**TUGAS BESAR GRAFIKA KOMPUTER**

**KANDANG HAMSTER 3D**

Diajukan untuk memenuhi

Tugas Mata Kuliah Grafika Komputer

Program Strata Satu Jurusan Teknik Informatika

Fakultas Teknik dan Ilmu Komputer

Universitas Komputer Indonesia

Dosen : Hendri Karisma, S. Kom

**KELOMPOK 8**

**OLEH :**

**ASEP ROJALI 10109411**

**ANGGI SOFYAN 10109408**

**EKA WIBAWA 10109407**



**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK DAN ILMU KOMPUTER**

**UNIVERSITAS KOMPUTER INDONESIA**

**BANDUNG**

**2013**

1. **Latar Belakang**

Kandang *hamster* saat ini semakin bervariasi dari mulai bahan, aksesoris dan ukuran, seperti bahan kawat atau logam, bahan kaca (aquarium), plastik dan bahan kayu dengan ukuran secara umum 75 x 50 x 25 cm untuk kandang besar, 38 x 25 x 25 untuk kandang kecil dan 40 x 20 x 12 cm untuk kandang sedang. Di kandang *hamster* tersebut terdapat berbagai macam aksesoris mainan seperti *jogging wheel,* bola , terowongan, ranjang tidur, pot tanaman, tangga, rumah dan sebagainya.

Untuk memenuhi salah satu tugas mata kuliah komputer grafik, kami akan mengimplementasikan kandang hamster beserta aksesorisnya ke dalam bentuk 3D, dimana objek yang akan di tuangkan ke dalam bentuk 3D adalah pot tanaman, tangga, tempat minum , tempat makan, ranjang tidur, jogging wheel, dan serbuk kayu.

Sehingga pada tugas ini kami beri judul “KANDANG HAMSTER 3D ”.

1. **Rumusan Masalah**

Rumusan masalah pada pembangunan kandang *hamster* 3D adalah sebagai berikut :

1. Bagaimana membuat kandang hamster 3D?
2. Bagaimana mengimplementasikan bahan ajar mata kuliah Komputer Grafika dalam membuat bangun ruang dengan menggunakan OpenGL?
3. **Batasan Masalah**

Agar pembahasan masalah tidak menyimpang dari pokok bahasan, maka batasan masalah dalam pembuatan kandang hamster 3D adalah :

1. Hanya menampilkan pohon, tangga, tempat minum, tempat tidur, tempat makan, *jogging wheel ,* rumah.
2. Objek ditampilkan ke dalam 3D.
3. **List Of Object**
4. Pohon `
5. Tangga
6. Tempat minum
7. Tempat tidur
8. Mainan berputar hamster (*jogging wheel*)
9. Rumah
10. Tempat makan
11. **List Object Pembangun :**

a.Pot tanaman yang dibuat dari objek lingkaran solid yang ditinggikan pada kordinat z- nya, tanaman diatas pot dibuat dengan mengabungkan kubus dengan kerucut.

b.Tangga di buat dengan kubus yang ditransformasikan sedimikian rupa sehingga terlihat seperti tangga.

c. Tempat minum dibuat dari gabungan dua lingkaran.

d. tempat tidur dibuat dari gabungan kubus-kubus.

e. Mainan berputar hamster dibuat dengan gabungan lingkaran

f. Rumah dibuat dari gabungan kubus, segitiga.

g. Tempat makan dibuat dari gabungan kubus-kubus.

**BAB II**

## 2.1 Job desk

## 

-Asep rojali

Membuat pohon,rumah,terain

-Anggi Sofyan

Membuat balok,tangga,juggling wheels

-Eka wibawa

Membuat galon,tempat air

**BAB III**

* 1. **Source Program**
* main.cpp

//\*

\* main.cpp

\*

\* Created on: Jun 27, 2013

\* Author: Asep Rojali,Anggi Sofyan,Eka Wibawa

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#ifdef \_\_APPLE\_\_

#include <OpenGL/OpenGL.h>

#include <GLUT/glut.h>

#else

#include <GL/glut.h>

#include <GL/glu.h>

#include <GL/gl.h>

#include "imageloader.h"

#include "vec3f.h"

#endif

static GLfloat spin = 0.0;

float angle = 0;

using namespace std;

float lastx, lasty;

