

**DOKUMENTASI RESEARCH PROJECT GAMELOFT INDONESIA
PROGRAMMER STUDENT COMPETITION 2019**

**PENERAPAN BASIC LIGHTING PADA OBJEK 3D DENGAN
MENGUNAKAN OPENGL ES2**



Oleh:

Asterik Rafael Winoto	311610002	Angkatan 2016
Hermawan Wirasantosa	311610009	Angkatan 2016
Reinaldo Sebastian Gunawan	311610015	Angkatan 2016

**UNIVERSITAS MA CHUNG
MALANG
2019**

Daftar Isi

1. Tujuan Penelitian	1
a. Topik Penelitian	1
b. Hasil dari Peneletian	1
2. Deskripsi Topik.....	1
a. Definisi Topik	1
i. Objek 3D	1
ii. Basic Lighting	2
iii. Open GL ES 2.0	6
b. Fitur Topik	6
c. Implementasi Aplikasi	7
d. Kelebihan	9
e. Pengembangan	9
3. Konklusi	9
4. Daftar Pustaka.....	11

Daftar Gambar

Gambar 1 Vertex Shader Lampu	2
Gambar 2 Fragment Shader Lampu	2
Gambar 3 Vertex Shader Lighting	3
Gambar 4 Fragment Shader Ambient Effect	3
Gambar 5 Fragment Shader Diffuse Effect	4
Gambar 6 Fragment Shader Specular Effect	5
Gambar 7 Fragment Shader Phong Effect	6
Gambar 8 Run Program	7
Gambar 9 Efek Ambient	7
Gambar 10 Efek Diffuse	8
Gambar 11 Efek Specular	8
Gambar 12 Efek Phong	9

1. Tujuan Penelitian

a. Topik Penelitian

Topik “Penerapan Basic Lighting pada Objek 3D dengan menggunakan OpenGL ES 2” berasal dari ketertarikan kami untuk mencari cara agar dapat menambahkan suatu efek cahaya kepada suatu objek 3D, sehingga riset kami meliputi seputar bagaimana lighting dibuat dan penerapannya pada *fragment* shader objek 3D.

b. Hasil dari Penelitian

Hasil dari penelitian yaitu diharapkan sudah dapat menerapkan Basic Lighting pada sebuah Objek 3D dengan benar dan membandingkan berbagai macam bentuk Lighting.

2. Deskripsi Topik

a. Definisi Topik

i. Objek 3D

Objek 3D adalah suatu gambar yang memiliki 3 sumbu utama yaitu, sumbu X, sumbu Y, dan sumbu Z dan terdiri dari *Vertex* sebagai titik-titik, *Edge* sebagai garis penghubung antar *Vertex*, dan *Face* sebagai permukaan yang terbentuk dari Edge (Fajar, n.d.). Sebuah Objek 3D di dalam OpenGL memiliki 3 bagian utama yaitu Model, *Texture*, serta *Shader*.

1. Model

Model merupakan kumpulan dari *Vertices data* dan *Indices Data*. *Vertices data* adalah kumpulan *Vertex* yang akan digunakan dalam pembentukan *Fragment*. Sedangkan *Indices Data* merupakan kumpulan segitiga yang *Vertex* nya berasal dari *Vertices data* dan segitiga-segitiga tersebut diurutkan sehingga membentuk suatu objek 3D.

2. Texture

Texture merupakan kumpulan detail warna yang akan di tempelkan pada sebuah model. *Texture* biasanya terdiri dari suatu file bergambar.

