LAPORAN PRAKTIKUM

Mata Kuliah : GRAFIKA DAN KOMPUTASI VISUAL

Praktikum Ke : 7 & 8

JUDUL PRAKTIKUM

TEKSTUR DENGAN GAMBAR DAN BAYANGAN

DISUSUN OLEH :

Nama : Aisabilla Dzuha Rahmahana

NIM : 24010314120018

Asisten : Mariza Putri, Rizki Syafwan, Rahmat Hidayat

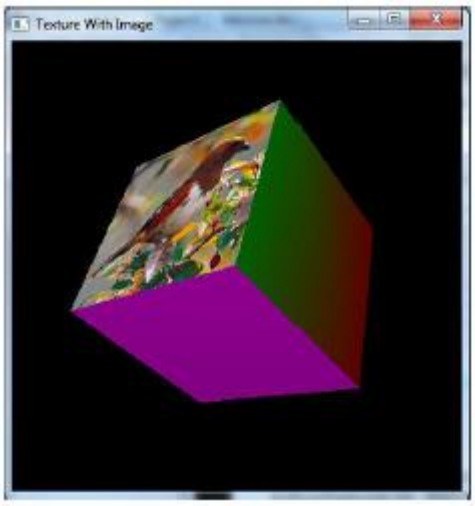
JURUSAN ILMU KOMPUTER / INFORMATIKA
FAKULTAS SAINS DAN MATEMATIKA
UNIVERSITAS DIPONEGORO
2016

LAPORAN PRAKTIKUM GRAFIKA DAN KOMPUTASI VISUAL KE TUJUH

1. Pertanyaan dan Jawaban

Pertanyaan :

1. Tekstur apa saja yang dibuat pada kubus tersebut!



Tugas :

1. Analisa dan Jelaskan alur / cara kerja teksture yang telah dikerjakan.
2. Buatlah sebuah kubus yang setiap sisinya berisi sebuah gambar.

Jawaban :

1. Kubus ini menggunakan tekstur gambar, untuk menempelkan gambar sebagai tekstur pada sisi kubus memanfaatkan procedure Gluint loadTexture(Image*image). Untuk menempelkan tekstur warna pada sisi kubus menggunakan tekstur Solid Color, selain itu tekstur gradient digunakan untuk membuat gradasi atau perpaduan warna. Terdapat beberapa fungsi yang dimanfaatkan untuk membuat tekstur kubus tersebut, antara lain:

```
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, _textureIdAtas);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

2. Kode Program dan Penjelasan Tugas

a. Analisa dan Jelaskan alur / cara kerja teksture yang telah dikerjakan.

Tahap pertama dalam program yaitu memasukkan file header imageloader.h dan imageloader.cpp. Imageloader.cpp merupakan library tambahan yang dibuat untuk memanggil fungsi – fungsi, dalam proses memuat gambar untuk tekstur.

Selanjutnya adalah inisialisasi panjang setiap sisi kubus yang akan diberikan tekstur menggunakan `const float BOX_SIZE = 7.0f;`

`float_angle = 0;` digunakan untuk merotasi kubus.

`Gluint_textureId;` digunakan untuk menginisialisasi ID OpenGL tekstur.

Selanjutnya procedure dibawah ini digunakan untuk memuat tekstur yang akan ditempelkan pada sisi kubus.

```
GLuint loadTexture(Image* image) {
GLuint textureId;
glGenTextures(1, &textureId);
glBindTexture(GL_TEXTURE_2D, textureId);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image->width, image->height, 0,
GL_RGB, GL_UNSIGNED_BYTE, image->pixels);
return textureId;
}
```

Kemudian prosedur init rendering digunakan untuk menjalankan fungsi – fungsi yang digunakan dalam tekstur. Gambar yang digunakan sebagai tekstur dimuat pada variabel penampung image dengan type Image*, kemudian variabel image ini dimuat pada variabel tekstureID dengan fungsi loadTexture.

```
void initRendering() {
glEnable(GL_DEPTH_TEST);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_NORMALIZE);
glEnable(GL_COLOR_MATERIAL);
Image* image = loadBMP("bg.bmp");
_textureId = loadTexture(image);
delete image;
}
```

Pengaturan posisi dari tekstur pada objek kubus dilakukan dengan menggunakan fungsi yang menunjukkan koordinat tekstur yaitu `glTextCoord2f`. Untuk menampilkan teksture, bagian sisi yang akan diberikan teksture diberikan perintah

```
glEnable(GL_TEXTURE_2D)
```

```
glBindTexture(GL_TEXTURE_2D, [VariabelTekstur]);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glColor3f(1.0f, 1.0f, 1.0f);
```

b. Tugas.cpp

Tugas.cpp adalah program yang berisi kubus dimana setiap sisi kubusnya telah diberi teksture gambar.

