Analisis Penerapan Aljabar Linear untuk Real-Time Shadow Rendering dalam Game 3D

Aryo Bama Wiratama – 13523088[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523088@std.stei.itb.ac.id, [2]aryobama09@gmail.com

***Abstract*— Makalah ini menganalisis penerapan aljabar linear dalam real-time shadow rendering untuk game 3D, dengan fokus pada teknik shadow mapping. Analisis mencakup dua jenis pencahayaan, yaitu point light dan directional light serta melihat perbedaannya dalam perhitungan bayangan. Menggunakan game engine Unity sebagai lingkungan pengujian, makalah ini mendemonstrasikan prinsip matematika di balik penentuan posisi bayangan melalui implementasi Python dan membandingkan hasilnya dengan sistem rendering bayangan pada game engine Unity. Hasil penelitian menunjukkan bahwa bayangan point light bergantung pada perhitungan posisi sumber cahaya, sementara bayangan directional light bergantung pada parameter rotasi, menjadikan keduanya sesuai untuk skenario pencahayaan yang berbeda dalam pengembangan game. Penelitian ini memberikan wawasan tentang bagaimana konsep aljabar linear, khususnya operasi vektor dan transformasi matriks, membentuk dasar sistem rendering bayangan game 3D modern.**

***Kata Kunci*---aljabar linear, game 3D, rendering real-time, shadow mapping, Unity engine**

I. pendahuluan

Di era modern saat ini, hampir semua aspek kehidupan telah beralih ke basis digital, salah satunya adalah aspek hiburan. Jika kita berkaca pada era sebelum adanya digitalisasi, anak – anak pada zaman tersebut biasanya mencari hiburan dengan bermain permainan tradisional, seperti petak-umpet, kelereng, layangan, dan masih banyak lagi. Zaman sekarang permainan tradisional sudah kurang relevan. Akibat digitalisasi, permainan anak – anak juga ikut berkembang dengan munculnya gim digital. Melalui gim digital, anak – anak dapat mendapat hiburan tanpa keluar rumah. Tidak cukup sampai situ saja, anak – anak bahkan dapat bermain atau berinteraksi dengan seluruh orang di dunia melalui gim digital.

Perkembangan gim digital dimulai pada tahun 1950-an. Pada awal tahun 1950, para ahli komputer menggunakan mesin elektronik untuk membuat sistem gim yang sederhana, seperti Bertie the Brain, mesin komputer yang dirancang khusus untuk bermain tic tac toe, atau Nimrod, untuk bermain nim. Kemudian permainan video semakin berkembang hingga pada tahun 1970 muncul permainan video arcade pertama. Nolan Bushnell bersama Ted Dabney menjadi orang pertama yang memiliki ide untuk membuat mesin permainan berbasis koin yang mengadaptasi game Spacewar! Kemudian game arcade menjadi semakin popouler di masyarakat hingga mencapai masa keemasannya di pertengahan tahun 1970 hingga tahun 1980. Pada tahun 1990 teknologi berkembang dengan pesat sehingga memengaruhi industri gim. Industri gim mengalami pergesaran dari grafik 2D menjadi grafik 3D yang lebih realistis.

Suksesnya industri gim membuat perusahaan pengembang gim di dunia berlomba – lomba menghadirkan gim yang menarik dengan memanfaatkan teknologi untuk menciptakan pengalaman bermain yang lebih berkesan. Mereka menghadirkan grafik yang lebih realistis, alur cerita yang mendalam, dan alur permainan yang inovatif sehingaa mampu menarik minat di semua kalangan. Dalam mengembangkan gim digital, para pengembang biasanya menggunakan suatu alat khusus yang biasa disebut *game engine* atau mesin permainan. *Game engine* adalah perangkat lunak yang dirancang khusus untuk pembuatan dan pengembangan gim. *Game engine* menyediakan berbagai macam pustaka yang memudahkan para pengembang untuk membuat fitur pada gim yang dikembangkannya, seperti rendering grafis, simulasi fisika dan masih banyak lagi. Terdapat beberapa game engine yang cukup populer di kalangan pengembang, yaitu unity, unreal engine, godot, dan cry engine.

Seperti yang telah disebutkan di atas, game engine mempunyai banyak pustaka yang membantu para pengembang untuk membuat fitur pada gim, salah satunya fitur shadowing. Fitur shadowing memungkinkan para pengembang untuk membuat bayangan yang realistis dalam gim mereka. Tujuan makalah ini adalah menganalisis penerapan aljabar linier pada fitur shadow game 3D. Makalah ini akan menggunakan salah satu game engine yang populer untuk eksperimen, yaitu unity. Alasannya adalah unity memiliki fitur shadowing untuk game 3D. Selain itu, unity memiliki antarmuka pengguna yang intuitif dan ramah bagi pemula sehingga memudahkan penulis dalam melakukan eksperimen.