GLint stencilBits;

static int viewx = 50;

static int viewy = 24;

static int viewz = 80;

float rot = 0;

//train 2D

//class untuk terain 2D

class Terrain {

private:

int w; //Width

int l; //Length

float\*\* hs; //Heights

Vec3f\*\* normals;

bool computedNormals; //Whether normals is up-to-date

public:

Terrain(int w2, int l2) {

w = w2;

l = l2;

hs = new float\*[l];

for (int i = 0; i < l; i++) {

hs[i] = new float[w];

}

normals = new Vec3f\*[l];

for (int i = 0; i < l; i++) {

normals[i] = new Vec3f[w];

}

computedNormals = false;

}

~Terrain() {

for (int i = 0; i < l; i++) {

delete[] hs[i];

}

delete[] hs;

for (int i = 0; i < l; i++) {

delete[] normals[i];

}

delete[] normals;

}

int width() {

return w;

}

int length() {

return l;

}

//Sets the height at (x, z) to y

void setHeight(int x, int z, float y) {

hs[z][x] = y;

computedNormals = false;

}

//Returns the height at (x, z)

float getHeight(int x, int z) {

return hs[z][x];

}

//Computes the normals, if they haven't been computed yet

void computeNormals() {

if (computedNormals) {

return;

}

//Compute the rough version of the normals

Vec3f\*\* normals2 = new Vec3f\*[l];

for (int i = 0; i < l; i++) {

normals2[i] = new Vec3f[w];

}

for (int z = 0; z < l; z++) {

for (int x = 0; x < w; x++) {

Vec3f sum(0.0f, 0.0f, 0.0f);

Vec3f out;

if (z > 0) {

out = Vec3f(0.0f, hs[z - 1][x] - hs[z][x], -1.0f);

}

Vec3f in;

if (z < l - 1) {

in = Vec3f(0.0f, hs[z + 1][x] - hs[z][x], 1.0f);

}

Vec3f left;

if (x > 0) {

left = Vec3f(-1.0f, hs[z][x - 1] - hs[z][x], 0.0f);

}

Vec3f right;

if (x < w - 1) {

right = Vec3f(1.0f, hs[z][x + 1] - hs[z][x], 0.0f);

}

if (x > 0 && z > 0) {

sum += out.cross(left).normalize();

}

if (x > 0 && z < l - 1) {

sum += left.cross(in).normalize();

}

if (x < w - 1 && z < l - 1) {

sum += in.cross(right).normalize();

}

if (x < w - 1 && z > 0) {

sum += right.cross(out).normalize();

}

normals2[z][x] = sum;

}

}

//Smooth out the normals

const float FALLOUT\_RATIO = 0.5f;

for (int z = 0; z < l; z++) {

for (int x = 0; x < w; x++) {

Vec3f sum = normals2[z][x];

if (x > 0) {

sum += normals2[z][x - 1] \* FALLOUT\_RATIO;

}

if (x < w - 1) {

sum += normals2[z][x + 1] \* FALLOUT\_RATIO;

}

if (z > 0) {

sum += normals2[z - 1][x] \* FALLOUT\_RATIO;

}

if (z < l - 1) {

sum += normals2[z + 1][x] \* FALLOUT\_RATIO;

}

if (sum.magnitude() == 0) {

sum = Vec3f(0.0f, 1.0f, 0.0f);

}

normals[z][x] = sum;

}

}

for (int i = 0; i < l; i++) {

delete[] normals2[i];

}

delete[] normals2;

computedNormals = true;

}

//Returns the normal at (x, z)

Vec3f getNormal(int x, int z) {

if (!computedNormals) {

computeNormals();

}

return normals[z][x];

}

};

//end class

void initRendering() {

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_NORMALIZE);

glShadeModel(GL\_SMOOTH);

}

//Loads a terrain from a heightmap. The heights of the terrain range from

//-height / 2 to height / 2.

