ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. А. Н. ТУПОЛЕВА – КАИКАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙКафедра: Автоматизированных систем обработки информации и управления

Институт компьютерных технологий и защиты информации

Г.М. НАБЕРЕЖНОВ, Н.Н. МАКСИМОВ

ЭЛЕМЕНТАРНОЕ ПРОГРАММИРОВАНИЕ ГРАФИКИ В OpenGL

Методическое пособие

к лабораторным работам по курсу

«Компьютерная геометрия и графика»

Студент, гр. 4210

Гауиш М.Г

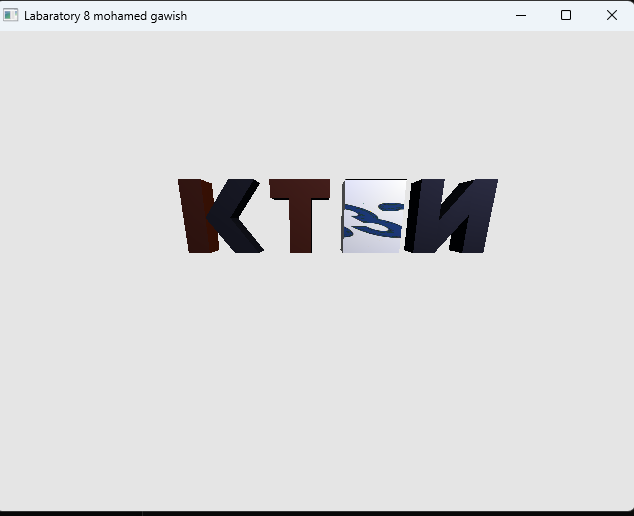
Преподаватель

Гаптуллазянова Гульшат Ильдусовна

Казань – 2023­­

Лабораторная работа №8 Логотип (Отчет)

Скриншот работы программы:



Код:­

#include <gl/glut.h>

#include <cmath>

#include <math.h>

#include <stdlib.h>

#include <C:\\Users\\Moe-gawish\\Desktop\\toool\\Лаба 5\\GLAUX.H>

#include "stb\_image.h"

#pragma comment(lib, "C:\\Users\\Moe-gawish\\Desktop\\toool\\Лаба 5\\GLAUX.lib")

#pragma comment(lib, "legacy\_stdio\_definitions.lib")

#include <Windows.h>

#include <stdlib.h>

#include <gl\glut.h>

#include <iostream>

#include <atlconv.h>

#include <string>

#include <cmath>

float thickness = 2;

float h1, h2, h3;

float angle\_light = 15.0;

float angle = 0, angle\_plus = 0;

bool check = false;

int list = 0;

void draw\_K();

void draw\_T();

void draw\_I();

void draw\_Z();

void init(void) {

float R = 640.0 / 480.0;

AUX\_RGBImageRec\* pImage = auxDIBImageLoad(L"C:/Users/Moe-gawish/source/repos/Lab 8/З.bmp");

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, pImage->sizeX, pImage->sizeY, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, pImage->data);

glEnable(GL\_TEXTURE\_2D);

glEnable(GL\_DEPTH\_TEST);

double eyeX = 1.0, eyeY = 5.0, eyeZ = 20.0;

double centerX = 1.0, centerY = 0.0, centerZ = 0.0;

double upX = 0.0, upY = 1.0, upZ = 0.0;

float n = 1.0f, f = 50.0f;

glClearColor(0.9, 0.9, 0.9, 1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(100, 1, 1, 120);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ);

list = glGenLists(1);

glNewList(list, GL\_COMPILE);

draw\_K();

draw\_T();

draw\_I();

draw\_Z();

glEndList();

}

void myDisplay()

{

float LightPos\_1[] = { 10.0, 15.0, 20.0, 1.0 };

float myDiffuse[] = { 0.15, 0.15, 0.2, 1 }; float myShininess[] = { 10.00 };

float myAmbient[] = { 0.01, 0.05, 0.11, 1 };

float mySpecular[] = { 0.33, 0.33, 0.52, 1 };

glPushMatrix(); //Сохраняем VM = 1

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); //вместе с очисткой буфера цветапредусмотреть очистку буфера глубины

glEnable(GL\_DEPTH\_TEST); //включение режима удаления невидимых (проверка глубины)