3. Shader

Shader adalah suatu program yang mengatur transformasi input dan output sebuah model dan texture. Shader bertumpu pada GPU

sehingga akan selalu digunakan pada proses Rendering (Shaders, n.d.). *Shader* terdiri dari 2 macam yaitu *Vertex Shader* dan *Fragment Shader*.

a. Vertex Shader

Vertex Shader adalah program yang menuliskan *behaviour* dari sebuah *vertex* dan menyusun transformasi *vertex*.

b. Fragment Shader

Fragment Shader adalah program yang memanipulasi pemberian warna di dalam *fragment*.

ii. Basic Lighting

Lighting merupakan efek cahaya terhadap suatu objek 3D. Tentunya Pencahayaan membutuhkan sumber cahaya untuk memperoleh cahaya. Terdapat *Vertex Shader* dan *Fragment Shader* untuk sumber cahaya. Di dalam *Vertex Shader* sumber cahaya, terdapat komponen yang diperlukan seperti vektor posisi *vertex*, matriks *World-Space*, dan matriks model objek sebagai pengatur transformasi letak objek. Untuk *Fragment Shader* sumber cahaya, hanya diperlukan vektor warna putih sebagai simbol bahwa objek tersebut merupakan sumber cahaya.

```
attribute vec3 a_posL;
uniform mat4 u_worldMatrix;
uniform mat4 u_model;
void main() {
    vec4 posL = vec4(a_posL, 1);
    gl_Position = u_worldMatrix * u_model * posL;
}
```

Gambar 1 Vertex Shader Lampu

```
precision mediump float;
void main() {
    gl_FragColor = vec4(1.0);
}
```

Gambar 2 Fragment Shader Lampu

Pencahayaan mempunyai beberapa komponen yang diperlukan, yaitu vektor normal objek, matriks *World-Space*, matriks model objek, matriks invers model objek vektor sebagai matriks model yang berada pada *World-Space*, vektor warna cahaya, dan vektor warna objek. Seluruh komponen

tersebut berguna untuk mencari nilai vektor posisi *fragment*, vektor Normal *World-Space*, dan posisi *vertex*.

```
1  attribute vec3 a_posl;
2  attribute vec3 a_norm;
3  varying vec3 Normal;
4  varying vec3 FragPos;
5  uniform mat4 u_worldMatrix;
6  uniform mat4 u_model;
7  uniform mat3 u_invModel;
8  uniform vec3 lightColor;
9  uniform vec3 objectColor;
10 void main() {
11     FragPos = vec3(u_model*vec4(a_posl,1.0));
12     Normal= u_invModel * a_norm;
13     gl_Position= u_worldMatrix*vec4(FragPos,1.0);
14 }
```

Gambar 3 Vertex Shader Lighting

Di dalam *Basic Lighting* terdiri atas 4 macam *lighting*, yaitu *Ambient*, *Diffuse*, dan *Specular* (Basic Lighting, n.d.).

1. Ambient

Efek cahaya yang memengaruhi warna seluruh objek 3D sehingga tampak seolah membaur dengan lingkungan nya dan arah cahaya nya tidak diketahui (Kusti, 2011). Penggunaan efek *Ambient* dilakukan dengan cara mengalikan nilai *Ambient* dengan nilai warna cahaya serta dikalikan dengan nilai warna objek.

```
precision mediump float;
uniform vec3 lightColor;
uniform vec3 objectColor;
void main() {
    float ambientStrength=0.3;
    vec3 ambient = ambientStrength*lightColor;
    vec3 result=ambient*objectColor;
    gl_FragColor = vec4(result,1.0);
}
```

Gambar 4 Fragment Shader Ambient Effect

2. Diffuse

Efek cahaya ini memengaruhi bagian objek 3D yang terkena sumber cahaya sehingga sudut cahaya terhadap objek sangat penting (Kusti, 2011). Semakin tegak lurus cahaya terhadap objek maka semakin terang pula cahaya yang menyentuh objek. Untuk memperoleh sudut cahaya tersebut dibutuhkan vektor normal.

```

void main() {
    vec3 norm= normalize(Normal);
    vec3 lightDir = normalize((lightPos)-FragPos);
    float diff= max(dot(norm,lightDir),0.0);
    vec3 diffuse= diff * lightColor;

    vec3 result = (diffuse)*objectColor;
    gl_FragColor = vec4(result,1.0);
}

