Aplikasi Quaternion pada Rotasi dalam Game Engine

Resa Kemal Saharso 13514109
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
resakemal@s.itb.ac.id

***Abstract*—Salah satu aplikasi quaternion adalah representasi rotasi pada bidang tiga dimensi. Karena kelebihannya dibandingkan representasi rotasi lainnya, quaternion sangat cocok digunakan pada *game engine* untuk menangani rotasi. Selain itu, quaternion juga digunakan untuk menghasilkan animasi rotasi yang halus dengan menggunakan Slerp.**

***Keywords*—Quaternion, Rotasi, *Game Engine*, 3D**

I. PENDAHULUAN

Pada masa globalisasi ini, teknologi sudah berkembang sangat pesat. Salah satu hasil dari kemajuan teknologi adalah permainan, terutama permainan digital. Karena hidup manusia tidak terlepas dari bantuan mesin, tidak dapat dimungkiri bahwa permainan digital akan sangat berpengaruh kepada kehidupan manusia. Manusia yang jenuh berkerja dan berusaha setiap hari akan mencari sarana rekreasi yang singkat dan tidak banyak memakan waktu. Permainan digital merupakan solusi pintar untuk menghilangkan penat yang menumpuk.

Salah satu topik yang sedang hangat pada pasar *game* saat ini adalah maraknya *indie game* yang muncul. Sangat bervariasi genre, konten, dan kualitasnya, *indie game* adalah game buatan individu/grup kecil yang biasanya menggunakan *game engine* komersial yang berbayar atau bahkan gratis.

Gambar 1.1 Cities: Skylines, salah satu game *city simulator* populer yang dibuat menggunakan *game engine* Unity
Sumber: http://i0.wp.com/geekdad.com/wp-content/uploads/2015/04/Skylines-San-Francisco.jpg

Pada permainan sendiri terdapat banyak elemen-elemen yang memengaruhi jalannya permainan. Salah satu elemen tersebut yang berhubungan dengan animasi adalah rotasi. Kebanyakan benda akan mempunyai animasi berputar, relatif terhadap pemain maupun objek lain. Pada permainan berorientasi *third-person* dimana pemain dapat melihat avatar/karakter yang dia mainkan, saat pemain memutar kamera pandangan pemain akan berubah mengikuti arah gerakan kamera. Gerakan kamera secara perlahan dari titik awal ke titik akhir adalah rotasi.

Ternyata, selain dapat direpresentasikan oleh matriks, rotasi terutama pada ruang 3 dimensi dapat juga direpresentasikan dengan quaternion. Dalam makalah ini penulis akan membahas kegunaan quaternion dalam mempermudah penanganan rotasi dalam *game engine*.

II. DASAR TEORI

A. Quaternion

1. Definisi

Quaternion adalah bentuk bilangan kompleks pada bidang tiga dimensi. Ditemukan oleh Wiliam Rowan Hamilton pada 1843, pada awalnya dia mengalikan 2 bilangan kompleks pada dimensi tiga (R^3) berupa

$$z = a + ib$$

yang menghasilkan

$$z1z2 = (a1 + ib1 + jc1)(a2 + ib2 + jc2)$$

diturunkan menjadi

$$z1z2 = a1a2 + ia1b2 + ja1c2 + ib1a2 + i2b1b2 + ijb1c2 + jc1a2 + jic1b2 + j2c1c2.$$

karena $i^2 = j^2 = k^2 = 1$,

$$z1z2 = (a1a2 - b1b2 - c1c2) + i(a1b2 + b1a2) + j(a1c2 + c1a2) + ijb1c2 + jic1b2$$

Namun ij dan ji tidak terdefinisi. Akhirnya, Hamilton menemukan solusinya dengan cara mengubah bentuk bilangan kompleks yang terdiri dari tiga bilangan menjadi empat bilangan;

$$z = a + ib + jc + kd$$

Yang diberi nama quaternion. Saat dua buah bilangan tersebut dikalikan,, hasilnya adalah

$$z1z2 = (a1 + ib1 + jc1 + kd1)(a2 + ib2 + jc2 + kd2)$$
$$= a1a2 + ia1b2 + ja1c2 + ka1d2$$
$$+ib1a2 + i2b1b2 + ijb1c2 + ikb1d2$$
$$+jc1a2 + jic1b2 + j2c1c2 + jkc1d2$$
$$+kd1a2 + kid1b2 + kjd1c2 + k2d1d2.$$