```
//NAMA          :Aisabilla Dzuha R
//NIM           :24010314120018

#include <iostream>
#include <stdlib.h>

#ifdef __APPLE__
#include <OpenGL/OpenGL.h>
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

#include "imageloader.h"
#include "imageloader.cpp"

using namespace std;

const float BOX_SIZE = 9.0f; //Panjang tiap sisi kubus
float _angle = 0;            //Rotasi terhadap box
GLuint
textureIdDepan,_textureIdBelakang,_textureIdAtas,_textureIdBawah,_texture
IdKanan,_textureIdKiri;           //ID OpenGL untuk tekstur

void handleKeypress(unsigned char key, int x, int y) {
   switch (key) {
         case 27:             //Tekan Escape untuk EXIT
               exit(0);
   }
}

//Membuat gambar menjadi tekstur kemudian berikan ID pada tekstur
GLuint loadTexture(Image* image) {
```

```
   GLuint textureId;
   glGenTextures(1, &textureId);
   glBindTexture(GL_TEXTURE_2D, textureId);
   glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image->width,
image->height, 0, GL_RGB, GL_UNSIGNED_BYTE, image->pixels);
   return textureId;
}
void initRendering() {
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_NORMALIZE);
   glEnable(GL_COLOR_MATERIAL);

   Image* imgdepan = loadBMP("bg.bmp");
   Image* imgkanan = loadBMP("bg2.bmp");
   Image* imgkiri = loadBMP("bg3.bmp");
   Image* imgatas = loadBMP("bg4.bmp");
   Image* imgbawah = loadBMP("bg5.bmp");
   Image* imgbelakang = loadBMP("bg6.bmp");
   _textureIdDepan = loadTexture(imgdepan);
   _textureIdBelakang = loadTexture(imgbelakang);
   _textureIdKanan = loadTexture(imgkanan);
   _textureIdKiri = loadTexture(imgkiri);
   _textureIdAtas = loadTexture(imgatas);
   _textureIdBawah = loadTexture(imgbawah);
   delete imgdepan;
   delete imgbelakang;
   delete imgatas;
   delete imgbawah;
   delete imgkanan;
   delete imgkiri;
}
void handleResize(int w, int h) {
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(45.0, (float)w / (float)h, 1.0, 200.0);
}

void drawScene() {
```

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

glTranslatef(0.0f, 0.0f, -20.0f);

GLfloat ambientLight[] = {0.3f, 0.3f, 0.3f, 1.0f};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);

GLfloat lightColor[] = {0.7f, 0.7f, 0.7f, 1.0f};
GLfloat lightPos[] = {-2 * BOX_SIZE, BOX_SIZE, 4 * BOX_SIZE, 1.0f};
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor);
glLightfv(GL_LIGHT0, GL_POSITION, lightPos);

glRotatef(-_angle, 1.0f, 1.0f, 0.0f);

glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, _textureIdAtas);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glColor3f(1.0f, 1.0f, 1.0f);

//Sisi atas
glBegin(GL_QUADS);
glNormal3f(0.0, 1.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(-BOX_SIZE / 2, BOX_SIZE / 2, -BOX_SIZE / 2);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(-BOX_SIZE / 2, BOX_SIZE / 2, BOX_SIZE / 2);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(BOX_SIZE / 2, BOX_SIZE / 2, BOX_SIZE / 2);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(BOX_SIZE / 2, BOX_SIZE / 2, -BOX_SIZE / 2);
 glEnd();

 glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, _textureIdBawah);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glColor3f(1.0f, 1.0f, 1.0f);
```

```
 glBegin(GL_QUADS);
//Sisi bawah
glNormal3f(0.0, -1.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(-BOX_SIZE / 2, -BOX_SIZE / 2, -BOX_SIZE / 2);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(BOX_SIZE / 2, -BOX_SIZE / 2, -BOX_SIZE / 2);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(BOX_SIZE / 2, -BOX_SIZE / 2, BOX_SIZE / 2);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(-BOX_SIZE / 2, -BOX_SIZE / 2, BOX_SIZE / 2);
 glEnd();


 glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, _textureIdKiri);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glColor3f(1.0f, 1.0f, 1.0f);

 glBegin(GL_QUADS);
//Sisi kiri
glNormal3f(-1.0, 0.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f);
glVertex3f(-BOX_SIZE / 2, -BOX_SIZE / 2, -BOX_SIZE / 2);
glTexCoord2f(1.0f, 0.0f);
glVertex3f(-BOX_SIZE / 2, -BOX_SIZE / 2, BOX_SIZE / 2);
glTexCoord2f(1.0f, 1.0f);
glVertex3f(-BOX_SIZE / 2, BOX_SIZE / 2, BOX_SIZE / 2);
glTexCoord2f(0.0f, 1.0f);
glVertex3f(-BOX_SIZE / 2, BOX_SIZE / 2, -BOX_SIZE / 2);
 glEnd();

 glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, _textureIdKanan);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glColor3f(1.0f, 1.0f, 1.0f);

 glBegin(GL_QUADS);
```

```
    //Sisi kanan
    glNormal3f(1.0, 0.0f, 0.0f);
    glTexCoord2f(0.0f, 0.0f);
    glVertex3f(BOX_SIZE / 2, -BOX_SIZE / 2, -BOX_SIZE / 2);
    glTexCoord2f(1.0f, 0.0f);
    glVertex3f(BOX_SIZE / 2, BOX_SIZE / 2, -BOX_SIZE / 2);
    glTexCoord2f(1.0f, 1.0f);
    glVertex3f(BOX_SIZE / 2, BOX_SIZE / 2, BOX_SIZE / 2);
    glTexCoord2f(0.0f, 1.0f);
    glVertex3f(BOX_SIZE / 2, -BOX_SIZE / 2, BOX_SIZE / 2);
     glEnd();


    glBindTexture(GL_TEXTURE_2D, _textureIdDepan);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glColor3f(1.0f, 1.0f, 1.0f);
    glBegin(GL_QUADS);

    //Sisi depan
    glNormal3f(0.0, 0.0f, 1.0f);
    glTexCoord2f(0.0f, 0.0f);
    glVertex3f(-BOX_SIZE / 2, -BOX_SIZE / 2, BOX_SIZE / 2);
    glTexCoord2f(1.0f, 0.0f);
    glVertex3f(BOX_SIZE / 2, -BOX_SIZE / 2, BOX_SIZE / 2);
    glTexCoord2f(1.0f, 1.0f);
    glVertex3f(BOX_SIZE / 2, BOX_SIZE / 2, BOX_SIZE / 2);
    glTexCoord2f(0.0f, 1.0f);
    glVertex3f(-BOX_SIZE / 2, BOX_SIZE / 2, BOX_SIZE / 2);
    glEnd();

    //Sisi belakang
    glBindTexture(GL_TEXTURE_2D, _textureIdBelakang);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glColor3f(1.0f, 1.0f, 1.0f);
    glBegin(GL_QUADS);
    glNormal3f(0.0, 0.0f, -1.0f);
    glTexCoord2f(0.0f, 0.0f);
    glVertex3f(-BOX_SIZE / 2, -BOX_SIZE / 2, -BOX_SIZE / 2);
    glTexCoord2f(1.0f, 0.0f);
```

```
   glVertex3f(-BOX_SIZE / 2, BOX_SIZE / 2, -BOX_SIZE / 2);
   glTexCoord2f(1.0f, 1.0f);
   glVertex3f(BOX_SIZE / 2, BOX_SIZE / 2, -BOX_SIZE / 2);
   glTexCoord2f(0.0f, 1.0f);
   glVertex3f(BOX_SIZE / 2, -BOX_SIZE / 2, -BOX_SIZE / 2);

   glEnd();
   glDisable(GL_TEXTURE_2D);

   glutSwapBuffers();
}

//Panggil setiap 25ms
void update(int value) {
   _angle += 1.0f;
   if (_angle > 360) {
         _angle -= 360;
   }
   glutPostRedisplay();
   glutTimerFunc(25, update, 0);
}

int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize(400, 400);

   glutCreateWindow("Texture With Image");
   initRendering();

   glutDisplayFunc(drawScene);
   glutKeyboardFunc(handleKeypress);
   glutReshapeFunc(handleResize);
   glutTimerFunc(25, update, 0);

   glutMainLoop();
   return 0;
}
```