```

Gambar 5 Fragment Shader Diffuse Effect

a. Vektor Normal

Vektor normal adalah suatu vektor yang tegak lurus dengan permukaan *fragment* (Basic Lighting, n.d.). Vektor ini nantinya akan digunakan untuk mengukur besar sudut cahaya terhadap *fragment* sehingga menghasilkan warna yang sesuai dengan letak sumber cahaya nya. Vektor normal ditempatkan pada data vertices bersamaan dengan vektor posisi.

b. Arah Cahaya

Arah cahaya adalah sebuah vektor yang bernilai selisih dari vektor posisi cahaya dan vektor posisi *fragment* (Basic Lighting, n.d.).

Untuk menghitung efek *Diffuse*, langkah awal yang harus dilakukan yaitu menormalisasikan nilai vektor normal sehingga menjadi suatu vektor satuan. Lalu menghitung nilai vektor arah cahaya dengan cara vektor posisi cahaya dikurangi vektor posisi *fragment*. Selanjutnya untuk mencari nilai besaran efek *Diffuse*, maka digunakanlah metode perkalian titik. Pada metode perkalian titik, dibutuhkan nilai vektor normal dan nilai vektor arah cahaya yang hasilnya berupa vektor satuan. Bila sudut dari kedua vektor lebih besar dari 90 derajat maka akan bernilai negatif sehingga diperlukan metode Max untuk menanggulangi terjadi nya nilai *Diffuse* negatif. Setelah itu nilai arah cahaya dikalikan dengan nilai warna cahaya dan nilai warna objek.

3. Specular

Efek cahaya ini memberi tampilan bayangan cahaya di suatu objek sehingga terlihat seperti mengkilap (Kusti, 2011). Mirip dengan efek *Diffuse*, efek *Specular* juga menggunakan vektor arah cahaya. Namun perbedaannya adalah efek *Specular* juga menggunakan posisi pandangan *User*. Kegunaannya untuk mengukur pantulan cahaya yang terdapat pada permukaan *fragment*.

```
void main() {  
    vec3 norm= normalize(Normal);  
    vec3 lightDir = normalize(lightPos-FragPos);  
    float specularStrength = 0.5;  
    vec3 viewDir = normalize(viewPos - FragPos);  
    vec3 reflectDir = reflect(-lightDir, norm);  
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32.0);  
    vec3 specular = specularStrength * spec * lightColor;  
  
    vec3 result = (specular) * objectColor;  
    gl_FragColor = vec4(result,1.0);  
}
```

Gambar 6 Fragment Shader Specular Effect

Langkah awal yang perlu dilakukan sama seperti pada efek *Diffuse* yaitu menormalisasikan nilai vektor normal dan mencari nilai vektor arah cahaya. Selanjutnya buat variabel yang berfungsi menyimpan nilai kekuatan cahaya *Specular*. Lalu seperti vektor arah cahaya, vektor arah pandangan dihitung dari nilai posisi pandangan dikurangi nilai posisi *fragment* dan hasilnya dinormalisasi.

Selanjutnya mencari nilai vektor refleksi dengan fungsi *reflect* dengan inputan nilai vektor arah cahaya dan nilai vektor normal. Pada inputan vektor arah cahaya, yang dibutuhkan adalah arah cahaya dari sumber cahaya kearah *fragment*. Namun nilai vektor arah cahaya saat ini adalah nilai dari *fragment* kearah sumber cahaya sehingga nilai nya dibuat negatif.

Berikutnya mencari nilai dampak efek *Specular* dengan mencari nilai vektor satuan antara vektor posisi pandangan dan vektor arah refleksi. Untuk menghindari nilai minus, digunakan metode *Max*. Nilai yang dihasilkan lalu di pangkatkan dengan nilai sorotan

kemilau. Semakin besar nilai sorotan kemilaunya maka semakin kecil pula pantulan sorotannya.

