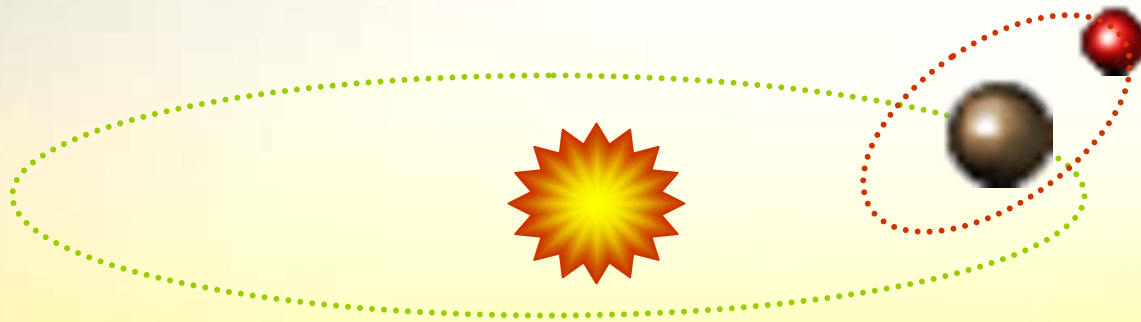


Grafik 3 Dimensi

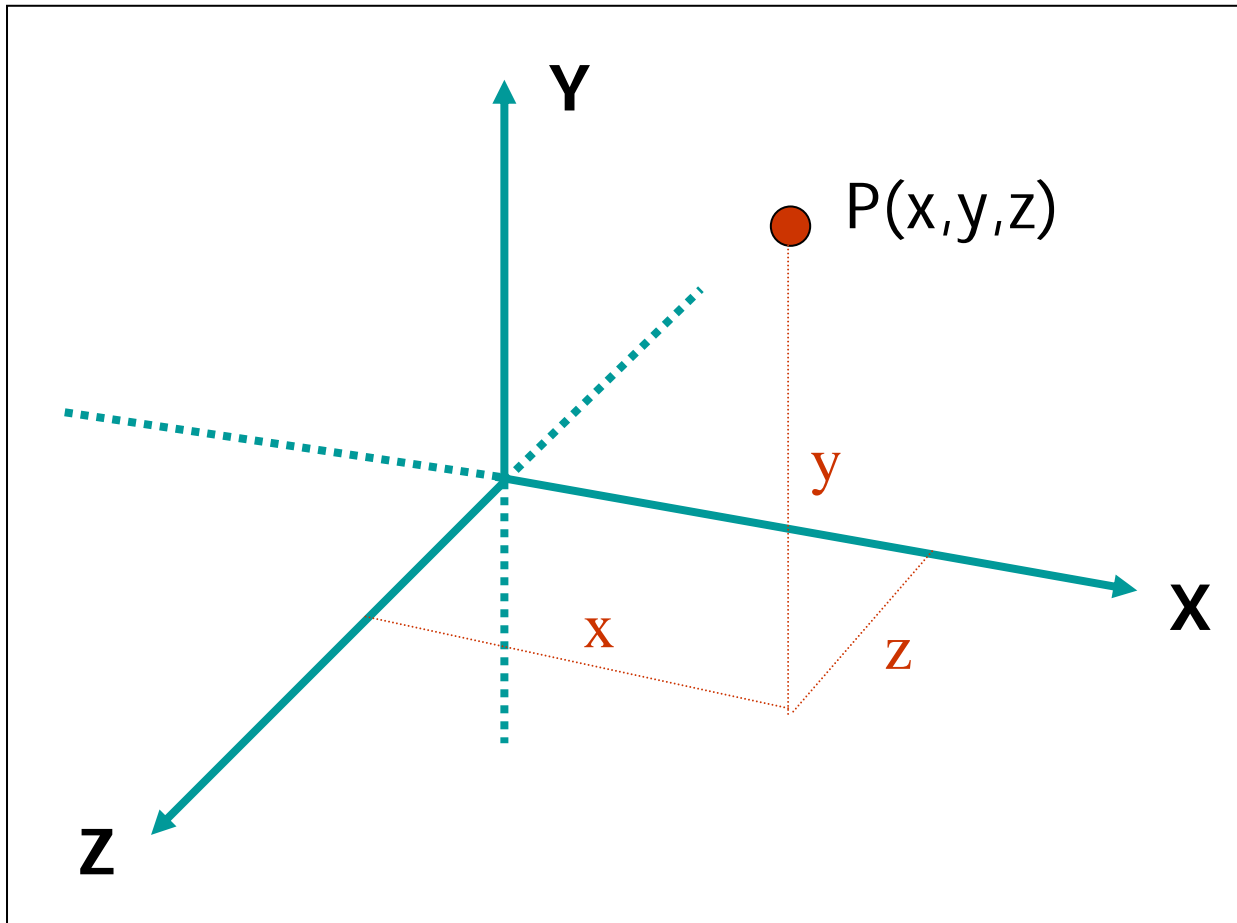


Achmad Basuki
Nana Ramadijanti

Materi

- *Sistem Koordinat 3D*
- *Definisi Obyek 3D*
- *Cara Menggambar Obyek 3D*
- *Konversi Vektor 3D menjadi Titik 2D*
