LAPORAN PRAKTIKUM

Disusun untuk Memenuhi Tugas Individu Praktikum

Mata Kuliah GKV

Disusun oleh:

Nama : Aryela Rachma Davina

NIM : 24060122140174

LAB : B1

DEPARTEMEN INFORMATIKA

FAKULTAS SAINS DAN MATEMATIKA

UNIVERSITAS DIPONEGORO

2024

Essay Cahaya dan Shadow

Apa itu Cahaya?

Hasil nyata dari grafika komputer adalah gambar yang secara alamiah adalah objek yang pada langkah akhirnya harus bisa oleh mata manusia. Secara teoritis penglihatan 154 manusia sangat dipengahruhi oleh banyak hal, salah satunya adalah adanya cahaya. Tanpa adanya cahaya maka manusia tidak bisa melihat objek. Kuat lemahnya pencahayaan juga mempengaruhi objek yang dilihat. Selanjutnya objek yang dilihat dipersepsikan oleh otak. Cahaya dapat dilihat sebagai gelombang energi atau artikel (photon). Cahaya yang dipandang sebagai gelombang engergi dapat dibagi menjadi dua bagian besar yaitu:

1. Cahaya terlihat / tampak (visible light).
2. Cahaya tidak tampak (invisible light).

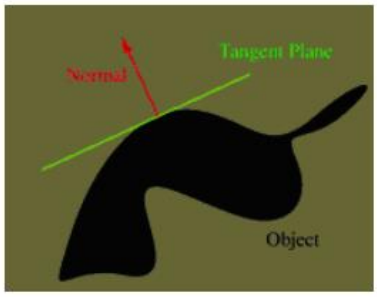
Cahaya tampak mempunyai panjang gelombang 390 s/d 720nm(nano meter). Mata kita hanya peka terhadap panjang gelombang 400 – 700nm. Cahaya tak tampak mempunyai panjang gelombang < 390nm atau panjang gelombang > 720nm. Gambar berikut memperlihatkan spektrum cahaya tampak dan tak tampak.

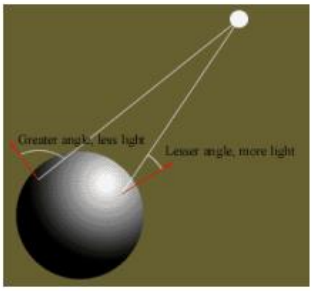
Sifat Materi Penyusun Benda

Sifat materi penyusun benda menentukan bagimana cahaya bereaksi terhadap materi penyusun benda. Secara umum cahaya yang mengenai permukaan suatu benda akan dipantulkan oleh permukaan benda tersebut. Gambar berikut mengilustrasikan perjalanan cahaya dari sumber cahaya. Berdasarkan kepada meteri penyusun benda maka ada tiga kemungkinan arah pentulancahaya ketika cahaya menimpa permukaan benda, yaitu:

1. Specular.

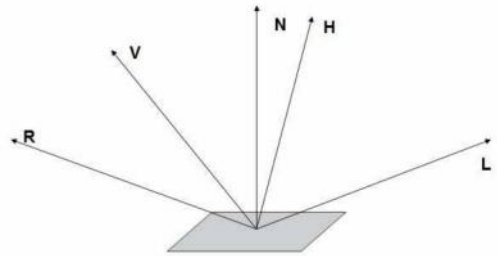
   Pada permukaan yang mulus, cahaya yang datang akan dipantulkan. Sesuai hukum pemantulan, sudut datang (angle of incident) akan sama dengan sudut yang dipantulkan. Dan apabila kita melihat dari sumber datanya cahaya maka kita melihat satu area yang relatif paling terang, area tersebut disebut dengan specular highlight.

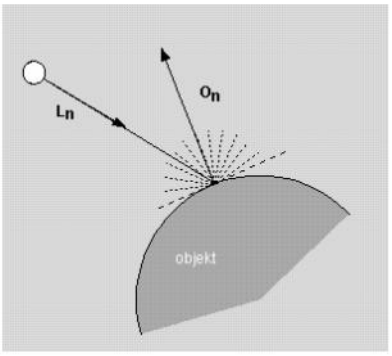




2. Diffuse.

   Diffuse merupakan sifat permukaan dimana cahaya yang datang dipantulkan ke segala arah, benda bersifat diffuse misalnya: batu, meja, tembok. Karena cahaya dipantulkan ke segala arah maka permukaan benda terlihat lebih kasar.





