BAB 9
SHADING DAN OPTICAL VIEW

9.1. Vektor Satuan

Vektor satuan (*unit vector*) adalah vektor yang besarnya satu. Untuk mendapatkan vektor satuan maka setiap elemen vektor dibagi dengan besarnya vektor tersebut dan dirumuskan dengan:

$$u = \frac{v}{|v|}$$

Bila $v = \{v_0, v_1, v_2\}$ maka besarnya vektor adalah:

$$|v| = \sqrt{v_0^2 + v_1^2 + v_2^2}$$

dan vektor satuannya adalah:

$$u = \frac{\{v_0, v_1, v_2\}}{\sqrt{v_0^2 + v_1^2 + v_2^2}}$$

Sehingga fungsi untuk menghasilkan vektor satuan adalah sebagai berikut:

```
vector3D_t unitVector(vector3D_t vec){
	int i;
	float vec2=0.;
	float vec1,invvec1;
	for (i=0;i<3;i++)
		vec2+=vec.v[i]*vec.v[i];
	vec1=sqrt(vec2);
	if (vec1!=0.) {
		invvec1=1./vec1;
		for (i=0;i<3;i++)
			vec.v[i]*=invvec1;
	}
	vec.v[3]=1.;
	return vec;
}
```

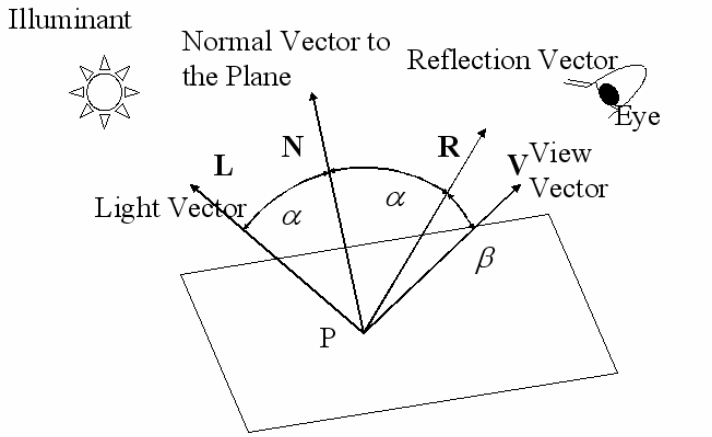
9.2. Optical View

Optical view adalah sebuah konsep mengenai pencahayaan yang jatuh pada sebuah benda. Pada opical view terdapat beberapa vektor yaitu:

- Vektor yang menyatakan arah cahaya (lightVector)
- Vektor yang menyatakan arah pandangan (viewVector)

- Vektor normal (vecNormal)
- Vektor pantulan cahaya yang jatuh atau refleksi (rVector)

Secara umum optical view dapat digambarkan sebagai berikut:



Gambar 9.1 Optical view

Di dalam optical view, ada satu syarat yang harus dipenuhi bahwa semua vektor harus vektor satuan.

$$|\mathbf{L}| = |\mathbf{N}| = |\mathbf{R}| = |\mathbf{V}| = 1$$

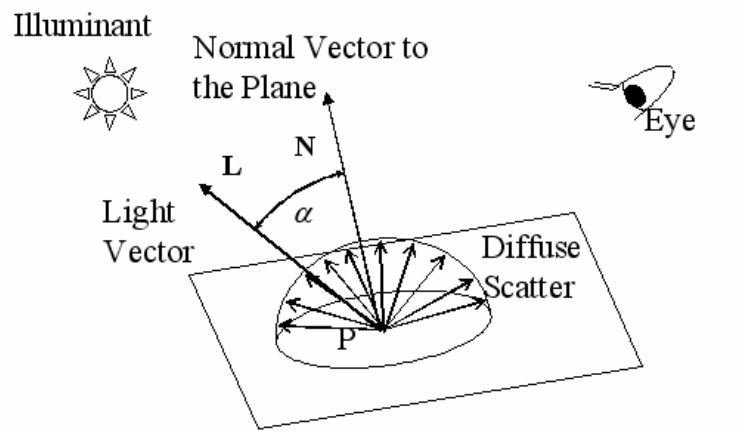
Ada beberapa macam metode optical view yang dapat dijadikan dasar sebagai pencahayaan yang jatuh pada benda, yaitu:

- Diffuse Scattering
- Specullar Reflection
- Ambient
- Lambert's Law
- Phong Model

Dari semua model pencahayaan tersebut, Phong model merupakan model pencahayaan yang paling lengkap. Jadi dalam buku grafika ini fungsi pencahayaan yang digunakan adalah Phong model.

9.2.1. Diffuse Scattering

Model pencahayaan menggunakan Diffuse Scattering dapat digambarkan sebagai berikut:



Gambar 9.2 Diffuse Scattering

Dari gambar 9.2 di atas, dapat dilihat bahwa besarnya cahaya yang jatuh pada benda dapat dirumuskan dengan:

$$I_d = I_s k_d \, \text{MAX}(\cos\alpha, 0)$$

$$\cos\alpha = \mathbf{L} \bullet \mathbf{N}$$

I_d : Intensity of the diffuse component

I_s : Intensity of the Light Source

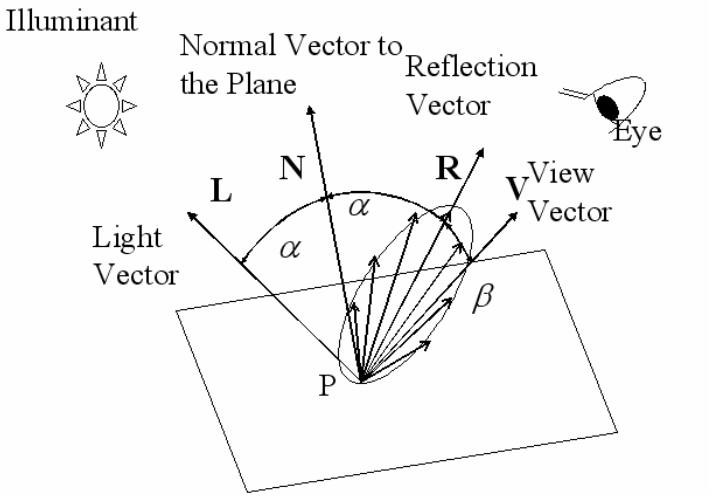
k_d : Diffuse reflection coefficient

Fungsi pencahayaan dengan Diffuse Scattering dapat dituliskan dengan:

```
color_t DiffuseScattering (vector3D_t Light, vector3D_t Normal,
vector3D_t View, color_t col) {
      // diffuse reflection coefficient
      float kdif=0.6;
      float tmp,NL,RV;
      color_t ColWhite={1,1,1};
      vector3D_t ReflectionVector=(2.*(Light*Normal)*Normal)-Light;
      tmp=Normal*Light;
      NL=funcPositive(tmp);
      tmp=ReflectionVector*View;
      RV=funcPositive(tmp);
      return kdif*NL*col;
}
```

9.2.2. Specular Reflection

Model pencahayaan menggunakan Specular Reflection dapat digambarkan sebagai berikut



Gambar 9.3. Specular Reflection

Dari gambar 9.3 di atas, dapat dilihat bahwa besarnya cahaya yang jatuh pada benda dapat dirumuskan dengan:

$$I_{sp} = I_s k_{sp} \text{MAX}(\cos^n \beta, 0)$$

$$\cos\beta = \mathbf{R} \bullet \mathbf{V}$$

$$\mathbf{R} = 2\mathbf{N}(\mathbf{L} \bullet \mathbf{N}) - \mathbf{L}$$

I_{sp} : Intensitas dari Specular Reflection

I_s : Intensitas dari sumber cahaya

k_{sp} : Koefisien Specular reflection

n : konstanta bulat dari experiment (1 s/d - 200)

Beberapa penjelasan rumus pada specular reflection adalah sebagai berikut:

$$\mathbf{R} = 2\mathbf{N}(\mathbf{L} \bullet \mathbf{N}) - \mathbf{L}$$

$$\because \mathbf{L} \bullet \mathbf{N} = \mathbf{R} \bullet \mathbf{N}, \quad \mathbf{R} = a\mathbf{L} + b\mathbf{N}$$

$$\mathbf{L} \bullet \mathbf{N} = (a\mathbf{L} + b\mathbf{N}) \bullet \mathbf{N}$$

$$b = (1-a)\mathbf{L} \bullet \mathbf{N}$$

$$\mathbf{R} = a\mathbf{L} + \{(1-a)\mathbf{L} \bullet \mathbf{N}\}\mathbf{N}$$

$$\mathbf{R}^2 = 1, (\mathbf{L}^2 = 1, \mathbf{N}^2 = 1)$$

$$\text{maka } a = \pm 1 \Rightarrow a = -1$$

Fungsi pencahayaan dengan Specural Reflection dapat dituliskan dengan:

```
color_t  SpeculalReflection(vector3D_t  Light,  vector3D_t  Normal,
vector3D_t View, color_t col) {
      // specular reflection coefficient
      float kspe=0.7;
      float tmp,NL,RV;
      color_t ColWhite={1,1,1};
      vector3D_t ReflectionVector=(2.*(Light*Normal)*Normal)-Light;
      tmp=Normal*Light;
      NL=funcPositive(tmp);
      tmp=ReflectionVector*View;
      RV=funcPositive(tmp);
      return kspe*power(RV,4)*ColWhite;
}
```

9.2.3. Ambient

Model pencahayaan menggunakan Ambient adalah model pencahayaan yang merata pada semua permukaan benda dan dirumuskan dengan:

$$I_a = I_s k_a$$

I_a : Intensitas dari Ambient Reflection

I_s : Intensitas dari sumber cahaya

k_{sp} : Koefisien ambient reflection

Fungsi pencahayaan dengan Ambient dapat dituliskan dengan:

```
color_t Ambient(vector3D_t Light,vector3D_t Normal, vector3D_t View,
color_t col) {
      // ambient light coefficient
      float kamb=0.4;
      return kamb*col;
}
```

9.2.4. Lambert's Law

Model pencahayaan menggunakan Lambert's Law adalah gabungan diffuse scattering dan ambient, dan dituliskan:

$$C_L = C_s\{k_d \ \mathrm{MAX}(\cos\alpha, 0) + k_a\}$$
$$\cos\alpha = \mathbf{L} \bullet \mathbf{N}$$
$$C_L : \text{Reflection Color}$$
$$C_s : \text{Surface Color}$$

Fungsi pencahayaan dengan Lambert's Law dapat dituliskan dengan:

```
color_t LambertLaw(vector3D_t Light, vector3D_t Normal, vector3D_t
View, color_t col) {
      // diffuse reflection coefficient
      float kdif=0.6;
      // ambient light coefficient
      float kamb=0.4;
      float tmp,NL,RV;
      color_t ColWhite={1,1,1};
      vector3D_t ReflectionVector=(2.*(Light*Normal)*Normal)-Light;
      tmp=Normal*Light;
      NL=funcPositive(tmp);
      tmp=ReflectionVector*View;
      RV=funcPositive(tmp);
      return kdif*NL*col+kamb*col;
}
```