- *Konversi Titik 2D menjadi Vektor 3D*
- *Visible dan Invisible*

Sistem Koordinat 3 Dimensi



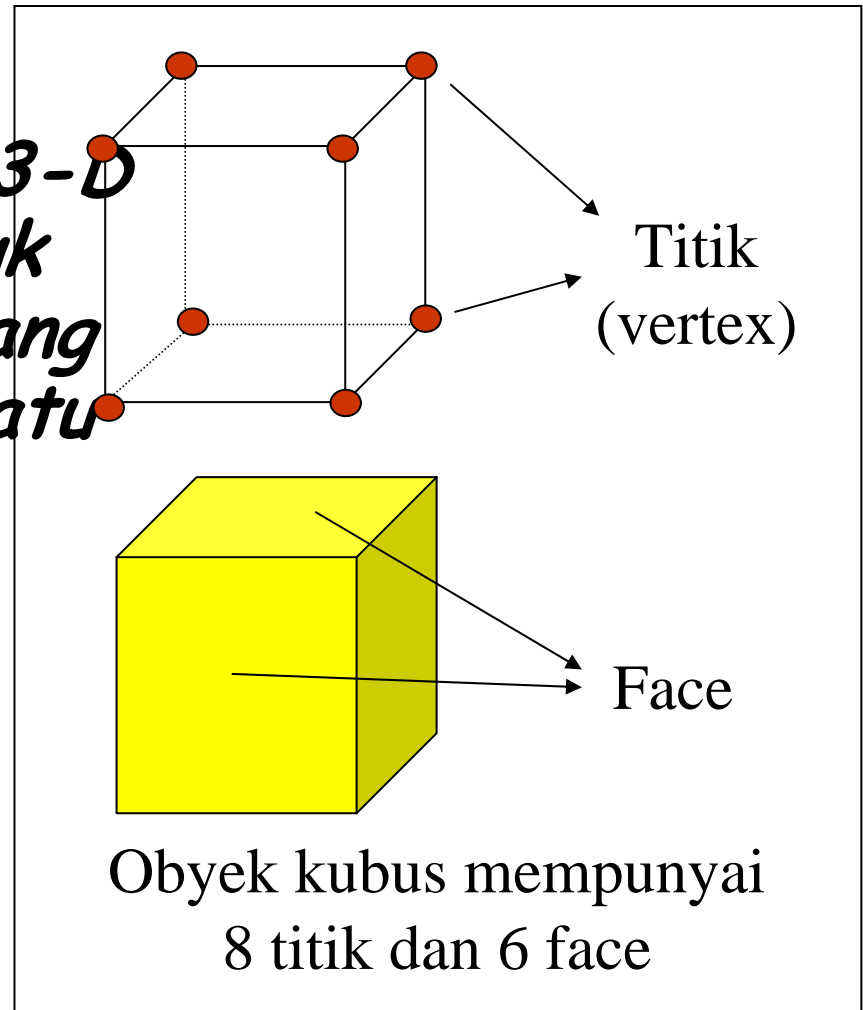
Titik 3D

Titik 3D dinyatakan dengan :
 $P(x,y,z)$

```
typedef struct {  
    float x,y,z;  
} point3D_t
```