4. Phong

Efek cahaya ini merupakan gabungan dari ketiga efek diatas. Sehingga langkah-langkah yang diterapkan pada shader efek *Phong* ini dimulai dari pembuatan efek *Ambient*, dilanjutkan dengan pembuatan efek *Diffuse*, lalu pembuatan efek *Specular*. Nilai dari ketiga efek tersebut dijumlahkan dan dimasukkan ke vektor hasil sehingga hasilnya merupakan gabungan dari tiga efek tersebut.

```
void main() {  
    float ambientStrength=0.3;  
    vec3 ambient = ambientStrength*lightColor*objectColor;  
  
    vec3 norm= normalize(Normal);  
    vec3 lightDir = normalize(lightPos-FragPos);  
    float diff= max(dot(norm,lightDir),0.0);  
    vec3 diffuse= diff * lightColor;  
  
    float specularStrength = 0.5;  
    vec3 viewDir = normalize(viewPos - FragPos);  
    vec3 reflectDir = reflect(-lightDir, norm);  
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32.0);  
    vec3 specular = specularStrength * spec * lightColor;  
  
    vec3 result = (ambient+diffuse+specular) * objectColor;  
    gl_FragColor = vec4(result,1.0);  
}
```

Gambar 7 Fragment Shader Phong Effect



iii. OpenGL ES 2

OpenGL ES 2 adalah sebuah API yang memberikan akses fungsi-fungsi yang memanipulasi gambar.

b. Fitur Aplikasi

Fitur yang terdapat pada aplikasi ini berupa key untuk mengubah posisi camera dan mengubah efek pencahayaan.

1. Key posisi Camera

Key yang digunakan untuk mengubah posisi Camera antara lain adalah “W” dan “S” untuk mengatur posisi sumbu Z, “A” dan “D” untuk mengatur posisi sumbu X, “E” dan “C” untuk mengatur posisi sumbu Y, “&” atau tombol  dan “)” atau tombol  untuk mengatur rotasi sumbu X, “%”

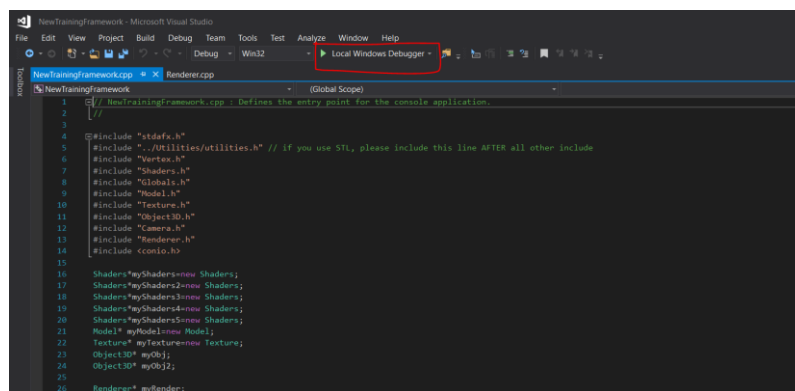
atau tombol ← dan “/” atau tombol → untuk mengatur rotasi sumbu Y, “Q” dan “Z” untuk mengatur rotasi sumbu Z.

2. Efek Pencahayaan

Key yang digunakan untuk mengganti efek pencahayaan antara lain adalah “1” untuk efek *Ambient*, “2” untuk efek *Diffuse*, “3” untuk efek *Specular*, dan “4” untuk efek *Phong*.

