties, such as mass and moment of inertia).
- Update velocities:
  - \( \mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \Delta t \mathbf{a}(t) \)
  - \( \mathbf{a}(t) = \mathbf{f}(t) / m(t) \)
- Update position and orientation:
  - \( \mathbf{p}(t + \Delta t) = \mathbf{p}(t) + \Delta t \mathbf{v}(t) \)
  - \( \mathbf{R}(t + \Delta t) = \mathbf{R}(t) \mathbf{R}(t + \Delta t) \)
  - Redraw your scene.

Collision Response:
- When the Katamari hits an object, compute the surface normal \( \mathbf{n} \) at the point of contact.
- Decompose the velocity vector \( \mathbf{v} \) into components:
  - \( \mathbf{v} = \mathbf{v}_n + \mathbf{v}_r \)
  - \( \mathbf{v}_n = p' - p \), where \( p' \) is parallel to \( \mathbf{n} \)
  - \( \mathbf{v}_r = \mathbf{v} - \mathbf{v}_n \)
- Set \( \mathbf{v}_n \) = 0, \( \mathbf{v}_r \) = \( \mathbf{v}_r \cdot \mathbf{n} \), where \( \mathbf{s} \leq q \leq 1 \), called the coefficient of restitution, indicates the loss of energy due to the non-elasticity of the collision.

Putting it together:
1. Given new linear velocity, compute new angular velocity and new position.
2. Given the new angular velocity, compute the quaternion derivative \( \frac{1}{2} \omega(t') \mathbf{q}(t) \).
3. Given the quaternions, update the orientation quaternion.
4. Given new position and orientation, draw object.
Summary

Summary:
- Rotation by Euler angles
- Rotation by Quaternions
- A bit of Physics
- Physics for the programming assignment