Definisi Obyek 3 Dimensi

- *Obyek 3-D adalah sekumpulan titik-titik 3-D (x, y, z) yang membentuk luasan-luasan (face) yang digabungkan menjadi satu kesatuan.*
- *Face adalah gabungan titik-titik yang membentuk luasan tertentu atau sering dinamakan dengan sisi.*



Implementasi Definisi Dari Struktur Faces

```
typedef struct {  
    int NumberOfVertices;  
    short int pnt[32];  
} face_t;
```

NumberOfVertices menyatakan jumlah titik pada sebuah face.

pnt[32] menyatakan nomor-nomor titik yang digunakan untuk membentuk face, dengan maksimum 32 titik

Implementasi Definisi Dari Struktur Obyek 3D

```
typedef struct {  
    int NumberOfVertices;  
    point3D_t pnt[100];  
    int NumberOfFaces;  
    face_t fc[32];  
} object3D_t;
```

NumberOfVertices menyatakan jumlah titik yang membentuk obyek.

pnt[100] menyatakan titik-titik yang membentuk face, dengan maksimum 100 titik

NumberOfFaces menyatakan jumlah face yang membentuk obyek

Fc[32] menyatakan face-face yang membentuk obyek.

Contoh Pernyataan Obyek Limas SegiEmpat

Titik-titik yang membentuk obyek:

Titik 0 \rightarrow (0,150,0)

Titik 1 \rightarrow (100,0,0)

Titik 2 \rightarrow (0,0,100)

Titik 3 \rightarrow (-100,0,0)

Titik 4 \rightarrow (0,0,-100)

Face yang membentuk obyek :