II. LANDASAN TEORI

A. Matriks

1. Definisi matriks

Matriks adalah kumpulan angka atau elemen yang disusun dalam bentuk tabel dengan baris dan kolom. Matriks digunakan untuk merepresentasikan dan memecahkan sistem persamaan linear dan aplikasi lainnya dalam matematika dan ilmu komputer.

2. Operasi matriks

Operasi penjumlahan dan pengurangan

Jika kita memiliki dua matriks A dan B yang memiliki ukuran yang sama, yaitu m×n (m baris dan n kolom), maka penjumlahan matriks A dan B menghasilkan matriks C yang juga berukuran m×n

$$C = A + B$$

Jika

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \text{ dan } B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

Maka

$$C = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{pmatrix}$$

Perkalian matriks

Jika

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \text{ dan } B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

Maka

$$A \cdot B = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

3. Matriks tranpos

Matriks tranpos didapat dengan ara mengubah elemen – elemen pada baris menjadi kolom dan kolom menjadi baris.

Jika

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

Maka matriks tranpos dari A adalah

$$A^T = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix}$$

4. Determinan matriks

Hanya matriks persegi yang memiliki nilai determinan.

Untuk matriks 2 x 2

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

Determinannya dapat dihitung sebagai

$$\det(A) = a_{11}a_{22} - a_{12}a_{21}$$

Untuk matriks 3 x 3

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Determinanya dihitung sebagai

$$\det(A) = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

B. Vektor

5. Definisi vektor

Vektor adalah konsep fundamental dalam matematika, fisika, dan ilmu komputer yang digunakan untuk merepresentasikan besaran yang memiliki magnitude (besar) dan arah.

2. Notasi vektor

Vektor biasanya disimbolkan menggunakan huruf kecil yang dicetak tebal atau dengan menambahkan tanda panah di atasnya, terutama jika ditulis dengan tangan.

3. Operasi dasar vektor

Vektor memiliki 3 operasi dasar, yaitu penjumlahan, pengurangan, dan perkalian dengan skalar.

Penjumlahan dua vektor dapat dituliskan sebagai berikut.

$$\vec{v} = (\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n)$$
$$\vec{w} = (\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_n)$$
$$\vec{v} + \vec{w} = (\vec{v}_1 + \vec{w}_1, \vec{v}_2 + \vec{w}_2, \ldots + \vec{v}_n + \vec{w}_n)$$

Pengurangan dua vektor dapat dituliskan sebagai berikut.

$$\vec{v} + \vec{w} = (\vec{v}_1 + \vec{w}_1, \vec{v}_2 + \vec{w}_2, \ldots + \vec{v}_n + \vec{w}_n)$$

Perkalian skalar dengan vektor dapat dituliskan sebagai barikut

$$\vec{v} = (\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n)$$
$$k\vec{v} = (k\vec{v}_1, k\vec{v}_2, \ldots, k\vec{v}_n)$$

6. Panjang vektor

Panjang vektor dapat dihitung menggunakan norma

$$\|\vec{v}\| = \sqrt{\vec{v}_1^2 + \vec{v}_2^2 + \ldots + \vec{v}_n^2}$$

7. Vektor yang dibentuk dari dua titik

Misalkan $P(x_1, y_2)$ *dan* $Q(x_2, y_2)$, *maka*

$$\vec{v} = \overrightarrow{PQ} = \overrightarrow{OQ} - \overrightarrow{OP}$$
$$= (x_2 - x_1, y_2 - y_1)$$

8. Hasil product vektor

Vektor memiliki 2 hasil product, yaitu dot product dan cross product. Dot product akan menghasilkan skalar, sedangkan cross product akan menghasilkan vektor yang tegak lurus dengan kedua vektor.

Dot product dapat dituliskan sebagai berikut

$$\vec{u} \cdot \vec{v} = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n$$

Cross product dapat dituliskan sebagai determinan suatu matriks

Misal terdapat vektor berikut

$$\vec{c} = \vec{a} \times \vec{b}$$

Dimana

$$\vec{a} = (a_x, a_y, a_z), \quad \vec{b} = (b_x, b_y, b_z)$$

Maka hasil cross product dapat dihitung

$$\vec{c} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

Ekspansi determinannya

$$\vec{c} = \hat{i}(a_y b_z - a_z b_y) - \hat{j}(a_x b_z - a_z b_x) + \hat{k}(a_x b_y - a_y b_x)$$

Sehingga

$$\vec{c} = \big((a_y b_z - a_z b_y), (a_z b_x - a_x b_z), (a_x b_y - a_y b_x)\big)$$

C. Shadow Mapping

Teknik menciptakan bayangan pada game 3D yang akan digunakan dan dibahas pada makalah ini kali ini adalah teknik Shadow Mapping. Shadow mapping merupakan salah satu teknik grafis yang digunakan untuk menghasilkan bayangan realistis pada game 3D. Teknik Shadow mapping membutuhkan informasi dari dua perspektif, perspektif sumber cahaya dan perspektif kamera atau sudut pandang player.