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, myDiffuse); // Источник света в CKw

glMaterialfv(GL\_FRONT, GL\_SHININESS, myShininess);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, myAmbient);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mySpecular);

glPushMatrix(); //Сохраняем VM=Fwe

glRotatef(angle\_light, 0, 1, 0);

glLightfv(GL\_LIGHT0, GL\_POSITION, LightPos\_1);

glEnable(GL\_LIGHTING); // Включение расчета освещенности

glEnable(GL\_LIGHT0); // включаем этот конкретный источник

glPopMatrix(); // Восстанавливаем VM=Fwe

glPushMatrix(); //Сохраняем VM=Fwe

glRotatef(angle, 0, 1, 0); // VM=Fwe\*R

glCallList(list);

glPopMatrix(); // Восстанавливаем VM=Fwe

glDisable(GL\_LIGHTING); //Выключаем освещение

glPopMatrix(); glutSwapBuffers();

glEndList(); //Завершение накопления команд в списке

Sleep(20);

}

void myReshape(int width, int height)

{

float R = 640.0 / 480.0; //Форматное соотношение

float w = 20; //Ширина мирового окна

float h = w / R, l = -w / 2, r = w / 2, b = -h / 2, t = h / 2, n, f;

if (width / height < 1) glViewport(0, 0, width, height);

else glViewport(0, 0, width, height);

}

void myIdle()

{

if (angle != 360) angle += angle\_plus;

else angle = 0;

glutPostRedisplay();

}

void key\_press(unsigned char key, int x, int y)

{

switch (key)

{

case 'o': //нажате клавишы 0 - начало или конец вращение объёкта

if (!check)

angle\_plus = 1;

else

angle\_plus = 0;

check = !check; break;

case 'l': //нажате клавишы 1 - вращение источника света

if (angle\_light != 360) angle\_light += 15;

else angle\_light = 0;

break;

case 'f': //нажате клавишы F - лицевый грани

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

break;

case 'b': //нажате клавишы B - нелицевый грани

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

break;

default:

break;

}

glutPostRedisplay();

}

void mouse\_press(int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON)

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

// устанавливает взаимодействие между текстурой и объектом - в данном случае текстурного отображения

// GL\_DECAL - аналог GL\_REPLACE, но для RGB. - создаёт эффект светящегося объекта

else if (button == GLUT\_RIGHT\_BUTTON)

glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);// GL\_MODULATE - модулирует коэффицент отражения для текстуры

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH); //устанавливая режим дисплея в функции main(), предусмотрено использование буфера глубины

glutInitWindowSize(640, 480);

glutInitWindowPosition(0, 0);

glutCreateWindow("Labaratory 8 mohamed gawish");

glutDisplayFunc(myDisplay);

glutKeyboardFunc(key\_press);

glutMouseFunc(mouse\_press);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

init();

glutMainLoop();

}

void draw\_K()

{

float myDiffuse[] = { 0.55, 0.15, 0.008, 1 };

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, myDiffuse); // Источник света в CKw

// параметры буквы К

float k\_1\_left = -8.0,

k\_1\_right = -6.5,

k\_2\_right = -3.0, k\_top = 8.5,

k\_bottom = 2.0,

k\_mid = (k\_top + k\_bottom) / 2;

{//red

glBegin(GL\_POLYGON); //face

glNormal3f(0, 0, 1);

glVertex3f(k\_1\_left, k\_bottom, thickness / 2);

glVertex3f(k\_1\_right, k\_bottom, thickness / 2);

glVertex3f(k\_1\_right, k\_top, thickness / 2);

glVertex3f(k\_1\_left, k\_top, thickness / 2);

glEnd();

glBegin(GL\_POLYGON); //back

glNormal3f(0, 0, -1);

glVertex3f(k\_1\_right, k\_bottom, -thickness / 2);

glVertex3f(k\_1\_left, k\_bottom, -thickness / 2);

glVertex3f(k\_1\_left, k\_top, -thickness / 2);

glVertex3f(k\_1\_right, k\_top, -thickness / 2);

glEnd();