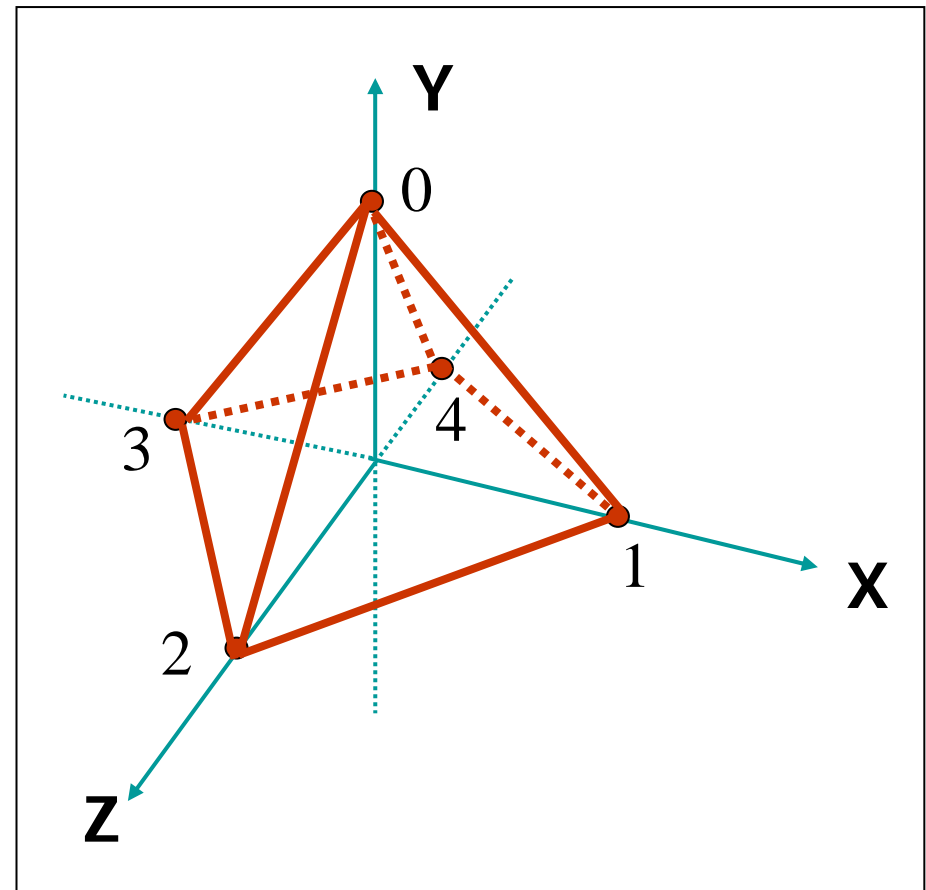
Face 0 \rightarrow 0,2,1

Face 1 \rightarrow 0,3,2

Face 2 \rightarrow 0,4,3

Face 3 \rightarrow 0,1,4

Face 4 \rightarrow 1,2,3,4



Implementasi Pernyataan Obyek 3 Dimensi

```
object3D_t prisma={5,  
    {{0,100,0},{100,0,0},{0,0,100},  
    {-100,0,0},{0,0,-100}},  
    5,  
    {{3,{0,1,2}},{3,{0,2,3}},  
    {3,{0,3,4}},{3,{0,4,1}}},  
    {4,{1,4,3,2}}};
```

*Pernyataan ini ditulis pada fungsi userdraw
sebagai nilai dari obyek 3D yang akan
digambarkan*

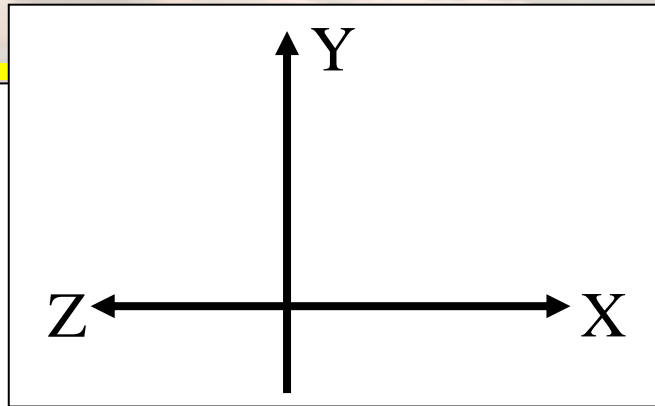
Cara Menggambar Obyek 3 Dimensi

- *Obyek 3 Dimensi terdiri dari titik-titik dan face-face.*
- *Penggambaran dilakukan pada setiap face menggunakan polygon.*
- *Polygon dibentuk dari titik-titik yang terdapat pada sebuah face.*
- *Titik-titik dinyatakan dalam struktur 3D, sedangkan layar komputer dalam struktur 2D. Sehingga diperlukan konversi dari 3D menjadi 2D.*

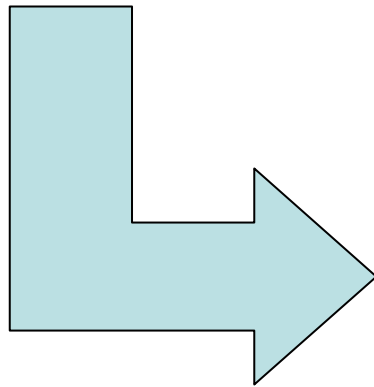
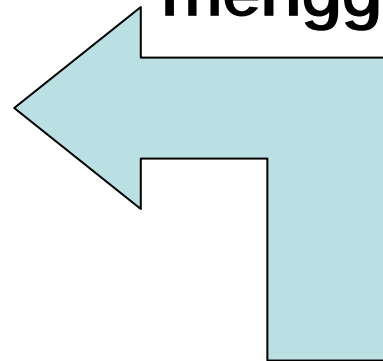
Konversi Vektor 3D menjadi 2D

- *Untuk menggambar obyek 3D, untuk setiap face perlu dilakukan pengubahan titik 3D menjadi vektor 3D, agar mudah dilakukan transformasi.*
- *Setelah proses pengolahan vektor, maka bentuk vektor 3D menjadi 2D.*
- *Sumbu Z adalah sumbu yang searah dengan garis mata, sehingga perlu transformasi untuk menampilkan sumbu ini. Untuk hal ini perlu dilakukan rotasi sumbu.*
- *Dalam konversi arah Z tidak diambil.*

