# Aplikasi Quaternion untuk Rotasi Objek di Unity Game Engine

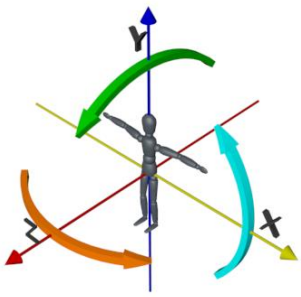
Muhammad Ra'if Alkautsar - 13523011[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
[1]*mraifalkautsar@gmail.com, 13523011@std.stei.itb.ac.id*

***Abstrak* — Rotasi adalah salah satu gerakan fundamental dalam kakas grafika dan animasi. Bilangan quaternion seringkali digunakan untuk melakukan rotasi. Pada makalah ini, dijelaskan mengenai aplikasi dan penerapan quaternion untuk melakukan rotasi objek di Unity Game Engine beserta dengan implementasi dan analisisnya secara langsung. Dilakukan juga demonstrasi dan analisis pada sejumlah teknik rotasi objek seperti linear interpolation (LERP) dan spherical linear interpolation (SLERP). Rotasi menggunakan quaternion mampu untuk digunakan secara efektif untuk rotasi *Unity Game Engine* tanpa mengalami gimbal lock seperti halnya rotasi dengan sudut euler.**

***Rotation is one of the fundamental movements in graphics and animation tools. Quaternions are often used to perform rotation. This paper explains the application and implementation of quaternions to rotate objects in the Unity Game Engine along with their direct implementation and analysis. Implementation and analysis are also carried out on a number of object rotation techniques such as linear interpolation (LERP) and spherical linear interpolation (SLERP). Rotation using quaternions can be used effectively for Unity Game Engine rotation without experiencing gimbal lock as with rotation with Euler angles.***

***Kata kunci* — quaternion, rotasi, game, unity**

## I. PENDAHULUAN

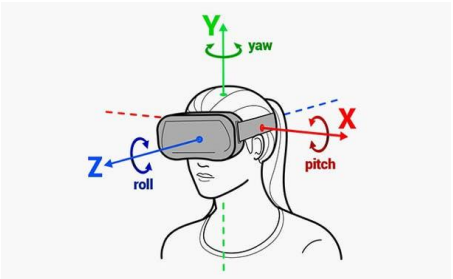



Gambar 1.
Rotasi sebuah objek dalam ruang tiga dimensi.
(Sumber: https://gamedev.stackexchange.com/questions/111690/tweening-back-from-arbitrary-3d-rotation-in-javascript)

Rotasi adalah salah satu gerakan paling fundamental pada sebuah game yang memuat ruang berdimensi tiga. Rotasi sendiri adalah sebuah konsep matematis yang berasal dari geometri. Sebuah rotasi adalah gerak terhadap suatu poros pada suatu ruang di mana setidaknya ada satu titik yang dipertahankan dan tidak terjadi perbesaran magnitude. Rotasi berbeda dengan jenis-jenis gerak lain seperti translasi yang tak memiliki titik tetap maupun refleksi.

Konsep rotasi telah diformalisasikan sejak geometri Euclid pada abad ke-3 sebelum Masehi. Kemudian, Rene Descartes memperkenalkan sistem koordinat Cartesian, yang memberi jalan ke representasi aljabar dari transformasi geometris, termasuk rotasi. Kemudian, pada abad ke-18, matriks dan aljabar linier mulai digunakan sebagai alat untuk merepresentasikan transformasi. Matriks menjadi sebuah pilihan untuk memodelkan rotasi baik di ruang 2D maupun 3D. Pada tahun 1843, Sir William Rowan Hamilton mengenalkan quaternion, sebuah sistem matematis yang memperluas bilangan kompleks ke ruang 3D. Quaternion lalu dikenal sebagai alat yang sangat efisien untuk merepresentasikan rotasi 3D, menghindari gimbal lock dan menyederhanakan implementasi. Dengan berkembangnya sistem-sistem komputasi, rotasi mulai digunakan untuk mensimulasikan berbagai hal seperti dinamika pesawat terbang. Teknologi dan ilmu grafika terus berkembang sehingga teknik-teknik seperti spherical linear interpolation (SLERP) juga dikenalkan untuk transisi rotasi yang halus dalam animasi. Saat ini, dalam era *real-time graphics*, integrasi rotasi dengan simulasi berbasis fisika, termasuk dinamika rotasi dan momentum sudut, menjadi standar di engine seperti Havok dan Unity. Teknik-teknik rotasi dimatangakn untuk *real-time rendering* di AR/VR, robotika, dan gaming. (Sejarah dari rotasi)