Langkah – langkah untuk melakukan shadow mapping adalah sebagai berikut:

1. Tentukan posisi sumber cahaya dan posisi benda untuk menentukan vektor cahaya.

$\mathbf{T}$ = *Posisi objek di koordinat dunia* (x_T, y_T, z_T)

$\mathbf{L}$ = *Posisi sumber cahaya di koordinat dunia* (x_L, y_L, z_L)

$$\mathbf{V}_{Light} = \mathbf{T} - \mathbf{L}$$

Normalisasikan vektor cahaya

$$\widehat{V}_{Light} = \frac{\mathbf{V}_{Light}}{\|\mathbf{V}_{Light}\|}$$

2. Kemudian tentukan matrix pandangan cahaya. Untuk menentukan matrix pandangan cahaya, kita membutuhkan vektor kanan dan vektor atas.

Hitung vektor kanan (**R**)

$$\mathbf{U}_{world} = (0, 1, 0)$$

$$\mathbf{R} = \frac{\mathbf{U}_{world} \times \widehat{\mathbf{V}}}{\|\mathbf{U}_{world} \times \widehat{\mathbf{V}}\|}$$

Hitung vektor atas sesungguhnya

$$\mathbf{U} = \widehat{\mathbf{V}} \times \mathbf{R}$$

Bentuk matriks rotasi

$$M_{rot} = \begin{bmatrix} \mathbf{R}_x & \mathbf{R}_y & \mathbf{R}_z & 0 \\ \mathbf{U}_x & \mathbf{U}_y & \mathbf{U}_z & 0 \\ \widehat{\mathbf{V}}_x & \widehat{\mathbf{V}}_y & \widehat{\mathbf{V}}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Bentuk matriks translasi

$$M_{trans} = \begin{bmatrix} 1 & 0 & 0 & -\mathbf{V}_{Light\,x} \\ 0 & 1 & 0 & -\mathbf{V}_{Light\,y} \\ 0 & 0 & 1 & -\mathbf{V}_{Light\,z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Maka matrix pandangan cahaya bisa didapatkan dari hasil perkalian matrix rotasi dengan translasi

$$M_{view} = M_{rot} \times M_{trans}$$

$$M_{view} = \begin{bmatrix} \mathbf{R}_x & \mathbf{R}_y & \mathbf{R}_z & -(\mathbf{R} \cdot \mathbf{V}_{Light}) \\ \mathbf{U}_x & \mathbf{U}_y & \mathbf{U}_z & -(\mathbf{U} \cdot \mathbf{V}_{Light}) \\ \widehat{\mathbf{V}}_x & \widehat{\mathbf{V}}_y & \widehat{\mathbf{V}}_z & -(\widehat{\mathbf{V}} \cdot \mathbf{V}_{Light}) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Kemudian kita bentuk matrix proyeksi perspektif untuk sumber cahaya. Untuk membentuk matrix, kita harus menentukan empat parameter terlebih dahulu, yaitu *FOV*, *aspect ratio*, *near, dan far*

fov = *field of view (lebar sudut pandang)*
aspect = *rasio lebar dan tinggi viewport*
near = *jarak bidang terdekat dari titik proyeksi*
far = *jarak bidang terjauh dari titik proyeksi*

matrix proyeksi perspektif dapat dihitung sebagai berikut

$$P_{persp} = \begin{bmatrix} \frac{1}{\tan\left(\frac{fov}{2}\right) \cdot aspect} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan\left(\frac{fov}{2}\right)} & 0 & 0 \\ 0 & 0 & -\frac{far + near}{far - near} & -\frac{2far \cdot near}{far - near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

4. Setelah itu kalikan matrix proyeksi dengan matrix pandangan cahaya untuk mendapatkan matrix ruang cahaya. Matrix ini menggambarkan bagaimana cahaya memandang dunia dari sudut pandangnya sendiri. Matriks pandangan cahaya adalah matrix transformasi yang mengubah koordinat dunia (world space) ke koordinat ruang cahaya (light space)..

$$M_{LSpace} = P_{LPersp} \times M_{Lview}$$

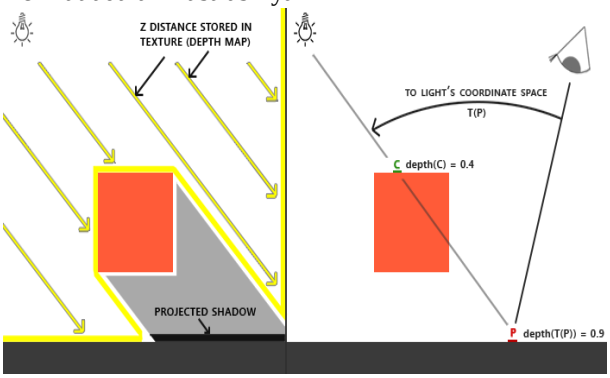
5. Setelah Mendapatkan matrix pandangan cahaya,

kita ubah semua posisi objek di ruang dunia ke ruang cahaya dengan cara mengalikannya dengan matrix pandangan cahaya.

