Aplikasi Transformasi Lanjar pada Pembuatan Obyek Tiga Dimensi

Hafizh Dary Faridhan Hudoyo - 13514072
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514072@std.stei.itb.ac.id

***Abstrak*— Makalah ini menjelaskan aplikasi transformasi lanjar dalam pembuatan obyek tiga dimensi dengan komputer. Obyek tersebut biasa digunakan dalam pembuatan *game* digital atau dalam *printing* tiga dimensi. Dalam makalah ini akan dibahas cara menerapkan transformasi lanjar dalam memprogram obyek tiga dimensi dengan bahasa pemrograman.**

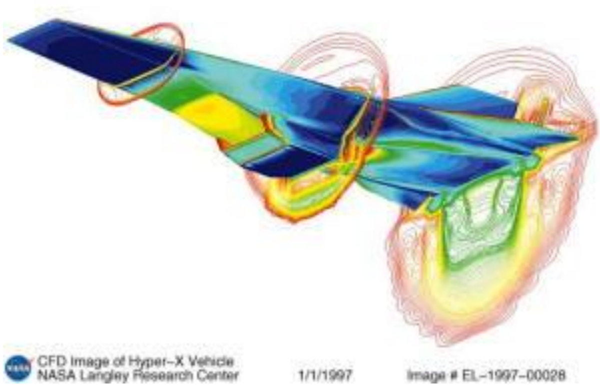
***Kata Kunci*—Tiga Dimensi, Transformasi Lanjar, Rotasi, Dilatasi**

I. PENDAHULUAN

Obyek tiga dimensi adalah obyek yang mempunyai ukuran panjang, lebar, dan tinggi. Karena mempunyai tiga ukuran panjang, maka obyek ini dapat dipaparkan ke dalam sumbu x, y, dan z. Setiap obyek yang dibuat menggunakan *software* aplikasi tiga dimensi akan mempunyai dimensi seperti yang disebutkan di atas.

Sejarah penemuan obyek tiga dimensi pada komputer berawal dari tahun 1963. Pada tahun tersebut telah ditemukan interaksi manusia dengan komputer (untuk memudahkan pengguna komputer) oleh Ivan Sutherland. Ia juga mengembangkan Sketchpad, yaitu salah satu *computer-aided design* (CAD) pada komputer yang berupa sebuah papan dengan pena elektronik untuk merealisasikan gambarnya pada komputer. Sekitar tujuh tahun setelahnya, yaitu tahun 1970-an, Ivan Sutherland mempunyai seorang murid yang bernama Edwin Catmull yang menjadi pionir pembuatan objek tiga dimensi pada komputer. Setelah itu, ia berperan besar dalam industri perfilman seperti Lucasfilm, Pixar, dan Disney.

CAD merupakan suatu alat untuk merealisasikan obyek dua dimensi maupun tiga dimensi ke dalam komputer. Obyek tersebut digambarkan dalam vektor dan *wireframe*, yaitu garis-garis yang membentuk kerangka obyek. Lalu, kerangka tersebut akan dibuat struktur dalam dan luar dari obyek tersebut. Langkah terakhir yang dilakukan sehingga menjadi obyek tiga dimensi adalah *texturing*, yaitu memberi warna, permukaan obyek, dan detail lainnya.



Gambar 1: Desain pesawat oleh NASA menggunakan CAD[1]

Pada era sekarang, sudah sangat banyak *software* pembuat obyek tiga dimensi yang telah dipasarkan. *Software* yang terkenal dan tidak asing bagi kita adalah Adobe Illustrator, Adobe Photoshop (fitur pembuatan tiga dimensinya tidak sebanyak Illustrator), Unity, dan Blender. Unity dan Blender biasa digunakan untuk membuat objek tiga dimensi yang akan dipakai di dalam *game* digital, sedangkan Adobe Illustrator dan Adobe Photoshop biasa digunakan untuk membuat poster, gambar untuk iklan, desain produk, dan media promosi lainnya.

II. DASAR TEORI

Dasar teori yang digunakan adalah aljabar lanjar, yang dikhususkan kepada matriks transformasi lanjar.

