Nama : Muhammad Raihan Wardhana

NIM : 24060122140166

Lighting dan Shadow - Laporan Praktikum GKV

1. Lighting

What (Apa itu Lighting?)

Dalam konteks Gravika Komputer dan Visual (GKV), "Lighting" merujuk pada proses dan teknik pencahayaan yang digunakan dalam produksi grafika komputer dan visual. Pencahayaan adalah aspek kunci dalam pembuatan grafika komputer realistik dan menarik. Ini melibatkan pengaturan cahaya dan bayangan untuk memberikan dimensi, kedalaman, dan nuansa yang sesuai dengan tujuan artistik atau tujuan komunikasi dari karya tersebut. Dalam GKV, teknik pencahayaan yang baik dapat membuat objek atau adegan terlihat lebih hidup, memberikan kesan emosi yang diinginkan, serta meningkatkan kesan visual secara keseluruhan.

Why (Mengapa mengimplementasi/mensimulasikan Lighting?)

Implementasi pencahayaan dalam grafika komputer memiliki signifikansi yang tak terbantahkan karena:

- **Realisme**: Pencahayaan menambahkan unsur realisme pada objek virtual, memperkuat kesan keberadaannya dan menarik minat pengguna.
- **Kedalaman dan Dimensi**: Dengan bantuan cahaya, ruang virtual dapat diberi kedalaman dan dimensi, memperkaya pengalaman visual dengan objek yang terlihat lebih hidup dan terlibat dengan lingkungan sekitarnya.
- **Atmosfer dan Mood**: Pencahayaan memungkinkan penciptaan atmosfer dan mood yang spesifik dalam suatu adegan, seperti suasana malam yang gelap atau kecerahan siang hari.
- **Visualisasi Data**: Dalam aplikasi visualisasi data, pencahayaan membantu dalam penyorotan informasi penting dan memudahkan pemahaman visualisasi secara keseluruhan.
- **Estetika**: Peran pencahayaan dalam menentukan nuansa warna dan komposisi secara keseluruhan memengaruhi estetika sebuah karya grafis, menjadikannya penting dalam mencapai hasil akhir yang memikat.

How (Bagaimana cara mengimplementasikannya?)

Ada beberapa metode untuk menerapkan pencahayaan dalam grafika komputer, termasuk:

- **Pencahayaan Difus**: Jenis pencahayaan ini terjadi ketika cahaya tersebar merata di semua arah setelah memantul dari permukaan yang kasar atau tidak rata. Untuk menerapkannya, perlu memperhitungkan sudut antara vektor normal permukaan dan arah cahaya. Intensitas cahaya di titik tertentu pada permukaan dipengaruhi oleh sudut antara vektor normal dan arah cahaya.

- **Pencahayaan Spekular**: Pencahayaan ini terjadi ketika cahaya dipantulkan secara teratur dari permukaan yang sangat halus atau mulus, menciptakan efek kilau atau sorotan pada objek. Untuk menerapkannya, perlu memperhitungkan sudut antara arah pandangan kamera (atau mata) dan arah cahaya yang dipantulkan.

- **Pencahayaan Ambien**: Jenis pencahayaan ini merujuk pada cahaya yang tersebar merata di sekitar lingkungan tanpa sumber cahaya yang jelas. Biasanya, nilai ambien yang tetap diberikan untuk memberikan efek pencahayaan minimal pada objek, independen dari sumber cahaya utama.

- **Pencahayaan Terarah**: Menggunakan sumber cahaya dengan arah tertentu, seperti sinar matahari, untuk memberikan iluminasi pada objek.

- **Pencahayaan Titik**: Mensimulasikan sumber cahaya tersebar dari titik tertentu dalam ruang, seperti lampu pijar, yang menciptakan bayangan dan sorotan pada objek.

- **Pencahayaan Area**: Menggunakan sumber cahaya dengan area tertentu, seperti layar, untuk menciptakan efek pencahayaan yang halus dan alami.

- **Pencahayaan Global**: Mengambil kalkulasi interaksi cahaya yang kompleks antara objek dalam adegan, termasuk pemantulan dan pembiasan cahaya dari berbagai permukaan.

Implementasi (Diffuse , Specular dan Ambient)

1. **Vertex Shader**

```
#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;


// Values that stay constant for the whole mesh.
uniform mat4 MVP;

uniform mat4 V;
uniform mat4 M;


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;


     // 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

```
#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;
```

```
        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 mirro



             MaterialSpecularColor * LightColor * LightPower *
pow(cosAlpha,5) / (distance*distance);


}
```

+ //;

// +

Shadow

What (Apa itu Shadow?)

Bayangan (shadow) dalam grafika komputer adalah efek visual yang dihasilkan ketika cahaya terhalang oleh suatu objek, menciptakan area gelap di belakang objek tersebut. Dalam dunia nyata, bayangan memberikan informasi tentang kedalaman dan posisi relatif objek dalam ruang. Dalam konteks grafika komputer, pembuatan bayangan membutuhkan pemodelan interaksi kompleks antara cahaya, objek, dan pencahayaan untuk menciptakan ilusi realisme.

Why (Mengapa Menerapkan Bayangan?)

Implementasi bayangan penting karena:

1. **Realisme**: Bayangan meningkatkan realisme visual dalam simulasi grafis dengan menciptakan ilusi kedalaman dan dimensi dalam sebuah adegan.
2. **Kedalaman**: Bayangan membantu memperjelas posisi relatif objek dalam ruang, membantu pemirsa memahami struktur dan jarak antar objek.
3. **Estetika**: Efek bayangan dapat digunakan untuk menciptakan komposisi visual yang menarik dan menambahkan nuansa dramatis atau artistik pada karya grafis.
4. **Informasi Spatial**: Bayangan memberikan informasi penting tentang bentuk dan posisi objek, memungkinkan pengguna untuk membuat keputusan yang lebih tepat dalam aplikasi seperti simulasi, permainan, dan desain.

How (Bagaimana Implementasi Bayangan?)

Terdapat beberapa teknik untuk mengimplementasikan bayangan dalam grafika komputer, di antaranya:

1. **Shadow Mapping**: Teknik yang paling umum digunakan untuk membuat bayangan dalam permainan video dan aplikasi 3D. Ini melibatkan membuat peta kedalaman (depth map) dari perspektif sumber cahaya, dan kemudian membandingkan posisi titik pandang pengamat dengan peta kedalaman untuk menentukan apakah suatu titik terkena bayangan atau tidak.

2. **Shadow Volumes**: Teknik ini menggunakan informasi geometris untuk menciptakan volume bayangan di sekitar objek yang memancarkan bayangan. Ini melibatkan pemotongan objek yang menerima bayangan dengan volume bayangan yang dihasilkan oleh sumber cahaya, menciptakan ilusi bayangan di permukaan objek tersebut.

    1. **Ray Tracing**: Metode ini menghasilkan bayangan dengan melacak jalur sinar cahaya dari sumber cahaya ke objek, dan kemudian menentukan apakah jalur sinar tersebut terhalangi oleh objek lain di sepanjang jalurnya. Meskipun ray tracing menghasilkan hasil yang sangat realistis, ini seringkali membutuhkan waktu komputasi yang lebih lama daripada teknik lainnya.