Konsep rotasi memiliki banyak aplikasi dalam matematika dan sains. Pada bidang informatika, rotasi berkaitan erat dengan grafika. Grafika komputer sendiri adalah sebuah cabang ilmu komputer yang mempelajari bagaimana gambar bisa digenerasi dengan bantuan komputer. Cabang tersebut adalah teknologi inti dalam bidang fotografi digital, film, seni digital, dan tak terkecuali video game. Beragam teknologi dan kakas pembuatan game mengutilisasi berbagai metode matematis untuk melakukan rotasi.



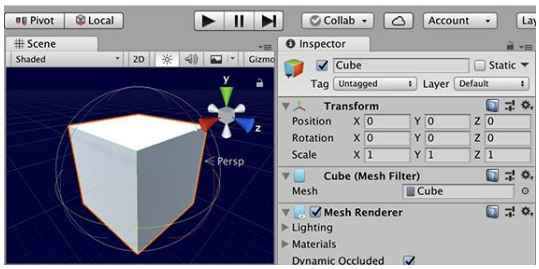

Gambar 2.

Rotasi dalam konteks perangkat *virtual reality*.
(Sumber: https://mecharithm.com/learning/lesson/what-is-virtual-reality-vr-101)

Pada industri hiburan, rotasi digunakan untuk mensimulasikan gerakan yang realistis, kendali kamera, character rigging, special effects, dan cinematic. Pada visualisasi saintifik, rotasi digunakan untuk berbagai disiplin melihat dan menganalisis struktur molekul 3D (seperti protein dan DNA), untuk mensimulasikan gerak planet, dan memvisualisasikan data. Pada bidang robotika, rotasi digunakan untuk kinematika dan dinamika, perencanaan jalur, dan sistem kendali. Pada AR/VR, rotasi digunakan untuk melacak gerak kepala, interaksi spasial, dan juga rendering.

Ada berbagai teknik-teknik umum dalam rotasi. Linear interpolation (LERP) adalah teknik sederhana untuk mencari titik-titik di antara dua posisi atau orientasi. Pada rotasi, hal ini bermanfaat untuk menginterpolasi antara dua sudut dan menciptakan pergerakan kamera dan transisi yang halus. Sementara itu, *spherical linear interpolation* (SLERP) adalah interpolasi antara dua quaternion (atau titik rotasi) melalui sebuah permukaan sferis, memastikan kecepatan sudut yang konsisten. Teknik ini diaplikasikan dalam transisi antar pose, rotasi halus, dan kamera sistem yang dinamis.

Pada kakas Unity Game Engine sendiri, secara mendasar, rotasi bisa dilakukan menggunakan konsep rotasi Euler atau Quaternion. Rotasi euler adalah rotasi yang didefiniskan dengan perpindahan suatu titik berdasarkan tiga sudut sekuensial, yang dikenal sebagai sudut Euler. Secara klasik, tiap sudut merepresentasikan rotasi mengitari salah satu sumbu utama (X, Y, Z) dan tiap rotasi dilakukan dalam urutan spesifik.




Gambar 2.
Rotasi di Unity Game Engine
(Sumber: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Transform.Rotate.html)

Namun, kelemahan dari rotasi menggunakan sudut Euler adalah adanya kemungkinan terjadinya gimbal lock. Gimbal lock adalah suatu fenomena ketika bidang atau permukaan dari sebuah sumbu menjadi sejajar atau paralel sehingga salah satu *degree of freedom*-nya hilang.

Kelemahan ini tak dimiliki oleh metode rotasi menggunakan quaternion. Quaternion sendiri adalah sebuah sistem bilangan dengan empat komponen yang biasa digunakan untuk merepresentasikan suatu rotasi. Pada makalah ini, akan dijelaskan teori mengenai quaternion, permasalahan yang diselesaikan oleh quaternion, serta studi kasus mengenai penggunaan quaternion pada Unity Game Engine beserta contoh dan demonstrasi.