Source code diatas, diguanakan untuk membuat kubus dimana setiap sisinya telah diberi tekstur gambar. Terdapat beberapa gambar yang akan disisipkan untuk setiap sisinya, yaitu:

1. bg.bmp untuk sisi depan.

2. bg2.bmp untuk sisi kanan.

3. bg3.bmp untuk sisi kiri.

4. bg4.bmp untuk sisi atas.

5. bg5.bmp untuk sisi bawah.

6. bg6.bmp untuk sisi belakang.

Tahap pertama dalam program yaitu memasukkan file header imageloader.h dan imageloader.cpp. Imageloader.cpp merupakan library tambahan yang dibuat untuk memanggil fungsi – fungsi, dalam proses memuat gambar untuk tekstur.

Selanjutnya adalah inisialisasi panjang setiap sisi kubus yang akan diberikan tekstur menggunakan `const float BOX_SIZE = 7.0f;`

`float_angle = 0;` digunakan untuk merotasi kubus.

`Gluint_textureId;` digunakan untuk menginisialisasi ID OpenGL tekstur.

Selanjutnya procedure dibawah ini digunakan untuk memuat tekstur yang akan ditempelkan pada sisi kubus.

```
GLuint loadTexture(Image* image) {
GLuint textureId;
glGenTextures(1, &textureId);
glBindTexture(GL_TEXTURE_2D, textureId);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image->width, image->height, 0,
GL_RGB, GL_UNSIGNED_BYTE, image->pixels);
return textureId;
}
```

Kemudian prosedur init rendering digunakan untuk menjalankan fungsi – fungsi yang digunakan dalam tekstur. Gambar yang digunakan sebagai tekstur dimuat pada variabel penampung image dengan type Image*, kemudian variabel image ini dimuat pada variabel tekstureID dengan fungsi loadTexture. Source code dibawah adalah inisialisasi untuk variabel yang menyimpan tekstur untuk menyimpan loadTeksture yaitu ID openGL untuk teksture _textureIdDepan, _textureIdBelakang, _textureIdAtas, _textureIdBawah, _textureIdKanan,_textureIdKiri.