3. Translucent

   Benda yang mempunyai permukaan translucent akan meneruskan cahaya yang datang dan sekaligus memantulkan cahaya tersebut.

Model Sumber Cahaya

Sumber cahaya dapat dikelompokkan menjadi dua macam, yaitu:

1. Cahaya lingkungan (Ambient Light).

   Cahaya Lingkungan (Ambient Light) Didalam dunia nyata semua benda memantulkan cahaya meskipun sedikit, cahaya lingkungan digunakan untuk memodelkan cahaya yang berasal dari berbagai sumber tersebut. Cahaya ambient tidak mempunyai arah dan lokasi.

2. Cahaya Titik (Point Light)

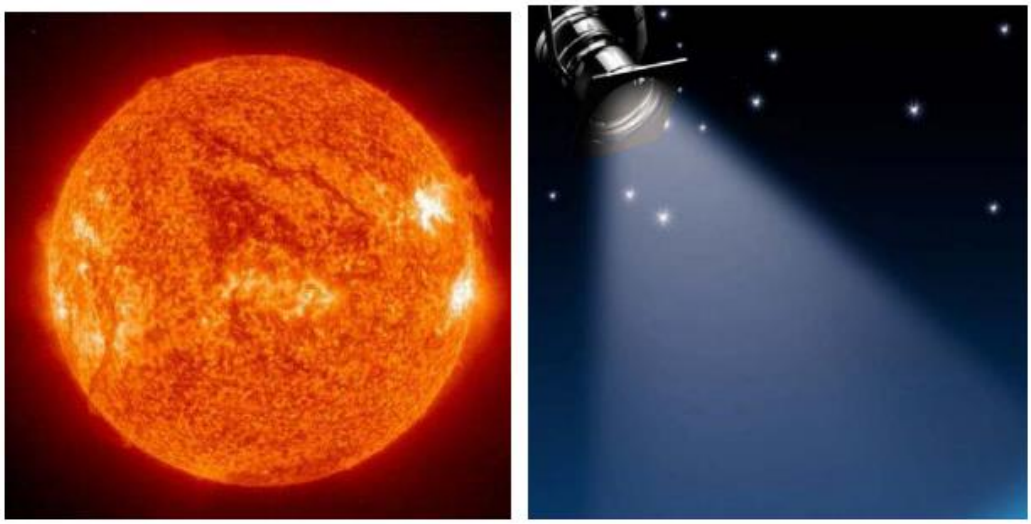
   Cahaya Titik (Point Light) Sumber cahaya ini mempunyi lokasi dan arah, dengan demikian jarak antara sumber cahaya terhadap benda akan berpengaruh terhadap kuat cahaya yang diterima oleh benda. Model cahaya ini dibedakan menjadi 2 bagian, yaitu:

   a. Directional

      Directional mempunyai krakteristik energi dari sumber tersebut menyebar ke semua arah dengan kekuatan yang sama. Contoh sumber cahaya ini adalah cahaya matahari.

   b. Positional

      Sumber cahaya ini mempunyai sifat dimana energi dari sumber cahaya tersebut akan melemah sebanding dengan jarak dan sudut terhadap sumber cahaya. Melemahnya kuat cahaya karena pengaruh jarak disebut attenuation. Apabila cahaya yang keluar dari sumber cahaya potensial dibatasi sudut penyebarannya maka akan memperoleh efek lampu sorot.



Gambar 10.8 Model Pencahayaan *Directional* dan *Positional*

Apa itu bayangan?

Dalam grafika komputer visual, bayangan (shadow) adalah efek optik yang ditampilkan untuk menirukan bagaimana objek melemparkan bayangan pada objek atau permukaan di sekitarnya. Efek bayangan penting untuk meningkatkan realisme dalam representasi visual, karena dalam kehidupan nyata, objek biasanya melemparkan bayangan tergantung pada posisi, bentuk, dan pencahayaan di sekitarnya.

Efek bayangan dalam grafika komputer tidak hanya menambahkan realisme visual, tetapi juga membantu memandu persepsi kedalaman dan orientasi dalam suatu adegan. Dengan menggunakan teknologi bayangan yang canggih, pengembang dapat menciptakan pengalaman visual yang lebih imersif dan mendalam bagi pengguna.

