Praktikum 4

Proyeksi dan Animasi

Tujuan :

1. Memahami Matrik dan Viewport

Pelaksanaan :

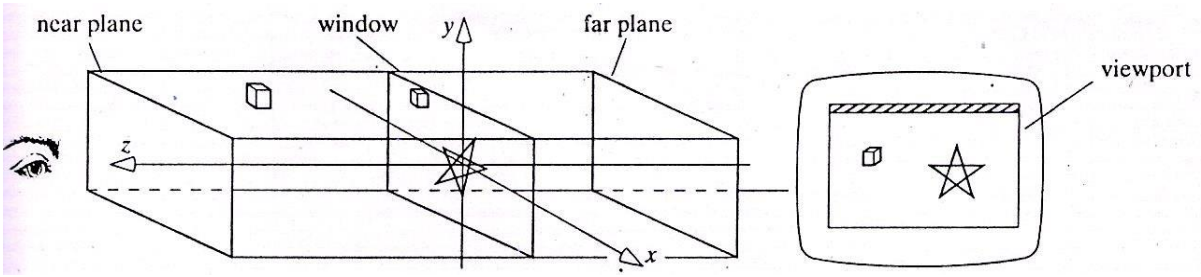
1. Penggunaan Pipeline (Proyeksi)
2. Membuat Animasi Benda

1. Penggunaan Pipeline (Proyeksi)

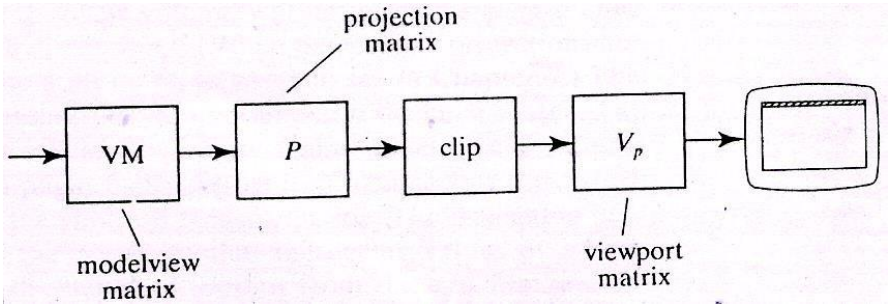
Sangat penting untuk dipahami bahwa sebenarnya dalam membangun objek menggunakan OpenGL yang terbentuk adalah objek 3D meskipun ditampilkan dalam layar 2D, hal tersebut merupakan transformasi 3D ke 2D. Ada 2 cara melakukan proses transformasi tersebut : "Projection Matrix and Modelview Matrix". Proyeksi merupakan proses bagaimana suatu objek ditransformasikan ke layar. Projection Matrix merupakan proyeksi yang melihat objek seperti segala sesuatu yang ada pada kehidupan sehari-hari. Maksudnya adalah bahwa makin jauh suatu objek maka akan makin kecil penampakan objek tersebut. Jika ingin menampilkan objek sesuai dengan kenyataan maka dipilihlah proyeksi perspektif ini. Sementara Modelview Matrix merupakan proyeksi yang menampilkan objek ke layar tanpa mengubah ukuran dari objek yang sesungguhnya yang akan ditampilkan.

Proyeksi ini lebih berpengaruh kepada penskalaan, rotasi, translasi, maupun pencahayaan.

Pada gambar 4.1 ditunjukkan proses komputer "melihat" obyek sampai menampilkannya di layar sebagai obyek 2D. "Mata" yang melihat pemandangan, mengarah ke window yang terletak pada bidang x-y, sepanjang sumbu z. Ruang pandang (*view volume*) dari kamera dibatasi oleh "balok berongga" yang luasnya sama dengan window, dan kedalamannya dibatasi oleh bidang dekat (*near plane*) dan bidang jauh (*far plane*). Setiap titik yang berada pada ruang pandang, diproyeksikan ke window sejajar dengan sumbu z. Dengan kata lain penggambaran titik 3D ke dalam window adalah memproyeksikan titik (x1, y1, z1) menjadi (x1, y2, 0). Titik yang berada diluar ruang pandang dipotong dan dibuang. Transformasi viewport yang terpisah memetakan titik yang diproyeksikan dari window ke peralatan tampilan (Layar komputer). Pada Gambar 4.2 menunjukkan proses penampilan gambar sampai ke view port. Proses ini dalam OpenGL disebut **pipelining**.