## II. DASAR TEORI

### A. Rotasi

Rotasi adalah sebuah transformasi geometris yang memutar sebuah object di antara sebuah titik atau poros pada ruang, sambil tetap menjaga bentuk dan ukurannya. Pada rotasi, jarak dan sudut di dalam objek tidak berubah.

Dalam ruang 2D, titik (x,y) diputar oleh sebuah sudut θ (theta) mengitari titik asalnya dengan matriks rotasi:

$$R(\theta) = \begin{bmatrix} cos\ \theta & -sin\ \theta \\ sin\ \theta & cos\ \theta \end{bmatrix}$$

Lalu, titik hasil transformasinya (x', y') dihitung dengan:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R(\theta) \begin{bmatrix} x \\ y \end{bmatrix}$$

Namun, pada ruang 3D, adanya lebih dari satu poros rotasi membuatnya lebih kompleks. Pada ruang 3D, rotasi dapat dilakukan dengan:

- Matriks rotasi
  Sebuah contoh matriks rotasi mengitari sumbu Z:

$$R_z(\theta) = \begin{bmatrix} cos\ \theta & -sin\ \theta & 0 \\ sin\ \theta & cos\ \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Quaternion
  Sebuah contoh quaternion rotasi mengitari sumbu X:

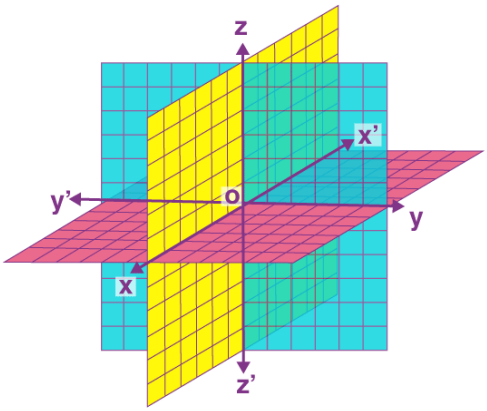
$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) \cdot i$$

  Rotasi kemudian dilakukan dengan operasi quaternion:

$$v' = qvq^{-1}$$

### B. Koordinat Cartesius

Dalam sistem koordinat Cartesius, titik dan vektor direpresentasikan dengan koordinat $(x, y, z)$ dalam ruang 3D. Rotasi melibatkan transformasi koordinat-koordinat ini sambil tetap menjaga hubungan spasial relatif yang ada.




Gambar 3.
Sebuah ruang tiga dimensi yang direpresentasikan dengan sistem koordinat Cartesius. [1]

(Sumber: https://byjus.com/maths/coordinate-axes-and-coordinate-planes-in-three-dimensionsional-space/)

C. Quaternion

Bilangan quaternion adalah sebuah bilangan yang dituliskan sebagai gabungan antara skalar dengan vektor, berbentuk

$$q = a + v = a + bi + cj + dk = (a, v)$$

Aturan dasar yang berlaku pada quaternion adalah

$$ij = k, jk = i, ki = j, ji = -k, kj = -i, ik = -j$$

beserta

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

Aturan perkalian dua quaternion $z_1 = a_1 + v_1$ dengan $z_2 = a_2 + v_2$ adalah

$$\begin{aligned} z_1 z_2 &= (a_1 + v_1)(a_2 + v_2) \\ &= a_1 a_2 - v_1 v_2 + a_1 v_2 + a_2 v_1 + v_1 \times v_2 \end{aligned}$$

Berbagai rumus dan persamaan yang berlaku untuk quaternion antara lain

▪ Norma (*magnitude*) quaternion

Quaternion: $q = a + v = a + bi + cj + dk$

Magnitude:

$$\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2}$$

▪ Quaternion satuan (*unit*)

Quaternion: $q = a + v = a + bi + cj + dk$

Quaternion satuan:

$$\hat{q} = \frac{1}{\|q\|}(a + bi + cj + dk)$$

▪ Quaternion murni (*pure quaternion*)

$$q = bi + cj + dk$$

Perkalian dua quaternion murni tidak bersifat tertutup, sebab hasilnya adalah quaternion yang tidak murni.