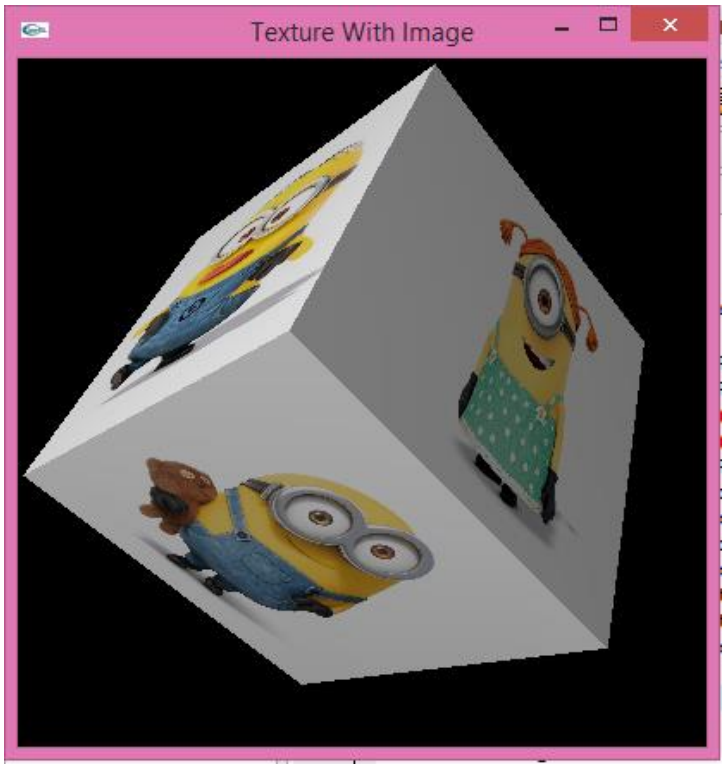
Kemudian memasukkan image menjadi teksture, digunakan pada bagian initRendering untuk meload image (loadBMP ke variabel bertipe Image*) yang digunakan sebagai tekstur. Setelah itu load teksture ke setiap variabel tekstur yang akan ditempelkan pada setiap sisi. Untuk membedakan tekstur setiap sisi, maka setiap sisi diawali dengan pendeklarasian tekstur dengan image masing-masing.

```
void initRendering() {
glEnable(GL_DEPTH_TEST);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_NORMALIZE);
glEnable(GL_COLOR_MATERIAL);
Image* imgdepan = loadBMP("bg.bmp");
Image* imgkanan = loadBMP("bg2.bmp");
Image* imgkiri = loadBMP("bg3.bmp");
Image* imgatas = loadBMP("bg4.bmp");
Image* imgbawah = loadBMP("bg5.bmp");
Image* imgbelakang = loadBMP("bg8.bmp");
_textureIdDepan = loadTexture(imgdepan);
_textureIdBelakang = loadTexture(imgbelakang);
_textureIdKanan = loadTexture(imgkanan);
_textureIdKiri = loadTexture(imgkiri);
_textureIdAtas = loadTexture(imgatas);
_textureIdBawah = loadTexture(imgbawah);
delete imgdepan;
delete imgbelakang;
delete imgatas;
delete imgbawah;
delete imgkanan;
delete imgkiri;}
```

Kemudian aktifkan fungsi teksture 2D pada setiap sisi kubus dengan menggunakan _tekstureId(namasisi) yang sesuai dengan sisinya. Sebagai contoh

```
//Sisi depan
 glNormal3f(0.0, 0.0f, 1.0f);
 glTexCoord2f(0.0f, 0.0f);
 glVertex3f(-BOX_SIZE / 2, -BOX_SIZE / 2, BOX_SIZE / 2);
 glTexCoord2f(1.0f, 0.0f);
 glVertex3f(BOX_SIZE / 2, -BOX_SIZE / 2, BOX_SIZE / 2);
 glTexCoord2f(1.0f, 1.0f);
 glVertex3f(BOX_SIZE / 2, BOX_SIZE / 2, BOX_SIZE / 2);
 glTexCoord2f(0.0f, 1.0f);
 glVertex3f(-BOX_SIZE / 2, BOX_SIZE / 2, BOX_SIZE / 2);
 glEnd();
```

3. Screenshot



Gambar 1. Kubus dengan setiap sisi bertekstur gambar.

LAPORAN PRAKTIKUM GRAFIKA DAN KOMPUTASI VISUAL KE DELAPAN

1. Pertanyaan dan Jawaban

Pertanyaan & Tugas :

1. Ubahlah untuk object yang berbeda
2. Analisis dan Jelaskan secara rinci proses membuat bayangannya.
3. Jelaskan Fungsi dan alur glShadowProjection.