9.2.5. Phong Model

Model pencahayaan Phong Model adalah model pencahayaan yang lengkap karena merupakan gabungan dari diffuse scattering, specular reflection dan ambient. Rumus dari Phong Model dituliskan dengan:

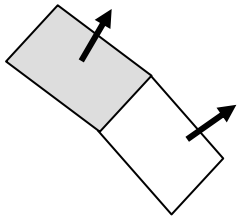
$$C_L = C_s\{k_d \ \mathrm{MAX}(\cos\alpha, 0) + k_a\} + C_{white} k_{sp} \mathrm{MAX}(\cos^n \beta, 0)$$
$$\cos\alpha = \mathbf{L} \bullet \mathbf{N}$$
$$\cos\beta = \mathbf{R} \bullet \mathbf{V}$$
$$\mathbf{R} = 2\mathbf{N}(\mathbf{L} \bullet \mathbf{N}) - \mathbf{L}$$
$$C_L : \text{Reflection Color}$$
$$C_s : \text{Surface Color}$$
$$C_{white} : \text{Specular Color (white)}$$

Fungsi pencahayaan dengan Phong Model dapat dituliskan dengan:

```
color_t  PhongModel(vector3D_t  Light,vector3D_t  Normal,  vector3D_t
View, color_t col) {
      // specular reflection coefficient
      float kspe=0.7;
      // diffuse reflection coefficient
      float kdif=0.6;
      // ambient light coefficient
      float kamb=0.4;
      float tmp,NL,RV;
      color_t ColWhite={1,1,1};
      vector3D_t ReflectionVector=(2.*(Light*Normal)*Normal)-Light;
      tmp=Normal*Light;
      NL=funcPositive(tmp);
      tmp=ReflectionVector*View;
      RV=funcPositive(tmp);
      return kdif*NL*col+kspe*power(RV,4)*ColWhite+kamb*col;
}
```

9.3. Flat Shading

Flat shading adalah bentuk pencahayaan dengan menganggap bahwa satu face mempunyai tingkat pencahayaan yang sama sehingga setiap face mempunyai satu warna saja sebagai akibat dari warna benda dan pencahayaan yang tetap. Flat shading dapat digambarkan seperti gambar 9.4 berikut.



Gambar 9.4. Flat shading.

Untuk menghasilkan sebuah warna akibat pencahayaan dapat dituliskan dengan:
colbuff=PhongModel(lightVector, vecNormal, viewVector, col);

dimana:

collbuf adalah warna hasil pencahayaan

col adalah warna benda.

Sehingga fungsi untuk menggambar dengan teknik obyek 3D menggunakan flat shading adalah sebagai berikut:

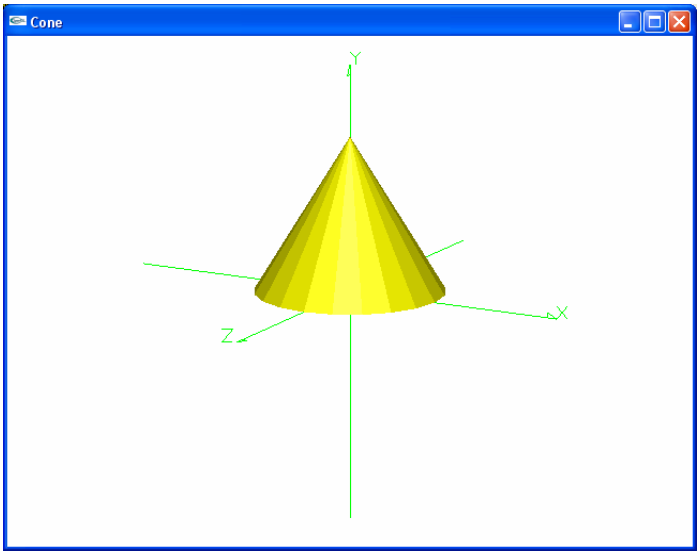
```
void draw3Dc(object3D_t obyek,matrix3D_t mat, color_t col){
vector3D_t vec[1600], vecbuff[50];
vector3D_t vecNormal;
vector3D_t lightVector={0,0,1,1};
vector3D_t viewVector={0,0,1,1};
color_t colbuff;
point2D_t p[50];
int i,j;
for(i=0;i<obyek.NumberofVertices;i++){
      vec[i]=Point2Vector(obyek.pnt[i]);
      vec[i]=mat*vec[i];
}
for(i=0;i<obyek.NumberofFaces;i++){
for(j=0;j<obyek.fc[i].NumberofVertices;j++)
      vecbuff[j]=vec[obyek.fc[i].pnt[j]];
vecNormal=(vecbuff[1]-vecbuff[0])^(vecbuff[2]-vecbuff[0]);
if(vecNormal.v[2]<0){
for(j=0;j<obyek.fc[i].NumberofVertices;j++){
      p[j]=Vector2Point2D(vecbuff[j]);
      }
vecNormal=unitVector(vecNormal);
colbuff=PhongModel(lightVector,vecNormal,viewVector,col);
fillPolygon(p,obyek.fc[i].NumberofVertices,colbuff);
}
for(i=0;i<obyek.NumberofFaces;i++){
for(j=0;j<obyek.fc[i].NumberofVertices;j++)
      vecbuff[j]=vec[obyek.fc[i].pnt[j]];
vecNormal=(vecbuff[1]-vecbuff[0])^(vecbuff[2]-vecbuff[0]);
if(vecNormal.v[2]>=0){
for(j=0;j<obyek.fc[i].NumberofVertices;j++){
      p[j]=Vector2Point2D(vecbuff[j]);
      }
vecNormal=unitVector(vecNormal);
colbuff=PhongModel(lightVector,vecNormal,viewVector,col);
fillPolygon(p,obyek.fc[i].NumberofVertices,colbuff);
}
}
}
```