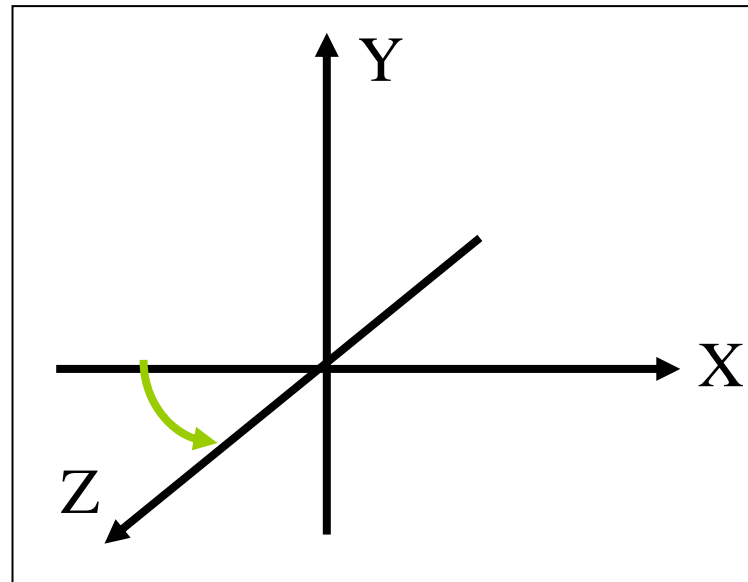
Konversi Vektor 3D menjadi 2D



Konversi untuk
menggambar obyek



Transformasi Sumbu
(*Tilting*)



Vektor 3D

$$vec = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

```
typedef struct {  
    float v[4];  
} vector3D_t;
```

Implementasi Konversi vektor 3D menjadi titik 2D

```
point2D_t Vector2Point2D(vector3D_t vec)
{
    point2D_t pnt;
    pnt.x=vec.v[0];
    pnt.y=vec.v[1];
    return pnt;
}
```

Implementasi Konversi titik 3D menjadi vektor 3D

```
vector3D_t Point2Vector(point3D_t pnt)
{
    vector3D_t vec;
    vec.v[0]=pnt.x;
    vec.v[1]=pnt.y;
    vec.v[2]=pnt.z;
    vec.v[3]=1.;
    return vec;
}
```

Implementasi Cara Menggambar Obyek 3D

```
mat=tilting;
for(i=0;i<prisma.NumberofVertices;i++)
{
    vec[i]=Point2Vector(prisma.pnt[i]);
    vec[i]=mat*vec[i];
}
for(i=0;i<prisma.NumberofFaces;i++)
{
    for(j=0;j<prisma.fc[i].NumberOfVertices;j++)
        vecbuff[j]=vec[prisma.fc[i].pnt[j]];
    for(j=0;j<prisma.fc[i].NumberOfVertices;j++)
        titik2D[j]=Vector2Point2D(vec[prisma.fc[i].pnt[j]]);
    drawPolygon(titik2D,prisma.fc[i].NumberOfVertices);
}
```

```
mat=tilting;  
for(i=0;i<prisma.NumberofVertices;i++)  
{  
    vec[i]=Point2Vector(prisma.pnt[i]);  
    vec[i]=mat*vec[i];  
}
```

Deklarasi mat sebagai matrik tilting menyatakan bahwa obyek yang digambar mengikuti pergerakan sumbu koordinat (*tilting*).

Setiap titik diubah menjadi vektor dengan memperhatikan matrik transformasi yang dinyatakan dalam mat.

Implementasi Tilting

Tilting adalah matrik rotasi dari sumbu koordinat dan semua obyek yang digambar di dalamnya.

```
float theta=0.5;  
matrix3D_t tilting=rotationXMTX(theta)*rotationYMTX(-theta);
```

Dalam deklarasi ini, matrik tilting adalah rotasi terhadap sumbu Y sebesar -0.5 rad dan rotasi terhadap sumbu X sebesar 0.5 rad.

```
for(i=0;i<prisma.NumberofFaces;i++)
{
    for(j=0;j<prisma.fc[i].NumberOfVertices;j++)
        vecbuff[j]=vec[prisma.fc[i].pnt[j]];
    for(j=0;j<prisma.fc[i].NumberOfVertices;j++)
        titik2D[j]=Vector2Point2D(vec[prisma.fc[i].pnt[j]]);
    drawPolygon(titik2D,prisma.fc[i].NumberOfVertices);
}
```

Untuk setiap face pada obyek 3D:

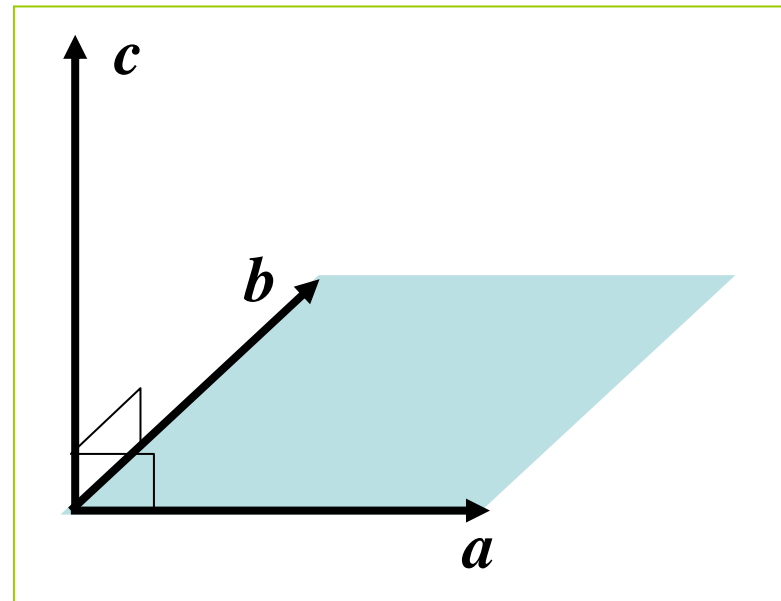
- (1) Ambil vektor dari setiap titik pada face tersebut
- (2) Konversikan setiap vektor 3D menjadi titik 2D
- (3) Dari hasil konversi digambarkan polygon

Visible dan Invisible

- *Visible dan invisible menyatakan apakah suatu face terlihat (visible) atau tidak terlihat (invisible)*
- *Pada obyek 3D tidak semua face terlihat, karena terdapat face-face yang berada di bagian belakang dan terhalang oleh face yang lainnya.*
- *Untuk menyatakan face visible dan invisible digunakan vektor normal pada face tersebut.*
- *Suatu face visible bila arah z pada vektor normal positif, dan invisible bila arah z pada vektor normalnya negatif*

Vektor Normal

- *Vektor normal adalah vektor yang arahnya tegak lurus dengan luasan suatu face*
- *Vektor normal adalah hasil perkalian silang vektor (cross-product) dari vektor-vektor yang ada pada luasan face*



$$c = a \times b$$

Perkalian Silang (Cross Product)

Perkalian silang (cross product) dari vektor $a=(a_x, a_y, a_z)$ dan $b=(b_x, b_y, b_z)$ didefinisikan dengan

$$\begin{aligned} c &= \begin{vmatrix} i & j & k \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} \\ &= i(a_y b_z - a_z b_y) + j(a_z b_x - a_x b_z) + k(a_x b_y - a_y b_x) \\ &= (a_y b_z - a_z b_y, a_z b_x - a_x b_z, a_x b_y - a_y b_x) \end{aligned}$$