Serupa dengan sebelumnya, $i^2 = j^2 = k^2 = 1$;

$$z1z2 = a1a2 - b1b2 - c1c2 - d1d2 + i(a1b2 + b1a2) + j(a1c2 + c1a2) + k(a1d2 + d1a2) + ijb1c2 + ikb1d2 + jic1b2 + jkc1d2 + kid1b2 + kjd1c2$$

Tetapi ij, jk, dan ki belum terdefinisi. Agar dapat disederhanakan, Hamilton menyatakan bahwa:

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

Sehingga persamaan menjadi:

$$z1z2 = a1a2 - (b1b2 + c1c2 + d1d2) + a1(ib2 + jc2 + kd2) + a2(ib1 + jc1 + kd1) + i(c1d2 - d1c2) + j(d1b2 - b1d2) + k(b1c2 - c1b2)$$

Akhirnya, bentuk awal bilangan tersebut diubah menjadi

$$z = s + v$$

Dimana s adalah komponen skalar dan v adalah komponen vektor dari bilangan. Hasil perkaliannya adalah:

$$z1z2 = s1s2 - v1 \cdot v2 + s1\,v2 + s2\,v1 + v1 \times v2$$

Dimana s1s2 adalah perkalian dua buah bilangan skalar, v1.v2 adalah *dot product* dari dua buah vektor, setta v1xv2 adalah *cross product* dari dua buah vektor.

2. Penjumlahan & Pengurangan Quaternion

Penjumlahan dan pengurangan dua buah quaternion adalah sebagai berikut.

$$q1 \pm q2 = [(s1 \pm s2) + i(x1 \pm x2) + j(y1 \pm y2) + k(z1 \pm z2)]$$

Contohnya adalah terdapat dua buah quaternion

$$q1 = 1 + i2 + j3 + k4$$
$$q2 = 2 - i + j5 - k2$$

Maka hasil penjumlahannya adalah

$$q1 + q2 = (1+2) + i(2-1) + j(3+5) + k(4-2)$$
$$= 3 + i + j8 + k2$$

3. Perkalian Quaternion

Perkalian dua buah quaternion seperti yang sudah dijelaskan pada definisi adalah sebagai berikut.

$$q1q2 = s1s2 - v1 \cdot v2 + s1v2 + s2v1 + v1 \times v2$$

Namun, quaternion tidak bersifat komutatif karena bila ditelusuri hasil perkalian q2q1 didapatkan

$$q1q2 = s1s2 - v1 \cdot v2 + s1v2 + s2v1 + v1 \times v2$$

Dimana v1xv2 tidak sama dengan v2xv1. Contohnya adalah hasil perkalian dua buah quaternion sebelumnya (pada penjumlahan dan pengurangan quaternion) adalah

$$q_1q_2 = (1 + i2 + j3 + k4)(2 - i + j5 - k2) \quad (5.36)$$
$$= [1 \times 2 - (2 \times (-1) + 3 \times 5 + 4 \times (-2))$$
$$+1(-i + j5 - k2) + 2(i2 + j3 + k4)$$
$$+i(3 \times (-2) - 4 \times 5) + j(4 \times (-1) - (-2) \times 2) + k(2 \times 5 - (-1) \times 3)]$$
$$= -3 + i3 + j11 + k6 - i26 + k13$$
$$q_1q_2 = -3 - i23 + j11 + k19$$

Dan hasil q2q1 adalah

$$q_2q_1 = (2 - i + j5 - k2)(1 + i2 + j3 + k4)$$
$$= [2 - ((-1) \times 2 + 5 \times 3 + (-2) \times 4)$$
$$+2(i2 + j3 + k4) + 1(-i + j5 - k2)$$
$$+i(5 \times 4 - 3 \times (-2)) + j((-2) \times 2 - 4 \times (-1)) + k((-1) \times 3 - 2 \times 5)]$$
$$q_2q_1 = -3 + i29 + j11 - k7$$

Terbukti bahwa $q1q2 \neq q2q1$.

4. Besaran Quaternion

Ukuran atau norma dari quaternion adalah $||q|| = \sqrt{s^2 + x^2 + y^2 + z^2}$

Contohnya adalah terdapat quaternion

$$q = 1 + i2 + j3 + k4$$

Maka besaran dari quaternion tersebut adalah

$$||q|| = \sqrt{1^2 + 2^2 + 3^2 + 4^2}$$
$$= \sqrt{30}$$

5. Unit Quaternion

Quaternion unit adalah quaternion yang serupa dengan vektor satuan. Unit sebuah quaternion adalah quaternion tersebut dibagi dengan besaran quaternion.

$$q = \frac{1}{||q||}(a + ib + jc + kd)$$

6. Quaternion Murni

Quaternion murni adalah quaternion yang tidak mempunyai nilai skalar. Contohnya adalah

$$z = ib + jc + kd$$

Bila dua buah quaternion murni dikalikan, hasilnya adalah

$q1q2 = -(\mathbf{x1x2 + y1y2 + z1z2}) + i(y1z2 - y2z1) + j(z1x2 - z2x1) + k(x1y2 - x2y1)$

dimana terdapat nilai skalar hasil dari *dot product* kedua quaternion, maka hasil perkalian quaternion murni bukanlah quaternion murni.