Contoh 9.1:

Membuat kerucut dengan n=20, warna kuning, r=90 dan h=150 menggunakan flat shading dapat dilakukan dengan menulisakan fungsi userdraw berikut:

```
void userdraw(void){
matrix3D_t tilting=rotationXMTX(0.25)*rotationYMTX(-0.5);
setColor(0,1,0);
drawAxes(tilting);
object3D_t kerucut;
createCone(kerucut,20,90,150);
color_t kuning={1,1,0};
draw3Dc(kerucut,tilting,kuning);
}
```

Hasilnya adalah:



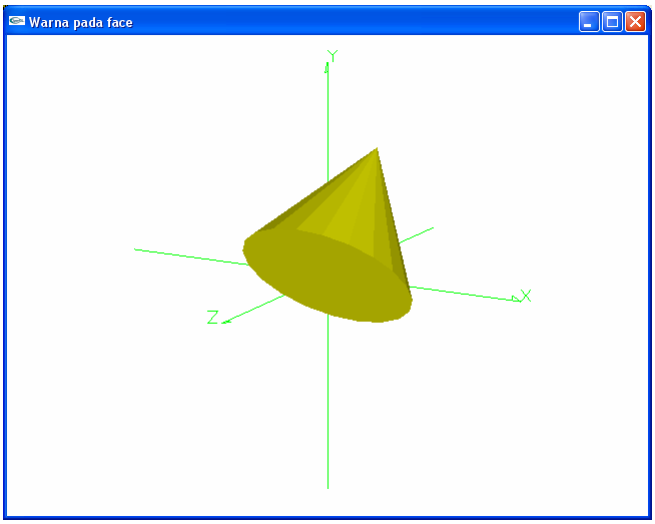
Gambar 9.5. Kerucut dengan flat shading

Pada gambar 9.5 di atas tidak memperlihatkan bagaimana keadaan pewarnaan pada face yang tidak terlihat, untuk itu perlu diputar sebesar -0.76 radian ke arah sumbu X sebagai berikut:

```
void userdraw(void){
matrix3D_t tilting=rotationXMTX(0.25)*rotationYMTX(-0.5);
setColor(0,1,0);
drawAxes(tilting);
object3D_t kerucut;
matrix3D_t mat=tilting*rotationXMTX(-0.76);
createCone(kerucut,20,90,150);
color_t kuning={1,1,0};
```

```
draw3Dc(kerucut,mat,kuning);
}
```

Hasilnya adalah:



Gambar 9.6. Contoh flat shading pada face invisible

9.4. Flat Shading Dengan Pewarnaan Face

Mengingat bahwa setiap face pada obyek 3 dimensi mempunyai warna yang berbeda-beda, maka diperlukan fungsi untuk menggambar efek flat shading pada obyek yang mempunyai warna berbeda pada setiap facenya sebagai berikut:

```
void draw3Dc(object3D_t obyek,matrix3D_t mat){
vector3D_t vec[1600], vecbuff[50];
vector3D_t vecNormal;
vector3D_t lightVector={0,0,1,1},viewVector={0,0,1,1};
color_t colbuff;
point2D_t p[50];
int i,j;
for(i=0;i<obyek.NumberofVertices;i++){
      vec[i]=Point2Vector(obyek.pnt[i]);
      vec[i]=mat*vec[i];
}
for(i=0;i<obyek.NumberofFaces;i++){
for(j=0;j<obyek.fc[i].NumberofVertices;j++)
      vecbuff[j]=vec[obyek.fc[i].pnt[j]];
vecNormal=(vecbuff[1]-vecbuff[0])^(vecbuff[2]-vecbuff[0]);
if(vecNormal.v[2]<0){
for(j=0;j<obyek.fc[i].NumberofVertices;j++){
      p[j]=Vector2Point2D(vecbuff[j]);
```