$$T' = M_{LSpace} \times T$$
$$T' = \textit{Koordinat objek di ruang cahaya}$$

6. Setelah mendapatkan Koordinat objek di ruang cahaya, kita simpan nilai z pada koordinat T' untuk setiap pixel ke dalam *depth map*. Nilai z akan mewakili jarak objek terhadap sumber cahaya.
7. Setalah itu masuk ke perspektif kamera, untuk mendapatkan koordinat objek dalam ruang kamera tentukan terlabih dahulu matriks tranformasi untuk ruang kamera dengan mengulangi langkah 1 hingga langkah 4.
8. Setelah itu gunakan matriks tranformasi ruang kamera untuk mendapatkan posisi objek relatif terhadap kamera (C').
9. Kemudian tranformasikan koordinat tersebut ke ruang cahaya sehingga didapatkan kedalaman yang dilihat oleh kamera.
10. Setelah itu bandingkan kedalaman yang dilihat oleh kamera dengan kedalaman yang ada di depth map. Apabila kedalaman yang dilihat kamera lebih besar daripada di depth map, artinya posisi tersebut masuk berada di bayangan.

Berikut adalah ilustrasinya



Gambar 2.1 Ilustrasi penentuan bayangan dalam teknik map shadowing
(Sumber: https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping)

III. IMPLEMENTASI

A. *Lingkungan Implementasi*

Dalam melakukan implementasi, penulis menggunakan bahasa pemrograman python versi 3.11.0 dalam melakukan perhitungan. Selain itu, penulis juga menggunakan pustaka numpy versi 2.2.1 untuk memudahkan dalam melakukan perhitungan serta pustaka matplotlib versi 3.10.0 untuk visualisasi.

B. *Perhitungan*

Perhitungan ini bertujuan memprediksi posisi jatuhnya bayangan suatu benda. Benda yang diprediksi pada perhitungan ini adalah sebuah kubus.

Berikut langkah – langkah perhitungan:

1. Membuat fungsi untuk membentuk objek berbentuk kubus.

```python
def create_cube(center, size):
    """Membuat vertices dan edges untuk kubus"""

    x, y, z = center
    vertices = np.array([
        [x-size/2, y-size/2, z-size/2],
        [x+size/2, y-size/2, z-size/2],
        [x+size/2, y+size/2, z-size/2],
        [x-size/2, y+size/2, z-size/2],
        [x-size/2, y-size/2, z+size/2],
        [x+size/2, y-size/2, z+size/2],
        [x+size/2, y+size/2, z+size/2],
        [x-size/2, y+size/2, z+size/2]
    ])


    faces = [
        [vertices[0], vertices[1], vertices[2], vertices[3]],
        [vertices[4], vertices[5], vertices[6], vertices[7]],
        [vertices[0], vertices[1], vertices[5], vertices[4]],
        [vertices[2], vertices[3], vertices[7], vertices[6]],
        [vertices[1], vertices[2], vertices[6], vertices[5]],
        [vertices[0], vertices[3], vertices[7], vertices[4]]
    ]

    return vertices, faces
```

Gambar 3.1 Implementasi fungsi untuk membuat objek kubus
(Sumber: arsip penulis)

Fungsi di atas menerima parameter *center* dan *size*. *Center* adalah koordinat titik pusat suatu kubus, sedangkan *size* adalah panjang rusuk suatu kubus.

2. Membuat fungsi untuk menghitung posisi jatuhnya bayangan.

Perhitungan dimulai dari menentukan vektor arah cahaya yang dapat dihitung sebagai berikut

$$\mathbf{T} = \textit{Posisi objek di koordinat dunia } (x_T, y_T, z_T)$$
$$\mathbf{L} = \textit{Posisi sumber cahaya di koordinat dunia } (x_L, y_L, z_L)$$
$$\mathbf{V}_{Light} = \mathbf{T} - \mathbf{L}$$

Kemudian untuk mendapatkan nilai parameter t Gunakan formula berikut

$$t = \frac{(f_z - L_z)}{\mathbf{V}_{Light\,z}}$$

$$f_z = \textit{ketinggian lantai}$$

Kemudian gunaka formula berikut untuk mencari posisi bayangan

$$\mathbf{S} = \mathbf{L} + t \cdot \mathbf{V}_{Light}$$

$$\mathbf{S} = \textit{posisi bayangan}$$

Berikut implementasinya dalam fungsi pada python

```python
def calculate_shadow_point(light_source, point, floor_height=0):

    direction = point - light_source

    t = (floor_height - light_source[2]) / direction[2]

    shadow_point = light_source + t * direction
    shadow_point[2] = floor_height

    return shadow_point
```

Gambar 3.2 Implementasi fungsi untuk menghitung posisi bayangan
(Sumber: arsip penulis)

Fungsi di atas menerima tiga parameter, yaitu *light_source*, *point*, dan *floor_height* kemudian akan mengembalikan nilai posisi bayangan. *Light_source* adalah posisi cahaya dalam ruang dunia. *Point* adalah posisi benda. *Floor_height* adalah ketinggian (nilai z)
dari lantai yang berfungsi sebagai tempat jatuhnya bayangan.

