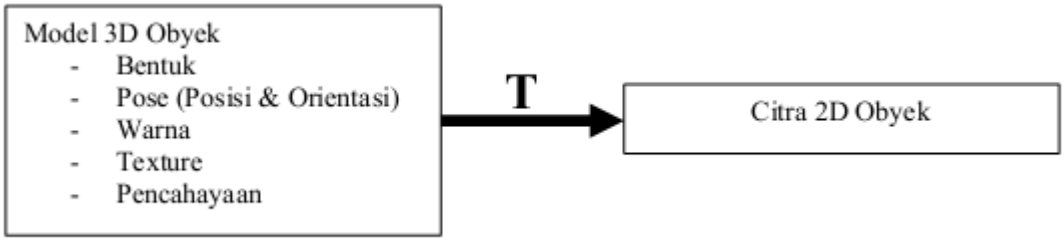
Tutorial 04

Modeling & Transformasi Proyeksi

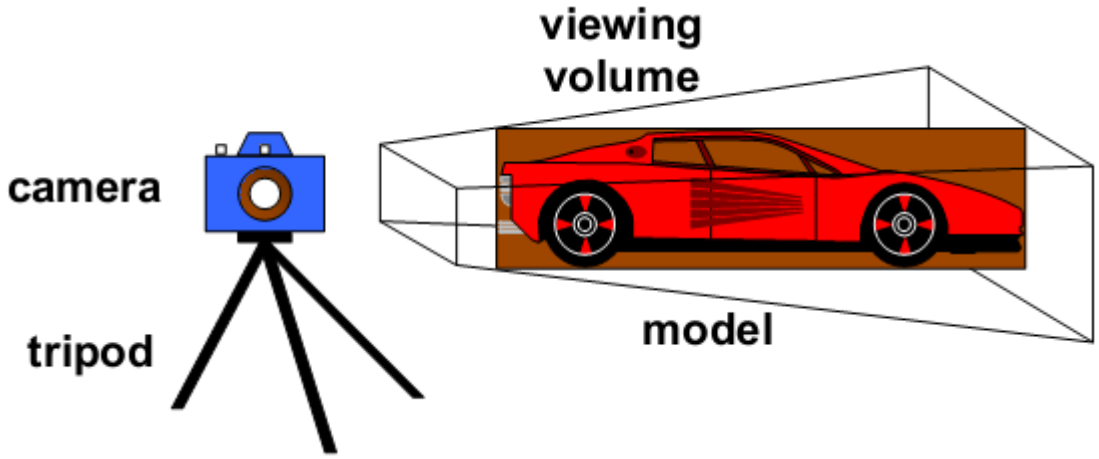
Secara substansi, Grafika Komputer adalah proses transformasi dari model 3D obyek berupa informasi geometri bentuk, informasi pose, warna, texture, dan pencahayaan menjadi citra 2D (cf. Gambar 3).

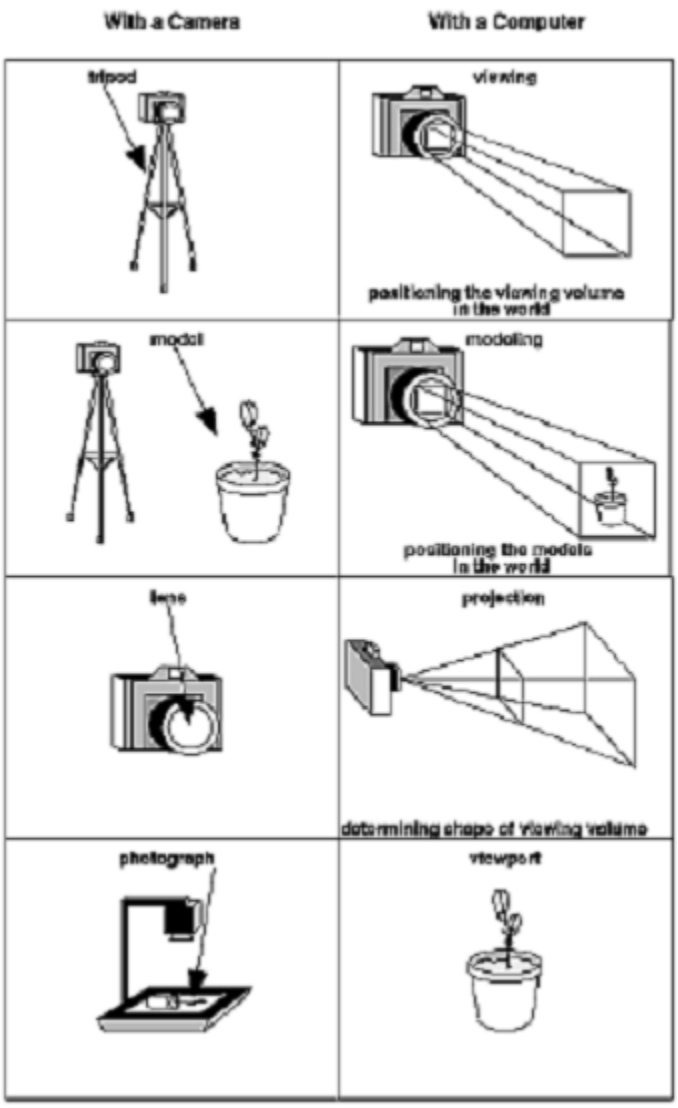
Gambar 3. Grafika Komputer: Transformasi dari Model 3D Obyek menjadi Citra