A. Aljabar Lanjar

Aljabar lanjar adalah studi tentang sistem persamaan lanjar dan sifat-sifat transformasinya. Aljabar linear memungkinkan untuk melakukan analisis rotasi dalam ruang, solusi dari pasangan persamaan diferensial, penentuan lingkaran melewati tiga titik yang diberikan, serta masalah-masalah lain yang terdapat dalam matematika, fisika, dan rekayasa. Aljabar lanjar berbeda dengan aljabar pada umumnya karena melibatkan vektor dan ruang vektor.

Matriks dan determinan adalah alat yang sangat berguna aljabar linier. Salah satu masalah utama dari aljabar linear adalah solusi dari persamaan matriks

$$\mathbf{Ax = b}$$

dengan solusi x. Secara teori, hal ini dapat diselesaikan dengan menggunakan invers matriks

$$\mathbf{x = A^{-1} b,}$$

teknik lain seperti eliminasi Gauss adalah teknik penghitungan yang lebih cepat.

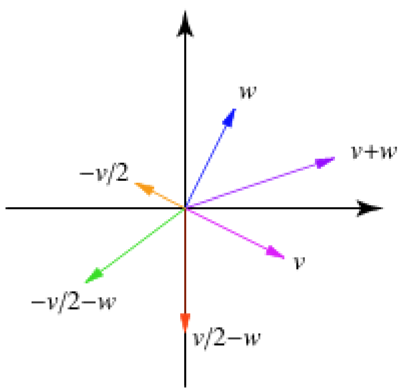
Selain digunakan untuk menggambarkan sistem linear persamaan, istilah aljabar lanjar juga digunakan untuk menggambarkan jenis tertentu dari aljabar. Secara khusus, aljabar lanjar **L** pada suatu bidang **F** memiliki struktur dari cincin dengan semua aksioma biasa untuk penjumlahan dan perkalian dalam dengan hukum distributif, sehingga memberikan struktur yang lebih dari cincin. Sebuah aljabar lanjar juga memakai operasi perkalian luar dengan skalar (yang merupakan elemen dasar dari bidang **F**).

B. Matriks Transformasi Lanjar

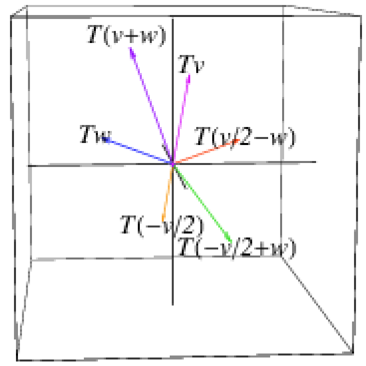
Sebuah transformasi lanjar antara dua ruang vektor **V** dan **W** adalah pemetaan **T : V → W** sehingga memiliki teorema berikut:

1. $T(\mathbf{v}_1 + \mathbf{v}_2) = T(\mathbf{v}_1) + T(\mathbf{v}_2)$ untuk vektor **v**1 dan **v**2 apapun pada ruang vektor **V**, dan
2. $T(\alpha\,\mathbf{v}) = \alpha\,T(\mathbf{v})$ untuk skalar α apapun.

Sebuah transformasi lanjar dapat atau tidak dapat berbentuk injektif atau surjektif. Saat V dan W mempunyai dimensi yang sama, dimungkinkan bahwa T dapat di-invers, yang berarti ada T^{-1} sedemikian sehingga $T\,T^{-1} = I$. Transformasi lanjar selalu memetakan garis vektor ke garis vektor (atau ke vektor nol).



Gambar 2: Penggambaran vektor pada koordinat Kartesius[6]



Gambar 3: Penggambaran vektor hasil transformasi T[6]