Gambar 4.1. Proses melihat dan menampilkan obyek 2D dengan OpenGL



Gambar 7.8. Proses Pipeline dalam OpenGL

Setiap vertek dari obyek harus melewati pipeline dengan pemanggilan fungsi glVertex3d(x, y, z). Vertek-vertek tersebut dikalikan dengan matriks, diantaranya adalah matrik modelview, matrik proyeksi dan matrik viewport. Dan terakhir dipetakan ke viewport.

Urutan perintah untuk proyeksi paralel adalah :

```
glMatrixMode(GL_PROJECTION);  // membangkitkan matrik proyeksi
   glLoadIdentity();          // inisialisasi matrik proyeksi
```

Untuk memahami Proyeksi ini cobalah kode berikut ini :

Kode : Proyeksi

```
#include <GL/glut.h>  void
Display(void)
{  glClear(GL_COLOR_BUFFER_BIT);
   glLoadIdentity();
   glBegin(GL_POLYGON);
   glColor3f(0.0,0.0,0.0);
   glVertex3f(-0.5,-0.5,-3.0);
   glColor3f(1.0,0.0,0.0);
   glVertex3f(0.5,-0.5,-3.0);
   glColor3f(0.0,0.0,1.0);
```

```
    glVertex3f(0.5,0.5,-3.0);
    glEnd();
    glFlush(); //Selesai rendering
}
void Reshape(int x, int y)
{  if (y == 0 || x == 0) return; //Jika tak ada lagi yang muncul,
maka berhenti
    //Mengatur proyeksi matrix baru
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();  //Sudut : 40
    derajat
    //Jarak potongan dataran terdekat: 0.5  //Jarak
    potongan dataran terjauh: 20.0
    gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);
    glMatrixMode(GL_MODELVIEW);
    glViewport(0,0,x,y); //gunakan seluruh window untuk rendering
}
int main (int argc, char **argv)
{
    //Inisialisasi GLUT  glutInit(&argc,
    argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(300,300);  glutCreateWindow("Intro");
    glClearColor(0.0,0.0,0.0,0.0);
    glutDisplayFunc(Display);
    glutReshapeFunc(Reshape); //mengatur kembali bentuk objek
    glutMainLoop();  return 0;
}
```

Anda dapat memanggilnya juga dengan perintah GL_PROJECTION or GL_MODELVIEW. Dalam fungsi Reshape Anda menspesifikasi sebuah projection matrix baru. Setelah menjalankan fungsi tadi, saat ini, anda sedang memuat identitas kedalam matrix, itu berarti “menghapus” proyeksi matrix sebelumya.

Dalam listing program diatas terdapat beberapa fungsi baru, **glLoadIdentity();** fungsi ini digunakan untuk menyimpan informasi yang akan kita masukkan. Penggunaan “if (y == 0

|| x == 0) return;” berarti fungsi tersebut hanya digunakan bila seluruh nilai x dan y sama dengan 0 selebihnya dilewatkan saja. gluPerspective(40.0, (GLdouble)x/(GLdouble)y, 0.5, 20.0); fungsi ini menunjukkan dari sudut mana kita melihat objek, tentu saja setelah di transformasi oleh projection dan modelview. glViewport(0,0,x,y); menunjukan besarnya area yang dapat dlihat oleh kita.

2. Membuat Animasi Benda

Selanjutnya, kita akan mencoba menggabungkan antara proses translasi dan rotasi yang telah dipelari sebelumnya dengan objek 3D. kali ini kita akan memanfaatkan sumbu z sehingga objek dapat berbentuk 3D. Selain memanfaatkan sumbu z, benda juga digerakkan melalui suatu proses / perhitungan tertentu.

Kode : Kubus Berotasi

```
#include <GL\glut.h>
GLfloat xRotated, yRotated, zRotated;  void
Display(void)
{  glClear(GL_COLOR_BUFFER_BIT);
   glLoadIdentity();
   glTranslatef(0.0,0.0,-4.0);
   glRotatef(xRotated,1.0,0.0,0.0);
   glRotatef(yRotated,0.0,1.0,0.0);
   glRotatef(zRotated,0.0,0.0,1.0);
   glScalef(2.0,1.0,1.0);
   glutWireCube(1.0);  glFlush();
   glutSwapBuffers();
}
void Reshape(int x, int y)
{  if (y == 0 || x == 0) return;
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);
   glMatrixMode(GL_MODELVIEW);  glViewport(0,0,x,y);
}
void Idle(void)
{  xRotated += 0.3;
   yRotated += 0.1;
   zRotated += -0.4;
   Display();
}

int main (int argc, char **argv)
{  glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize(300,300);
   glutCreateWindow("Cube example");
   glPolygonMode(GL_FRONT_AND_BACK,GL_LINE);
   xRotated = yRotated = zRotated = 0.0;
   glClearColor(0.0,0.0,0.0,0.0);
   glutDisplayFunc(Display);
   glutReshapeFunc(Reshape);  glutIdleFunc(Idle);
   glutMainLoop();  return 0;
}
```