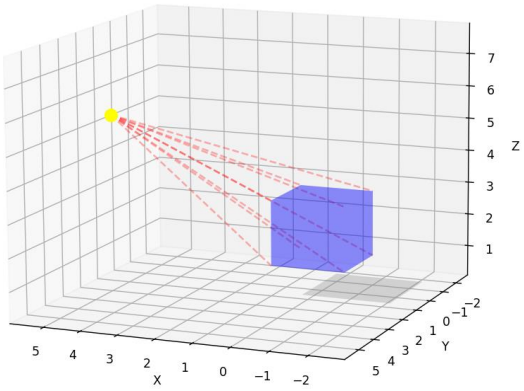
3. Selanjutnya, dibuat visualisasi untuk hasil perhitungan di atas.
   Berikut kode untuk membuat visualisasi

```python
def plot_cube_and_shadow(cube_center=[0, 0, 2], cube_size=2, light_source=[4, 4, 6]):

    fig = plt.figure(figsize=(10, 10))
    ax = fig.add_subplot(111, projection='3d')

    # Buat kubus
    vertices, faces = create_cube(cube_center, cube_size)

    cube = Poly3DCollection(faces, alpha=0.25)
    cube.set_facecolor('blue')
    ax.add_collection3d(cube)

    shadow_vertices = []
    light_source = np.array(light_source)

    print("\nPosisi vertex kubus dan bayangannya:")
    print("====================================")
    vertex_names = ['Depan-Kiri-Bawah', 'Depan-Kanan-Bawah', 'Belakang-Kanan-Bawah',
                    'Belakang-Kiri-Bawah', 'Depan-Kiri-Atas', 'Depan-Kanan-Atas',
                    'Belakang-Kanan-Atas', 'Belakang-Kiri-Atas']

for i, vertex in enumerate(vertices):
    shadow_point = calculate_shadow_point(light_source, vertex)
    shadow_vertices.append(shadow_point)

    # Print koordinat
    print(f"\nVertex {vertex_names[i]}:")
    print(f"  Posisi asli  : ({vertex[0]:.2f}, {vertex[1]:.2f}, {vertex[2]:.2f})")
    print(f"  Posisi bayang: ({shadow_point[0]:.2f}, {shadow_point[1]:.2f}, {shadow_point[2]:.2f})")

    ax.plot([light_source[0], vertex[0]],
            [light_source[1], vertex[1]],
            [light_source[2], vertex[2]],
            'r--', alpha=0.3)

shadow_vertices = np.array(shadow_vertices)

print(f"\nPosisi sumber cahaya: ({light_source[0]:.2f}, {light_source[1]:.2f}, {light_source[2]:.2f})")


    shadow_faces = [[shadow_vertices[0], shadow_vertices[1],
                     shadow_vertices[2], shadow_vertices[3]]]

    shadow = Poly3DCollection(shadow_faces, alpha=0.3)
    shadow.set_facecolor('gray')
    ax.add_collection3d(shadow)

    ax.scatter(*light_source, color='yellow', s=100, label='Light Source')

    # Set batas plot
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
```

```python
    ax.view_init(elev=20, azim=45)

    # Set batas axis
    ax.set_xlim([-3, 6])
    ax.set_ylim([-3, 6])
    ax.set_zlim([0, 8])

    plt.title('Cube Shadow Projection')
    plt.legend()
    plt.show()

plot_cube_and_shadow()
```

Gambar 3.3 Implementasi fungsi untuk visualisasi
(Sumber: arsip penulis)

Berikut contoh visualisasi dengan posisi sumber cahaya = (4, 4, 6), tinggi lantai = 0, dan posisi pusat kubus = (0, 0, 2).