Contoh dasar dari sebuah transformasi lanjar adalah perkalian matriks. Diberikan sebuah matriks A berukuran $n \times m$, definisikan $T(\mathbf{v}) = \mathsf{A}\,\mathbf{v}$, dimana $\mathbf{v}$ adalah vektor berbentuk kolom (dengan m koordinat). Sebagai contoh, dimisalkan

$$\mathsf{A} = \begin{bmatrix} 0 & 1 \\ -2 & 2 \\ 1 & 0 \end{bmatrix},$$

dan T adalah fungsi transformasi lanjar dari $\mathbb{R}^2$ ke $\mathbb{R}^3$, didefinisikan sebagai berikut

$$T(x, y) = (y, -2x + 2y, x).$$

Saat V dan W memiliki dimensi yang terdefinisi, transformasi lanjar dapat dituliskan sebagai perkalian matriks hanya setelah basis ruang vektor untuk V dan W dispesifikasikan. Saat V dan W memiliki hasil kali dalam dan basis vektor ruangnya, $\{v_1, \ldots, v_m\}$ dan $\{w, \ldots, w_n\}$, orthonormal, akan mudah menuliskan matriks korespondensinya, yaitu $A = (a_{ij})$ dengan $a_{ij} = \langle w_i, T(v_j) \rangle$. Sebagai catatan bahwa saat menggunakan basis standar untuk $\mathbb{R}^n$ dan $\mathbb{R}^m$, kolom ke-j berkoresponden dengan vektor basis ke-j.

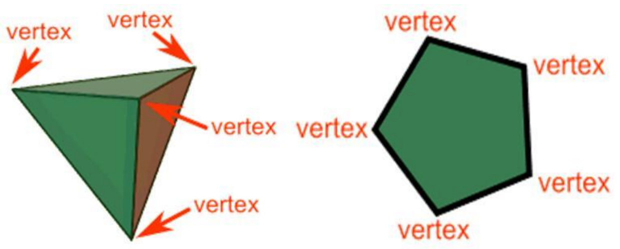
Saat V dan W memiliki dimensi yang tidak terdefinisi, maka dimungkinkan transformasi lanjar tidak kontigu. Sebagai contoh, misal V adalah ruang polinom dalam satu variabel, dan T dapat diturunkan. Maka $T(x^n) = n\,x^{n-1}$, di mana transformasi tersebut tidak kontigu karena $x^n / n \to 0$ saat $T(x^n / n)$ tidak konvergen.

III. Pembuatan Obyek Tiga Dimensi dengan Pemrograman

A. Latar Belakang

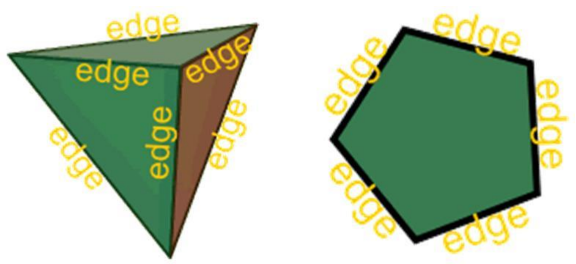
Membuat gambar tiga dimensi (perspektif) adalah kegiatan menuangkan ide atau angan-angan dalam ukuran tiga dimensi ke dalam kertas dua dimensi, baik dengan pengamatan langsung atau berdasarkan kenyataan. Dalam gambar benda yang lebih dekat dengan mata akan tergambar lebih besar bila dibandingkan dengan yang jauh dengan mata. Semakin jauh dari pandangan mata semakin kecil dan akhirnya menghilang di titik horizontal. Adapun bagian-bagian dalam objek tiga dimensi, yaitu

1. *Vertex*. Dalam objek 3 dimensi, *vertex* (plural : *Vertices*), dapat diistilahkan sebagai titik (jamak : simpul), adalah titik di mana dua atau lebih garis lurus bertemu.



Gambar 4: *Vertex*[1]

2. *Edge*. Dalam obyek tiga dimensi, *edge*, dapat diistilahkan sebagai rusuk, adalah ruas garis yang menghubungkan dua simpul.



Gambar 5: *Edge*[1]

3. *Face*. Dalam obyek tiga dimensi, *face*, dapat diistilahkan sebagai wajah/sisi/permukaan, adalah salah satu permukaan individual dari sebuah benda padat.

Gambar 6: *Face*[1]

Topologi di dalam dunia dimensi tiga dapat diartikan sebagai pemetaan bidang-bidang (*polygons/ faces*) yang dibangun dari setiap titik-titik (*vertices*) atau rusuk (*edge*) yang membentuk kesinambungan secara menyeluruh pada obyeknya. Sehingga pada prinsipnya adalah semua obyek benda dibangun dari minimal dua titik (*vertices*) yang membentuk rusuk (*edge*) dan minimal ada tiga rusuk yang akan membentuk satu sisi (*face*).