// left

glBegin(GL\_POLYGON);

glNormal3f(-1, 0, 0);

glVertex3f(k\_1\_left, k\_bottom, -thickness / 2);

glVertex3f(k\_1\_left, k\_bottom, thickness / 2);

glVertex3f(k\_1\_left, k\_top, thickness / 2);

glVertex3f(k\_1\_left, k\_top, -thickness / 2);

glEnd();

// top part

h1 = 2; h2 = k\_top - k\_mid; h3 = sqrt(h1 \* h1 + h2 \* h2);

glBegin(GL\_POLYGON);

glNormal3f(0, 1, 0); glVertex3f(k\_1\_left, k\_top, thickness / 2);

glVertex3f(k\_1\_right, k\_top, thickness / 2);

glVertex3f(k\_1\_right, k\_top, -thickness / 2);

glVertex3f(k\_1\_left, k\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(1, 0, 0);

glVertex3f(k\_1\_right, k\_top, thickness / 2);

glVertex3f(k\_1\_right, k\_mid, thickness / 2);

glVertex3f(k\_1\_right, k\_mid, -thickness / 2);

glVertex3f(k\_1\_right, k\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(1, 0, 0);

glVertex3f(k\_1\_right, k\_mid, thickness / 2);

glVertex3f(k\_1\_right, k\_bottom, thickness / 2);

glVertex3f(k\_1\_right, k\_bottom, -thickness / 2);

glVertex3f(k\_1\_right, k\_mid, -thickness / 2);

glEnd();

}

// FACE

{

float myDiffuse[] = { 0.15, 0.15, 0.2, 1 };

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, myDiffuse); // Источник света в CKw

glBegin(GL\_POLYGON);

glNormal3f(0, 0, 1);

glVertex3f(k\_1\_right, k\_mid, thickness / 2);

glVertex3f(k\_1\_right + 1.7, k\_mid, thickness / 2);

glVertex3f(k\_2\_right, k\_top, thickness / 2);

glVertex3f(k\_2\_right - 1.7, k\_top, thickness / 2); glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0, 1);

glVertex3f(k\_2\_right - 1.7, k\_bottom, thickness / 2);

glVertex3f(k\_2\_right, k\_bottom, thickness / 2);

glVertex3f(k\_1\_right + 1.7, k\_mid, thickness / 2);

glVertex3f(k\_1\_right, k\_mid, thickness / 2);

glEnd();

}

// BACK

{

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(k\_1\_right + 1.7, k\_mid, -thickness / 2);

glVertex3f(k\_1\_right, k\_mid, -thickness / 2);

glVertex3f(k\_2\_right - 1.7, k\_top, -thickness / 2);

glVertex3f(k\_2\_right, k\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(k\_2\_right, k\_bottom, -thickness / 2);

glVertex3f(k\_2\_right - 1.7, k\_bottom, -thickness / 2);

glVertex3f(k\_1\_right, k\_mid, -thickness / 2);

glVertex3f(k\_1\_right + 1.7, k\_mid, -thickness / 2);

glEnd();

}

// BREDGES

{

glBegin(GL\_POLYGON);

glNormal3f(-h1 / h3, h2 / h3, 0);

glVertex3f(k\_1\_right, k\_mid, thickness / 2); glVertex3f(k\_2\_right - 1.7, k\_top, thickness / 2);

glVertex3f(k\_2\_right - 1.7, k\_top, -thickness / 2);

glVertex3f(k\_1\_right, k\_mid, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 1, 0);

glVertex3f(k\_2\_right - 1.7, k\_top, thickness / 2);

glVertex3f(k\_2\_right, k\_top, thickness / 2);

glVertex3f(k\_2\_right, k\_top, -thickness / 2);

glVertex3f(k\_2\_right - 1.7, k\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(h1 / h3, -h2 / h3, 0);

glVertex3f(k\_2\_right, k\_top, thickness / 2);

glVertex3f(k\_1\_right + 1.7, k\_mid, thickness / 2);

glVertex3f(k\_1\_right + 1.7, k\_mid, -thickness / 2);

glVertex3f(k\_2\_right, k\_top, -thickness / 2);

glEnd();

