**TUGAS BESAR**

**PROGRAM ANIMASI 3D PONSEL NOKIA 700**

**DENGAN OPENGL DEVC++**

Diajukan Untuk Memenuhi

Salah Satu Tugas Mata Kuliah Komputer Grafika

Dosen : Hendri Karisma, S. Kom.



Disusun Oleh : 10108384 – Winda Tania

10108387 – Hidayatuaulia Wiris Wichaksani

10108391 – Verra Verdianti

10108399 – Yuni Retna Ningsih

10108421 – Marcelina Girsang

Kelas : IF-8/S1/VIII

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK DAN ILMU KOMPUTER**

**UNIVERSITAS KOMPUTER INDONESIA**

**2012**

**SOURCE CODE**

#include <iostream>

#include <stdlib.h>

#include <gl/glut.h>

#include <math.h>

#include "imageloader.h"

#include "vec3f.h"

using namespace std;

int w=600, h=600, z=10;

int a=0 , b=0, c=0, d=0;

int x1=0, y2=0, sudut=0, z1=0;

float skalaX=1, skalaY=1, skalaZ=1;

int cx, cy;

class Terrain {

private:

int w; //Lebar

int l; //Panjang

float\*\* hs; //Tinggi

Vec3f\*\* normals;

bool computedNormals;

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;

}

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

hs[z][x] = y;

computedNormals = false;

}

float getHeight(int x, int z) {

return hs[z][x];

}

//Perhitungan keadaan normal

void computeNormals() {

if (computedNormals) {

return;

}

//Perhitungan pembulatan keadaan normal

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

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;

}

//Mengembalikan nilai normal pada (x, z)

Vec3f getNormal(int x, int z) {

if (!computedNormals) {

computeNormals();

}

return normals[z][x];

}

};

GLuint loadTexture(Image\* image) {

GLuint textureId;

glGenTextures(1, &textureId);

glBindTexture(GL\_TEXTURE\_2D, textureId);

//Mapping gambar pada tekstur

glTexImage2D(GL\_TEXTURE\_2D,

0,

GL\_RGB,

image->width, image->height,

0,

GL\_RGB, //Karena gambar disimpan dalam format RGB

GL\_UNSIGNED\_BYTE,

image->pixels);

return textureId;

}

GLuint \_textureId;

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;

Terrain\* \_terrain;

void cleanup() {

delete \_terrain;

}

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

if (key =='i') z+=5;

else if (key == 'o') z-=5;

else if (key == 'l') {

x1=0;

y2=-1;

z1=0;

sudut+=-10;

}

else if (key == 'r') {

y2=1;

x1=0;

z1=0;

sudut+=-10;

}

}

void init(){

glShadeModel(GL\_SMOOTH);

GLfloat light\_ambient[] = { 0.0f, 0.0f, 0.0f, 0.0f };

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

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

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

glEnable(GL\_NORMALIZE);

glClearColor(0.0f,0.0f,0.0f,0.0f);

glClearDepth(1.0f);

glEnable(GL\_DEPTH\_TEST);

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

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

return;

}

void renderScene(void){

glClear (GL\_COLOR\_BUFFER\_BIT);

glFlush();

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

glClearColor(0.9, 0.9, 0.9, 0.9);//Warna background

glLoadIdentity();

glTranslatef(10,0,z-80);//Posisi awal kamera saat program dicompile

glRotatef(sudut,x1,y2,z1);

glScalef(skalaX, skalaY, skalaZ);

glPushMatrix();

float scale = 80.0f / max(\_terrain->width() - 1, \_terrain->length() - 1);

glScalef(scale, scale, scale);

glTranslatef(-(float)(\_terrain->width() - 1) / 2, 0.0f, -(float)(\_terrain->length() - 1) / 2);

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.3f, 0.4f, 1.0f);//Warna terrain

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

glBegin(GL\_TRIANGLE\_STRIP);

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

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

glNormal3f(normal[2], normal[1], normal[2]);//Tekstur terrain

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();

} glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

//Badan HP

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

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

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(0,-0.9,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(0,-0.9,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(0,-0.9,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(0,-0.9,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(0,-0.9,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(0,-0.9,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(4,0,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(0,5,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(0,3,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(-1,0,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(-3,0,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(0,3,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(0,2,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.6,0,0);

glTranslatef(4,0,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

//Bingkai Merah

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0,0,0);

glTranslatef(0.2,0,-0.5);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0,0,0);

glTranslatef(-4.4,0,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0,0,0);

glTranslatef(0,-5,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0,0,0);

glTranslatef(0,-5,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0,0,0);

glTranslatef(0,-1.9,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0,0,0);

glTranslatef(4.4,0,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0,0,0);

glTranslatef(0,3,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0,0,0);

glTranslatef(0,5,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

//Layar

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(1,1,1);

glTranslatef(-1,1.7,-0.2);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(1,1,1);

glTranslatef(-2.2,0,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(1,1,1);

glTranslatef(0,-5,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(1,1,1);

glTranslatef(0,-3,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(1,1,1);

glTranslatef(2.2,0,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(1,1,1);

glTranslatef(0,3,0);

glRotatef(360,360,360,360);

glutSolidCube(5);

glDisable(GL\_COLOR\_MATERIAL);

//Tombol

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.2,0.2,0.2);

glTranslatef(2,-6.3,-2.2);

glRotatef(360,360,360,360);

glutSolidCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.2,0.2,0.2);

glTranslatef(-1,0,0);

glRotatef(360,360,360,360);

glutSolidCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.2,0.2,0.2);

glTranslatef(-1,0,0);

glRotatef(360,360,360,360);

glutSolidCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.2,0.2,0.2);

glTranslatef(-1,0,0);

glRotatef(360,360,360,360);

glutSolidCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.2,0.2,0.2);

glTranslatef(-1,0,0);

glRotatef(360,360,360,360);

glutSolidCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.2,0.2,0.2);

glTranslatef(-1,0,0);

glRotatef(360,360,360,360);

glutSolidCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.2,0.2,0.2);

glTranslatef(-1,0,0);

glRotatef(360,360,360,360);

glutSolidCube(1);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.2,0.2,0.2);

glTranslatef(-0.3,0,0);

glRotatef(360,360,360,360);

glutSolidCube(1);

glDisable(GL\_COLOR\_MATERIAL);

//Speaker

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.1,0.1,0.1);

glTranslatef(2,15,-0.25);

glRotatef(360,360,360,360);

glutSolidCube(0.3);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.1,0.1,0.1);

glTranslatef(0.3,0,0);

glRotatef(360,360,360,360);

glutSolidCube(0.3);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.1,0.1,0.1);

glTranslatef(0.3,0,0);

glRotatef(360,360,360,360);

glutSolidCube(0.3);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.1,0.1,0.1);

glTranslatef(0.3,0,0);

glRotatef(360,360,360,360);

glutSolidCube(0.3);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.1,0.1,0.1);

glTranslatef(0.3,0,0);

glRotatef(360,360,360,360);

glutSolidCube(0.3);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.1,0.1,0.1);

glTranslatef(0.3,0,0);

glRotatef(360,360,360,360);

glutSolidCube(0.3);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.1,0.1,0.1);

glTranslatef(0.3,0,0);

glRotatef(360,360,360,360);

glutSolidCube(0.3);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.1,0.1,0.1);

glTranslatef(0.3,0,0);

glRotatef(360,360,360,360);

glutSolidCube(0.3);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0.1,0.1,0.1);

glTranslatef(0.3,0,0);

glRotatef(360,360,360,360);

glutSolidCube(0.3);

glDisable(GL\_COLOR\_MATERIAL);

//Kamera

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0,0,0);

glTranslatef(-1,-3,5);

glRotatef(360,360,360,360);

glutSolidCube(1.5);

glDisable(GL\_COLOR\_MATERIAL);

glPushMatrix();

glEnable(GL\_COLOR\_MATERIAL);

glColor3f(0,0,0);

glTranslatef(-1,0,0);

glRotatef(360,360,360,360);

glutSolidCube(1.5);

glDisable(GL\_COLOR\_MATERIAL);

glPopMatrix();

glutSwapBuffers();

}

void resize(int w1, int h1){

glViewport(0,0,w1,h1);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0,(float) w1/(float) h1, 1.0,300.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

/\*void timer(int value){

glutPostRedisplay();

glutTimerFunc(50,timer,0);

}\*/

void update(int value) {

\_angle += 1.5f;

if (\_angle > 360) {

\_angle -= 360;

}

glutPostRedisplay();

glutTimerFunc(25, update, 0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_DEPTH | GLUT\_RGBA);

glutInitWindowPosition(170,10);

glutInitWindowSize(1024,768);

glutCreateWindow("Nokia 700");

init();

glutDisplayFunc(renderScene);