Jawaban :

1. Mengubah untuk objek yang berbeda

```
#include <math.h>
#include <stdio.h>
#include <GL/glut.h>

double rx = 0.0;
double ry = 0.0;
float l[] = { 0.0, 80.0, 0.0 }; // koordinat sumber cahaya
float n[] = { 0.0, -40.0, 0.0 };
float e[] = { 0.0, -60.0, 0.0 };

void help();

// obyek yang akan digambar
void balok(double panjang, double lebar, double tinggi)
{
glScalef(panjang, tinggi, lebar);
glutSolidCube(1.0);
}
//-----------------procedure membuat silider ----------------------//
void silinder(float alas,float tutup,float tinggi)
{
float i;
glPushMatrix();
glTranslatef(1.0,0.0,-alas/8);
glutSolidCone(alas,0,32,4);
for(i=0;i<=tinggi;i+=alas/24)
{
glTranslatef(0.0,0.0,alas/24);
glutSolidTorus(alas/4,alas-((i*(alas-tutup))/tinggi),16,16);
}
glTranslatef(0.0,0.0,alas/4);
glutSolidCone(tutup,0,20,1);
glColor3f(1.,0.,0.);
glPopMatrix();
}

void draw()
{
glPushMatrix();
glRotated(-180,0,1,0);
```

```
glTranslatef(0,4.0,0.0);
glPushMatrix();
//body mobil
//Bagian tengah atas
glPushMatrix();
glColor3f(0.0, 0.0, 0);
balok(11, 10, 5);
glPopMatrix();
//Bagian bawah
glPushMatrix();
glColor3f(0.0, 0.0, 0);
glTranslatef(1, -4, 0);
balok(30, 10, 5);
glPopMatrix();
//Bagian jendela depan
glPushMatrix();
glRotatef(-35, 0, 0, 15);
glTranslatef(6.75, 3, 0);
balok(8,10,4);
glPopMatrix();
//Bagian jendela belakang
glPushMatrix();
glRotatef(40, 0, 0, 15);
glTranslatef(-6.5, 3.25, 0);
balok(8,10,4);
glPopMatrix();
//kaca belakang
glPushMatrix();
glRotatef(55, 0, 0, 15);
glTranslatef(5,-6,0);
glColor3f(0.0, 0.0, 0);
balok(0.25, 8, 5);
glPopMatrix();
//kaca depan
glPushMatrix();
glRotatef(-50, 0, 0, 15);
glTranslatef(-5.25, -6,0);
glColor3f(0.0, 0.0, 0);
balok(0.25, 8, 5);
glPopMatrix();
//roda
glPushMatrix();
glColor3f(0.0, 0.0, 0);
glTranslatef(8, -6,-4.80);
silinder(2, 2,1);
glPopMatrix();
glPushMatrix();
glColor3f(0,0,0);
glTranslatef(-8, -6,-4.80);
silinder(2, 2,1);
glPopMatrix();
glPushMatrix();
glColor3f(0.0, 0.0, 0);
glTranslatef(8,-6,4);
silinder(2, 2,1);
glPopMatrix();
glPushMatrix();
glColor3f(0,0,0);
glTranslatef(-8,-6,4);
silinder(2, 2,1);
glPopMatrix();
```

```
//velg
glPushMatrix();
glColor3f(0.0, 0.0, 0);
glTranslatef(-7.93,-5.93,4.75);
silinder(1.5, 1.5,0.5);
glPopMatrix();
glPushMatrix();
glColor3f(0.0, 0.0, 0);
glTranslatef(7.93,-5.93,4.75);
silinder(1.5, 1.5,0.5);
glPopMatrix();
glPushMatrix();
glRotatef(180, 0, 0, 0);
glColor3f(0.0, 0.0, 0);
glTranslatef(-9.8,6,4.8);
silinder(1.5, 1.5,0.5);
glPopMatrix();
glPushMatrix();
glRotatef(180, 0, 0, 0);
glColor3f(0.0, 0.0, 0);
glTranslatef(6,6,4.8);
silinder(1.5, 1.5,0.5);
glPopMatrix();
//lampu depan
glPushMatrix();//kiri
glRotatef(90,0,1,0);
glColor3f(0.0, 0.0, 0);
glTranslatef(-3.5, -4,16);
balok(2,0.1,1);
glPopMatrix();
glPushMatrix();//kanan
glRotatef(90,0,1,0);
glColor3f(0.0, 0.0, 0);
glTranslatef(3.5, -4,16);
balok(2,0.1,1);
glPopMatrix();
//lampu sign
glPushMatrix(); //kanan
glRotatef(90,0,1,0);
glColor3f(0.0, 0.0, 0);
glTranslatef(-3.5, -4.7, 16);
balok(2,0.1,0.5);
glPopMatrix();
glPushMatrix(); //kiri
glRotatef(90,0,1,0);
glColor3f(0.0, 0.0, 0);
glTranslatef(3.5, -4.7, 16);
balok(2,0.1,0.5);
glPopMatrix();
//lampu belakang
glPushMatrix();
glRotatef(90,0,1,0);
glColor3f(0.0, 0.0, 0);
glTranslatef(-2.5, -3,-14);
balok(2,0.1,1.2);
glPopMatrix();
glPushMatrix();
glRotatef(90,0,1,0);
glColor3f(0.0, 0.0, 0);
glTranslatef(2.5, -3,-14);
balok(2,0.1,1.2);
```

```
glPopMatrix();
//knalpot
glPushMatrix();
glRotatef(90,0,1,0);
glColor3f(0.0, 0.0, 0);
glTranslatef(-3.7, -6,-14);
silinder(0.6,0.5,4);
glPopMatrix();

//jendela kanan
glPushMatrix();
glColor3f(0.0, 0.0, 0);
glTranslatef(2.5, 1.5,5);
balok(4.5, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glTranslatef(-2.5, 1.5,5);
balok(4.5, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glTranslatef(3.75, -1,5);
balok(7, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glTranslatef(-3.75, -1,5);
balok(7, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glRotatef(90, 0, 0, 15);