// bottom part

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(k\_1\_left, k\_bottom, -thickness / 2);

glVertex3f(k\_1\_right, k\_bottom, -thickness / 2);

glVertex3f(k\_1\_right, k\_bottom, thickness / 2);

glVertex3f(k\_1\_left, k\_bottom, thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(-h1 / h3, -h2 / h3, 0);

glVertex3f(k\_2\_right - 1.7, k\_bottom, thickness / 2);

glVertex3f(k\_1\_right, k\_mid, thickness / 2);

glVertex3f(k\_1\_right, k\_mid, -thickness / 2); glVertex3f(k\_2\_right - 1.7, k\_bottom, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(k\_2\_right, k\_bottom, thickness / 2);

glVertex3f(k\_2\_right - 1.7, k\_bottom, thickness / 2);

glVertex3f(k\_2\_right - 1.7, k\_bottom, -thickness / 2);

glVertex3f(k\_2\_right, k\_bottom, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(h1 / h3, h2 / h3, 0);

glVertex3f(k\_1\_right + 1.7, k\_mid, thickness / 2);

glVertex3f(k\_2\_right, k\_bottom, thickness / 2);

glVertex3f(k\_2\_right, k\_bottom, -thickness / 2);

glVertex3f(k\_1\_right + 1.7, k\_mid, -thickness / 2);

glEnd();

}

}

// параметры буквы T

void draw\_T()

{

float t\_1\_left = -2.0,

t\_1\_right = 2.0,

t\_top = 8.5,

t\_bottom = 2.0,

t\_mid = -0.75;

float myDiffuse[] = { 0.55, 0.15, 0.008, 1 };

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, myDiffuse); // Источник света в CKw// FACE

{

glBegin(GL\_POLYGON);

glNormal3f(0, 0, 1);

glVertex3f(t\_1\_left, t\_top, thickness / 2);

glVertex3f(t\_1\_right, t\_top, thickness / 2);

glVertex3f(t\_1\_right, t\_top - 1.5, thickness / 2);

glVertex3f(t\_1\_left, t\_top - 1.5, thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0, 1);

glVertex3f(t\_mid, t\_top - 1.5, thickness / 2);

glVertex3f(t\_mid, t\_bottom, thickness / 2);

glVertex3f(t\_mid + 1.5, t\_bottom, thickness / 2);

glVertex3f(t\_mid + 1.5, t\_top - 1.5, thickness / 2);

glEnd();

}

// BACK

{

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(t\_1\_left, t\_top, -thickness / 2);

glVertex3f(t\_1\_right, t\_top, -thickness / 2);

glVertex3f(t\_1\_right, t\_top - 1.5, -thickness / 2);

glVertex3f(t\_1\_left, t\_top - 1.5, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(t\_mid, t\_top - 1.5, -thickness / 2); glVertex3f(t\_mid, t\_bottom, -thickness / 2);

glVertex3f(t\_mid + 1.5, t\_bottom, -thickness / 2);

glVertex3f(t\_mid + 1.5, t\_top - 1.5, -thickness / 2);

glEnd();

}

// BRIDGES

{

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(t\_1\_left, t\_top, -thickness / 2);

glVertex3f(t\_1\_left, t\_top - 1.5, -thickness / 2);

glVertex3f(t\_1\_left, t\_top - 1.5, thickness / 2);

glVertex3f(t\_1\_left, t\_top, thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0, 1);

glVertex3f(t\_1\_right, t\_top, -thickness / 2);

glVertex3f(t\_1\_right, t\_top - 1.5, -thickness / 2);

glVertex3f(t\_1\_right, t\_top - 1.5, thickness / 2);

glVertex3f(t\_1\_right, t\_top, thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(t\_mid, t\_bottom, -thickness / 2);

glVertex3f(t\_mid, t\_top - 1.5, -thickness / 2);

glVertex3f(t\_mid, t\_top - 1.5, thickness / 2);

glVertex3f(t\_mid, t\_bottom, thickness / 2);

glEnd();

glBegin(GL\_POLYGON); glNormal3f(0, 0, -1);

glVertex3f(t\_mid + 1.5, t\_bottom, -thickness / 2);

glVertex3f(t\_mid + 1.5, t\_top - 1.5, -thickness / 2);

