LIGHTING AND SHADOW

Disusun untuk Memenuhi Tugas Individu pada praktikum
Mata Kuliah Grafika dan Komputasi Visual Laboratorium B1

Disusun oleh:

Indana Najwa Ramadhani

24060122130070

DEPARTEMEN ILMU KOMPUTER/INFORMATIKA

FAKULTAS SAINS DAN MATEMATIKA

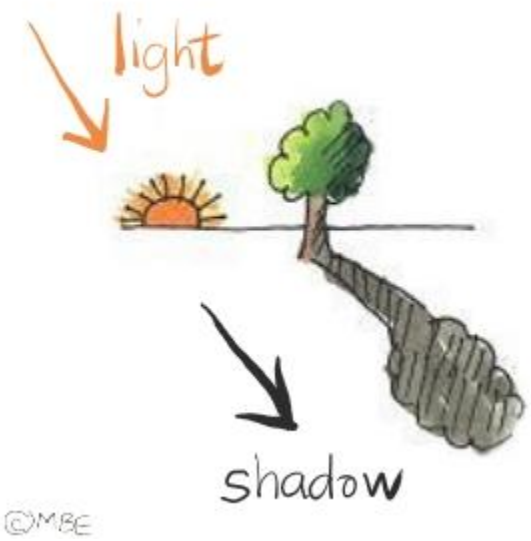
UNIVERSITAS DIPONEGORO

2024

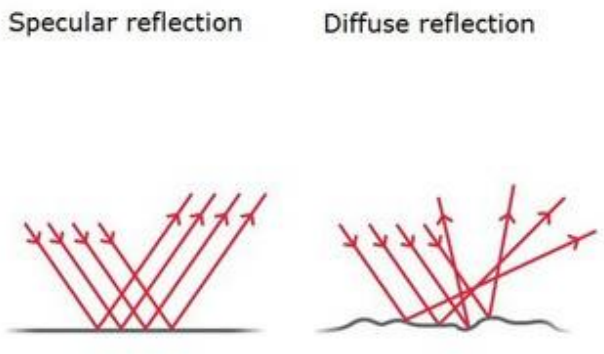
Lighting & Shadow

1. Mengapa Perlu Adanya Pencahayaan dan Bayangan?

Bayangan kerap kali dikaitkan dengan pencahayaan karena kehadiran cahaya yang memicu timbulnya bayangan. Cahaya adalah sinar atau terang yang berasal dari sesuatu yang bersinar, yang memungkinkan mata menangkap bayangan benda-benda disekitarnya. Ketika suatu cahaya menyinari suatu permukaan, secara otomatis cahaya akan memantulkan pantulannya pada permukaan tersebut. Namun, terdapat permukaan tertentu yang tidak menerima cahaya secara langsung, permukaan itulah yang disebut bayangan.



Ketika sinar cahaya menabrak suatu permukaan yang halus dan tembus pandang, sinar-sinar itu akan memantul kembali dari permukaan tersebut. Sehingga terjadi fenomena yang dikenal sebagai refleksi cahaya, di mana cahaya dipantulkan dari suatu permukaan saat bertemu dengan permukaan tersebut. Dalam refleksi cahaya, terdapat dua jenis utama, yaitu refleksi specular dan refleksi diffuse. Perhatikan gambar di bawah.



a. Specular Reflection

Specular Reflection atau biasa disebut dengan pemantulan teratur adalah pemantulan yang terjadi ketika cahaya direfleksikan sepenuhnya dari permukaan pada sudut yang sama. Refleksi ini akan menciptakan pantulan objek secara sempurna, seperti pada permukaan cermin.

b. Diffuse Reflection

Berbeda dengan specular reflection, diffuse reflection terjadi ketika cahaya menembus permukaan, yang kemudian dipantulkan oleh mikro struktur di permukaan tersebut. Hal ini menciptakan sudut-sudut yang tidak teratur dan mengakibatkan gambar terlihat kabur.