7. Konjugasi Quaternion

Konjugasi dari sebuah quaternion adalah elemen skalar ditambah negasi dar i elemen vektornya.

$$q = s - \mathrm{v} = s - (ix + jy + kz)$$

Contohnya adalah quaternion

$$\mathrm{q} = 1 + i2 + j3 + k4$$

memiliki konjugasi

$$q = 1 - i2 - j3 - k4$$

7. Invers Quaternion

Invers dari sebuah quaternion adalah

$$q^{-1} = \frac{s - ix - jy - kz}{||q||}$$

dikarenakan q^{-1} memenuhi persamaan

$$qq^{-1} = \frac{(s + ix + jy + kz)(s - ix - jy - kz)}{||q||^2} = 1$$

Contohnya adalah quaternion sebelumnya (pada konjugasi quaternion) memiliki invers berupa

$$q^{-1} = \frac{1 - i2 - j3 - k4}{30}$$

B.Rotasi

Rotasi adalah pergerakan melingkar sebuah benda yang berpusat pada sebuah poros. Benda tersebut bergerak tegak lurus terhadap pusat, namun karena terikat oleh poros pergerakan benda berbelok searah dengan porosnya.

Pada aljabar linear, untuk merepresentasikan rotasi digunakan matriks rotasi. Matriks rotasi adalah matriks yang berisi nilai-nilai sehingga bila matriks elemen aljabar dikalikan dengan matriks rotasi maka elemen aljabar tersebut akan berubah karena rotasi sebesar θ. Besar matriks adalah n x n dengan n sebagai ruang, contohnya matriks rotasi berukuran 2x2 digunakan pada ruang 2 dimensi.

$$\mathbf{A}_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$

$$\mathbf{A}_Y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$\mathbf{A}_Z = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \end{bmatrix}$$

Gambar 2.1 Matriks rotasi yang digunakan untuk rotasi dengan poros sumbu x, y dan z.

C.Rotasi dengan Quaternion

1. Definisi

Quaternion dapat digunakan dalam menghitung rotasi vektor dengan operasi

$$\mathrm{p'} = \mathrm{qpq}^{-1}$$

dimana p adalah vektor awal sebelum rotasi, p' adalah vektor hasil rotasi, q dan q^{-1} adalah

$$\mathrm{q} = \cos(\theta/2) + \sin(\theta/2)\ \hat{}\mathrm{u}$$
$$\mathrm{q}^{-1} = \cos(\theta/2) - \sin(\theta/2)\ \hat{}\mathrm{u}$$

dimana ^u adalah poros rotasi berupa

$$\hat{}\mathrm{u} = xu\mathrm{i} + yu\mathrm{j} + zu\mathrm{k}$$

Contohnya adalah diketahui sebuah vektor P(0,0,1) yang mengalami rotasi sebesar 90^o dengan poros ^u = j.

$p = 0 + i0 + j + k$ (5.81)

$$q = \cos 45^o + \sin 45(i0 + j + k0)$$
$$q = \frac{\sqrt{2}}{2}(1 + i0 + j + k0)$$
$$q^{-1} = \cos 45^o - \sin 45(i0 + j + k0)$$
$$q^{-1} = \frac{\sqrt{2}}{2}(1 - i0 - j - k0)$$

Hasil rotasi adalah

$$\mathrm{p'} = \mathrm{qpq}^{-1}$$

$$\mathrm{p}' = \frac{\sqrt{2}}{2}(1 + i0 + j + k0)(0 + i0 + j + k)\frac{\sqrt{2}}{2}(1 - i0 - j - k0)$$

Pertama hitunglah pq,

$$qp = \frac{\sqrt{2}}{2}(1 + i0 + j + k0)(0 + i0 + j + k)$$
$$qp = \frac{\sqrt{2}}{2}(-1 + i + j + k).$$

Lalu hitung $(pq)q^{-1}$,

$$(qp)q^{-1} = \frac{\sqrt{2}}{2}(-1 + i + j + k)\frac{\sqrt{2}}{2}(1 - i0 - j - k0)$$
$$= \frac{1}{2}(-1 + 1 + j + i + j + k + i - k)$$
$$= \frac{1}{2}(0 + i2 + j2 + k0)$$
$$(qp)q^{-1} = 0 + i + j + k0$$

Hasil rotasi P adalah elemen vektor pada quaternion yaitu (1,1,0)

3. Konkatenasi Quaternion

Proses menghitung hasil beberapa rotasi pada quaternion adalah sebagai berikut.

