Praktikum 5
Menggunakan kamera dalam openGL

Tujuan :
1. Memahami kamera dalam openGL
2. Menggerakkan kamera

I. Menggunakan kamera dalam openGL

Pertama : Tidak ada kamera dalam OpenGL. Tapi, tentu saja kita bisa mensimulasikannya. Yang kita perlu ketahui dan pahami seperti halnya pada dunia nyata : menggerakan sebuah objek sama saja dengan menggerakan diri sendiri dari arah yang berlawan. Itu berarti: jika Anda mau mensimulasikan pandangan dari posisi (9|-3|10) Anda hanya cukup melakukan transformasi glTranslate(-9, 3, -10). Sama halnya dengan rotasi. glScale secara logika tidak menggunakan simulasi kamera, namun mendekati.

Anda mungkin tahu game seperti halflife, UnrealTournament dan lainnya. Semuanya memiliki “ghost mode”. Pemain dapat bergerak kesegala arah, melihat kemanapun mereka mau. Apa yang sebenarnya pemain lakukan adalah bergerak (maju atau mundur) atau berputar dengan derajat tertentu. Lalu mengapa tidak melakukan ini semua dengan perintah OpenGL ? Masalahnya adalah – sekali lagi – bahwa operasi matix harus di swapped. Jadi seandainya Anda tidak menggunakan glLoadIdentity tapi menggunakan matrix stack, Anda tidak akan bisa menjalankan translasi ketika pemain mau bergerak maju. Apa yang harus Anda lakukan yaitu mengambil matrix yang aktif saat ini (ada perintah openGL untuk melakukan ini), muat identitas matrix, panggil perintah tranformasi Anda dan gandakan matrix yang tersimpan dalam modelview matrix.tapi ada kemungkina lain, di

C++ Anda buat kelas kamera dengan metode “move”, “rotate”, and so on. Sebuah objek dari kelas ini akan menyimpan posisi dan arah dari kamera.