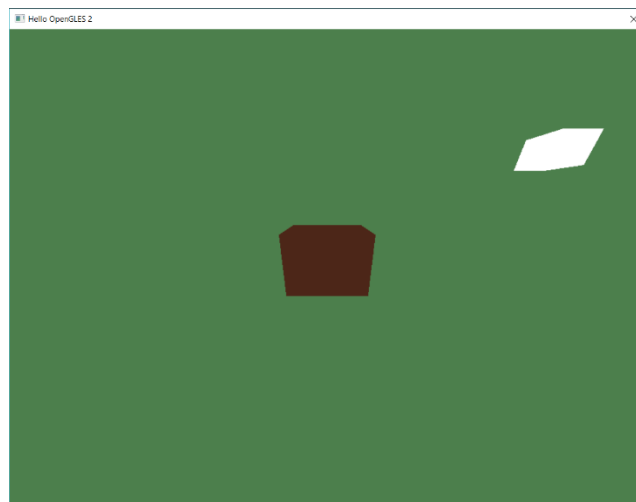
c. Implementasi Aplikasi

- Buka file NewTrainingFramework.sln didalam folder ResearchOpenGLES2
- Run program tersebut



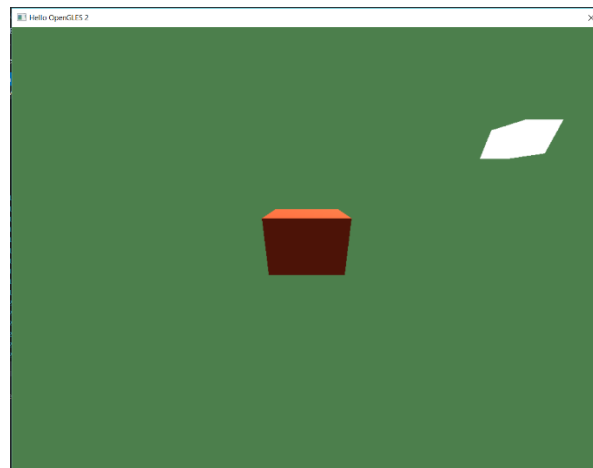
Gambar 8 Run Program

- Klik tombol 1 untuk mengganti efek cahaya menjadi *Ambient*



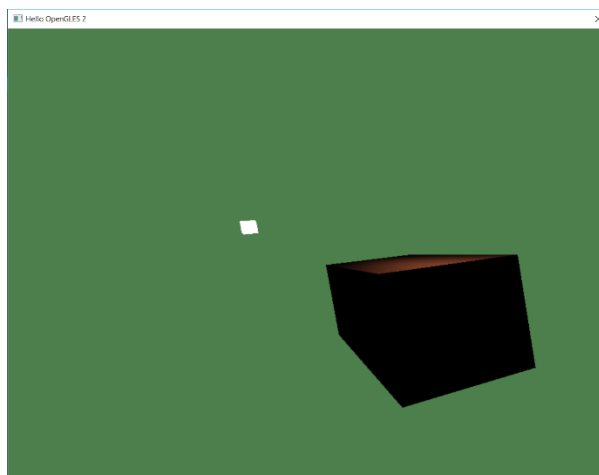
Gambar 9 Efek Ambient

- iv. Klik tombol 2 untuk mengganti efek cahaya menjadi *Diffuse*



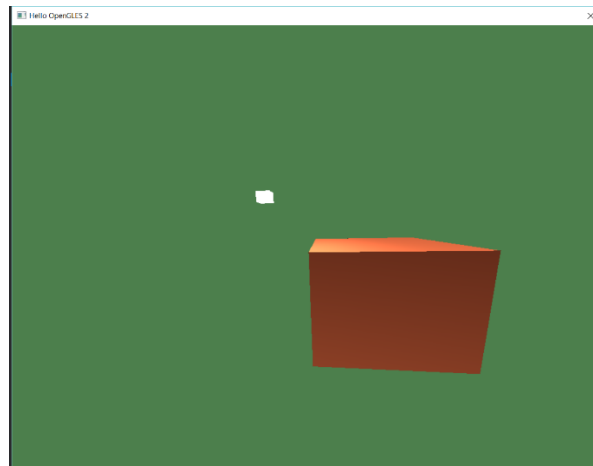
Gambar 10 Efek Diffuse

- v. Klik tombol 3 untuk mengganti efek cahaya menjadi *Specular*



Gambar 11 Efek Specular

- vi. Klik tombol 4 untuk mengganti efek cahaya menjadi *Phong*



Gambar 12 Efek Phong

d. Kelebihan dan Kekurangan

Efek cahaya *Ambient* cocok digunakan sebagai efek cahaya rambu reflektor namun kurang cocok digunakan sebagai efek simulasi pencahayaan objek terhadap sumber cahaya. Efek *Diffuse* cocok untuk efek cahaya objek yang tidak mengkilap seperti kayu, gedung, jalan, dan lain-lain. Efek *Specular* cocok untuk objek mengkilap seperti besi, gelas, keramik, dan lain-lain namun efek baur cahaya tidak ada dalam efek *Specular* sehingga seperti objek hitam yang terpantul sinar cahaya. Efek *Phong* dapat digunakan di berbagai macam objek karena efek nya yang lengkap.

e. Pengembangan

Pengembangan aplikasi yang dapat dibuat adalah dengan membuat suatu efek cahaya dengan berbagai macam warna sehingga efek cahaya yang dihasilkan terlihat adanya gradasi warna. Selain itu dapat juga ditambahkan efek cahaya bila sumber cahaya tertutup oleh suatu objek lain di depan nya seperti sinar matahari yang tertutup oleh awan.

3. Konklusi

Pencahayaan merupakan suatu efek yang memiliki banyak faktor penentu sehingga efek pencahayaan dapat bermacam-macam bentuknya. Gabungan dari beberapa macam efek dapat menjadi efek pencahayaan yang baru. Sehingga pencahayaan dapat disesuaikan sesuai kebutuhan. Terdapat 4 efek pencahayaan sederhana yang umum dipakai, yaitu efek *Ambient*, *Diffuse*, *Specular*, dan *Phong*. Efek *Ambient* sebagai pencahayaan yang menyebar sehingga satu sumber cahaya dapat menerangi seluruh permukaan objek. Efek *Diffuse*