▪ Bilangan quaternion sekawan (conjugate)

Quaternion: $q = a + v = a + bi + cj + dk$

Conjugate: $q = a - v = a - bi - cj - dk$

Dapat ditunjukkan bahwa:

$$q\bar{q} = a^2 + b^2 + c^2 + d^2$$

▪ Balikan (inverse) quaternion

Quaternion: $q = a + v = a + bi + cj + dk$

Balikan:

$$q^{-1} = \frac{1}{q} = \frac{\bar{q}}{\|q\|^2}$$

Dapat ditunjukkan bahwa:

$$qq^{-1} = q^{-1}q = 1$$

▪ Balikan (inverse) quaternion

Quaternion: $q = a + v = a + bi + cj + dk$

Balikan:

$$q^{-1} = \frac{1}{q} = \frac{\bar{q}}{\|q\|^2}$$

Dapat ditunjukkan bahwa:

$$qq^{-1} = q^{-1}q = 1$$

▪ Rotasi quaternion

Quaternion: $q = a + v = a + bi + cj + dk$

Rotasi dengan quaternion dapat dilakukan dengan merepresentasikan vektor $v = (v_x i, v_y j, v_z k)$ sebagai sebuah quaternion murni, lalu mengaplikasikan:

$$p' = qpq^*$$

dengan $p'$ adalah quaternion yang telah dirotasi (mengandung vektor yang telah dirotasi pada bagian vektornya) dan $q^* = (w, -x, -y, -z)$ sebagai sekawan atau conjugate dari $q$. Dengan mengambil bagian vektor dari quaternion yang telah dirotasi, kita memperoleh vektor hasil rotasinya.

$$v' = (p'_x, p'_y, p'_z)$$

D. Teknik-Teknik Rotasi

Ada beberapa teknik-teknik rotasi yang sering digunakan dalam game engine maupun kakas animasi lainnya.

- *Linear interpolation* (LERP) adalah sebuah teknik untuk secara linier menginterpolasi antara dua titik atau nilai melewati sebuah interval. Untuk dua titik a dan b dengan faktor interpolasi t ($0 \leq t \leq 1$):

$$LERP(a, b, t) = a + t \cdot (b - a)$$

- *Spherical linear interpolation* (SLERP) adalah sebuah teknik untuk menginterpolasi antara dua rotasi melewati sebuah permukaan bola satuan. Berbeda dengan LERP, yang bergerak melewati garis lurus, SLERP bergerak pada sebuah busur yang harus, membuatnya ideal untuk menginterpolasi rotasi. Untuk dua quaternion satuan $q_1$ dan $q_2$, serta sebuah faktor interpolasi t (di mana $0 \leq t \leq 1$):

$$SLERP(q_1, q_2, t) = \frac{\sin((1-t)\theta)}{\sin(\theta)} q_1 + \frac{\sin(t\theta)}{\sin(\theta)} q_2$$

## III. ROTASI DENGAN SUDUT EULER

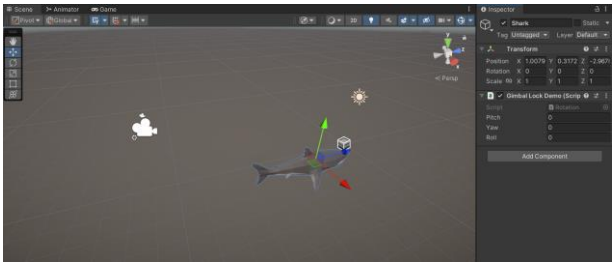
### A. Eksperimen

Rotasi dasar di Unity dapat dilakukan dengan menggunakan metode sudut Euler maupun quaternion. Sebuah contoh rotasi dasar menggunakan metode sudut Euler ada pada kode berikut:

```
using UnityEngine;

public class GimbalLockDemo : MonoBehaviour
{
    public float pitch = 0f;
    public float yaw = 0f;
    public float roll = 0f;
```

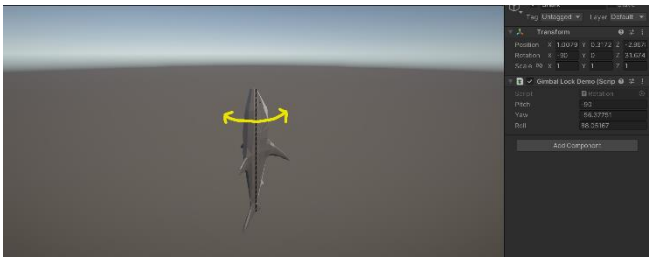
```
    void Update()
    {
        pitch += Input.GetAxis("Vertical") *
50f * Time.deltaTime;
        yaw += Input.GetAxis("Horizontal") *
50f * Time.deltaTime;
        roll += Input.GetKey(KeyCode.Q) ? 50f
* Time.deltaTime : 0f;

        pitch = Mathf.Clamp(pitch, -90f,
90f);
        transform.eulerAngles = new
Vector3(pitch, yaw, roll);
        Debug.Log($"Pitch: {pitch}, Yaw:
{yaw}, Roll: {roll}");
    }
}
```