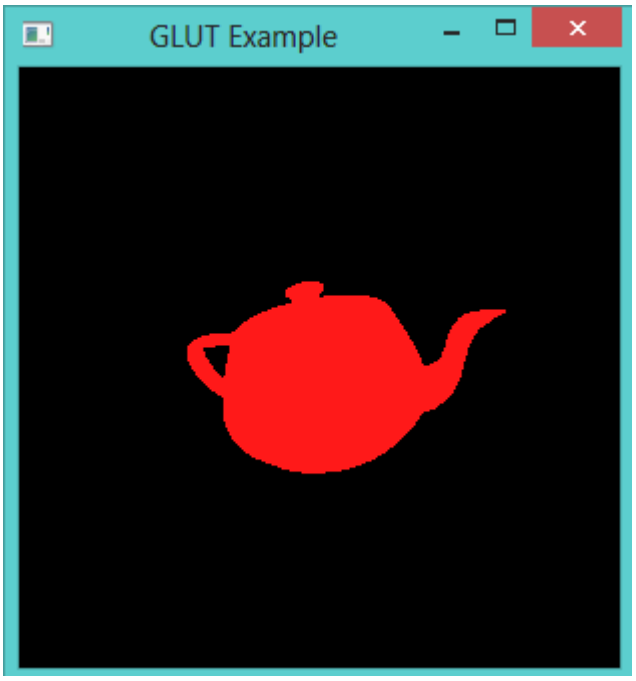
II. Menggerakkan kamera dalam openGL

```
#include <stdlib.h> // standard definitions
#include <stdio.h> // C I/O (for sprintf)
#include <math.h> // standard definitions
#include <GL/glut.h> // GLUT

double rotAngle = 10; // rotation angle (BEWARE: Global)
double rotAngle1 = 10; // rotation angle (BEWARE: Global)
//--------------------------------------------------------------------
// init
// Sets up some default OpenGL values.
//--------------------------------------------------------------------
void init()
{
  glClearColor(0, 0, 0, 0); // background color
  glClearDepth(1.0); // background depth value
```

```
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(60, 1, 1, 1000); // setup a perspective projection
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  gluLookAt( // set up the camera
        0.0, 0.0, 5.0, // eye position
        0.0, 0.0, 0.0, // lookat position
        0.0, 1.0, 0.0); // up direction
}
//-----------------------------------------------------------------
// display callback function
// This is called each time application needs to redraw itself.
// Most of the opengl work is done through this function.
//-----------------------------------------------------------------
void display()
{
  glClear(
    GL_COLOR_BUFFER_BIT | // clear the frame buffer (color)
    GL_DEPTH_BUFFER_BIT); // clear the depth buffer (depths)
  glPushMatrix(); // save the current camera transform
    glRotated(rotAngle, 0, 1, 0); // rotate by rotAngle about y-axis
    glRotated(rotAngle1, 1, 0, 0); // rotate by rotAngle about y-
axis
    glEnable(GL_COLOR_MATERIAL); // specify object color
    glColor3f(1.0, 0.1, 0.1); // redish
    glutSolidTeapot(1); // draw the teapot
  glPopMatrix(); // restore the modelview matrix
  glFlush(); // force OpenGL to render now
  glutSwapBuffers(); // make the image visible
}
//-----------------------------------------------------------------
// keyboard callback function
// This is called whenever a keyboard key is hit.
//-----------------------------------------------------------------
void keyboard(unsigned char k, int x, int y)
{
  switch (k)
  {
   case 'a':
     rotAngle += 5; // increase rotation by 5 degrees
     break;
   case 'y':
     rotAngle1 += 5; // increase rotation by 5 degrees
     break;
   case 'b':
     rotAngle1 -= 5; // increase rotation by 5 degrees
     break;
```

```
   case 'l':
    rotAngle -= 5; // decrease rotation by 5 degrees
    break;
  case 'q':
    exit(0); // exit
  }
  glutPostRedisplay(); // redraw the image now
}
//-----------------------------------------------------------------
// main program
// Where everything begins.
//-----------------------------------------------------------------
int main()
{
  glutInitDisplayMode( // initialize GLUT
     GLUT_DOUBLE | // use double buffering
     GLUT_DEPTH | // request memory for z-buffer
     GLUT_RGB ); // set RGB color mode
  glutCreateWindow("GLUT Example"); // create the window
  glutDisplayFunc(display); // call display() to redraw window
  glutKeyboardFunc(keyboard); // call keyboard() when key is hit
  init(); // our own initializations
  glutMainLoop(); // let GLUT take care of everything
  return 0;
}
```



Pertanyaan :

1. Jelaskan apa saja yang harus dilakukan agar dapat menggerakkan objek dengan keyboard?

2. Jelaskan Fungsi GluLookAt!
3. Dari kode di atas, terdapat dua proyeksi (Pojection, dan Model View), Jelaskan cara kerjanya. Mengapa keduanya digunakan?

Tugas :

Cobalah untuk mengubah-ubah kode di atas untuk :

a. Mengubah sudut geser
b. Mengubah papan tombol yang digunakan

Praktikum 6
Depth dan Lighting di OpenGL

Tujuan :
1. Memahami Dept dan Lighting di OpenGL
2. Scripting Dept dan Lighting

I. Memahami Dept dan Lighting di OpenGL

Dept testing digunakan untuk menghilangkan permukaan yang tersembunyi. Kita belum membutuhkan saat ini, karena kita selalu me-render objek berkerangka. Lighting jauh lebih terasa jika semua objek telah terisi dan sisanya objek berkerangka tetapi tidak realistis semuanya.

Dept testing susah – susah gampang untuk digunakan :

Berikan satu lagi konstanta ke glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH); Sekarang sebuah depth buffer telah dibuat dimana informasi dari depth telah tersimpan dalam tiap pixel. Hal berikunya yaitu mengaktifkan glEnable(GL_DEPTH_TEST); Sekarang Anda dapat menggambar benda-benda Anda seperti biasa.