glTranslatef(0.5, -0.5,5);
balok(2.5, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glRotatef(90, 0, 0, 15);
glTranslatef(0.5, 0.5,5);
balok(2.5, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glRotatef(45, 0, 0, 1);
glTranslatef(-4, 4.35, 5);
balok(3.5, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glRotatef(-40, 0, 0, 1);
glTranslatef(4.4, 4, 5);
balok(4, 0.5, 0.5);
glPopMatrix();
//jendela kiri
glPushMatrix();
glTranslatef(2.5, 1.5,-5);
balok(4.5, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glTranslatef(-2.5, 1.5,-5);
balok(4.5, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glTranslatef(3.75, -1,-5);
balok(7, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
```

```
glTranslatef(-3.75, -1,-5);
balok(7, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glRotatef(90, 0, 0, 15);
glTranslatef(0.5, -0.5,-5);
balok(2.5, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glRotatef(90, 0, 0, 15);
glTranslatef(0.5, 0.5,-5);
balok(2.5, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glRotatef(45, 0, 0, 1);
glTranslatef(-4, 4.35, -5);
balok(3.5, 0.5, 0.5);
glPopMatrix();
glPushMatrix();
glRotatef(-40, 0, 0, 1);
glTranslatef(4.5, 4, -5);
balok(4, 0.5, 0.5);
glPopMatrix();
glPopMatrix();
glPopMatrix();
}

//membuat proyeksi bayangan

void glShadowProjection(float * l, float * e, float * n)
{
 float d, c;
 float mat[16];
 d = n[0]*l[0] + n[1]*l[1] + n[2]*l[2];
 c = e[0]*n[0] + e[1]*n[1] + e[2]*n[2] - d;
 mat[0] = l[0]*n[0]+c; // membuat matrik. OpenGL menggunakan kolom matrik
 mat[4] = n[1]*l[0];
 mat[8] = n[2]*l[0];
 mat[12] = -l[0]*c-l[0]*d;
 mat[1] = n[0]*l[1];
 mat[5] = l[1]*n[1]+c;
 mat[9] = n[2]*l[1];
 mat[13] = -l[1]*c-l[1]*d;
 mat[2] = n[0]*l[2];
 mat[6] = n[1]*l[2];
 mat[10] = l[2]*n[2]+c;
 mat[14] = -l[2]*c-l[2]*d;
 mat[3] = n[0];
 mat[7] = n[1];
 mat[11] = n[2];
 mat[15] = -d;
 glMultMatrixf(mat); // kalikan matrik
}

void render()
{
 glClearColor(0.0,0.6,0.9,0.0);
 glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
 glLightfv(GL_LIGHT0, GL_POSITION, l);
 glDisable(GL_CULL_FACE);
 glDisable(GL_LIGHTING);
```

```
 glColor3f(1.0,1.0,0.0);
 glBegin(GL_POINTS);
 glVertex3f(l[0],l[1],l[2]);
 glEnd();
 glColor3f(0.8,0.8,0.8);
 glBegin(GL_QUADS);
 glNormal3f(0.0,1.0,0.0);
 glVertex3f(-1300.0,e[1]-0.1, 1300.0);
 glVertex3f( 1300.0,e[1]-0.1, 1300.0);
 glVertex3f( 1300.0,e[1]-0.1,-1300.0);
 glVertex3f(-1300.0,e[1]-0.1,-1300.0);
 glEnd();
 // gambar bayangan
 glPushMatrix();
 glRotatef(ry,0,1,0);
 glRotatef(rx,1,0,0);
 glEnable(GL_LIGHTING);
 glColor3f(0.0,0.0,0.8);
 draw();
 glPopMatrix();
//sekarang gambar bayangan yang muncul
 glPushMatrix();
 glShadowProjection(l,e,n);
 glRotatef(ry,0,1,0);
 glRotatef(rx,1,0,0);
 glDisable(GL_LIGHTING);
 glColor3f(0.4,0.4,0.4);
 draw();
 glPopMatrix();
 glutSwapBuffers();
}

void keypress(unsigned char c, int a, int b)
{
 if ( c==27 ) exit(0);
 else if ( c=='s' ) l[1]-=5.0;
 else if ( c=='w' ) l[1]+=5.0;
 else if ( c=='a' ) l[0]-=5.0;
 else if ( c=='d' ) l[0]+=5.0;
 else if ( c=='q' ) l[2]-=5.0;
 else if ( c=='e' ) l[2]+=5.0;
 else if ( c=='h' ) help();
}

void help()
{
 printf("proyeksi contoh bayangan sebuah obyek teapot\n");
}

void idle()
{
 rx+=1;
 ry+=1;
 render();
}

void resize(int w, int h)
{
 glViewport(0, 0, w, h);
}
```

```
int main(int argc, char * argv[])
{
 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
 glutInitWindowPosition(300,30);
 glutCreateWindow("proyeksi bayangan");
 glutReshapeFunc(resize);
 glutReshapeWindow(400,400);
 glutKeyboardFunc(keypress);
 glutDisplayFunc(render);
 glutIdleFunc(idle);
 glEnable(GL_NORMALIZE);
 glEnable(GL_LIGHTING);
 glEnable(GL_COLOR_MATERIAL);
 glEnable(GL_DEPTH_TEST);
 glEnable(GL_LIGHT0);
 glEnable(GL_TEXTURE_2D);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 gluPerspective(60.0f, 1.0, 1.0, 400.0);
 // Reset koordinat sebelum dimodifikasi/diubah
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
 glTranslatef(0.0, 0.0, -150.0);
 glutMainLoop();
 return 0;
}
```