Kode tersebut adalah sebuah script yang dapat digunakan pada Unity Script Editor lalu di-*attach* ke sebuah game object. Cara kerja script tersebut adalah memeriksa apakah ada input vertikal, horizontal, atau tombol 'Q' pada tiap frame game dijalankan. Ketika ada masukan, program akan melakukan rotasi dengan sumbu yang sesuai (input vertikal atau *pitch* berarti rotasi bersumbu X, input horizontal atau *yaw* berarti rotasi bersumbu Y, dan Q atau *roll* berarti rotasi bersumbu Z). Nilai dari pitch di-clamp pada ±90° supaya fenomena *gimbal lock* bisa diperlihatkan.



Gambar 4.
*Setup* dari scene eksperimen *gimbal lock*.

Fenomena gimbal lock terjadi ketika kita melakukan pitch sejauh ±90°. Pada derajat tersebut, bidang permukaan dari sumbu Z menjadi sejajar dengan bidang permukaan dari sumbu Y. Sehingga, ketika dilakukan input *yaw* maupun *roll*, gerakan yang dihasilkan sama, yaitu rotasi di sumbu Y.



Gambar 5.
Fenomena gimbal lock yang terjadi ketika pitch sejauh ±90°. Panah kuning adalah arah rotasi *yaw*/*roll* dan garis hitam adalah sumbunya.

B. Penjelasan

Berdasarkan definisi, *gimbal lock* terjadi ketika dua dari tiga sumbu rotasi menjadi sejajar, sehingga derajat kebebasan dari rotasi berkurang dari tiga menjadi dua.

Sudut euler merepresentasikan rotasi secara sekuensial mengitari tiga sumbu. Urutan yang digunakan dalam kakas Unity adalah konvensi Tait-Bryan, yaitu Y (*yaw*), lalu X (*pitch*), lalu Z (*roll*). Urutan rotasi tidaklah komutataif dan akan memengaruhi orientasi akhir.

Jika sudut euler dikonversikan menjadi matriks rotasi, maka untuk pitch $\theta$, yaw $\psi$, dan roll $\phi$ serta R adalah matriks rotasi, persamaannya:

$$R = R_y(\psi)R_x(\psi)R_z(\phi)$$

Ketika $\theta = \pm 90°$, akan terbentuk matriks R berikut:

$$R = \begin{bmatrix} sin\ \psi \sin\phi & sin\ \psi \sin\phi & \pm cos\ \psi \\ \sin\phi & cos\ \phi & 0 \\ cos\ \psi\ \sin\phi & \mp cos\psi\ cos\phi & sin\ \psi \end{bmatrix}$$

Pada matriks ini, mengubah $\psi$ (yaw) maupun $\phi$ (roll) sama-sama akan merotasikan objek mengitari sumbu Y.

Dalam konteks scene atau kode, ketika pitch mencapai ±90°:

- Sumbu-y lokal dari objek sejajar dengan sumbu-z dunia.
- Sumbu-z lokal dari objek sejajar dengan sumbu-y dunia.
- Rotasi apapun yang diaplikasikan pada *yaw* tak bisa dibedakan dengan *roll* dalam konteks lokal.