Bagaimana cara mengaktifkan pencahayaan ?

Dengan memanggil fungsi glEnable(GL_LIGHTING); Kemudian Anda harus mengaktifkan masing-masing lampu yang Anda gunakan, misalnya glEnable (GL_LIGHT0);

Bagaimana cara menentukan sumber cahaya?

Pencahayaan adalah topik yang sangat kompleks dan Anda dapat mensimulasikan setiap kemungkinan cahaya dengan OpenGL. Untuk menetapkan sifat cahaya, Anda harus menggunakan perintah glLight*(), di mana * diisi dengan I atau f, jika anda ingin menentukan nilai sebagai vektor, gunakan "v". Untuk menentukan posisi pada OpenGL Anda pertama kali perlu mendefinisikan sebuah vektor empat-dimensi (x, y, z dan w). Nilai x, y dan z dibagi oleh w, itu berarti bahwa jika Anda menetapkan nilai "w" sebesar 0.0, posisi akan meluas sangat jauh. Contoh untuk ini adalah matahari. Jika Anda ingin menggunakan apa yang disebut "posisi" cahaya (yaitu w adalah nol), Anda harus menetapkan w menjadi 1,0.

Contoh : static GLfloat LightPos[] = {0.5, 0.5, 1.0, 1.0};

Kemudian panggil glLightfv(GL_LIGHT0, GL_POSITION, LightPos); Tentu saja Anda bisa menggunakan nilai lain untuk GL_LIGHT0.

Bagaimana cara menentukan sifat materi / material properties?

Material properties bekerja seperti light properties. Tentu saja mereka tidak memiliki posisi, tetapi memiliki properties ambient, diffuse dan specular. Terutama nilai specular, berbeda dari bahan ke bahan: Sebuah benda logam jauh lebih mengkilap dari kayu.