Jika dilihat secara analogi, hal di atas mirip dengan cara kerja kamera dalam mengambil foto dalam bidang fotografi (cf. Gambar 4). Model ini disebut model sintesis kamera.

Gambar 4. Analogi Pengambilan Gambar oleh Kamera





Untuk menghasilkan gambar dari obyek dengan skenario tertentu kita harus melakukan beberapa proses, yaitu:

- melakukan pengesetan kamera dalam bentuk setting lensa kamera (Transformasi Proyeksi),
- mengarah kamera dengan mengatur letak tripod (Transformasi Viewing),
- mengatur letak obyek (Transformasi Modeling), dan
- mengatur skala dan layout dari foto (Transformasi Viewport)

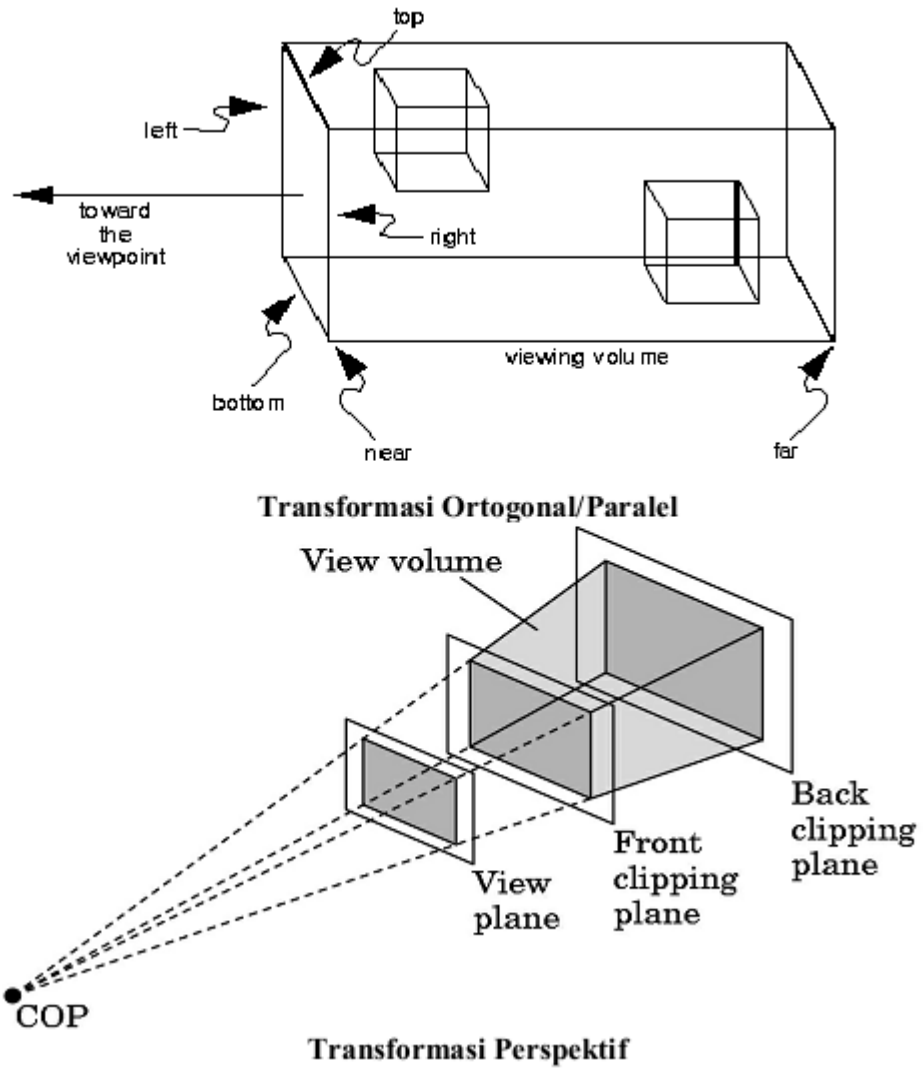
Kita telah mempelajari Transformasi Viewport pada tutorial sebelumnya dengan menggunakan perintah glViewport(). Pada tutorial ini, kita akan mempelajari transformasi-transformasi lainnya.