$q_{net} = q_3q_2q_1;$
$v' = q_3q_2q_1 \; v \; q_1^{-1}q_2^{-1}q_3^{-1}$
$= q_{net} \; v \; q_{net}^{-1}$

Dapat dilihat bahwa urutan perkalian quaternion dibalik ($q_3q_2q_{1)}$. Ini disebabkan invers sebuah quaternion adalah invers semua elemennya sehingga

$$q_{net}^{-1} = q_1^{-1}q_2^{-1}q_3^{-1}$$

4. Konversi

a. Quaternion – Matriks

Jika diketahui quaternion q = w + ix + jy + kz, maka matriks rotasinya adalah:

$$\begin{matrix} 1 - 2y^2 - 2z^2 & 2xy + 2zw & 2x - 2yw \\ 2xy - 2zw & 1 - 2x^2 - 2z^2 & 2yz + 2xw \\ 2xz + 2yw & 2yz - 2xw & 1 - 2x^2 - 2y^2 \end{matrix}$$

b. Matriks – Quaternion

Konversi matriks rotasi menjadi quaternion lebih sulit daripada sebaliknya, sehingga dibuat algoritma berikut.

```
MatToQuat(float m[4][4], QUAT * quat)
{
float  tr, s, q[4];
int    i, j, k;
int nxt[3] = {1, 2, 0};
tr = m[0][0] + m[1][1] + m[2][2];
// check the diagonal
if (tr > 0.0) {
s = sqrt (tr + 1.0);
quat->w = s / 2.0;
s = 0.5 / s;
quat->x = (m[1][2] - m[2][1]) * s;
quat->y = (m[2][0] - m[0][2]) * s;
quat->z = (m[0][1] - m[1][0]) * s;
} else {
// diagonal is negative
i = 0;
if (m[1][1] > m[0][0]) i = 1;
if (m[2][2] > m[i][i]) i = 2;
j = nxt[i];
k = nxt[j];
s = sqrt ((m[i][i] - (m[j][j] + m[k][k])) + 1.0);
q[i] = s * 0.5;
if (s != 0.0) s = 0.5 / s;
q[3] = (m[j][k] - m[k][j]) * s;
q[j] = (m[i][j] + m[j][i]) * s;
q[k] = (m[i][k] + m[k][i]) * s;
quat->x = q[0];
quat->y = q[1];
quat->z = q[2];
quat->w = q[3];
}
}
```

Gamb ernion

Sumber: http://www.gamasutra.com/view/feature/131686/rotating_objects_using_quaternions.php

5. Kelebihan Rotasi dengan Quaternion

Beberapa kelebihan rotasi dengan quaternion dibandingkan dengan matriks rotasi adalah:

- Representasi rotas i pada ruang 3D lebih simpel karena hanya menggunakan 4 variabel (1 skalar dan 3 vektor pada ruang 3 dimensi) dibandingkan matriks rotasi yang menggunakan 9 variabel (matriks 3x3)
- Membuat quaternion berdasarkan sudut dan poros rotasi serta mengetahui sudut dan poros rotasi dari quaternion lebih mudah bila dibandingkan dengan matriks
- Nilai pada matriks lebih rawan daripada quaternion jika dilakukan pembulatan nilai yang memiliki *margin error*.

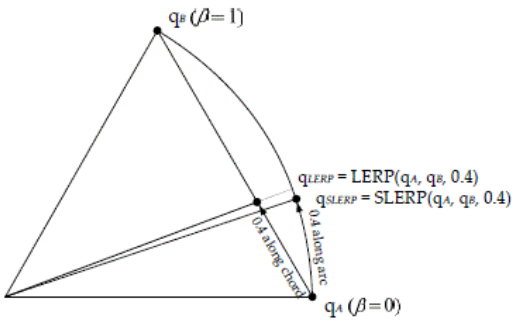
III. APLIKASI QUATERNION PADA ROTASI DALAM *GAME ENGINE*

A. Slerp

Slerp adalah *spherical linear interpolation* yaitu interpolasi yang menghasilkan lengkungan dari 2 titik. Slerp memiliki rumus

$$SLERP(p; q; \beta) = wpp + wqq$$

Dimana p adalah titik awal, q adalah titik akhir dan β adalah nilai dimana $0 \leq \beta \leq 1$. Nilai β digunakan untuk membuat titik-titik yang nantinya akan membentuk



lengkungan.