//load terain di procedure inisialisasi

Terrain\* loadTerrain(const char\* filename, float height) {

Image\* image = loadBMP(filename);

Terrain\* t = new Terrain(image->width, image->height);

for (int y = 0; y < image->height; y++) {

for (int x = 0; x < image->width; x++) {

unsigned char color = (unsigned char) image->pixels[3

\* (y \* image->width + x)];

float h = height \* ((color / 255.0f) - 0.5f);

t->setHeight(x, y, h);

}

}

delete image;

t->computeNormals();

return t;

}

float \_angle = 60.0f;

//buat tipe data terain

Terrain\* \_terrain;

Terrain\* \_terrainTanah;

Terrain\* \_terrainAir;

const GLfloat light\_ambient[] = { 0.3f, 0.3f, 0.3f, 1.0f };

const GLfloat light\_diffuse[] = { 0.7f, 0.7f, 0.7f, 1.0f };

const GLfloat light\_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };

const GLfloat light\_position[] = { 1.0f, 1.0f, 1.0f, 1.0f };

const GLfloat light\_ambient2[] = { 0.3f, 0.3f, 0.3f, 0.0f };

const GLfloat light\_diffuse2[] = { 0.3f, 0.3f, 0.3f, 0.0f };

const GLfloat mat\_ambient[] = { 0.8f, 0.8f, 0.8f, 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 cleanup() {

delete \_terrain;

delete \_terrainTanah;

}

//untuk di display

void drawSceneTanah(Terrain \*terrain, GLfloat r, GLfloat g, GLfloat b) {

// glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

/\*

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(0.0f, 0.0f, -10.0f);

glRotatef(30.0f, 1.0f, 0.0f, 0.0f);

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

GLfloat ambientColor[] = {0.4f, 0.4f, 0.4f, 1.0f};

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambientColor);

GLfloat lightColor0[] = {0.6f, 0.6f, 0.6f, 1.0f};

GLfloat lightPos0[] = {-0.5f, 0.8f, 0.1f, 0.0f};

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, lightColor0);

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPos0);

\*/

float scale = 200.0f / max(terrain->width() - 1, terrain->length() - 1);

glScalef(scale, scale, scale);

glTranslatef(-(float) (terrain->width() - 1) / 2, 0.0f,

-(float) (terrain->length() - 1) / 2);

glColor3f(r, g, b);

for (int z = 0; z < terrain->length() - 1; z++) {

//Makes OpenGL draw a triangle at every three consecutive vertices

glBegin(GL\_TRIANGLE\_STRIP);

for (int x = 0; x < terrain->width(); x++) {

Vec3f normal = terrain->getNormal(x, z);

glNormal3f(normal[0], normal[1], normal[2]);

glVertex3f(x, terrain->getHeight(x, z), z);

normal = terrain->getNormal(x, z + 1);

glNormal3f(normal[0], normal[1], normal[2]);

glVertex3f(x, terrain->getHeight(x, z + 1), z + 1);

}

glEnd();

}

}

//Segitiga

void segitiga() {

glBegin(GL\_QUADS);

glVertex3f(-2, -2, 2);

glVertex3f(-2, -2, -2);

glVertex3f(-2, 2, -2);

glVertex3f(-2, 2, 2);

glEnd();

//Sisi-sisi Prisma

glBegin(GL\_TRIANGLES);

glColor3d(1.0f, 1.0f, 1.0f);

//Segitiga Warna Merah

glVertex3f(-2, -2, 2);

glVertex3f(8, 0, 0);

glVertex3f(-2, 2, 2);

//Segitiga Warna Hijau

glVertex3f(-2, 2, 2);

glVertex3f(8, 0, 0);

glVertex3f(-2, 2, -2);

//Segitiga Warna Biru

glVertex3f(-2, 2, -2);

glVertex3f(8, 0, 0);

glVertex3f(-2, -2, -2);

//Segitiga Warna Putih

glVertex3f(-2, -2, -2);

glVertex3f(8, 0, 0);

glVertex3f(-2, -2, 2);

glEnd();

}

//cylinder dibuat beberapa objek dari solid cone sama solidtorus (payung)

void cylinder(float alas, float atas, 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 - atas)) / tinggi), 16, 16);

}

glTranslatef(0.0, 0.0, alas / 4);

glutSolidCone(atas, 0, 20, 1);

glPopMatrix();

}

//Galon

void galon() {

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

//warna biru laut

glColor3d(0.803921568627451, 0.5215686274509804, 0.2470588235294118);

//wadah atas

glTranslated(0.0, 12, 0.0);

glutSolidCube(20);

glPopMatrix();