Transformasi Proyeksi

Lensa kamera dan mata manusia memiliki daerah penglihatan (viewing volume) yang berbentuk kerucut, namun karena bentuk display yang biasanya berbentuk segiempat membuat OpenGL (dan hampir semua API grafika komputer lain) lebih efisien memodelkan daerah penglihatan sebagai volume berbentuk piramida.

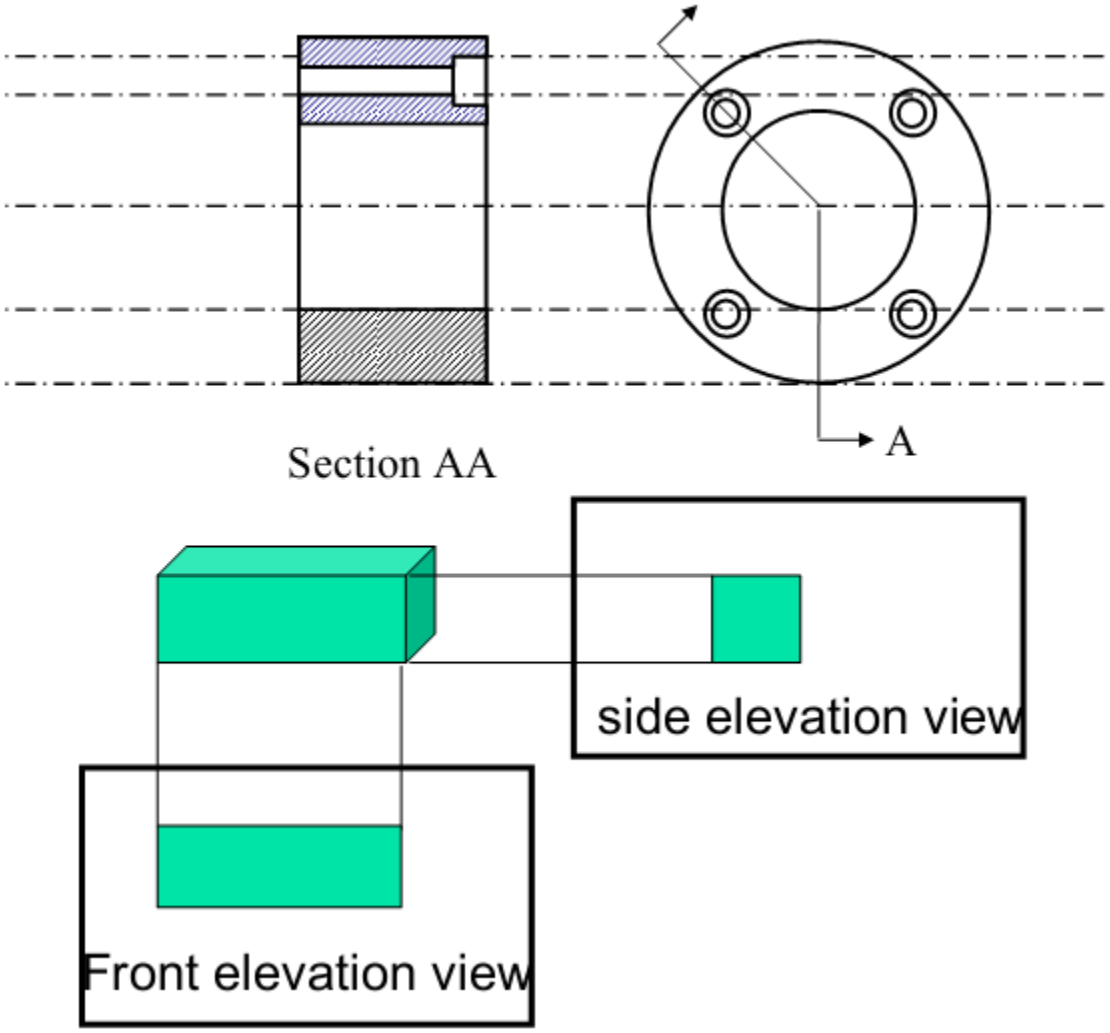
Tipe transformasi proyeksi ada dua macam, bergantung pada parameter dan bentuk piramidanya. Dua tipe transformasi tersebut adalah Transformasi Ortogonal/Paralel (Orthogonal Transformation) dan Transformasi Perspektif(Perspective Transformation) (cf. Gambar 5).