Gambar 3.4 Hasil Visualisasi
(Sumber: arsip penulis)

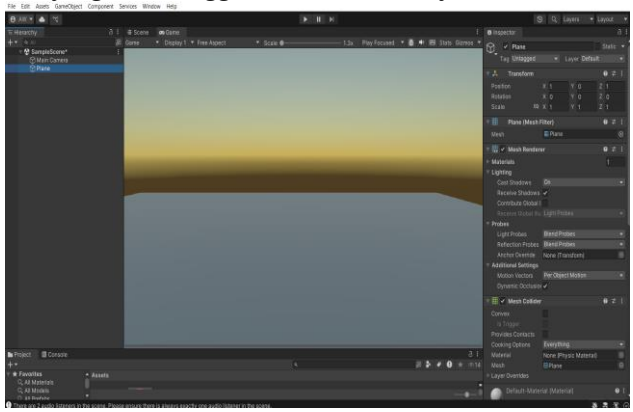
IV. EKSPERIMEN DAN PEMBAHASAN

A. Persiapan

Untuk melakukan eksperimen, penulis menggunakan *game engine* Unity versi 3.10.0. *Game engine* ini dipilih penulis karena memiliki antarmuka yang ramah pemula serta fitur yang lengkap. Dalam eksperimen ini, hasil perhitungan akan dibandingkan dengan hasil dari *game engine* Unity.

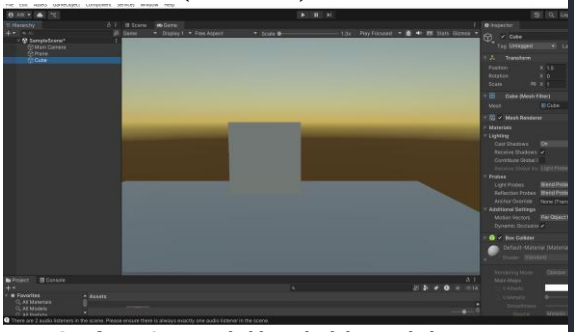
Langkah awal untuk melakukan eksprimen ini adalah melakukan persiapan pada unity. Berikut persiapan yang dilakukan.

1. Mempersiapkan lantai sebagai tempat jatuhnya bayangan. Ketinggian lantai diatur di y = 0.



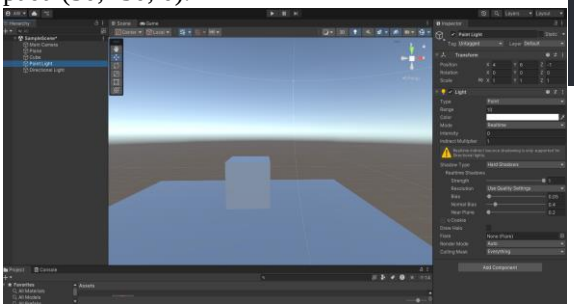
Gambar 4.1 Menambahkan lantai
(Sumber: arsip penulis)

2. Menambah suatu kubus dengan posisi titik pusat kubus berada di (1.5, 1.5, 0)

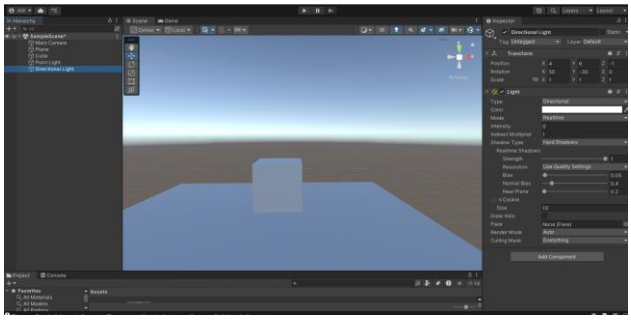


Gambar 4.2 Menambahkan objek berupa kubus
(Sumber: arsip penulis)

3. Menambah dua sumber cahaya yang berbeda. Cahaya pertama bertipe directional light, sedangkan cahaya kedua bertipe point. Posisi kedua sumber cahaya sama, yaitu (4, 6, -1). Pada cahay bertipe directional light atur rotation pada (50, -30, 0).



Gambar 4.3 Menambahkan cahaya bertipe point light
(Sumber: arsip penulis)



Gambar 4.4 Menambahkan cahaya bertipe direction light
(Sumber: arsip penulis)

Sebelum melakukan perbandingan terdapat penyesuaian pada posisi objek yang harus dimasukkan pada kode python yang telah dibuat. Pada kode python, ketinggian objek direpresentasikan oleh nilai pada sumbu-z , sedangkan pada unity ketinggian objek direpresentasikan oleh nilai pada sumbu-y. Oleh karena itu, sumbu-y pada unity akan menjadi sumbu-z pada kode python, begitupun sebaliknya.