glVertex3f(t\_mid + 1.5, t\_top - 1.5, thickness / 2);

glVertex3f(t\_mid + 1.5, t\_bottom, thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(t\_mid, t\_top - 1.5, -thickness / 2);

glVertex3f(t\_mid, t\_top - 1.5, thickness / 2);

glVertex3f(t\_1\_left, t\_top - 1.5, thickness / 2);

glVertex3f(t\_1\_left, t\_top - 1.5, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(t\_mid + 1.5, t\_top - 1.5, -thickness / 2);

glVertex3f(t\_mid + 1.5, t\_top - 1.5, thickness / 2);

glVertex3f(t\_1\_right, t\_top - 1.5, thickness / 2);

glVertex3f(t\_1\_right, t\_top - 1.5, -thickness / 2);

glEnd();

}

}

void draw\_Z()

{

float myDiffuse[] = { 1, 1, 1, 1 };

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, myDiffuse); // Источник света в CKw

float z\_left = 3.0,

z\_right = 7.0,

z\_top = 8.5,

z\_bottom = 2.0; glBegin(GL\_POLYGON); //передняя

glNormal3f(0, 0, 1);

glTexCoord2f(0, 0); glVertex3f(z\_left, z\_bottom, thickness / 2);

glTexCoord2f(0, 1); glVertex3f(z\_left, z\_top, thickness / 2);

glTexCoord2f(1, 1); glVertex3f(z\_right, z\_top, thickness / 2);

glTexCoord2f(1, 0); glVertex3f(z\_right, z\_bottom, thickness / 2);

glEnd();

glPixelZoom(1, -1);//переворачивается

glBegin(GL\_POLYGON); //задняя

glNormal3f(0, 0, -1);

glTexCoord2f(0, 0); glVertex3f(z\_left, z\_bottom, -thickness / 2);

glTexCoord2f(0, 1); glVertex3f(z\_left, z\_top, -thickness / 2);

glTexCoord2f(1, 1); glVertex3f(z\_right, z\_top, -thickness / 2);

glTexCoord2f(1, 0); glVertex3f(z\_right, z\_bottom, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON); //левая сторона

glNormal3f(1, 0, 0);

glVertex3f(z\_left, z\_top, thickness / 2);

glVertex3f(z\_left, z\_bottom, thickness / 2);

glVertex3f(z\_left, z\_bottom, -thickness / 2);

glVertex3f(z\_left, z\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);//правная

glNormal3f(-1, 0, 0);

glVertex3f(z\_right, z\_top, thickness / 2); glVertex3f(z\_right, z\_bottom, thickness / 2);

glVertex3f(z\_right, z\_bottom, -thickness / 2);

glVertex3f(z\_right, z\_top, -thickness / 2);

glEnd();

glEndList();

}

void draw\_I()

{

float myDiffuse[] = { 0.15, 0.15, 0.2, 1 };

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, myDiffuse); // Источник света в CKw

// параметры буквы И

float i\_1\_left = 8.0,

i\_1\_right = 9.5,

i\_2\_left = 11.5,

i\_2\_right = 13.0;

float i\_top = 8.5,

i\_bottom = 2.0;

// FACE

{

glBegin(GL\_POLYGON);

glNormal3f(0, 0, 1);

glVertex3f(i\_1\_left, i\_bottom, thickness / 2);

glVertex3f(i\_1\_right, i\_bottom, thickness / 2);

glVertex3f(i\_1\_right, i\_top, thickness / 2);

glVertex3f(i\_1\_left, i\_top, thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0, 1); glVertex3f(i\_1\_right, i\_bottom, thickness / 2);

glVertex3f(i\_2\_left, i\_top - 3, thickness / 2);

glVertex3f(i\_2\_left, i\_top, thickness / 2);

glVertex3f(i\_1\_right, i\_bottom + 3, thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0, 1);

glVertex3f(i\_2\_left, i\_bottom, thickness / 2);

glVertex3f(i\_2\_right, i\_bottom, thickness / 2);

glVertex3f(i\_2\_right, i\_top, thickness / 2);

glVertex3f(i\_2\_left, i\_top, thickness / 2);

glEnd();