Jenis Bayangan

Ada beberapa jenis bayangan yang umum ditemui dalam grafika komputer visual, di antaranya:

1. Bayangan Tampak (Cast Shadows): Jenis bayangan ini terjadi ketika objek menghalangi sumber cahaya dan menciptakan bayangan pada objek atau permukaan di belakangnya. Bayangan ini seringkali dihasilkan dengan menggunakan algoritma seperti shadow mapping atau ray tracing.
2. Bayangan Terima (Receive Shadows): Bayangan ini muncul pada objek atau permukaan yang menerima cahaya yang dihalangi oleh objek lain. Ini menciptakan efek dimana objek terlihat "menangkap" bayangan yang dilemparkan oleh objek lain di sekitarnya.

3. Bayangan Lunak (Soft Shadows): Bayangan lunak terjadi ketika sumber cahaya tidak memiliki tepi yang tajam, sehingga bayangan yang dihasilkan memiliki tepi yang kabur atau terdifusi. Ini terjadi ketika cahaya dipantulkan atau ditempa oleh objek-objek lain di sekitarnya, sehingga menciptakan efek pencahayaan yang lebih realistis.
4. Bayangan Global (Global Shadows): Bayangan global mencakup efek bayangan yang melibatkan interaksi antara beberapa sumber cahaya, objek, dan material di dalam suatu adegan. Ini sering melibatkan teknik rendering kompleks seperti ray tracing.

Mengapa perlu mengimplementasikan cahaya dan bayangan

Implementasi atau simulasi pencahayaan pada objek 3D penting karena hal tersebut memberikan tingkat realisme yang lebih tinggi pada hasil akhir. Pencahayaan adalah salah satu aspek penting dalam visualisasi grafis karena tidak hanya dapat memberikan dimensi dan tekstur pada objek melainkan juga menciptakan permainan kontras yang menarik sehingga mempengaruhi tampilan, suasana, dan persepsi sebuah objek atau lingkungan.

Berikut adalah beberapa alasan mengapa implementasi atau simulasi pencahayaan penting dalam objek 3D:

1. Realisme: Pencahayaan yang tepat dapat membuat objek 3D terlihat lebih nyata dan alami. Pencahayaan yang buruk atau tidak akurat dapat membuat objek terlihat datar atau tidak beraturan.
2. Atmosfer: Pencahayaan juga dapat membantu menciptakan atmosfer yang sesuai dengan suasana yang diinginkan. Misalnya, pencahayaan yang lembut dan hangat dapat menciptakan suasana yang romantis atau nyaman, sementara pencahayaan yang tajam dan kontras dapat menciptakan suasana yang dramatis atau menakutkan.
3. Memperjelas Detail: Pencahayaan yang baik dapat membantu memperjelas detail-detail pada objek 3D, sehingga memudahkan penonton untuk melihat dan menghargai desain atau karya tersebut.
4. Pengaruh Emosi: Pencahayaan dapat memengaruhi emosi penonton. Dengan memanfaatkan efek pencahayaan yang tepat, penggambar dapat mengarahkan perhatian penonton pada bagian-bagian tertentu dari sebuah karya atau menciptakan suasana emosional yang diinginkan.

5. Konsistensi Visual: Dalam lingkungan virtual yang kompleks, konsistensi pencahayaan sangat penting untuk memastikan bahwa semua objek dalam suatu adegan terlihat seperti bagian dari lingkungan yang sama.
6. Komunikasi Informasi: Pencahayaan juga dapat digunakan untuk membantu mengkomunikasikan informasi kepada penonton. Misalnya, pencahayaan yang fokus pada objek tertentu dapat menyoroti objek tersebut sebagai pusat perhatian.