B. Titik dan Vektor

Sebelum membuat fungsi transformasi lanjar, harus dibuat definisi titik dan vektor terlebih dahulu. Gambar di bawah ini adalah definisi titik dan vektor dalam bahasa pemrograman.

```
Point Class
{
  Variables:
    num tuple[3]; //(x,y,z)

  Operators:
    Point AddVectorToPoint(Vector);
    Point SubtractVectorFromPoint(Vector);
    SubtractPointFromPoint(Point);

  Functions:
    //draw a point at its position tuple with your favorite graphics API
    drawPoint;
}

Vector Class
{
  Variables:
    num tuple[3]; //(x,y,z)

  Operators:
    Vector AddVectorToVector(Vector);
    Vector SubtractVectorFromVector(Vector);
}
```

Gambar 7: Pendefinisian tipe bentukan (*class*) titik dan vektor[4]

Dua tipe bentukan di atas selalu menjadi basis pembuatan gambar tiga dimensi, dengan tipe bentukan yang pertama adalah *point* (merepresentasikan nilai koordinat x,y, dan z), dan yang kedua adalah *vector* (jarak dan arah dari dua *point*).

C. Rotasi

Rotasi secara definisi adalah perpindahan objek dengan lintasan menyerupai lingkaran di sekitar titik rotasi. Titik pusat rotasi di dalam ruang dimensi tiga kita definisikan terdapat tiga kemungkinan: titik pada bidang XY, bidang XZ, atau bidang YZ (di mana setiap bidang dibentuk dari dua vektor basis).

Matriks rotasi tersebut kita definisikan sebagai berikut

- Matriks Rotasi XY

$$\begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Matriks Rotasi XZ

$$\begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

- Matriks Rotasi YZ

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}$$

Untuk merotasi suatu titik A dengan titik pusat rotasi di bidang XY sebesar 90 derajat ($\pi/2$), ikuti langkah-langkah di bawah ini.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} cos\frac{\pi}{2} & -sin\frac{\pi}{2} & 0 \\ sin\frac{\pi}{2} & cos\frac{\pi}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

$$= \begin{bmatrix} cos\frac{\pi}{2}a_0 + -sin\frac{\pi}{2}a_1 + 0a_2 \\ sin\frac{\pi}{2}a_0 + cos\frac{\pi}{2}a_1 + 0a_2 \\ 0a_0 + 0a_1 + 1a_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0a_0 + -1a_1 + 0a_2 \\ 1a_0 + 0a_1 + 0a_2 \\ 0a_0 + 0a_1 + 1a_2 \end{bmatrix}$$

$$= \begin{bmatrix} -a_1 \\ a_0 \\ a_2 \end{bmatrix}$$

Jika titik A bernilai (3,4,5), maka keluaran titik B adalah (−4,3,5).

D. Dilatasi

Dilatasi adalah proses membesarkan atau mengecilkan obyek. Proses ini lebih mudah dibandingkan dengan rotasi karena matriks transformasinya yang sangat praktis.

Sebuah transformasi dilatasi membutuhkan dua masukan: vektor dan sebuah 3-tuple dilatasi, yang mendefinisikan bagaimana vektor masukan diubah ukurannya berdasarkan tiga sumbu yang diberlakukan sebelumnya, yaitu x, y, dan z.

Sebagai contoh, tuple dilatasi sebagai (s0,s1,s2), s0 merepresentasikan dilatasi pada sumbu X, s1 dengan sumbu Y, dan s2 dengan sumbu Z.