## IV. ROTASI DENGAN QUATERNION

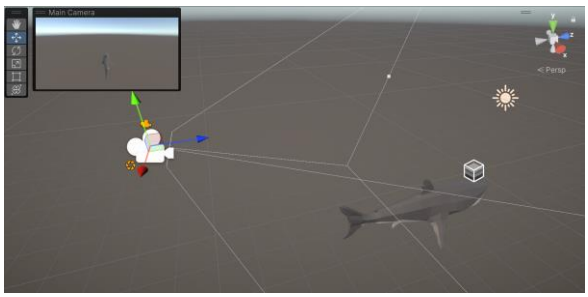
### A. Eksperimen

Dalam rotasi menggunakan quaternion, tidak ada kemungkinan terjadinya *gimbal lock* seperti halnya menggunakan sudut euler. Rotasi menggunakan quaternion dapat didemonstrasikan di Unity Game Engine.

```
using UnityEngine;

public class QuaternionRotationDemo :
MonoBehaviour
{
    private Quaternion currentRotation;

    void Start()
    {
        currentRotation = Quaternion.identity;
    }

    void Update()
    {
        float pitchDelta =
Input.GetAxis("Vertical") * 50f * Time.deltaTime;
```

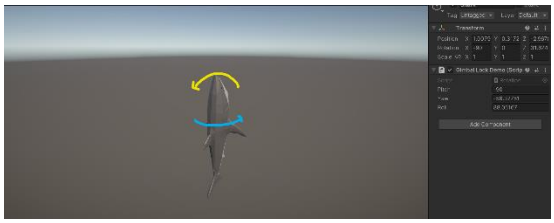
```
        float yawDelta =
Input.GetAxis("Horizontal") * 50f *
Time.deltaTime;
        float rollDelta = Input.GetKey(KeyCode.Q)
? 50f * Time.deltaTime : 0f;

        Quaternion pitchRotation =
Quaternion.AngleAxis(pitchDelta, Vector3.right);
        Quaternion yawRotation =
Quaternion.AngleAxis(yawDelta, Vector3.up);
        Quaternion rollRotation =
Quaternion.AngleAxis(rollDelta, Vector3.forward);

        currentRotation = yawRotation *
pitchRotation * rollRotation * currentRotation;

        transform.rotation = currentRotation;
    }
}
```

Script dipasangkan ke sebuah game object yang ingin dirotasikan,



Gambar 4.
Setup dari scene eksperimen rotasi dengan quaternion.

Sama seperti kode sebelumnya, kode tersebut dapat di-attach ke sebuah *game object*. Lalu, ketika ada masukan, program akan melakukan rotasi dengan sumbu yang sesuai (input vertikal atau *pitch* berarti rotasi bersumbu X, input horizontal atau *yaw* berarti rotasi bersumbu Y, dan Q atau *roll* berarti rotasi bersumbu Z). Pada rotasi menggunakan implementasi quaternion, tidak terjadi *gimbal lock* ketika pitch berada pada ±90°. Pitch, yaw, maupun roll tetap memberikan pergerakan yang berbeda.



Gambar 5.
Rotasi dari game object (kuning adalah arah roll, biru adalah arah dari yaw)

## B. Penjelasan

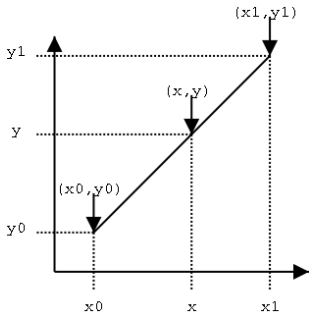
Rotasi dengan quaternion tidak mengakibatkan gimbal lock karena alasan-alasan berikut:

- Rotasi menggunakan quaternion tidak memiliki *sequence* atau urutan. Sebuah quaternion merepresentasikan orientasi rotasi secara keseluruhan sebagai satu, bukan sebagai tiga rotasi berbeda yang bergantung pada sumbu.
- Rotasi menggunakan quaternion tidak memiliki sumbu perantara. Rotasi dengan sebuah quaternion relatif ke sebuah 4D *hypersphere*, menghindari ketergantungan pada kerangka koordinat. Sumbu-sumbu yang ada tidak mungkin untuk sejajar dan menghilangkan derajat kebebasan.
- Jika sejumlah quaternion digabungkan (contoh: $q_1 \cdot q_2$), quaternion yang dihasilkan akan tetap merepresentasikan suatu rotasi yang valid.

# V. BERBAGAI TEKNIK ROTASI DENGAN QUATERNION

## A. Linear Interpolation (Lerp)

Pada kakas Unity Game Engine, dapat dilakukan linear interpolation, yaitu sebuah teknik untuk menginterpolasi titik-titik antara titik a dan b dengan sebuah variabel interpolan t. Parameter t di-*clamp* di antara rentang [0, 1]. Teknik ini sering digunakan untuk memindahkan sebuah objek secara *gradual* di antara dua titik tersebut.