II. Scripting Dept dan Lighting

```
// Template untuk mainan objek 3D
// Kamera bisa maju mundur kiri kanan
// Sudah dilengkapi pencahayaan
// Sekarang pake texture

#include <math.h>
#include <GL/glut.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

float angle=0.0, deltaAngle = 0.0, ratio;
float x=0.0f,y=1.75f,z=15.0f; // posisi awal kamera
float lx=0.0f,ly=0.0f,lz=-1.0f;
int deltaMove = 0,h,w;
int bitmapHeight=12;

void Reshape(int w1, int h1)
{
  // Fungsi reshape
  if(h1 == 0) h1 = 1;
  w = w1;
  h = h1;
  ratio = 1.0f * w / h;
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glViewport(0, 0, w, h);
  gluPerspective(45,ratio,0.1,1000);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  gluLookAt(
     x, y, z,
     x + lx,y + ly,z + lz,
     0.0f,1.0f,0.0f);
}

void orientMe(float ang)
{
  // Fungsi ini untuk memutar arah kamera (tengok kiri/kanan)
  lx = sin(ang);
  lz = -cos(ang);
  glLoadIdentity();
  gluLookAt(x, y, z,
  x + lx,y + ly,z + lz,
  0.0f,1.0f,0.0f);
}

void moveMeFlat(int i)
{
  // Fungsi ini untuk maju mundur kamera
  x = x + i*(lx)*0.1;
  z = z + i*(lz)*0.1;
  glLoadIdentity();
```

```
  gluLookAt(x, y, z,
  x + lx,y + ly,z + lz,
  0.0f,1.0f,0.0f);
}

void Grid() {
  // Fungsi untuk membuat grid di "lantai"
  double i;
  const float Z_MIN = -50, Z_MAX = 50;
  const float X_MIN = -50, X_MAX = 50;
  const float gap = 1.5;
  glColor3f(0.5, 0.5, 0.5);
  glBegin(GL_LINES);
    for(i=Z_MIN;i<Z_MAX;i+=gap)
    {
       GlVertex3f(i, 0, Z_MIN);
       glVertex3f(i, 0, Z_MAX);
    }

    for(i=X_MIN;i<X_MAX;i+=gap)
    {
       glVertex3f(X_MIN, 0, i);
       glVertex3f(X_MAX, 0, i);
    }
  glEnd();
}

void KotakKayu()
{
  //depan
  glPushMatrix();
    glTranslatef(0,0,3);
    glBegin(GL_QUADS);
      glVertex3f(-3.0f,-3.0f,0.0f);
      glVertex3f(3.0f,-3.0f,0.0f);
      glVertex3f(3.0f,3.0f,0.0f);
      glVertex3f(-3.0f,3.0f,0.0f);
    glEnd();
  glPopMatrix();

  // atas
  glPushMatrix();
    glRotated(-90, 1, 0, 0);
    glTranslatef(0,0,3);
    glBegin(GL_QUADS);
      glVertex3f(-3.0f,-3.0f,0.0f);
      glVertex3f(3.0f,-3.0f,0.0f);
      glVertex3f(3.0f,3.0f,0.0f);
      glVertex3f(-3.0f,3.0f,0.0f);
    glEnd();
  glPopMatrix();

  // belakang
  glPushMatrix();
    glRotated(-180, 1, 0, 0);
```

```
    glTranslatef(0,0,3);
   glBegin(GL_QUADS);
     glTexCoord2f(0.0f,0.0f); // kiri bawah
     glVertex3f(-3.0f,-3.0f,0.0f);
     glTexCoord2f(1.0f,0.0f); // kanan bawah
     glVertex3f(3.0f,-3.0f,0.0f);
     glTexCoord2f(1.0f,1.0f); // kanan atas
     glVertex3f(3.0f,3.0f,0.0f);
     glTexCoord2f(0.0f,1.0f); // kanan bawah
     glVertex3f(-3.0f,3.0f,0.0f);
   glEnd();
  glPopMatrix();

  // bawah
  glPushMatrix();
    glRotated(90, 1, 0, 0);
    glTranslatef(0,0,3);
    glBegin(GL_QUADS);
      glTexCoord2f(0.0f,0.0f); // kiri bawah
      glVertex3f(-3.0f,-3.0f,0.0f);
      glTexCoord2f(1.0f,0.0f); // kanan bawah
      glVertex3f(3.0f,-3.0f,0.0f);
      glTexCoord2f(1.0f,1.0f); // kanan atas
      glVertex3f(3.0f,3.0f,0.0f);
      glTexCoord2f(0.0f,1.0f); // kanan bawah
      glVertex3f(-3.0f,3.0f,0.0f);
    glEnd();
  glPopMatrix();

  // kiri
  glPushMatrix();
    glRotated(-90, 0, 1, 0);
    glTranslatef(0,0,3);
    glBegin(GL_QUADS);
      glVertex3f(-3.0f,-3.0f,0.0f);
      glVertex3f(3.0f,-3.0f,0.0f);
      glVertex3f(3.0f,3.0f,0.0f);
      glVertex3f(-3.0f,3.0f,0.0f);
    glEnd();
  glPopMatrix();

  // kanan
  glPushMatrix();
    glRotated(90, 0, 1, 0);
    glTranslatef(0,0,3);
    glBegin(GL_QUADS);
      glVertex3f(-3.0f,-3.0f,0.0f);
      glVertex3f(3.0f,-3.0f,0.0f);
      glVertex3f(3.0f,3.0f,0.0f);
      glVertex3f(-3.0f,3.0f,0.0f);
    glEnd();
  glPopMatrix();