Implementasi Cross-Product

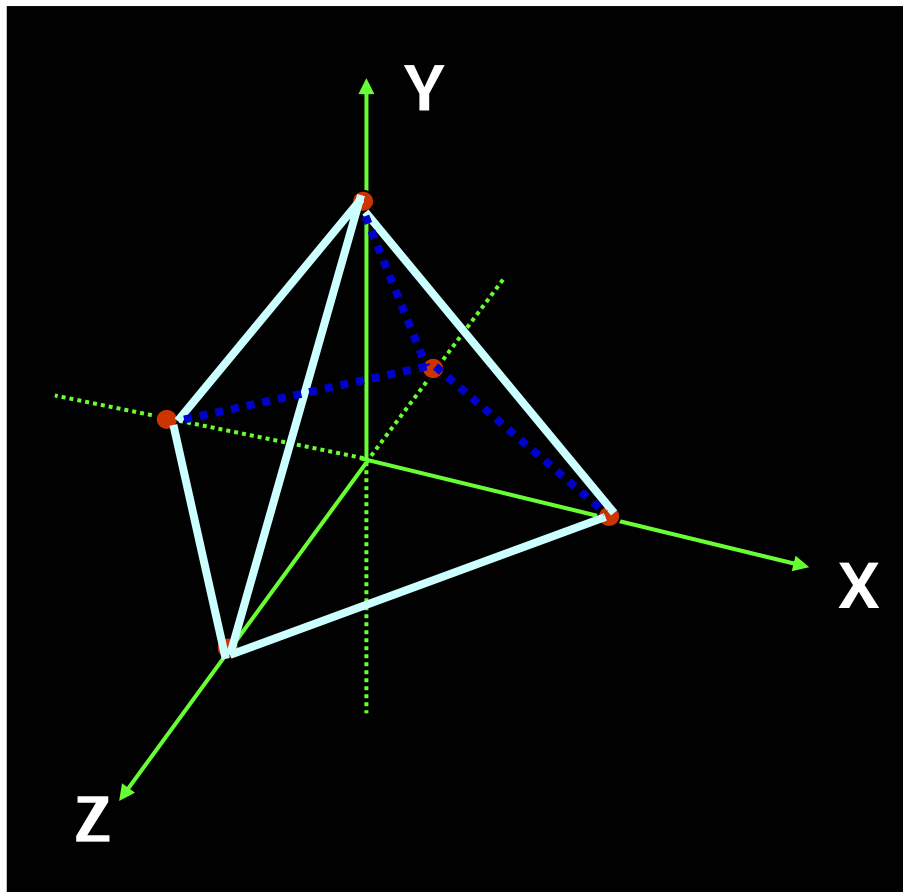
```
vector3D_t operator ^ (vector3D_t a, vector3D_t b)
{
    vector3D_t c; //c=a*b
    c.v[0]=a.v[1]*b.v[2]-a.v[2]*b.v[1];
    c.v[1]=a.v[2]*b.v[0]-a.v[0]*b.v[2];
    c.v[2]=a.v[0]*b.v[1]-a.v[1]*b.v[0];
    c.v[3]=1.;
    return c;
}
```

Cross product disimbolkan dengan operator \wedge
 $a=(a_x, a_y, a_z)$ diubah sesuai struktur data dari vektor 3D menjadi $(a.v[0], a.v[1], a.v[2])$
 $b=(b_x, b_y, b_z)$ diubah sesuai struktur data dari vektor 3D menjadi $(b.v[0], b.v[1], b.v[2])$

Implementasi Visible dan Invisible

- *Untuk mengimplementasikan face visible dan invisible maka dilakukan penggambaran dua kali*
- *Pertama digambar dulu face-face yang invisible ($NormalVector.v[2] < 0$)*
- *Kedua digambar face-face yang visible ($NormalVector.v[2] > 0$)*

Contoh Visible dan Invisible



```

setColor(0,0,1);
for(i=0;i<prisma.NumberofFaces;i++)
{
    for(j=0;j<prisma.fc[i].NumberOfVertices;j++)
        vecbuff[j]=vec[prisma.fc[i].pnt[j]];
    NormalVector=(vecbuff[1]-vecbuff[0])^(vecbuff[2]-vecbuff[0]);
    normalzi=NormalVector.v[2];
    if(normalzi<0.)
    {
        for(j=0;j<prisma.fc[i].NumberOfVertices;j++)
            titik2D[j]=Vector2Point2D(vec[prisma.fc[i].pnt[j]]);
        drawPolygon(titik2D,prisma.fc[i].NumberOfVertices);
    }
}

```

Menghitung vektor normal dari setiap face (*NormalVector*)

Menghitung arah z dari vektor normal (*normalzi*)

Menentukan apakah face invisible (*normalize < 0*)

Bagian invisible diberi warna biru (0,0,1)

```

setColor(0,1,1);
for(i=0;i<prisma.NumberofFaces;i++)
{
    for(j=0;j<prisma.fc[i].NumberOfVertices;j++)
        vecbuff[j]=vec[prisma.fc[i].pnt[j]];
    NormalVector=(vecbuff[1]-vecbuff[0])^(vecbuff[2]-vecbuff[0]);
    normalzi=NormalVector.v[2];
    if(normalzi>0.)
    {
        for(j=0;j<prisma.fc[i].NumberOfVertices;j++)
            titik2D[j]=Vector2Point2D(vec[prisma.fc[i].pnt[j]]);
        drawPolygon(titik2D,prisma.fc[i].NumberOfVertices);
    }
}

```

Menghitung vektor normal dari setiap face (*NormalVector*)

Menghitung arah z dari vektor normal (*normalzi*)

Menentukan apakah face visible (*normalize>0*)

Bagian visible diberi warna cyan (0,1,1)