}

// BACK

{

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(i\_1\_right, i\_bottom, -thickness / 2);

glVertex3f(i\_1\_left, i\_bottom, -thickness / 2);

glVertex3f(i\_1\_left, i\_top, -thickness / 2);

glVertex3f(i\_1\_right, i\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(i\_2\_left, i\_top - 3, -thickness / 2);

glVertex3f(i\_1\_right, i\_bottom, -thickness / 2);

glVertex3f(i\_1\_right, i\_bottom + 3, -thickness / 2);

glVertex3f(i\_2\_left, i\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(i\_2\_right, i\_bottom, -thickness / 2); glVertex3f(i\_2\_left, i\_bottom, -thickness / 2);

glVertex3f(i\_2\_left, i\_top, -thickness / 2);

glVertex3f(i\_2\_right, i\_top, -thickness / 2);

glEnd();

}

// BRIDGES

{

glBegin(GL\_POLYGON);

glNormal3f(-1, 0, 0);

glVertex3f(i\_1\_left, i\_bottom, thickness / 2);

glVertex3f(i\_1\_left, i\_top, thickness / 2);

glVertex3f(i\_1\_left, i\_top, -thickness / 2);

glVertex3f(i\_1\_left, i\_bottom, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 1, 0);

glVertex3f(i\_1\_left, i\_top, thickness / 2);

glVertex3f(i\_1\_right, i\_top, thickness / 2);

glVertex3f(i\_1\_right, i\_top, -thickness / 2);

glVertex3f(i\_1\_left, i\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(1, 0, 0);

glVertex3f(i\_1\_right, i\_top, thickness / 2);

glVertex3f(i\_1\_right, i\_bottom, thickness / 2);

glVertex3f(i\_1\_right, i\_bottom, -thickness / 2);

glVertex3f(i\_1\_right, i\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

h1 = -2; h2 = i\_top - i\_bottom - 2; h3 = sqrt(h1 \* h1 + h2 \* h2);

glNormal3f(h1 / h3, h2 / h3, 0);

glVertex3f(i\_1\_right, i\_bottom + 3, thickness / 2); glVertex3f(i\_2\_left, i\_top, thickness / 2);

glVertex3f(i\_2\_left, i\_top, -thickness / 2);

glVertex3f(i\_1\_right, i\_bottom + 3, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 1, 0);

glVertex3f(i\_2\_left, i\_top, thickness / 2);

glVertex3f(i\_2\_right, i\_top, thickness / 2);

glVertex3f(i\_2\_right, i\_top, -thickness / 2);

glVertex3f(i\_2\_left, i\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(1, 0, 0);

glVertex3f(i\_2\_right, i\_top, thickness / 2);

glVertex3f(i\_2\_right, i\_bottom, thickness / 2);

glVertex3f(i\_2\_right, i\_bottom, -thickness / 2);

glVertex3f(i\_2\_right, i\_top, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(i\_2\_right, i\_bottom, thickness / 2);

glVertex3f(i\_2\_left, i\_bottom, thickness / 2);

glVertex3f(i\_2\_left, i\_bottom, -thickness / 2);

glVertex3f(i\_2\_right, i\_bottom, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(-1, 0, 0);

glVertex3f(i\_2\_left, i\_bottom, thickness / 2);

glVertex3f(i\_2\_left, i\_top, thickness / 2);

glVertex3f(i\_2\_left, i\_top, -thickness / 2);

glVertex3f(i\_2\_left, i\_bottom, -thickness / 2);

glEnd(); glBegin(GL\_POLYGON);

glNormal3f(-h1 / h3, -h2 / h3, 0);

glVertex3f(i\_2\_left, i\_top - 3, thickness / 2);

glVertex3f(i\_1\_right, i\_bottom, thickness / 2);

glVertex3f(i\_1\_right, i\_bottom, -thickness / 2);

glVertex3f(i\_2\_left, i\_top - 3, -thickness / 2);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(i\_1\_right, i\_bottom, thickness / 2);

glVertex3f(i\_1\_left, i\_bottom, thickness / 2);

glVertex3f(i\_1\_left, i\_bottom, -thickness / 2);

glVertex3f(i\_1\_right, i\_bottom, -thickness / 2);

glEnd();

}

}