Matriks transformasi dilatasi adalah seperti berikut (di mana s0, s1, and s2 adalah elemen dari 3-tuple dilatasi):

$$\begin{bmatrix} s0 & 0 & 0 \\ 0 & s1 & 0 \\ 0 & 0 & s2 \end{bmatrix}$$

Untuk membuat vektor masukan A (a0,a1,a2) menjadi dua kali lebih besar sepanjan sumbu X (di sini kita gunakan 3-tuple dilatasi S=(2,1,1)), operasi matematikanya akan menjadi seperti berikut:

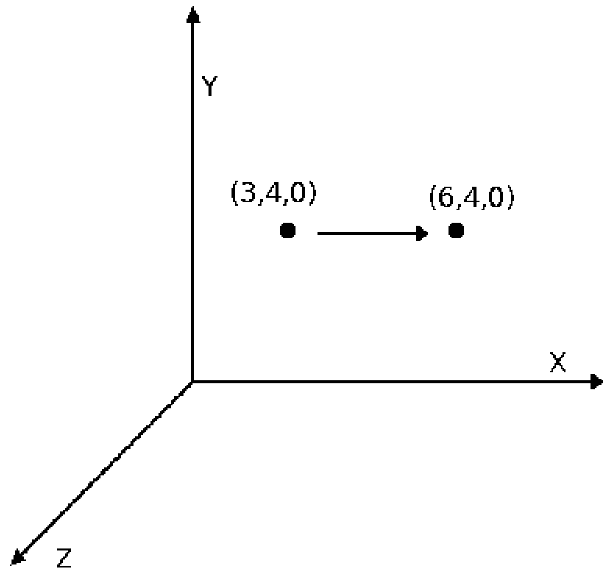
$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} s0 & 0 & 0 \\ 0 & s1 & 0 \\ 0 & 0 & s2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

$$= \begin{bmatrix} 2a_0 + 0a_1 + 0a_2 \\ 0a_0 + 1a_1 + 0a_2 \\ 0a_0 + 0a_1 + 1a_2 \end{bmatrix}$$

$$= \begin{bmatrix} 2a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

Jika diberikan vektor masukan **A**=(3,4,0), maka vektor keluaran **B** akan menjadi (6,4,0).



Gambar 8: Dilatasi vektor A menjadi vektor B[4]

E. Implementasi dalam Bahasa Pemrograman

Contoh programnya adalah sebagai berikut:

- Program akan menyimpan maksimal seratus *point* ke dalam *array*.
- Saat tombol D ditekan, program akan melakukan *clear screen* lalu menggambar ulang *point*.
- Saat tombol A ditekan, program akan melakukan dilatasi semua titik sebesar 0,5 kali.
- Saat tombol S ditekan, program akan melakukan dilatasi semua titik sebesar 2 kali.
- Saat tombol R ditekan, program akan melakukan rotasi kepada semua titik sebesar 15 derajat pada bidang XY.

- Saat tombol Esc ditekan, program akan berhenti dan keluar.

Tipe bentukannya adalah seperti berikut.

```
Point Class
{
    Variables:
        num tuple[3]; //(x,y,z)

    Operators:
        Point AddVectorToPoint(Vector);
        Point SubtractVectorFromPoint(Vector);
        Vector SubtractPointFromPoint(Point);
        //sets the current point's position to that of the inputted point
        Null SetPointToPoint(Point);
    Functions:
        //draw a point at its position tuple with your favorite graphics API
        drawPoint;
}

Vector Class
{
    Variables:
        num tuple[3]; //(x,y,z)

    Operators:
        Vector AddVectorToVector(Vector);
        Vector SubtractVectorFromVector(Vector);
        Vector RotateXY(degrees);
        Vector RotateYZ(degrees);
        Vector RotateXZ(degrees);
        Vector Scale(s0,s1,s2);
}
```

Gambar 9: Pendefinisian tipe bentukan (*class*) titik dan vektor yang diperbarui[4]