GLfloat xRotated, yRotated, zRotated berarti kita menginisialisasi/mendeklarasikan variable xRotated, yRotated, dan zRotated dengan tipe data float. Agar proses transformasi

diatas dapat bekerja maka memerlukan fungsi *glutWireCube()*, ini juga berfungsi membentuk kubus itu sendiri, satu-satunya parameter yang digunakan dalam fungsi tersebut adalah ukuran. Semakin besar nilai yang diberikan maka semakin besar kubus yang terbentuk.

Untuk menambah pengetahuan tentang animasi yang bisa dilakukan, coba juga kode di bawah ini. Cobalah untuk menjalankan. gunakan keyboard S, s, E, e untuk menggerakkan benda tersebut.

Kode : Lengan Bergerak

```
#include <GL/glut.h>
static int shoulder = 0, elbow = 0;

 void init(void) {
      glClearColor (0.0, 0.0, 0.0, 0.0);
glShadeModel (GL_FLAT);
 }

 void display(void) {
      glClear (GL_COLOR_BUFFER_BIT);
glPushMatrix();
          glTranslatef (-1.0, 0.0, 0.0);
          glRotatef ((GLfloat) shoulder, 0.0, 0.0, 1.0);
glTranslatef (1.0, 0.0, 0.0);            glPushMatrix();
              glScalef (2.0, 0.4, 1.0);
glutWireCube (1.0);            glPopMatrix();
          glTranslatef (1.0, 0.0, 0.0);
          glRotatef ((GLfloat) elbow, 0.0, 0.0, 1.0);
glTranslatef (1.0, 0.0, 0.0);            glPushMatrix();
              glScalef (2.0, 0.4, 1.0);
glutWireCube (1.0);
glPopMatrix();       glPopMatrix();
glutSwapBuffers();
 }

 void reshape (int w, int h) {
      glViewport (0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode (GL_PROJECTION);        glLoadIdentity
();
      gluPerspective(65.0, (GLfloat) w/(GLfloat) h, 1.0, 20.0);
glMatrixMode(GL_MODELVIEW);        glLoadIdentity();
      glTranslatef (0.0, 0.0, -5.0);
 }
```

```
 void keyboard(unsigned char key, int x, int y) {
switch (key) {
```

```
            case 's': shoulder = (shoulder + 5) % 360;
glutPostRedisplay();                    break;
            case 'S': shoulder = (shoulder - 5) % 360;
glutPostRedisplay();                    break;
            case 'e': elbow = (elbow + 5) % 360;
glutPostRedisplay();                    break;
            case 'E': elbow = (elbow - 5) % 360;
glutPostRedisplay();                    break;
case 27: exit(0);                   break;
default: break;
      }
 }

 int main(int argc, char** argv) {
glutInit(&argc, argv);
      glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
glutInitWindowSize (700, 600);
glutInitWindowPosition (100, 100);
glutCreateWindow (argv[0]);        init();
      glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutKeyboardFunc(keyboard);
glutMainLoop();        return 0;
 }
```

Pertanyaan :

1. Apakah fungsi glutReshapeFunc dan glutPostRedisplay? Jelaskan!
2. Apa tugas prosedur Idle pada kode untuk kubus bergerak?
3. Jelaskan secara sederhana yang dimaksud dengan proyeksi, proyeksi projection dan modelview!

Tugas :

1. Jelaskan Bagaimana cara kerja Kode Lengan di atas!
2. Tambahkanlah telapak tangan beserta jari-jari dari lengan di atas. Telapak tangan dan jari-jari tersebut bisa digerakan menggunakan keyboard!
3. Simulasikan menggunakan sumbu x,y,z bagaimana operasi yang dilakukan ketika keyboard di ketik (dalam bentuk screenshot dan milimeter blok).