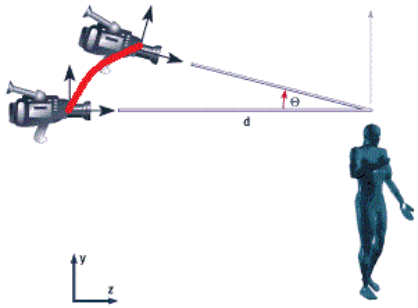
Gambar 3.1 Representasi slerp

Sumber: Gregory, Jason. 2015. Game Engine Architecture. Taylor & Francis Group

B. Quaternion Slerp

Slerp digunakan untuk menghasilkan rotasi yang halus dan biasanya digunakan untuk pergerakan kamera pada game 3D dengan orientasi *third-person*. Ketika pemain menggerakkan kamera, game akan mencatat titik awal dan

titik akhir pergerakan kamera dan membuat lengkung menggunakan Slerp sebagai jalur pergerakan kamera.



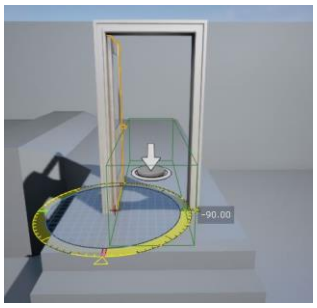
Gambar 3.1 Pergerakan kamera pada game dengan orientasi *third-person*. Garis merah adalah jalur pergerakan kamera yang dibuat menggunakan SLERP.
Sumber:http://www.gamasutra.com/view/feature/131686/rotating_objects_using_quaternions.php

Beberapa bentuk quaternion yang ekivalen dengan slerp adalah:

$$\begin{aligned} Slerp(q_0,q_1, t) &= q_0(q_0^{-1},q_1)^t \\ &= q_1(q_1^{-1} q_0)^{1-t} \\ &= (q_0 q_1^{-1})^{1-t} q_1 \\ &= (q_1 q_0^{-1})^t q_0 \end{aligned}$$

C. Aplikasi pada Game Engine

Selain Slerp, aplikasi utama quaternion adalah sebagai representasi rotasi menggantikan matriks rotasi. Sebagai contohnya tinjau aplikasi quaternion pada *game engine* Unity. Unity adalah salah satu *game engine* paling populer berbasis 3D yang cukup mudah untuk dipakai. Hampir semua *indie game* sekarang memakai Unity. Pada dokumentasi Unity, terlihat bahwa terdapat *struct* Quaternion yang mempunyai fungsi-fungsi utama Quaternion.LookRotation,Quaternion.Angle, Quaternion. Euler, Quaternion.Slerp, Quaternion.FromToRotation, dan Quaternion.identity. Lalu pada *game engine* Unreal Engine 4, *game engine* populer lainnya yang dapat menghasilkan grafis yang sangat nyata dan realistis, mempunyai struct bernama Fquat yang mempunyai beberapa fungsi seperti GetAxisX, GetAxisY, Inverse, Normalized, Rotator, Slerp, dsb.



Gambar 3.2 Pengaturan rotasi pintu pada Unreal Engine 4
Sumber:https://docs.unrealengine.com/latest/images/Engine/Matinee/HowTo/MHT_1/MHT1_DoorOpen.jpg

V. KESIMPULAN

Quaternion sangat berguna untuk representasi rotasi pada *game engine* untuk menggantikan matriks rotasi, serta aplikasinya pada slerp untuk pergeseran kamera maupun animasi.

REFERENSI

[1] Vince, John. 2008. Geometric Algebra for Computer Graphics. Springer-Verlag London Limited.

[2] Gregory, Jason. 2015. Game Engine Architecture. Taylor & Francis Group

[3] http://www.gamasutra.com/view/feature/131686/rotating_objects_using_quaternions.php Diakses pada 14 Desemer 2015 pukul 16.31

[4] http://run.usc.edu/cs520-s12/assign2/p245-shoemake.pdf Diakses pada 14 Desember pukul 17.02

[5] http://www.3dgep.com/understanding-quaternions/ Diakses pada 14 Desember pukul 17.05

[6] http://docs.unity3d.com/ScriptReference/Quaternion.html Diakses pada 15 Desember 2015 pukul 11.25

[7] https://docs.unrealengine.com/latest/INT/API/Runtime/Core/Math/FQuat/index.html Diakses pada 15 Desember 2015 pukul 11.37

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, 15 Desemeber 2015

Resa Kemal Saharso 13514109