Gambar 5. Transformasi Ortogonal dan Transformasi Proyektif.



Pada tutorial sebelumnya digunakan transformasi orthogonal dengan parameter default. Transformasi ini membuat jarak benda relatif terhadap kamera tidak berpengaruh pada citra benda tersebut. Biasanya transformasi ini digunakan pada aplikasi-aplikasi teknik seperti gambar teknik (cf. Gambar 6). Untuk merubah parameter transformasi ortogonal dapat menggunakan perintah glOrtho() dengan didahului proses merubah status OpenGL ke mode proyeksi dengan perintah glMatrixMode(GL_PROJECTION).

Gambar 6. Contoh Transformasi Ortogonal/Paralel



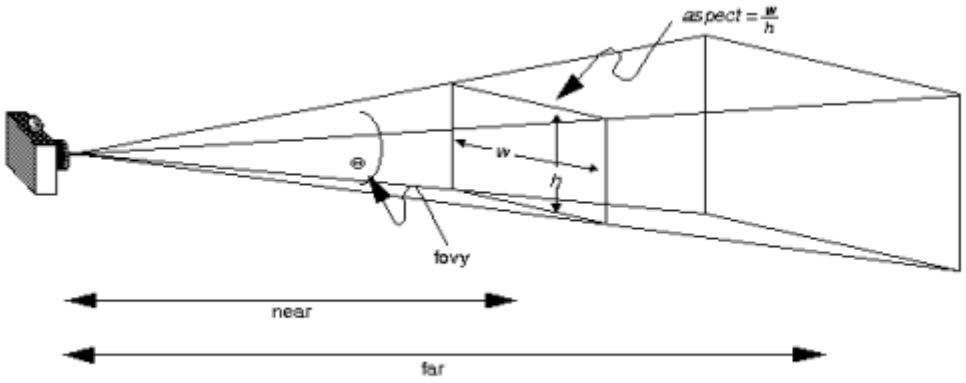
Pada tutorial ini dan selanjutnya, kita akan memfokuskan diri pada transformasi yang banyak digunakan yaitu transformasi perspektif. Pada transformasi jenis ini jarak benda akan mempengaruhi gambar yang di buat. Parameter transformasi jenis ini dapat dirubah dengan menggunakan gluPerspective()/glFrustum() , juga dengan didahului proses merubah status OpenGL ke mode proyeksi dengan perintah glMatrixMode(GL_PROJECTION).

```
glMatrixMode(GL_PROJECTION);

glLoadIdentity( );

gluPerspective(fovy, aspect, near, far);
```

fovy adalah sudut antara bidang bottom dan up.



Transformasi Viewing

Untuk menghasilkan gambar, kamera perlu diletakkan pada posisi yang tepat didepan pemandangan yang diinginkan. Secara default, dalam OpenGL kemera akan berada pada posisi (0,0,0) dengan menghadap ke arah z = -1 dengan sumbu y mengarah ke atas kamera. Hal ini dapat dilakukan dengan menggunakan perintah gluLookAt() dengan didahului proses merubah status OpenGL ke mode proyeksi dengan perintah glMatrixMode(GL_MODELVIEW).