//galon

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.1, 1.0, 1.0);

glTranslated(0. - 1, 25.0, 0.0);

glRotated(-90.0, 1.0, 0.0, 0.0);

cylinder(10.0, 10.0, 15.0);

glPopMatrix();

//corong

glPushMatrix();

glTranslated(5.0, 18.0, 0.0);

glRotated(90.0, 0.0, 1.0, 0.0);

glRotated(40.0, 1.0, 0.0, 0.0);

cylinder(1.0, 1.0, 15.0);

glPopMatrix();

}

//pohon dibuat dari cylinder

void pohon() {

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.0, 1.0, 0.0);

glRotated(-90.0, 1.0, 0.0, 0.0);

glutSolidCone(15.0, 15.0, 15, 10);

glPopMatrix();

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.803921568627451, 0.5215686274509804, 0.2470588235294118);

glRotated(-90.0, 1.0, 0.0, 0.0);

glTranslated(-1.0, 0.0, -20.0);

cylinder(6.0, 3.0, 20.0);

glPopMatrix();

}

void rumahMewah() {

//atap

glPushMatrix();

glTranslated(0.0, 50.0, 16.0);

glRotated(90.0, 0.0, 0.0, 1.0);

glScaled(2.0, 15.0, 15.0);

segitiga();

glPopMatrix();

//rumah bawah atas

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.903921568627451, 0.5215686274509804, 0.2470588235294118);

glTranslated(0.0, 40.0, 17.5);

glutSolidCube(30);

glPopMatrix();

//rumah bawah

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.803921568627451, 0.5215686274509804, 0.2470588235294118);

glTranslated(0.0, 20.0, 0.0); //cube

//glScaled(1.5, 1.5, 1.5);

glutSolidCube(30);

glTranslated(0.0, 0.0, 30.0);

glutSolidCube(30);

glPopMatrix();

}

void muterMuteran() {

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.903921568627451, 0.5215686274509804, 0.2470588235294118);

glTranslated(00.0, 20.0, -1.0);

glRotated(90, 1.0, 0.0, 0.0);

glRotated(30.0, 1.0, 0.0, 0.0);

cylinder(3.0, 3.0, 20.0);

glPopMatrix();

//kaki

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.903921568627451, 0.5215686274509804, 0.2470588235294118);

glTranslated(00.0, 20.0, 1.0);

glRotated(90, 1.0, 0.0, 0.0);

glRotated(-30.0, 1.0, 0.0, 0.0);

cylinder(3.0, 3.0, 20.0);

glPopMatrix();

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.0, 0.0, 0.0);

glutSolidCube(10);

glTranslated(0.0, 0.0, 10.0);

glutSolidCube(10);

glTranslated(0.0, 0.0, -20.0);

glutSolidCube(10);

glPopMatrix();

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.1, 1.0, 1.0);

glTranslated(0.0, 40.0, 0.0);

glRotated(-90, 0.0, 1.0, 0.0);

cylinder(20.0, 20.0, 5.0);

glPopMatrix();

}

void hiasanTengah() {

}

void atapRumah() {

glPushMatrix();

glRotated(90, 0.0, 0.0, 1.0);

glScaled(5.0, 25.0, 25.0);

segitiga();

glPopMatrix();

}

unsigned int LoadTextureFromBmpFile(char \*filename);

void display(void) {

glClearStencil(0); //clear the stencil buffer

glClearDepth(1.0f);

glClearColor(0.0, 0.6, 0.8, 1);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT | GL\_STENCIL\_BUFFER\_BIT); //clear the buffers

glLoadIdentity();

gluLookAt(viewx, viewy, viewz, 0.0, 0.0, 5.0, 0.0, 1.0, 0.0);

//Tempat Minum galon

glPushMatrix();

glTranslated(20.0, 0.0, -85.0);

glRotated(-90.0, 0.0, 1.0, 0.0);

glTranslated(120.0, 0.0, -65.0);

glRotated(180.0, 0.0, 1.0, 0.0);

glRotated(90.0, 0.0, 1.0, 0.0);

galon();

glPopMatrix();

//Rumah

glPushMatrix();

glTranslated(75.0, -3.0, -25.0);

glRotated(220.0, 0.0, 1.0, 0.0);

rumahMewah();

glPopMatrix();