2. Mengapa Mengimplementasi/Mensimulasikan Lighting Penting?

Pencahayaan tidak hanya memberikan suasana tambahan yang memberikan dimensi dan tekstur pada objek, tetapi juga memainkan peran dalam menciptakan permainan kontras yang menarik. Dengan pengaturan yang tepat, pencahayaan dapat menyoroti detail penting, menciptakan atmosfer yang sesuai dengan cerita, dan meningkatkan pengalaman pemain.

Sedangkan, bayangan juga merupakan elemen penting dalam proses pembuatan estetika dan narasi visual. Bayangan dapat digunakan untuk menambahkan kedalaman, memunculkan perasaan misteri, atau menyoroti aspek tertentu dari lingkungan game. Dengan mengatur bayangan secara tepat, objek akan menciptakan efek realistis.

Pencahayaan dan bayangan tidak hanya penting untuk memberikan dimensi visual pada objek, tetapi juga untuk menciptakan suasana yang sesuai dalam sebuah grafik. Cahaya yang dipancarkan dapat memberikan focus pada detail tertentu, sementara pada bayangan akan memberikan kesan realisme terhadap objek. Dengan adanya bayangan, objek akan terlihat memiliki kedudukan yang mempertegas hubungan objek dengan lingkungan sekitar.

3. Cara Mengimplementasikan Lighting dan Shadow

Untuk menciptakan efek 3D, tidak cukup hanya mengandalkan bayangan, namum perlu adanya penerapan shading. Shading memainkan peran penting untuk mengatur warna, kecerahan, dan tekstur objek, sehingga menciptakan ilusi kedalaman dan dimensi yang lebih realistis (daerah terang dan gelap).

Shader merupakan bagian dari program yang mengatur tampilan dan efek visual dalam suatu grafis. Shader digunakan untuk mensimulasikan efek cahaya seperti pencahayaan, bayangan, hingga refleksi. Vertex shader digunakan untuk mengubah posisi titik (vertex) dari world space ke camera space, serta menghitung dan mengatur posisi untuk menghasilkan efek cahaya yang realistis. Perhatikan kode berikut.

```
#version 330 core

// Input vertex data.
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;
}
```

Pada kode di atas, terdapat 3 kode utama yang digunakan untuk mengatur posisi dari tiap-tiap titik (vertex) dalam sebuah objek.

1. LightPosition_cameraspace

   Menunjukkan arah cahaya dari titik vertex ke kamera dalam camera space.
2. Normal_cameraspace

   Normal (tegak lurus) dari permukaan di sekitar vertex dalam camera space.
3. EyeDirection_cameraspace

   Menunjukkan arah dari vertex ke camera (untuk menghitung specular reflection) dalam camera space.

Salah satu bagian lain dari shader adalah Fragment Shader. Fragment shader digunakan untuk menghitung warna dan property lain dari setiap fragmen (pixel) dalam sebuah objek. Frgament shader menghitung beberapa efek cahaya seperti diffuse, specular, dan ambient.

a. Diffuse

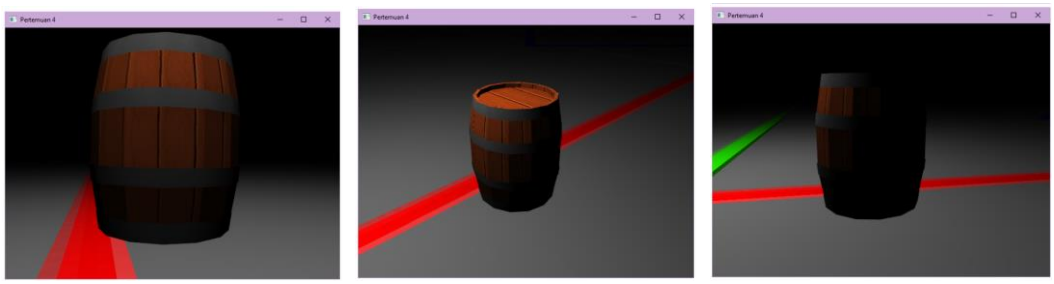
Diffuse reflection adalah proses untuk menghitung seberapa banyak cahaya yang dipantulkan oleh permukaan benda ke arah mata kita. Diffuse dihitung berdasarkan sudut insiden. Apabila sudut insiden lebih kecil (tegak lurus dengan benda), maka sinar cahaya akan lebih banyak mengeluarkan warna dari objek tersebut. Untuk menghitung banyaknya cahaya yang dipantulkan oleh sebuah permukaan benda melalui diffuse reflection, perhatikan rumus berikut.