\_terrain = loadTerrain("heightmap.bmp",13);

glutReshapeFunc(resize);

glutKeyboardFunc(myKeyboard);

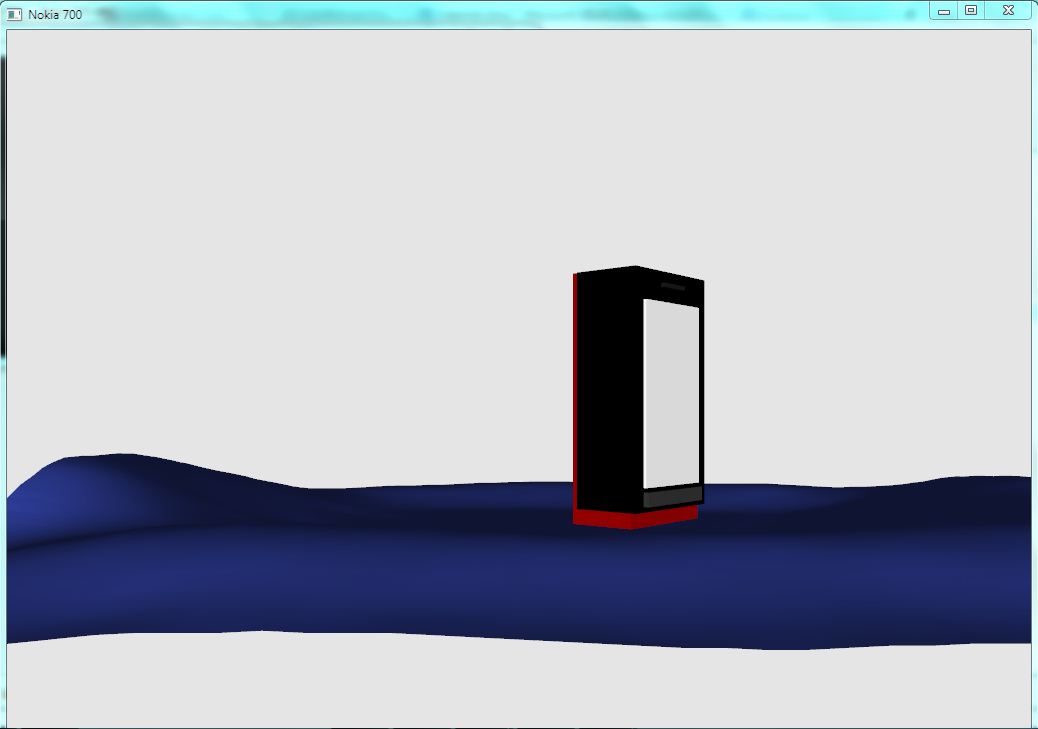
// glutTimerFunc(1,timer,0);

glutTimerFunc(25, update, 0);

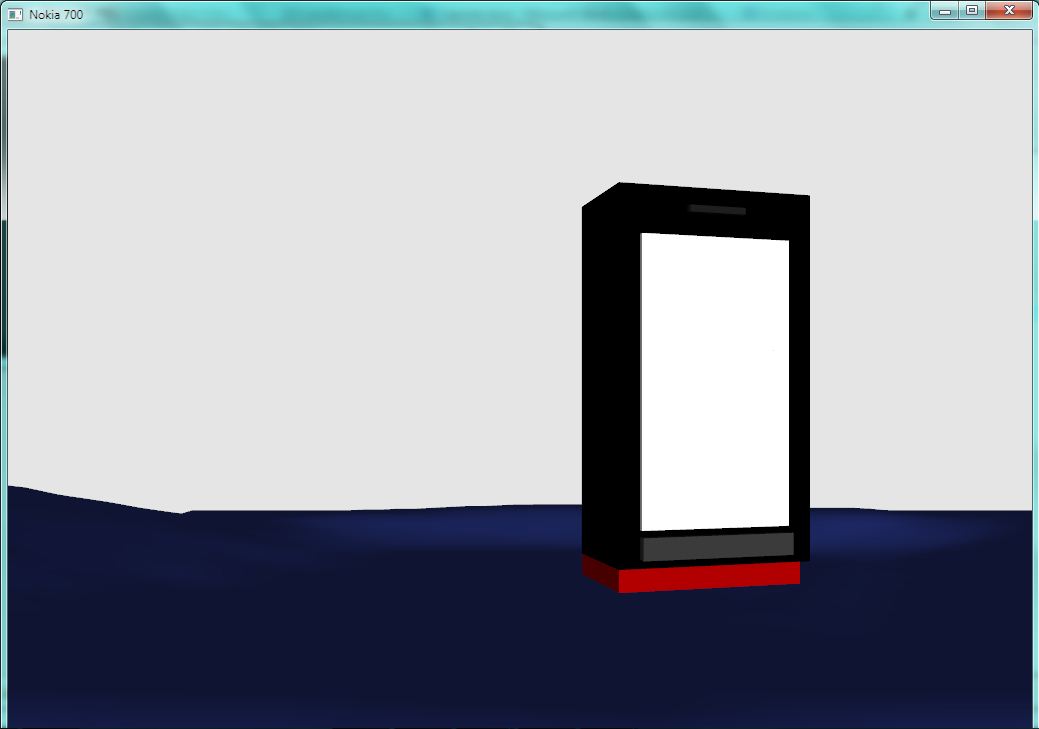
glutMainLoop();