2. Analisa cara kerja proses pembuatan bayangan.

```
void glShadowProjection(float * l, float * e, float * n)
{
 float d, c;
 float mat[16];
 d = n[0]*l[0] + n[1]*l[1] + n[2]*l[2];
 c = e[0]*n[0] + e[1]*n[1] + e[2]*n[2] - d;
 mat[0] = l[0]*n[0]+c; // membuat matrik. OpenGL menggunakan kolom matrik
 mat[4] = n[1]*l[0];
 mat[8] = n[2]*l[0];
 mat[12] = -l[0]*c-l[0]*d;
 mat[1] = n[0]*l[1];
 mat[5] = l[1]*n[1]+c;
 mat[9] = n[2]*l[1];
 mat[13] = -l[1]*c-l[1]*d;
 mat[2] = n[0]*l[2];
 mat[6] = n[1]*l[2];
 mat[10] = l[2]*n[2]+c;
 mat[14] = -l[2]*c-l[2]*d;
 mat[3] = n[0];
 mat[7] = n[1];
 mat[11] = n[2];
 mat[15] = -d;
 glMultMatrixf(mat); // kalikan matrik
}

void render()
{
 glClearColor(0.0,0.6,0.9,0.0);
 glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
 glLightfv(GL_LIGHT0, GL_POSITION, l);
```

```
 glDisable(GL_CULL_FACE);
 glDisable(GL_LIGHTING);
 glColor3f(1.0,1.0,0.0);
 glBegin(GL_POINTS);
 glVertex3f(l[0],l[1],l[2]);
 glEnd();
 glColor3f(0.8,0.8,0.8);
 glBegin(GL_QUADS);
 glNormal3f(0.0,1.0,0.0);
 glVertex3f(-1300.0,e[1]-0.1, 1300.0);
 glVertex3f( 1300.0,e[1]-0.1, 1300.0);
 glVertex3f( 1300.0,e[1]-0.1,-1300.0);
 glVertex3f(-1300.0,e[1]-0.1,-1300.0);
 glEnd();
 // gambar bayangan
 glPushMatrix();
 glRotatef(ry,0,1,0);
 glRotatef(rx,1,0,0);
 glEnable(GL_LIGHTING);
 glColor3f(0.0,0.0,0.8);
 draw();
 glPopMatrix();
//sekarang gambar bayangan yang muncul
 glPushMatrix();
 glShadowProjection(l,e,n);
 glRotatef(ry,0,1,0);
 glRotatef(rx,1,0,0);
 glDisable(GL_LIGHTING);
 glColor3f(0.4,0.4,0.4);
 draw();
 glPopMatrix();
 glutSwapBuffers();
}
```

Dalam pembuatan bayangan langkah awalnya yaitu menentukan koordinat sumber cahaya yang mengenai benda yaitu :

l = sumber cahaya

n= normal vector

e= dasar untuk letak bayangan objek

Untuk membuat proyeksi digunakan procedure `void glShadowProjection(float * l, float * e, float * n)`, prosedur ini berisi perkalian matriks – matriks dengan variabel yang merupakan hasil perkalian koordinat sumber cahaya dan ModelView-Matriks. Object digambarkan setelah pemanggilan prosedur ini akan diproyeksikan ke bawah.