$$\text{DiffuseColor} = \frac{\text{MaterialDiffuseColor} \cdot \text{LightColor} \cdot \text{LightPower} \cdot \cos\theta}{\text{distance}^2}$$

1. MaterialDiffuseColor: adalah warna yang dipantulkan oleh permukaan benda ketika terjadi diffuse reflection. Misalnya, jika permukaan benda berwarna putih, maka MaterialDiffuseColor akan mencerminkan warna putih tersebut.
2. LightColor: adalah warna dari cahaya datang yang mengenai permukaan benda
3. LightPower: adalah kekuatan/intensitas cahaya yang dating dari sumber cahaya. Semakin besar nilai LightPower, maka akan semakin terang juga cahayanya.
4. cosTheta: adalah sudut insiden cahaya dan normal (dihitung dengan melakukan perkalian dot, dengan adalah vector normal, dan vector arah cahaya). Ketika cosTheta mendekati 1, berarti cahaya datang hampir tegak lurus dengan permukaan benda.

5. distance: adalah jarak dari posisi titik (vertex) ke sumber cahaya. Jarak ini digunakan untuk menghitung penurunan intensitas cahaya seiring dengan jaraknya dari sumber cahaya.

Singkatnya, diffuse reflection adalah proses pemantulan cahaya dari permukaan benda ke arah mata kita. Semakin tegak lurus cahaya dengan permukaan, semakin terang objek terlihat. Hal ini dihitung dengan rumus yang memperhitungkan warna permukaan, warna cahaya, kekuatan cahaya, sudut cahaya dan normal, serta jarak dari sumber cahaya.



b. Specular

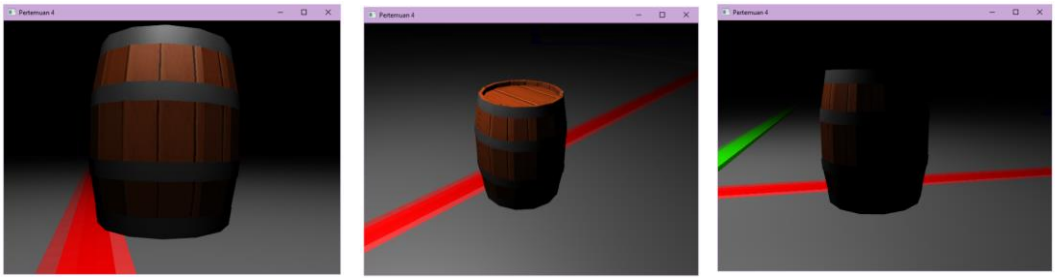
$$\text{SpecularColor} = \frac{\text{MaterialSpecularColor} \cdot \text{LightColor} \cdot \text{LightPower} \cdot (\cos\alpha)^k}{\text{distance}^2}$$

1. MaterialDiffuseColor: adalah warna yang dipantulkan oleh permukaan benda ketika terjadi specular reflection. Misalnya, jika permukaan benda berwarna putih, maka MaterialDiffuseColor akan mencerminkan warna putih tersebut.
2. LightColor: adalah warna dari cahaya datang yang mengenai permukaan benda
3. LightPower: adalah kekuatan/intensitas cahaya yang dating dari sumber cahaya. Semakin besar nilai LightPower, maka akan semakin terang juga cahayanya.
4. cosAlpha: adalah sudut insiden cahaya dan normal (dihitung dengan melakukan perkalian dot, dengan adalah vector normal, dan vector arah cahaya). Ketika cosAlpha mendekati 1, berarti arah pandang kamera hampir sejajar dengan arah pantulan cahaya, yang menghasilkan specular highlight yang kuat.
5. k: adalah angka yang mengatur berapa besar sudut yang dapat memantulkan cahaya secara teratur. Semakin besar nilai k, maka akan semakin kecil sudut yang dapat

memantulkan cahaya secara teratur, dan menghasilkan specular highlight yang lebih fokus.