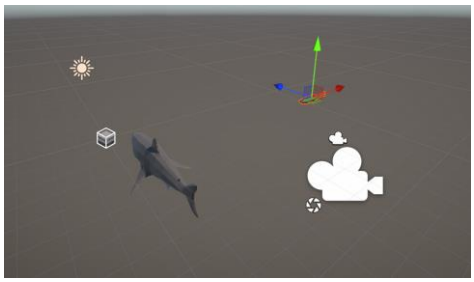

Gambar 6.
Sebuah grafik interpolasi linear.
(Sumber: https://www.toppr.com/guides/maths-formulas/linear-interpolation-formula/)

Untuk mendemonstrasikan aplikasi *linear interpolation* di Unity, dibuat sebuah script C# di Unity untuk merotasikan sebuah game object secara halus dari rotasi awal ke target rotasi seiring waktu menggunakan *Quaternion.Lerp*.

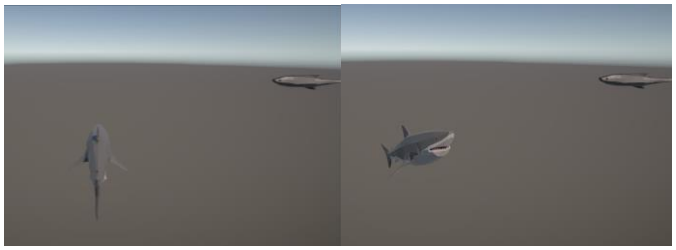
```
using UnityEngine;

public class LerpRotationDemo : MonoBehaviour
{
    public Transform target;
    public float lerpSpeed = 1.0f;
    private Quaternion startRotation;
    private Quaternion targetRotation;
    private Quaternion defaultRotation =
Quaternion.Euler(-90,0,0);
    private float lerpProgress = 0f;
    private bool isRotating = false;

    void Start()
    {
        startRotation = transform.rotation;
```

```
        if (target != null)
        {
            Vector3 directionToTarget =
target.position - transform.position;
            targetRotation =
Quaternion.LookRotation(directionToTarget);
        }
        else
        {
            targetRotation = defaultRotation;
        }
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            isRotating = true;
            lerpProgress = 0f;
            startRotation = transform.rotation;
        }

        if (isRotating)
        {
            lerpProgress += Time.deltaTime *
lerpSpeed;
            lerpProgress =
Mathf.Clamp01(lerpProgress);
            transform.rotation =
Quaternion.Lerp(startRotation, targetRotation,
lerpProgress);

            if (lerpProgress >= 1f)
            {
                isRotating = false;
            }
        }
    }
}
```

Script dipasangkan ke sebuah objek yang ingin dirotasikan, lalu, dipasangkan target (untuk diambil properti transform-nya) pada inspektor. Target ini adalah sebuah objek yang akan menjadi arah tujuan dari rotasi. Apabila target kosong, target rotasi akan bernilai default (dalam kasus ini di-har Quaternion.Lerp(startRotation, targetRotation, t) menginterpolasikan antara dua orientasi rotasi, dikendalikan dengan parameter *t*, yang merentang dari 0 ke 1. Tombol spasi lalu ditekan untuk memulai interpolasi. Objek akan berotasi dari orientasi awal ke arah target.

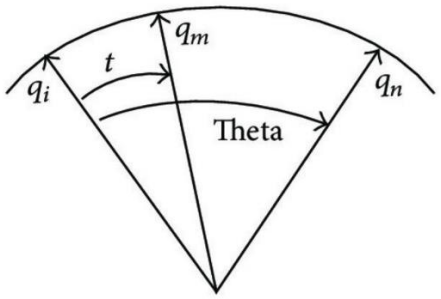


Gambar 7.
Setup scene dengan model ikan di kanan atas sebagai target rotasi.



Gambar 8.
Sebelum dan sesudah rotasi dijalankan.