//pintu rumah 1

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.0, 0.0, 0.0);

glRotated(318.0, 0.0, 1.0, 0.0);

glTranslated(18.0, 20.0, -60.0);

cylinder(10.0, 10.0, 5.0);

glPopMatrix();

//pintu rumah 2

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.1, 1.0, 1.0);

glRotated(-225.0, 0.0, 1.0, 0.0);

glTranslated(-18.0, 20.0, 57.0);

cylinder(14.0, 14.0, 5.0);

glPopMatrix();

//Hiasan Tengah

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.803921568627451, 0.5215686274509804, 0.2470588235294118);

glRotated(-80.0, 0.0, 0.0, 1.0);

glRotated(-35.0, 1.0, 0.0, 0.0);

glRotated(13.0, 0.0, 0.0, 1.0);

glTranslated(-15.0, -10.0, -3.0);

cylinder(4.0, 4.0, 20.0);

for (int t = 0; t < 4; t++) {

glTranslated(0.0, 8.0, 0.0);

cylinder(4.0, 4.0, 20.0);

}

for (int t = 0; t < 3; t++) {

glTranslated(8, 8.0, 0);

cylinder(4.0, 4.0, 20.0);

}

glPopMatrix();

//Pagar di tengah

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.803921568627451, 0.5215686274509804, 0.2470588235294118);

glRotated(-90.0, 1.0, 0.0, 0.0);

glTranslated(90.0, -65.0, 5.0);

cylinder(5.0, 5.0, 10.0);

glTranslated(-15.0, 0.0, 0.0);

cylinder(5.0, 5.0, 10.0);

glTranslated(-15.0, 0.0, 0.0);

cylinder(5.0, 5.0, 10.0);

glTranslated(-15.0, 5.0, 0.0);

cylinder(5.0, 5.0, 10.0);

glTranslated(-15.0, 5.0, 0.0);

cylinder(5.0, 5.0, 10.0);

glPopMatrix();

//pohon

glPushMatrix();

glTranslated(-80.0, 25.0, 60.0);

pohon();

glTranslated(30.0, 0.0, 0.0);

pohon();

glTranslated(30.0, 0.0, 0.0);

pohon();

glTranslated(30.0, 0.0, 0.0);

pohon();

glPopMatrix();

glPushMatrix();

glTranslated(-90.0, 25.0, -60.0);

pohon();

glTranslated(30.0, 0.0, 0.0);

pohon();

glTranslated(30.0, 0.0, 0.0);

pohon();

glPopMatrix();

//Muter"an

glPushMatrix();

glTranslated(-85.0, 5.0, 20.0);

muterMuteran();

glPopMatrix();

//Pagar ditengah lurus

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.803921568627451, 0.5215686274509804, 0.2470588235294118);

glRotated(-90.0, 1.0, 0.0, 0.0);

glTranslated(10.0, 65.0, 5.0);

cylinder(6.0, 6.0, 15.0);

for (int x = 0; x < 2; x++) {

glTranslated(0.0, -15.0, 0.0);

cylinder(6.0, 6.0, 15.0);

}

glPopMatrix();

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.803921568627451, 0.5215686274509804, 0.2470588235294118);

glRotated(-90.0, 1.0, 0.0, 0.0);

glTranslated(-90.0, 40.0, 5.0);

cylinder(6.0, 6.0, 15.0);

for (int x = 0; x < 2; x++) {

glTranslated(0.0, -15.0, 0.0);

cylinder(6.0, 6.0, 15.0);

}

glPopMatrix();

//Pagar di pinggir

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.803921568627451, 0.5215686274509804, 0.2470588235294118);

glRotated(-90.0, 1.0, 0.0, 0.0);

glTranslated(90.0, -10.0, 5.0);

cylinder(5.0, 5.0, 10.0);

for (int x = 0; x < 2; x++) {

glTranslated(-15.0, 0.0, 0.0);

cylinder(5.0, 5.0, 10.0);

}

glPopMatrix();

//Tempat Minum

glPushMatrix();

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

glColor3d(0.1, 1.0, 1.0);

glTranslated(-60.0, 10.0, -30.0);

glRotated(-270.0, 1.0, 0.0, 0.0);

cylinder(15.0, 10.0, 3.0);

glPopMatrix();