6. distance: adalah jarak dari posisi titik (vertex) ke sumber cahaya. Jarak ini digunakan untuk menghitung penurunan intensitas cahaya seiring dengan jaraknya dari sumber cahaya.

Apabila diffuse dan specular digabungkan, kita dapat menghasilkan tampilan yang lebih realistis dan detail pada permukaan objek. Diffuse reflection memberikan tampilan umum dari warna dan kecerahan objek berdasarkan arah cahaya datang, sementara specular reflection akan menambahkan highlight yang lebih terfokus pada area yang lebih reflektif dari permukaan objek.



c. Ambient Light

Ambient light adalah fenomena di mana lingkungan tetap memiliki sedikit cahaya meskipun tidak ada sumber cahaya secara langsung. Hal ini terjadi karena cahaya yang memantul di dalam ruangan atau lingkungan tertutup akan terus memantul dari permukaan ke permukaan, menyebarkan cahaya di sekitarnya. Dalam dunia nyata, bahkan dalam ruangan yang gelap, cahaya masih dapat terlihat karena efek ambient ini.

Untuk menyelesaikan kalkulasi pencahayaan ini dalam fragment shader, kita hanya perlu menjumlahkan ketiga nilai yang telah dihitung sebelumnya, yaitu ambient, diffuse, dan specular. Perhatikan kode berikut.

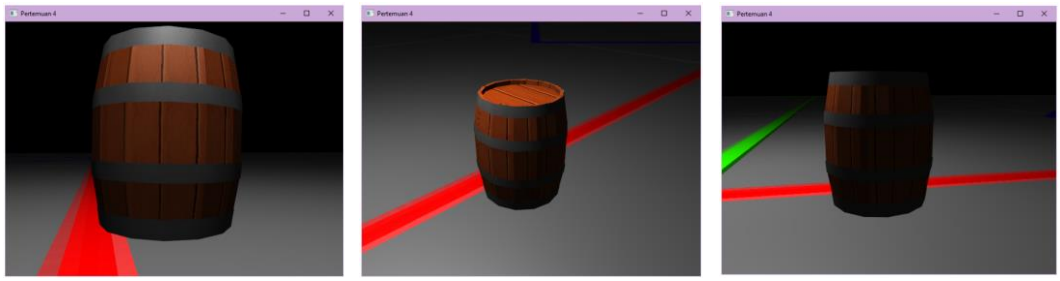
```
MaterialAmbientColor = vec3(0.1,0.1,0.1) * MaterialDiffuseColor;


#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);

}
```

Ambient light memberikan cahaya latar yang merata pada seluruh permukaan objek, tanpa memperhatikan arah cahaya datang secara spesifik. Dengan kombinasi ketiga elemen ini: diffuse (berdasarkan sudut insiden cahaya dan normal), specular (berdasarkan sudut antara arah pandang mata dan arah pantulan), dan ambient, objek akan tampak lebih

hidup dan realistis, dengan detail yang lebih mendalam dan kesan yang lebih alami dalam berbagai kondisi pencahayaan.



4. Kesimpulan

Dapat disimpulkan bahwa pencahayaan dan bayangan merupakan elemen penting dalam pembuatan grafika komputer. Cahaya memungkinkan kita untuk melihat objek dan lingkungan secara visual, sementara bayangan menambah kedalaman dan dimensi dalam pengalaman visual. Mengimplementasikan pencahayaan memungkinkan kita untuk menyoroti detail, menciptakan suasana yang sesuai, dan meningkatkan pengalaman pengguna dalam sebuah grafik. Kombinasi antara diffuse, specular, dan ambient light dalam pengaturan pencahayaan juga menciptakan tampilan yang lebih realistis dan mendalam pada permukaan objek. Dengan komposisi yang tepat, pencahayaan dan bayangan dapat meningkatkan kualitas visual dan memberikan pengalaman yang lebih berkesan bagi pengguna.