Fungsi transformasinya adalah sebagai berikut.

```
main{
    //setup for your favorite Graphics API here
    //setup for keyboard input (may not be required) here

    //create an array of 100 points
    Point Array  pointArray[100];

    for (int x = 0; x < pointArray.length; x++)
    {
        //Set its location to a random point on the screen
        pointArray[x].tuple = [random(0,screenWidth), random(0,screenHeight), random(0,desiredDepth));
    }

    //this function clears the screen and then draws all of the points
    function redrawScreen()
    {
        //use your Graphics API's clear screen function
        ClearTheScreen();

        for (int x = 0; x < pointArray.length; x++)
        {
            //draw the current point to the screen
            pointArray[x].drawPoint();
        }
    }

    // while the escape is not being pressed, carry out the main loop
    while (esc != pressed)
    {
        // perform various actions based on which key is pressed
        if (key('d') == pressed)
        {
            redrawScreen();
        }
        if (key('a') == pressed)
        {
            //create the space's origin as a point
            Point origin = new Point(0,0,0);

            Vector tempVector;
            for (int x = 0; x < pointArray.length; x++)
            {
                //store the current vector address for the point, and set the point
                tempVector = pointArray[x].subtractPointFromPoint(origin);
                //reset the point so that the scaled vector can be added
                pointArray[x].setPointToPoint(origin);
                //scale the vector and set the point to its new, scaled location
                pointArray[x].addVectorToPoint(tempVector.scale(0.5,0.5,0.5));
            }
            redrawScreen();
        }

        if(key('s') == pressed)
        {
            //create the space's origin as a point
            Point origin = new Point(0,0,0);

            Vector tempVector;
            for (int x = 0; x < pointArray.length; x++)
            {
                //store the current vector address for the point, and set the point
                tempVector = pointArray[x].subtractPointFromPoint(origin);
                //reset the point so that the scaled vector can be added
                pointArray[x].setPointToPoint(origin);
                //scale the vector and set the point to its new, scaled location
                pointArray[x].addVectorToPoint(tempVector.scale(2.0,2.0,2.0));
            }
            redrawScreen();
        }

        if(key('r') == pressed)
        {
            //create the space's origin as a point
            Point origin = new Point(0,0,0);
            Vector tempVector;
            for (int x = 0; x < pointArray.length; x++)
            {
                //store the current vector address for the point, and set the point
                tempVector = pointArray[x].subtractPointFromPoint(origin);
                //reset the point so that the scaled vector can be added
                pointArray[x].setPointToPoint(origin);
                //scale the vector and set the point to its new, scaled location
                pointArray[x].addVectorToPoint(tempVector.rotateXY(15));
            }
            redrawScreen();
        }
    }
}
```

Gambar 10: Fungsi transformasi lanjar[4]

IV. SIMPULAN

Membuat suatu obyek tiga dimensi tidaklah sulit. Walaupun tanpa GUI untuk memudahkan visualisasinya, obyek tiga dimensi dapat dibuat hanya dengan bahasa pemrograman yang sering kita pakai. Transformasi lanjar juga dapat kita buat dengan mudah apabila kita telah mengetahui dasar-dasarnya pada materi aljabar lanjar.

V. UCAPAN TERIMA KASIH

Pertama-tama penulis mengucapkan terima kasih kepada Allah SWT karena telah memberikan rahmat dan berkat sehingga penulis dapat menyelesaikan makalah ini. Terima kasih juga kepada dosen pengajar Mata Kuliah IF2123 Aljabar Geometri yaitu Drs. Judhi Santoso, M.Sc dan Dr. Ir. Rinaldi Munir, M.T. atas segala bimbingannya selama perkuliahan di semester III ini. Terima kasih juga kepada teman-teman mahasiswa Teknik Informatika yang selalu memberi dukungan dalam pengerjaan makalah ini.

REFERENSI

[1] http://oprekzone.com/pengertian-objek-3-dimensi/ diakses pada tanggal 15 Desember 2015 20:51 WIB.

[2] http://www.explainthatstuff.com/computer-graphics.html diakses pada tanggal 15 Desember 2015 21:32 WIB.

[3] http://www.animationcareerreview.com/articles/top-20-most-essential-software-artists-and-designers diakses pada tanggal 15 Desember 2015 21:41 WIB.

[4] http://gamedevelopment.tutsplus.com/tutorials/lets-build-a-3d-graphics-engine-linear-transformations--gamedev-7716 diakses pada tanggal 15 Desember 2015 22:55 WIB.

[5] http://mathworld.wolfram.com/LinearAlgebra.html diakses pada tanggal 16 Desember 2015 00:43 WIB.

[6] http://mathworld.wolfram.com/LinearTransformation.html diakses pada tanggal 16 Desember 2015 01:32 WIB.

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, 16 Desember 2015

Hafizh Dary Faridhan Hudoyo
13514072