//tiang 1

// glPushMatrix();

// glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE);

// glColor3d(0.0, 0.0, 0.0);

// glRotated(-90, 1.0, 0.0, 0.0);

// glTranslated(90.0, 100.0, 0.0);

// cylinder(3.0, 3.0, 100);

// glPopMatrix();

//

// //tiang 2

// glPushMatrix();

// glRotated(-90, 1.0, 0.0, 0.0);

// glTranslated(90.0, -100.0, 0.0);

// cylinder(3.0, 3.0, 100);

// glPopMatrix();

//

// //tiang 3

// glPushMatrix();

// glRotated(-90, 1.0, 0.0, 0.0);

// glTranslated(-100.0, 100.0, 0.0);

// cylinder(3.0, 3.0, 100);

// glPopMatrix();

//

// //tiang 4

// glPushMatrix();

// glRotated(-90, 1.0, 0.0, 0.0);

// glTranslated(-100.0, -100.0, 0.0);

// cylinder(3.0, 3.0, 100);

// glPopMatrix();

glPushMatrix();

//glBindTexture(GL\_TEXTURE\_3D, texture[0]);

drawSceneTanah(\_terrain, 0.3f, 0.9f, 0.0f);

glPopMatrix();

// glPushMatrix();

// //glBindTexture(GL\_TEXTURE\_3D, texture[0]);

// drawSceneTanah(\_terrainTanah, 0.7f, 0.2f, 0.1f);

// glPopMatrix();

//

// glPushMatrix();

// //glBindTexture(GL\_TEXTURE\_3D, texture[0]);

// drawSceneTanah(\_terrainAir, 0.0f, 0.2f, 0.5f);

// glPopMatrix();

glutSwapBuffers();

glFlush();

rot++;

angle++;

}

void init(void) {

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glDepthFunc(GL\_LESS);

glEnable(GL\_NORMALIZE);

glEnable(GL\_COLOR\_MATERIAL);

glDepthFunc(GL\_LEQUAL);

glShadeModel(GL\_SMOOTH);

glHint(GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_NICEST);

glEnable(GL\_CULL\_FACE);

\_terrain = loadTerrain("heightmap.bmp", 20);

\_terrainTanah = loadTerrain("heightmapTanah.bmp", 20);

\_terrainAir = loadTerrain("heightmapAir.bmp", 20);

//binding texture

}

static void kibor(int key, int x, int y) {

switch (key) {

case GLUT\_KEY\_HOME:

viewy++;

break;

case GLUT\_KEY\_END:

viewy--;

break;

case GLUT\_KEY\_UP:

viewz--;

break;

case GLUT\_KEY\_DOWN:

viewz++;

break;

case GLUT\_KEY\_RIGHT:

viewx++;

break;

case GLUT\_KEY\_LEFT:

viewx--;

break;

case GLUT\_KEY\_F1: {

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

}

;

break;

case GLUT\_KEY\_F2: {

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient2);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse2);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);

}

;

break;

default:

break;

}

}

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

if (key == 'd') {

spin = spin - 1;

if (spin > 360.0)

spin = spin - 360.0;

}

if (key == 'a') {

spin = spin + 1;

if (spin > 360.0)

spin = spin - 360.0;

}

if (key == 'q') {

viewz++;

}

if (key == 'e') {

viewz--;

}

if (key == 's') {

viewy--;

}

if (key == 'w') {

viewy++;

}

}

void reshape(int w, int h) {

glViewport(0, 0, (GLsizei) w, (GLsizei) h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, (GLfloat) w / (GLfloat) h, 0.1, 1000.0);

glMatrixMode(GL\_MODELVIEW);

}

int main(int argc, char \*\*argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_STENCIL | GLUT\_DEPTH); //add a stencil buffer to the window

glutInitWindowSize(800, 600);

glutInitWindowPosition(100, 100);

glutCreateWindow("Sample Terain");

init();

glutDisplayFunc(display);

glutIdleFunc(display);

glutReshapeFunc(reshape);

glutSpecialFunc(kibor);

glutKeyboardFunc(keyboard);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT, GL\_SHININESS, high\_shininess);

glColorMaterial(GL\_FRONT, GL\_DIFFUSE);

glutMainLoop();

return 0;

}