Bagian prosedur render digunakan untuk mengatur seluruh adegan dari sudut sumber cahaya dan meletakkannya di z-penyangga. Langkah pada render ini menggunakan 2 buah objek yang sama, objek pertama yaitu objek yang tidak terkena proyeksi bayangan (objek

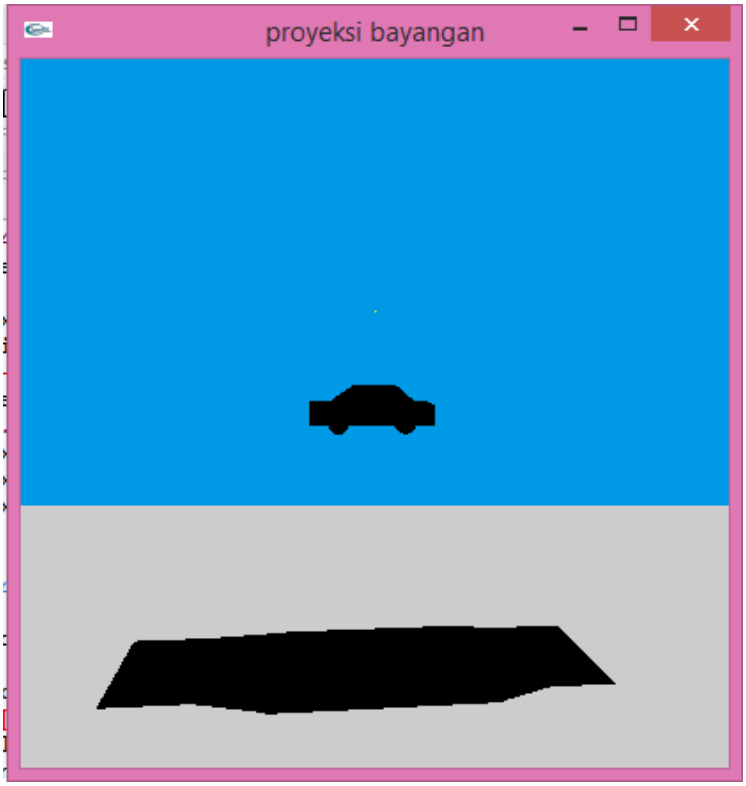
aslinya), objek kedua yaitu objek duplikasi dari objek pertama yang dikenai proyeksi bayangan sehingga nampak sebagai bayangan bendanya, kemudian bayangan akan muncul dengan melakukan shadowProjection pada benda aslinya.

3. Fungsi dari prosedur glShadowProection digunakan untuk membuat proyeksi dari benda/ objek membentuk sebuah bayangan. Fungsi ini menerima 3 nilai masukan yang bertipe float/ real, nilai masukan ini adalah letak koordinat sumber cahaya.

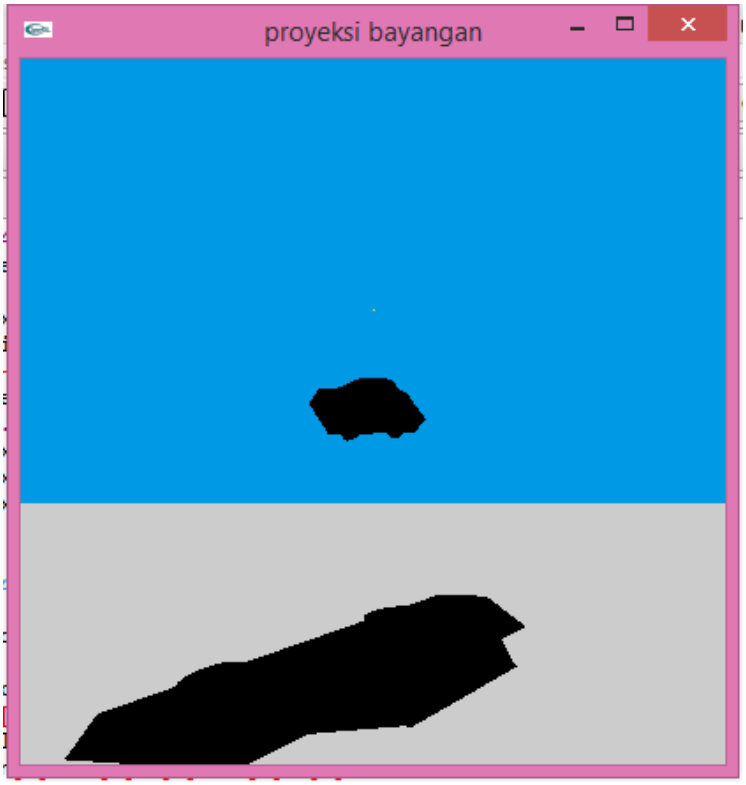
   Dalam prosesnya, prosedure ini berjalan dengan mengalikan ketiga variabel inputan dengan nilai pada matriks. Kemudian glShadowProjection akan menghasilkan sebuah matriks untuk proyeksi cahaya sebagai bayangan.

   Terdapat 2 variabel yang diinisialisasi yaitu variabel d dan c, digunakan untuk mencari dot Products antara vector sumber cahaya (I) dengan vector normal (n), dan vector latar/ ground (e). Kemudian membuat matriks proyeksi, untuk isi setiap kolom 0-13, dihitung menggunakan semua rumus tersebut, lalu matriks dikalikan. Matriks yang dihasilkan membentuk proyeksi cahaya yaitu proyeksi untuk bangun/ benda real membentuk bayangannya.

2. Screenshot



Gambar 2. Tampak samping mobil serta bayangannya



Gambar 3. Tampak mobil serta bayangannya