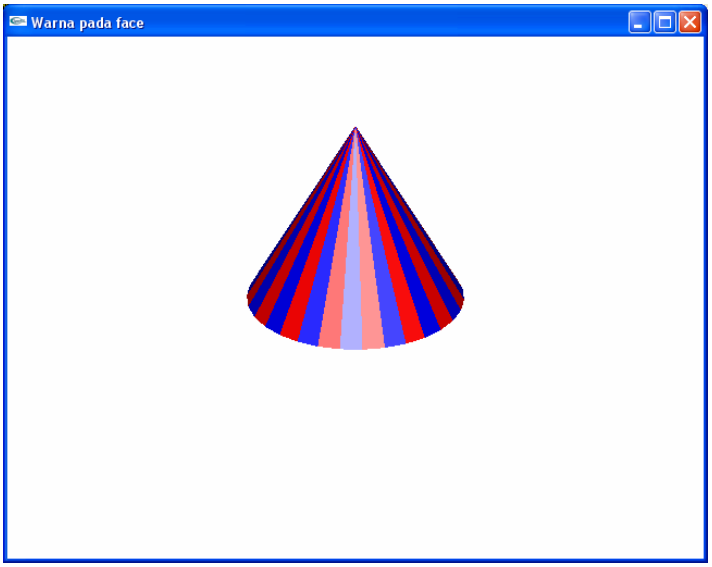
```
}
vecNormal=unitVector(vecNormal);
colbuff=PhongModel(lightVector,vecNormal,viewVector,obyek.fc[i].col)
;
fillPolygon(p,obyek.fc[i].NumberofVertices,colbuff);
}}
for(i=0;i<obyek.NumberofFaces;i++){
for(j=0;j<obyek.fc[i].NumberofVertices;j++)
      vecbuff[j]=vec[obyek.fc[i].pnt[j]];
vecNormal=(vecbuff[1]-vecbuff[0])^(vecbuff[2]-vecbuff[0]);
if(vecNormal.v[2]>=0){
for(j=0;j<obyek.fc[i].NumberofVertices;j++){
      p[j]=Vector2Point2D(vecbuff[j]);
}
vecNormal=unitVector(vecNormal);
colbuff=PhongModel(lightVector,vecNormal,viewVector,obyek.fc[i].col)
;
fillPolygon(p,obyek.fc[i].NumberofVertices,colbuff);
}}}
```

Contoh 9.2:

Membuat kerucut yang warnanya berselang-seling merah dan biru dengan efek flat shading, dengan fungsi userdraw sebagai berikut:

```
void userdraw(void){
matrix3D_t tilting=rotationXMTX(0.5)*rotationYMTX(-0.25);
object3D_t o;
color_t merah={1,0,0};
color_t biru={0,0,1};
createCone(o,30,100,180);
for(int i=0;i<o.NumberofFaces;i++)
if(i%2==0) o.fc[i].col=merah;
else o.fc[i].col=biru;
draw3Dc(o,tilting);
}
```

Hasilnya adalah



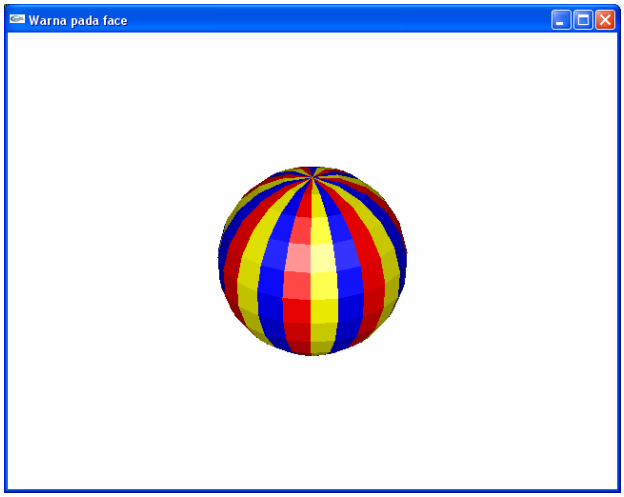
Gambar 9.7. Efek flat shading pada warna face

Contoh 9.3:

Membuat bola dengan 3 warna merah, biru dan kuning dengan efek shading. Agar bola tampak bagus maka jumlah n harus kelipatan 3 (karena 3 warna). Contoh userdraw berikut ini menggunakan n=21.

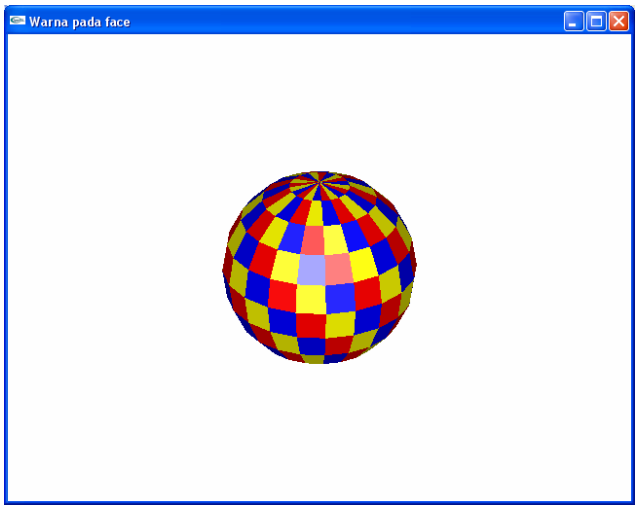
```
void userdraw(void){
matrix3D_t tilting=rotationXMTX(0.5)*rotationYMTX(-0.25);
object3D_t o;
color_t merah={1,0,0}; color_t biru={0,0,1};
color_t kuning={1,1,0};
createSphere(o,21,100);
for(int i=0;i<o.NumberofFaces;i++)
      if(i%3==0)  o.fc[i].col=merah;
      else if(i%3==1) o.fc[i].col=biru;
      else o.fc[i].col=kuning;
draw3Dc(o,tilting);
}
```

Hasilnya adalah:



Gambar 9.8. Contoh bola 3 warna

Bila jumlah n bukan kelipatan 3, misalnya n=20, maka hasilnya adalah sebagai berikut:



Gambar 9.9. Contoh bola 3 warna dengan n=20

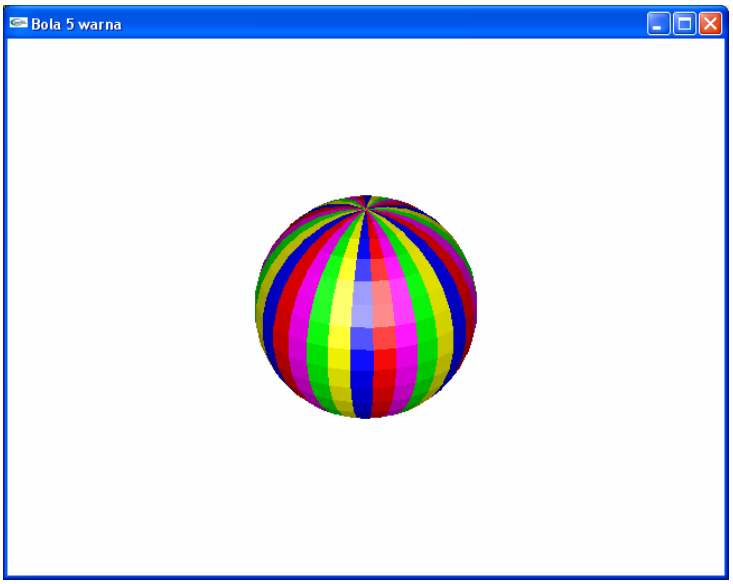
Contoh 9.4:

Membuat bola dengan 5 warna merah, biru dan kuning dengan efek shading. Agar bola tampak bagus maka jumlah n harus kelipatan 5 (karena 5 warna). Contoh userdraw berikut ini menggunakan n=30.

```
void userdraw(void){
matrix3D_t tilting=rotationXMTX(0.5)*rotationYMTX(-0.25);
object3D_t o;
color_t merah={1,0,0};
color_t biru={0,0,1};
color_t kuning={1,1,0};
color_t hijau={0,1,0};
```

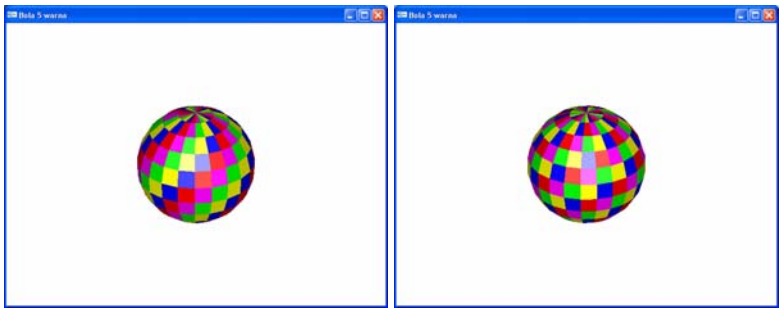
```
color_t ungu={1,0,1};
createSphere(o,30,100);
for(int i=0;i<o.NumberofFaces;i++)
      if(i%5==0)  o.fc[i].col=merah;
      else if(i%5==2) o.fc[i].col=biru;
      else if(i%5==3) o.fc[i].col=kuning;
      else if(i%5==4) o.fc[i].col=hijau;
      else o.fc[i].col=ungu;
draw3Dc(o,tilting);
}
```

Hasilnya adalah:



Gambar 9.10. Contoh bola 5 warna

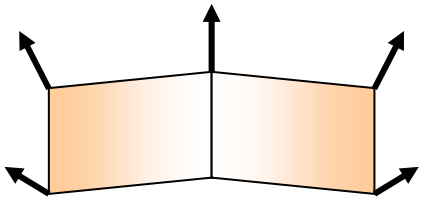
Bila jumlah n bukan kelipatan 5, misalnya n=23 atau n=24, maka hasilnya adalah sebagai berikut:



Gambar 9.11. Contoh bola 5 warna dengan n=21,22,23 dan 24

9.5. Gouraud Shading

Gouraud shading adalah bentuk pencahayaan dengan menganggap bahwa satu titik mempunyai tingkat pencahayaan yang sama sehingga setiap face mempunyai warna bergradiasi sejumlah titik pembentuknya sebagai akibat dari warna benda dan pencahayaan yang tetap. Gouraud shading dapat digambarkan sebagai berikut:



Gambar 9.12. Gouraud Shading

Sebelum menggunakan fungsi gouraud shading maka perlu diganti definisi obyeknya menjadi:

```
typedef struct {
      int NumberofVertices; //in the face
      short int pnt[50];
} face_t;
typedef struct {
      int NumberofVertices; //of the object
      point3D_t pnt[1600];
      color_t col[1600];
      int NumberofFaces; //of the object
      face_t fc[1000];
} object3D_t;
```

Fungsi untuk menghasilkan gouraud shading adalah sebagai berikut:

```
void draw3Dcg(object3D_t obyek,matrix3D_t mat){
vector3D_t vec[1600], vecbuff[50];
vector3D_t vecNormal;
```

```
vector3D_t lightVector={0,0,1,1},viewVector={0,0,1,1};
color_t colbuff[50];
point2D_t p[50];
int i,j;
for(i=0;i<obyek.NumberofVertices;i++){
      vec[i]=Point2Vector(obyek.pnt[i]);
      vec[i]=mat*vec[i];
      }
for(i=0;i<obyek.NumberofFaces;i++){
for(j=0;j<obyek.fc[i].NumberofVertices;j++)
      vecbuff[j]=vec[obyek.fc[i].pnt[j]];
vecNormal=(vecbuff[1]-vecbuff[0])^(vecbuff[2]-vecbuff[0]);
if(vecNormal.v[2]<0){
      for(j=0;j<obyek.fc[i].NumberofVertices;j++){
      p[j]=Vector2Point2D(vecbuff[j]);
      vecNormal=unitVector(vecbuff[j]);
      colbuff[j]=PhongModel(lightVector,vecNormal,viewVector,obyek.c
ol[obyek.fc[i].pnt[j]]);
      gradatePolygon(p,obyek.fc[i].NumberofVertices,colbuff);
      }
}
for(i=0;i<obyek.NumberofFaces;i++){
for(j=0;j<obyek.fc[i].NumberofVertices;j++)
vecbuff[j]=vec[obyek.fc[i].pnt[j]];
vecNormal=(vecbuff[1]-vecbuff[0])^(vecbuff[2]-vecbuff[0]);
if(vecNormal.v[2]>=0){
      for(j=0;j<obyek.fc[i].NumberofVertices;j++){
      p[j]=Vector2Point2D(vecbuff[j]);
      vecNormal=unitVector(vecbuff[j]);
      colbuff[j]=PhongModel(lightVector,vecNormal,viewVector,obyek.c
ol[obyek.fc[i].pnt[j]]);
      gradatePolygon(p,obyek.fc[i].NumberofVertices,colbuff);
      }
}
}
```