B. Eksperimen

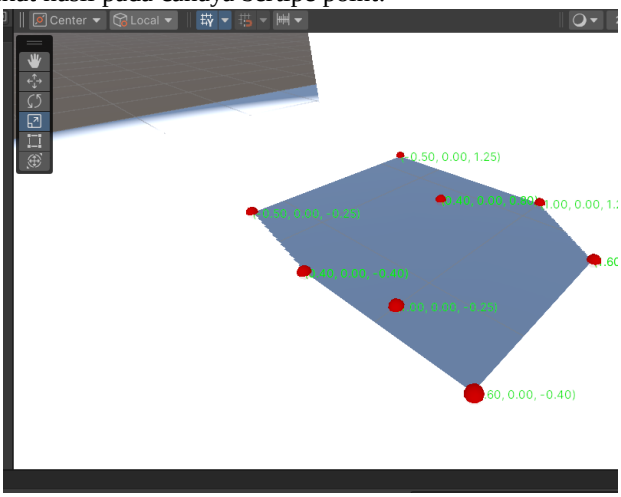
Posisi pusat kubus yang akan diuji adalah (1.5, 1.5, 0) dan posisi cahaya yang akan diuji adalah (4, 6, -1)

Berikut hasil kode python:

```
Posisi vertex kubus dan bayangannya:
==================================

Vertex Depan-Kiri-Bawah:
  Posisi asli  : (1.00, -0.50, 1.00)
  Posisi bayang: (0.40, -0.40, 0.00)

Vertex Depan-Kanan-Bawah:
  Posisi asli  : (2.00, -0.50, 1.00)
  Posisi bayang: (1.60, -0.40, 0.00)

Vertex Belakang-Kanan-Bawah:
  Posisi asli  : (2.00, 0.50, 1.00)
  Posisi bayang: (1.60, 0.80, 0.00)

Vertex Belakang-Kiri-Bawah:
  Posisi asli  : (1.00, 0.50, 1.00)
  Posisi bayang: (0.40, 0.80, 0.00)

Vertex Depan-Kiri-Atas:
  Posisi asli  : (1.00, -0.50, 2.00)
  Posisi bayang: (-0.50, -0.25, 0.00)

Vertex Depan-Kanan-Atas:
  Posisi asli  : (2.00, -0.50, 2.00)
  Posisi bayang: (1.00, -0.25, 0.00)

Vertex Belakang-Kanan-Atas:
  Posisi asli  : (2.00, 0.50, 2.00)
  Posisi bayang: (1.00, 1.25, 0.00)

Vertex Belakang-Kiri-Atas:
  Posisi asli  : (1.00, 0.50, 2.00)
  Posisi bayang: (-0.50, 1.25, 0.00)

Posisi sumber cahaya: (4.00, -1.00, 6.00)
```

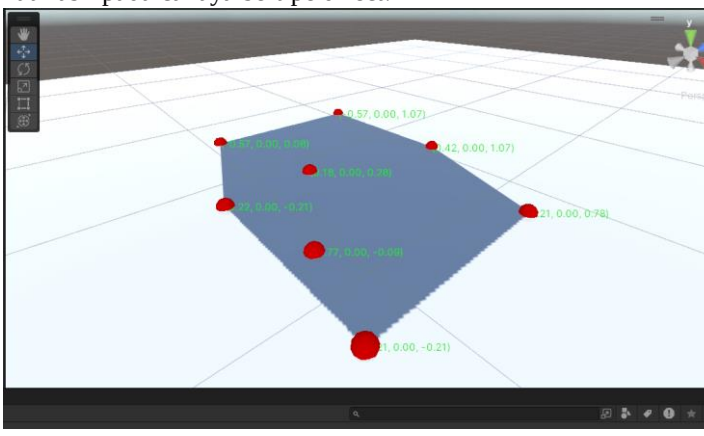
Gambar 4.5 Hasil perhitungan dengan kode python
(Sumber: arsip penulis)

Berikut hasil pada cahaya bertipe point:



Gambar 4.6 Hasil dari cahaya bertipe point light
(Sumber: arsip penulis)

Berikut hasil pada cahaya bertipe direct:



Gambar 4.7 Hasil dari cahaya bertipe direction light
(Sumber: arsip penulis)

Berikut tabel perbandingannya

Tabel 4.1 Perbandingan Posisi bayangan

Posisi asli kubus	Posisi bayangan kubus		
	Kode python	Point light	Directional light
(1.00, 1.00, -0.50)	(0.40, 0.00, -0.40)	(0.40, 0.00, -0.40)	(0.224, 0, -0.209)
(2.00, 1.00, -0.50)	(1.60, 0.00, -0.40)	(1.60, 0.00, -0.40)	(1.205, 0, -0.209)
(2.00, 1.00, 0.50)	(1.60, 0.00, 0.80)	(1.60, 0.00, 0.80)	(1.205, 0, 0.7774)
(1.00, 1.00, 0.50)	(0.40, 0.00, 0.80)	(0.40, 0.00, 0.80)	(0.424, 0, 1.066)
(1.00, 2.00, -0.50)	(-0.50, 0.00, -0.25)	(-0.50, 0.00, -0.25)	(-0.57, 0, 0.0774)
(2.00, 2.00, -0.50)	(1.00, 0.00, -0.25)	(1.00, 0.00, -0.25)	(0.773, 0, -0.086)
(2.00, 2.00, 0.50)	(1.00, 0.00, 1.25)	(1.00, 0.00, 1.25)	(0.184, 0, 0.283)
(1.00, 2.00, 0.50)	(-0.50, 0.00, 1.25)	(-0.50, 0.00, 1.25)	(-0.57, 0, 1.066)