Transformasi Modeling

Selain posisi dan orientasi kamera yang dapat dirubah-rubah, secara natural obyek juga dapat berpindah posisi dan orientasi relatif terhadap yang lain.Transformasi obyek dapat direpresentasikan dengan dua cara, yaitu:

- menggunakan matriks transformasi (glLoadMatrix)
- menggunakan operasi transformasi (glRotate , glTranslate)

dengan didahului proses merubah status OpenGL ke mode proyeksi dengan perintah glMatrixMode(GL_MODELVIEW).

Program dibawah memberi ilustrasi tentang bagaimana transformasi di atas diimplementasikan. Sebagai tambahan juga diberikan tentang callback keyboard untuk menangani input keyboard. Obyek ditranslasikan pada sumbu z dengan menggunakan tombol keyboard "," dan ".". Callback timer digunakan untuk timer yang di sini digunakan untuk animasi berputar.

Program 07. Proyeksi Perspektif

```
// - Viewing Volume of Perspective Projection
// - Try the keyboard callback
// - Reshape callback
// - Timer
#include  <stdio.h>
#include  <stdlib.h>
#include  <string.h>
#include  <stdarg.h>
#include  <GL/glut.h>

float  z_pos=0.0f;
float  rot=0.0f;

void mydisplay(){
    glClear(GL_COLOR_BUFFER_BIT);

 glLoadIdentity();
 glTranslatef(0.0,0.0f,z_pos);
  glRotatef(rot, 0, 0, 1);

 glBegin(GL_POLYGON);
```

```
  glColor3f(0, 1, 0);
    glVertex3f(-0.5, -0.5, -5);
  glColor3f(0, 0, 1);
  glVertex3f(-0.75, 0, -5);
  glColor3f(1, 0, 0);
  glVertex3f(-0.5, 0.5, -5);
  glColor3f(0, 1, 0);
    glVertex3f(0, 0.75, -5);
  glColor3f(0, 0, 1);
  glVertex3f(0.5, 0.5, -5);
  glColor3f(1, 0, 0);
  glVertex3f(0.75, 0, -5);
  glColor3f(0, 1, 0);
    glVertex3f(0.5, -0.5, -5);
  glColor3f(0, 0, 1);
  glVertex3f(0,-0.75, -5);
 glEnd();

 glFlush();
 glutSwapBuffers();
}

void init( void )
{

  glClearColor( 1.0, 0.0, 0.0, 1.0 ); // A Background Clear Color

 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
```

```
  gluPerspective(45, (GLdouble)500.0/(GLdouble)500.0, 0, 100);
 glMatrixMode(GL_MODELVIEW);
}

void resize( int  w, int  h )
{
   glViewport( 0, 0, (GLsizei) w, (GLsizei) h );
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();
  gluPerspective(45, (GLdouble)w/(GLdouble)h, 0, 100);
   glMatrixMode( GL_MODELVIEW );
}

void myTimeOut( int  id)
{
  // called if timer event
  // ...advance the state of animation incrementally...
 rot+=10;
 glutPostRedisplay();    // request redisplay
  glutTimerFunc(100, myTimeOut, 0);  // request next timer event
}

void myKeyboard(unsigned   char key, int  x, int  y)
{
  if((key=='<')||(key==',')) z_pos-=0.1f;
  if((key=='>')||(key=='.')) z_pos+=0.1f;
}

int  main( int  argc, char** argv)
```

```
{
 glutInit(&argc,argv);
  //glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
 glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);

  glutInitWindowSize(500,500);
 glutInitWindowPosition(0,0);
 glutCreateWindow("simple");

// callbacks
 glutDisplayFunc(mydisplay);
 glutKeyboardFunc(myKeyboard);
  glutTimerFunc(100, myTimeOut, 0);
 glutReshapeFunc(resize);
 init();
 glutMainLoop();
}
```

Tambahan:

Konsep Double Buffer. Pada program di atas mode display menggunakan tipe GLUT_DOUBLE yang diikuti oleh glutSwapBuffers() . Hal ini merupakan teknik yang disebut Double Buffer untuk menghindari flicker . Untuk mengetahui apa itu flicker, ubah mode display menjadi GLUT_SINGLE dan hapus/commented perintah glutSwapBuffer().