}
```

```
void display() {
  // Kalau move dan angle tidak nol, gerakkan kamera...
  if (deltaMove)
    moveMeFlat(deltaMove);
  if (deltaAngle) {
    angle += deltaAngle;
    orientMe(angle);
  }
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  // Gambar grid
  Grid();

  // Gambar objek di sini...
  KotakKayu();
  glutSwapBuffers();
  glFlush();
}

void pressKey(int key, int x, int y) {
  // Fungsi ini akan dijalankan saat tombol keyboard ditekan dan
belum dilepas
  // Selama tombol ditekan, variabel angle dan move diubah => kamera
bergerak
  switch (key) {
    case GLUT_KEY_LEFT : deltaAngle = -0.01f;break;
    case GLUT_KEY_RIGHT : deltaAngle = 0.01f;break;
    case GLUT_KEY_UP : deltaMove = 1;break;
    case GLUT_KEY_DOWN : deltaMove = -1;break;
  }
}

void releaseKey(int key, int x, int y) {
  // Fungsi ini akan dijalankan saat tekanan tombol keyboard dilepas
  // Saat tombol dilepas, variabel angle dan move diset nol =>
kamera berhenti
  switch (key) {
    case GLUT_KEY_LEFT :
       if (deltaAngle < 0.0f)
          deltaAngle = 0.0f;
       break;
    case GLUT_KEY_RIGHT : if (deltaAngle > 0.0f)
       deltaAngle = 0.0f;
       break;
    case GLUT_KEY_UP : if (deltaMove > 0)
       deltaMove = 0;
       break;
    case GLUT_KEY_DOWN : if (deltaMove < 0)
       deltaMove = 0;
       break;
  }
}

// Variable untuk pencahayaan
```

```
const GLfloat light_ambient[] = { 0.5f, 0.5f, 0.5f, 0.0f };
const GLfloat light_diffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_position[] = { 0.0f, 20.0f, 10.0f, 1.0f };
const GLfloat mat_ambient[] = { 0.7f, 0.7f, 0.7f, 1.0f };
const GLfloat mat_diffuse[] = { 0.8f, 0.8f, 0.8f, 1.0f };
const GLfloat mat_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat high_shininess[] = { 100.0f };

void lighting(){
  // Fungsi mengaktifkan pencahayaan
  glEnable(GL_DEPTH_TEST);
  glDepthFunc(GL_LESS);
  glEnable(GL_LIGHT0);
  glEnable(GL_NORMALIZE);
  glEnable(GL_COLOR_MATERIAL);
  glEnable(GL_LIGHTING);
  glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
  glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
  glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
  glLightfv(GL_LIGHT0, GL_POSITION, light_position);
  glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
  glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
  glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
  glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
}

void init(void)
{
  glEnable (GL_DEPTH_TEST);
  glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
}

int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
  glutInitWindowPosition(100,100);
  glutInitWindowSize(640,480);
  glutCreateWindow("3D Lighting");
  glutIgnoreKeyRepeat(1); // Mengabaikan key repeat (saat tombol
keyboard dipencet terus)
  glutSpecialFunc(pressKey);
  glutSpecialUpFunc(releaseKey);
  glutDisplayFunc(display);
  glutIdleFunc(display); // Fungsi display-nya dipanggil terus-
menerus
  glutReshapeFunc(Reshape);
  lighting();
  init();
  glutMainLoop();
  return(0);
}
```

Pertanyaan :

1. Jelaskan Lighting yang ada di OpenGL !
2. Jelaskan apa kegunaan void lighting di atas !
3. Analisislah Bagaimana Kubus, Grid dan pencahayaan tersebut dapat dibuat!

Tugas :

Apabila sebelumnya sudah pernah membuat sebuah mobil 2D, sekarang buatlah sebuah mobil 3D yang memanfaatkan Depth dan Lighting.

Catatan :

1. Mobil yang dibangun boleh tidak menggunakan roda (jika dengan roda nilai tambah)
2. Agar depth dan Lighting bekerja, mobil yang dibuat harus terbentuk dari sebuah rangka. (seperti membuat rangka kubus yang terdiri dari 6 kotak kemudian dirotasi dan ditranslasi menjadi kubus)