sebagai pencahayaan yang mensimulasikan dampak dari sumber cahaya terhadap objek penerima cahaya sehingga permukaan yang tidak menghadap sumber cahaya akan gelap sedangkan yang menghadap cahayanya disesuaikan dengan sudut dan jarak objek terhadap sumber cahaya. Efek *Specular* adalah pencahayaan yang berupa sebuah pantulan cahaya yang terfokus pada satu titik sehingga terlihat permukaan objek seakan mengkilap. Efek *Phong* merupakan gabungan dari keempat efek diatas sehingga pencahayaan yang dihasilkan juga gabungan dari keistimewaan keempat efek diatas. Diantara keempat efek tersebut, dibutuhkan 2 objek utama yaitu objek sumber cahaya dan objek penerima cahaya. Di dalam objek tersebut terdapat komponen yang selalu digunakan untuk mengatur pencahayaan yaitu vektor normal suatu objek, vektor posisi cahaya, dan vektor posisi pandangan. Dari keempat efek tersebut, terlihat bila semakin kompleks efek yang dihasilkan, maka semakin kompleks pula metode yang digunakan seperti semakin banyak metode perhitungan vektor dan matriks seperti refleksi, perkalian titik, transpose, invers, dan lain-lain.

4. Daftar Pustaka

Basic Lighting. (n.d.). Retrieved from learnopengl.com:

<https://learnopengl.com/Lighting/Basic-Lighting>

Fajar, A. (n.d.). *Objek 3 dimensi*. Retrieved from lintasmateri.blogspot.com:

<https://lintasmateri.blogspot.com/2016/05/objek-3-dimensi.html>

Kusti, N. (2011, 10 19). *Pencahayaan Ambient, Diffuse, Specular*. Retrieved from

nabarakusti.blogspot.com: <http://nabarakusti.blogspot.com/2011/10/pencahayaan-Ambient-Diffuse-Specular.html>

Shaders. (n.d.). Retrieved from learnopengl.com: <https://learnopengl.com/Getting-started/Shaders>