Contoh 9.4:

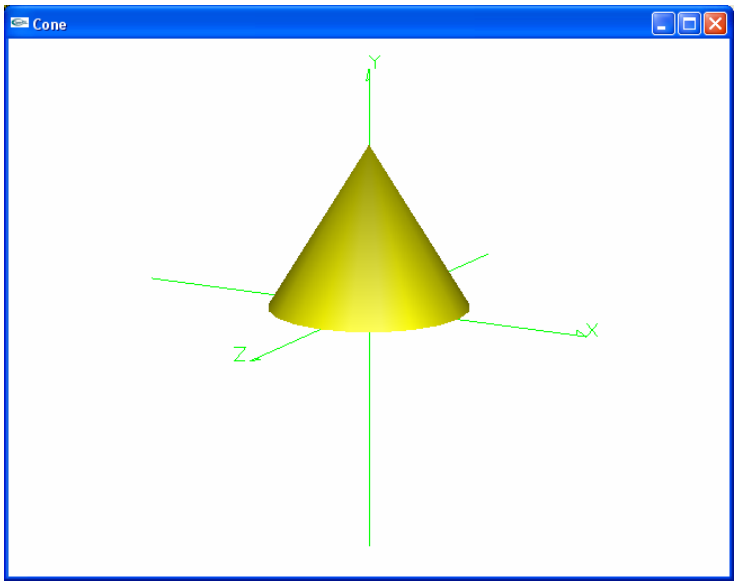
Menggambar kerucut dengan n=20, r=90 dan h=120 dengan gouraud shading dapat dilakukan dengan menuliskan fungi userdraw sebagai berikut:

```
void userdraw(void){
```

```
matrix3D_t tilting=rotationXMTX(0.25)*rotationYMTX(-0.5);
setColor(0,1,0);
drawAxes(tilting);
object3D_t kerucut;
createCone(kerucut,20,90,150);
color_t kuning={1,1,0};
// Memberi warna kuning
for(int i=0;i<kerucut.NumberofVertices;i++)
      kerucut.col[i]=kuning;
draw3Dcg(kerucut,tilting);
}
```

Untuk menghasilkan gauroud shading maka nilai warna adalah sejumlah titik, karena pada dasarnya gauroud shading menggunakan gradatePolygon yang mensyaratkan jumlah warna sama dengan jumlah titiknya.

Hasilnya adalah:



Gambar 9.13. Contoh kerucut dengan gouraud shading

Contoh 9.5:

Membedakan dua buah lingkaran dengan jari-jari masing-masing 100 dan n=20 dengan efek flat shading dan gauroud shading.

(1) Buat lingkaran berwarna merah dengan posisi titik pusat (-150,0,0) dengan flat shading

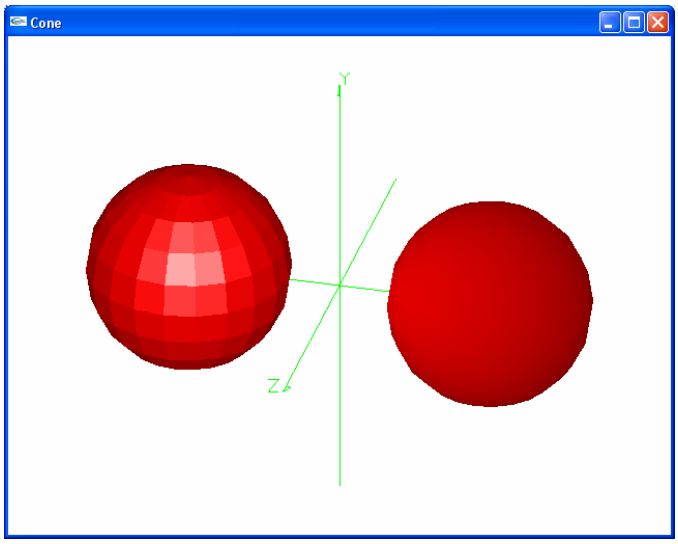
(2) Buat lingkaran berwarna merah dengan posisi titik pusat (150,0,0) dengan gauroud shading. Untuk bisa membuah ini, maka pada setiap titik diberi warna merah sebelum obyek tersebut digambar.

Peletakan posisi di sebuah obyek 3 dimensi membutuhkan transformasi translasi karena tidak ada obyek lingkaran dengan titik pusat tertentu, hanya ada titik pusat di pusat koordinat.