B. Spherical Linear Interpolation (Slerp)



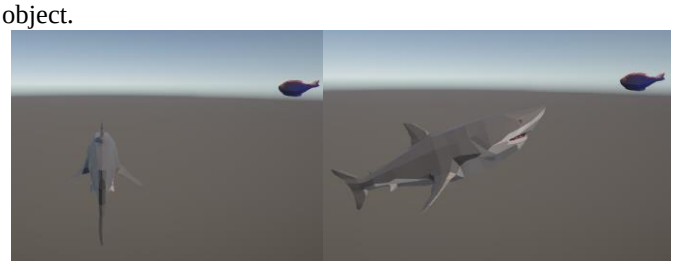

Gambar 7.
Visualisasi sederhana dari spherical linear interpolation.
(Sumber: https://www.researchgate.net/figure/Spherical-linear-interpolation_fig6_321743256)

*Spherical linear interpolation* (SLERP) adalah sebuah metode untuk secara halus menginterpolasi antara dua rotasi atau dua orientasi yang direpresentasikan sebagai quaternion atau vektor satuan pada permukaan sebuah bola. Sebuah kode untuk script Unity yang mendemonstrasikan SLERP adalah sebagai berikut:

```
using UnityEngine;

public class SlerpRotationDemo : MonoBehaviour
{
    public Transform target;
    public float slerpSpeed = 1.0f;
    private Quaternion startRotation;
    private Quaternion targetRotation;
    private float slerpProgress = 0f;
    private bool isRotating = false;

    void Start()
    {
        startRotation = transform.rotation;

        if (target != null)
        {
            Vector3 directionToTarget =
target.position - transform.position;
            targetRotation =
Quaternion.LookRotation(directionToTarget);
        }
        else
        {
            targetRotation = Quaternion.Euler(0,
90, 0);
        }
    }
```

```
void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space)) {
            isRotating = true;
            slerpProgress = 0f;
            startRotation = transform.rotation;
        }

        if (isRotating) {
            slerpProgress += Time.deltaTime *
slerpSpeed;
            slerpProgress =
Mathf.Clamp01(slerpProgress);
            transform.rotation =
Quaternion.Lerp(startRotation, targetRotation,
slerpProgress);

            if (slerpProgress >= 1f)
            {
                isRotating = false;
            }
        }
    }
}
```

Cara kerja dari script ini sama seperti dengan script LERP. Dipasangkan sebuah target object di *inspector* Unity, lalu ketika ditekan Spasi, objek akan berotasi untuk menghadap ke target object.



Gambar 8.
Sebelum dan sesudah rotasi dijalankan.

## V. KESIMPULAN

Berdasarkan analisis dan eksperimen yang telah dilakukan, ditemukan sejumlah hal terkait rotasi dalam Unity Game Engine, baik secara umum maupun secara spesifik untuk rotasi menggunakan quaternion.

Rotasi dengan quaternion mampu untuk menghindari fenomena *gimbal lock* yang dapat terjadi pada rotasi menggunakan sudut euler. Oleh karena kelebihan tersebut, rotasi menggunakan quaternion menjadi lebih fleksibel dan *robust.*

Selain itu, rotasi dengan quaternion juga dapat digunakan untuk berbagai teknik-teknik rotasi, seperti LERP (*linear interpolation*) dan SLERP (*spherical linear interpolation*). Teknik-teknik rotasi ini bisa digunakan untuk berbagai penerapan ketika mengembangkan sebuah proyek di Unity atau game engine lain yang menerapkan prinsip sama seperti Unity.

## VI. LAMPIRAN

a. Video Demonstrasi dan Penjelasan Makalah
https://youtu.be/ut6kcuuWDb0
b. *Repository* GitHub
https://github.com/mraifalkautsar/MakalahQuaternionAlgeo

## VII. UCAPAN TERIMA KASIH

Makalah ini tidak akan selesai tanpa bantuan dari pihak lain. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Allah SWT,
2. orang tua dan keluarga penulis,
3. dosen pengampu mata kuliah Aljabar Linier dan Geometri, dan
4. teman-teman penulis

yang senantiasa mendukung penulis selama pembuatan makalah ini berlangsung.

## REFERENSI

[1] Howard Anton & Chris Rorres, *Elementary Linear Algebra 12th Ed.*, John Wiley and Sons, 2019.
[2] John Vince, *Geometric Algebra for Computer Graphics*, Springer, 2015.
[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-25-Aljabar-Quaternion-Bagian1-2023.pdf. Diakses pada tanggal 30 Desember 2024.
[4] https://docs.unity.com/. Diakses pada tanggal 30 Desember 2024.

## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 2 Januari 2025

Muhammad Ra'if Alkautsar
13523011