Simulasi pada shader

1. Vertex Shader

   Vertex shader adalah jenis shader yang paling awal dieksekusi dalam pipeline rendering. Tugas utamanya adalah memanipulasi posisi setiap titik (vertex) dalam objek 3D. Misalnya, vertex shader dapat digunakan untuk mentransformasi koordinat objek dari ruang objek ke ruang kamera, menerapkan efek deformasi, atau mengatur warna dan tekstur.

```
#version 330 core

// Input vertex data, different for all executions of this shader.
layout(location = 0) in vec3 vertexPosition_modelspace;
layout(location = 1) in vec2 vertexUV;
layout(location = 2) in vec3 vertexNormal_modelspace;

// Output data ; will be interpolated for each fragment.
out vec2 UV;
out vec3 Position_worldspace;
out vec3 Normal_cameraspace;
out vec3 EyeDirection_cameraspace;
out vec3 LightDirection_cameraspace;

// Values that stay constant for the whole mesh.
uniform mat4 MVP;
uniform mat4 V;
uniform mat4 M;
uniform vec3 LightPosition_worldspace;
```

```
void main(){

    // Output position of the vertex, in clip space : MVP * position
    gl_Position =  MVP * vec4(vertexPosition_modelspace,1);

    // Position of the vertex, in worldspace : M * position
    Position_worldspace = (M * vec4(vertexPosition_modelspace,1)).xyz;

    // Vector that goes from the vertex to the camera, in camera space.
    // In camera space, the camera is at the origin (0,0,0).
    vec3 vertexPosition_cameraspace = ( V * M *
vec4(vertexPosition_modelspace,1)).xyz;
    EyeDirection_cameraspace = vec3(0,0,0) - vertexPosition_cameraspace;

    // Vector that goes from the vertex to the light, in camera space. M is
ommited because it's identity.
    vec3 LightPosition_cameraspace = ( V *
vec4(LightPosition_worldspace,1)).xyz;
    LightDirection_cameraspace = LightPosition_cameraspace +
EyeDirection_cameraspace;

    // Normal of the the vertex, in camera space
    Normal_cameraspace = ( V * M * vec4(vertexNormal_modelspace,0)).xyz; //
Only correct if ModelMatrix does not scale the model ! Use its inverse
transpose if not.

    // UV of the vertex. No special space for this one.
    UV = vertexUV;
}
```

2. Fragment shader

   Fragment shader, juga dikenal sebagai pixel shader, beroperasi pada level piksel. Setelah vertex shader selesai, objek 3D dirender sebagai segitiga atau poligon. Fragment shader diterapkan pada setiap piksel yang dihasilkan dari triangulasi objek 3D. Tugasnya termasuk menentukan warna, tekstur, dan efek visual lainnya untuk setiap piksel. Fragment shader

juga dapat menghasilkan efek kompleks seperti bayangan, efek cahaya, refleksi, dan efek partikel.

```
#version 330 core

// Interpolated values from the vertex shaders
in vec2 UV;
in vec3 Position_worldspace;
in vec3 Normal_cameraspace;
in vec3 EyeDirection_cameraspace;
in vec3 LightDirection_cameraspace;

// Output data
out vec3 color;

// Values that stay constant for the whole mesh.
uniform sampler2D textureSampler;
uniform mat4 MV;
uniform vec3 LightPosition_worldspace;

void main(){

    // Light emission properties
    // You probably want to put them as uniforms
    vec3 LightColor = vec3(1,1,1);
    float LightPower = 100.0f;

    // Material properties
    vec3 MaterialDiffuseColor = texture( textureSampler, UV ).rgb;
    vec3 MaterialAmbientColor = vec3(0.1,0.1,0.1) * MaterialDiffuseColor;
    // vec3 MaterialAmbientColor =  MaterialDiffuseColor;
    vec3 MaterialSpecularColor = vec3(0.3,0.3,0.3);

    // Distance to the light
    float distance = length( LightPosition_worldspace -
Position_worldspace);
```

```
    // Normal of the computed fragment, in camera space
    vec3 n = normalize( Normal_cameraspace );

    // Direction of the light (from the fragment to the light)
    vec3 l = normalize( LightDirection_cameraspace );
    // Cosine of the angle between the normal and the light direction,
    // clamped above 0
    //  - light is at the vertical of the triangle -> 1
    //  - light is perpendicular to the triangle -> 0
    //  - light is behind the triangle -> 0
    float cosTheta = clamp( dot( n,l ), 0,1 );

    // Eye vector (towards the camera)
    vec3 E = normalize(EyeDirection_cameraspace);

    // Direction in which the triangle reflects the light
    vec3 R = reflect(-l,n);

    // Cosine of the angle between the Eye vector and the Reflect vector,
    // clamped to 0
    //  - Looking into the reflection -> 1
    //  - Looking elsewhere -> < 1
    float cosAlpha = clamp( dot( E,R ), 0,1 );

    color =
        // Ambient : simulates indirect lighting
        MaterialAmbientColor +
        // Diffuse : "color" of the object
        MaterialDiffuseColor * LightColor * LightPower * cosTheta
/(distance*distance) + //; // +
        // Specular : reflective highlight, like a mirror
        MaterialSpecularColor * LightColor * LightPower * pow(cosAlpha,5)
/(distance*distance);

}
```