C. Pembahasan

Pada tabel 4.1. posisi bayangan antara kode python dengan point light sama persis. Hal itu menandakan metode yang digunakan untuk penentuan bayangan point light pada unity dengan kode python sama sehingga bayangan yang dihasilkan juga sama. Sedangkan pada directioanl light memiliki posisi bayangan yang menandakan metode yang digunakan antara directional light dengan point light pada unity berbeda. Hal ini akibat pengaruh dari paramter *rotation* pada directioanl light. Di awal paramater *rotation* pada directional light diatur pada (50, -30, 0) yang dimana nilai tersebut merupakan nilai default dari unity. Pada directional light, penentuan bayangan tidak ditentukan berdasarkan posisi sumber cahaya. Hal itu terjadi karena directional light diasumsikan sebagai cahaya paralel yang mempunyai jarak yang sangat jauh dengan benda sehingga pengaturan posisi tidak akan berpengaruh. Untuk mengatur posisi bayangan pada directioanl light, paramater *rotation*. Oleh karena itu, directioal light biasanya dipakai oleh sumber cahaya yang sangat jauh, seperti matahari. Bayangan yang dihasilkan oleh matahari pada sore hari tentu berbeda dengan bayangan pada pagi hari. Perbedaan ini ditentukan oleh arah datangnya sinar matahari, yang diatur melalui parameter *rotation* pada Direct Light. Parameter ini memungkinkan untuk mengontrol sudut dan arah cahaya, sehingga menciptakan efek visual yang mencerminkan suasana pagi, siang, atau sore hari.

V. KESIMPULAN

Terdapat lebih dari satu teknik yang digunakan untuk menghasilkan shadow dalam game 3D. Shadow mapping merupakan salah satu contohnya. Kemudian terdapat 2 macam tipe cahaya, yaitu point light dan directional light. Point light digunakan untuk cahaya yang mempunyai jarak yang terbatas sehingga posisi sumber cahaya menentukan bayangan yang dihasilkan. Point light biasanya digunakan untuk menghasilkan cahaya dari lampu. Directional light digunakan untuk meghasilkan cahaya yang paralel dan cahay yang memiliki jarak yang sangat jauh. Oleh karena itu, penentuan bayangan bukan ditentukan oleh posisi sumber cahaya karena pada yang jarak yang sangat jauh posisi sumber cahay tidak berpengaruh. Oleh karena itu, directional light biasanya digunakan untuk cahaya yang dihasilkan oleh matahari.

VI. LAMPIRAN

Source code yang digunakan untuk melakukan perhitungan posisi bayangan
https://github.com/AryoBama/ImplementasiAlgeo/blob/main/main.py

VII. UCAPAN TERIMA KASIH

Puji syukur dipanjatkan kepada Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan karunia-Nya sehingga penyusunan makalah ini dapat diselesaikan dengan lancar. Penulis ingin menyampaikan rasa terima kasih kepada keluarga serta teman – teman karena atas dukungan mereka selama proses pengerjaan makalah sehingga makalah dapat selesai dengan baik. Ucapan terima kasih juga diucapkan kepada para pengajar mata kuliah IF2123, khususnya Dr. Ir. Rinaldi, M.T. selaku dosen pengampu kelas 02 yang telah memberikan ilmu yang bermanfaat dalam penyusunan makalah ini. Harapan penulis, makalah ini dapat menjadi sumber referensi yang berguna, baik bagi para pembelajar yang tertarik dengan bidang ilmu ini maupun sebagai acuan bagi penulis sendiri di masa mendatang.

VIII. REFERENSI

[1] Rinaldi Munir, “Vektor di Ruang Euclidean (bagian 1)”, https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2024.pdf (diakses 01 Januari 2025 pukul 20.00)

[2] Rinaldi Munir, “Review Matriks”, https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-01-Review-Matriks-2023.pdf (diakses 01 Januari 2025 pukul 20.10)
[3] https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping (diakses 01 Januari 2025 pukul 21.00)
[4] https://www.cg.tuwien.ac.at/courses/CG23/slides/tutorium/CG2LU_Tutorium.pdf (diakses 01 Januari 2025 pukul 21.20)
[5] https://www.cg.tuwien.ac.at/courses/CG23/slides/tutorium/CG2LU_Tutorium.pdf (diakses 01 Januari 2025 pukul 21.50)
[6] Smith, Alexander. “*They Create Worlds: The Story of the People and Companies That Shaped the Video Game Industry*”, Vol. I: 1971-1982 pp. 119–20, 188–91

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, 02 Desember 2025

Aryo Bama Wiratama, 13523088