}

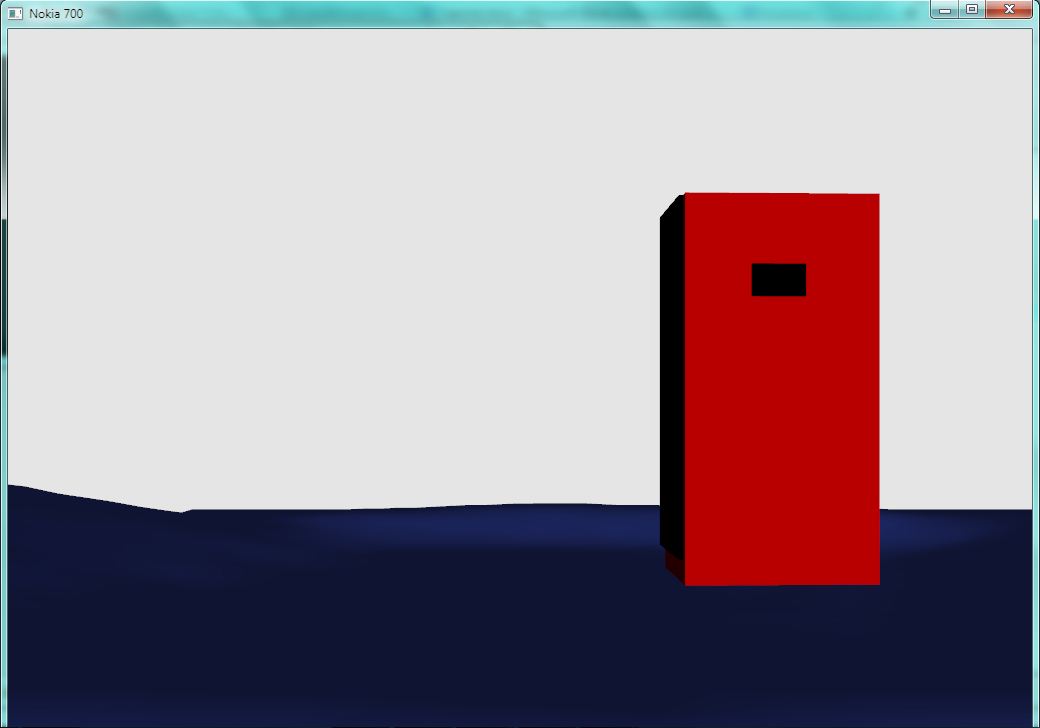
**PRINTSCREEN**

****

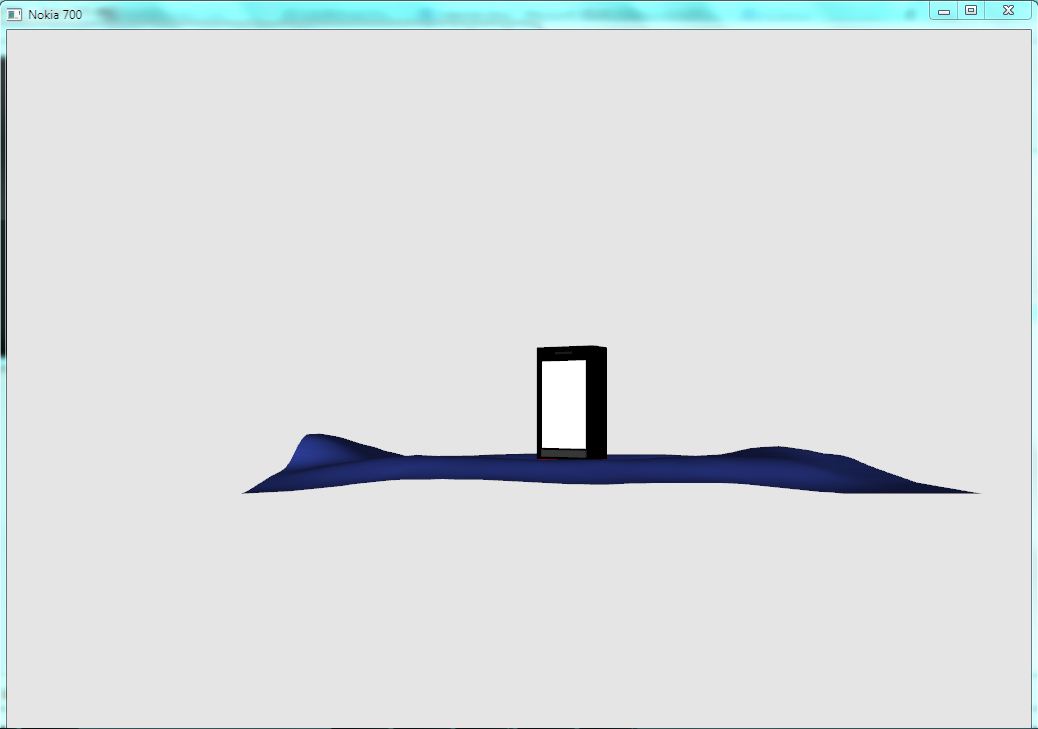
**Gambar 1 Keadaan Awal**

****

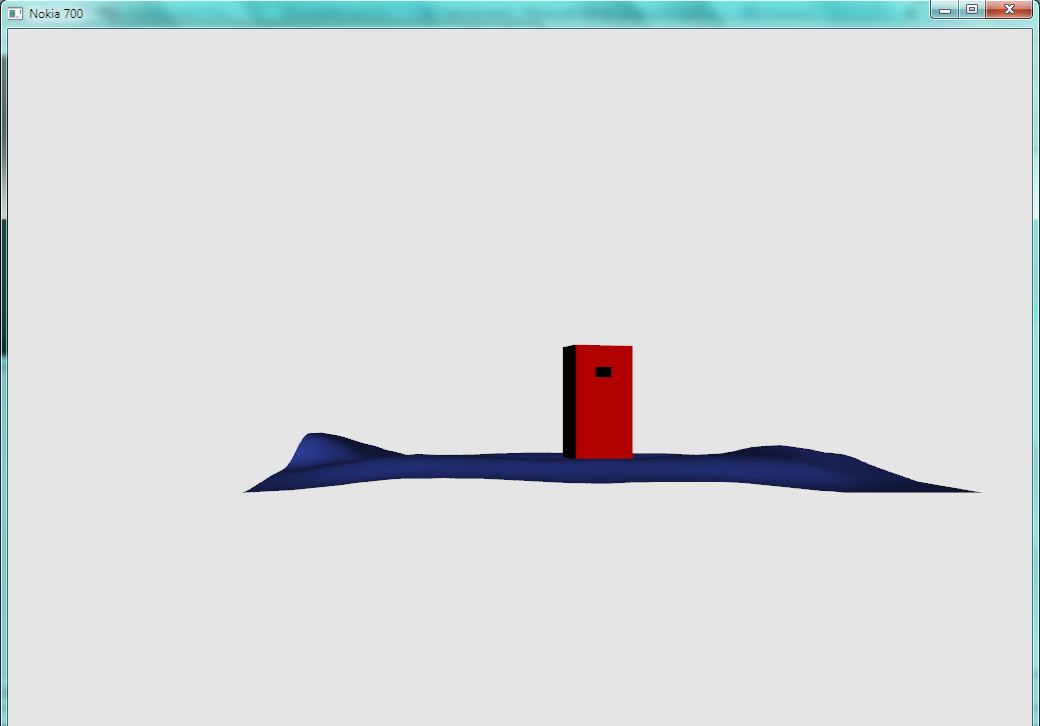
**Gambar 2 Keadaan Saat Dilakukan *Zoom In* Tampak Depan**

****

**Gambar 3 Keadaan Saat *Zoom In* Tampak Belakang**

****

**Gambar 4 Keadaan Saat *Zoom Out* Tampak Depan**

****

**Gambar 5 Keadaan Saat *Zoom Out* Tampak Belakang**