Programnya pada userdraw adalah sebagai berikut:

```
void userdraw(void){
matrix3D_t tilting=rotationXMTX(0.5)*rotationYMTX(-0.25);
object3D_t lingkaran;
matrix3D_t mat;
color_t merah={1,0,0};
createSphere(lingkaran,20,100);
// lingkaran dengan flat shading
mat=tilting*translationMTX(-150,0,0);
draw3Dc(lingkaran,mat,merah);
// lingkaran dengan gauroud shading
mat=tilting*translationMTX(150,0,0);
for(int i=0;i<lingkaran.NumberofVertices;i++)
      lingkaran.col[i]=merah;
draw3Dcg(lingkaran,mat);
}
```

Hasilnya adalah:



Gambar 9.14. Perbedaan flat shading dan gauroud shading

Pada gambar 9.14 di atas, terlihat bahwa hasil dari gauroud shading lebih halus meskipun jumlah titiknya sama, tetapi hasil flat shading lebih cerah hal ini karena banyaknya warna titik yang sama.

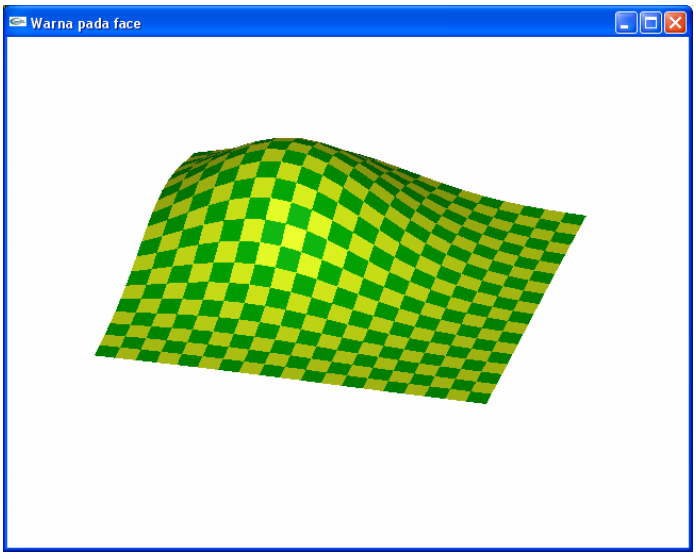
Contoh 9.6:

Menggambar landscape dengan model plane berbentuk papan catur dengan efek flat shading, dapat dilakukan dengan: membuat plane berukuran 20x20, Memberi warna pada face berselang-seling antara hijau (0,0.7,0) dan coklat (0.9, 1,0.1), dan mengubah nilai y (tinggi) pada titik (6,y,6) dengan efek eksponensial, terakhir menggambar obyek.

Fungsi userdraw untuk obyek landscape ini adalah sebagai berikut:

```
void userdraw(void){
matrix3D_t tilting=rotationXMTX(0.5)*rotationYMTX(-0.25);
object3D_t o;
color_t hijau={0,0.7,0};
color_t coklat={0.9,1,0.1};
createPlane(o,20,20,20,20);
for(int i=0;i<o.NumberofFaces;i++)
if(i%2==0) o.fc[i].col=hijau;
  else o.fc[i].col=coklat;
int k=0;
for(int x=0;x<20;x++)
for(int y=0;y<20;y++){
o.pnt[k].y=100*(float)exp(-(float)(x-6)*(x-6)/40-(float)(y-6)*(y-6)/40);
k++; }
matrix3D_t mat=tilting*translationMTX(-200,0,-200);
draw3Dc(o,mat);
}
```

Hasilnya adalah:



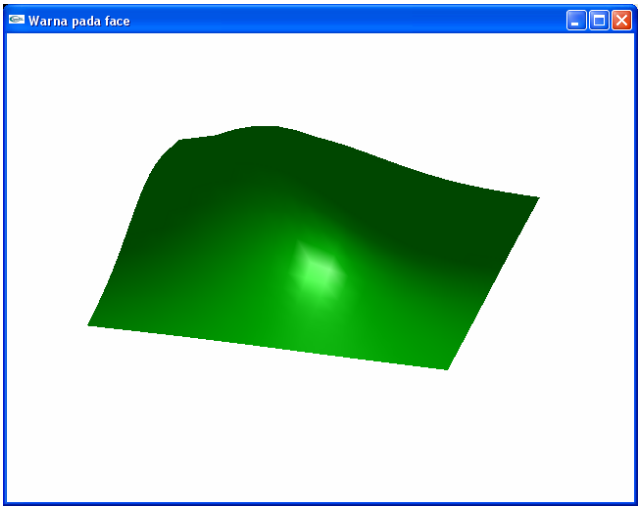
Gambar 9.15. Landscape

Untuk menggambar landscape dengan efek gauroud shading, fungsi userdraw diganti menjadi:

```
void userdraw(void){
matrix3D_t tilting=rotationXMTX(0.5)*rotationYMTX(-0.25);
```

```
object3D_t o;
color_t hijau={0,0.7,0};
createPlane(o,20,20,20,20);
int k=0;
for(int x=0;x<20;x++)
for(int y=0;y<20;y++){
o.pnt[k].y=100*(float)exp(-(float)(x-6)*(x-6)/40-(float)(y-6)*(y-6)/40);
k++; }
for(int i=0;i<o.NumberofVertices;i++) o.col[i]=hijau;
matrix3D_t mat=tilting*translationMTX(-200,0,-200);
draw3Dcg(o,mat);
}
```

Hasilnya adalah:



Gambar 9.16. Landscape dengan efek gauroud shading