Министерство образования и науки Российской Федерации Федеральное агентство по образованию

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №8 дисциплины

«Компьютерная графика»

Выполнил студент группы ИВТ-22 /Крючков И. С/ Проверил /Коржавина А. С./

Киров 2021

**Цель работы:**

# Научиться программно выводить трёхмерные объекты.

# Задание:

### Написать на языке Pascal программу, которая выводит каркас векторной полигональной модели.

**Теория:**

Для описания пространственных объектов здесь используются следующие элементы: вершины: отрезки прямых (векторы); полилинии. полигоны, полигональные поверхности (рис. 1). Элемент «вершина» (vertex) — главный элемент описания, все другие являются производными. При использовании трехмерной декартовой системы координаты вершин определяются как (x,y,z). Каждый объект однозначно определяется координатами собственных вершин.

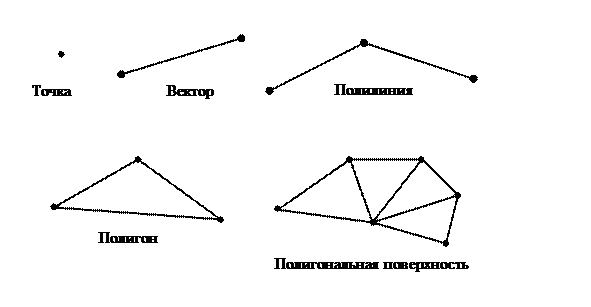


Рисунок 1 – Элементы описания пространственных объектов

Вершина может моделировать отдельный точечный объект, размер которого не имеет значения, а также может использоваться в качестве конечных точек для линейных объектов и полигонов. Двумя вершинами задается вектор. Несколько векторов составляют полилинию. Полилиния может моделировать отдельный линейный объект, толщина которого не учитывается, а также мо­жет представлять контур полигона. Полигон моделирует площадный объект Один полигон может описывать плоскую грань объемного объекта. Несколь­ко граней составляют объемный объект в виде полигональной поверхности – многогранник или незамкнутую поверхность (в литературе часто употребляется название «полигональная сетка»).

**Схемы алгоритмов:**

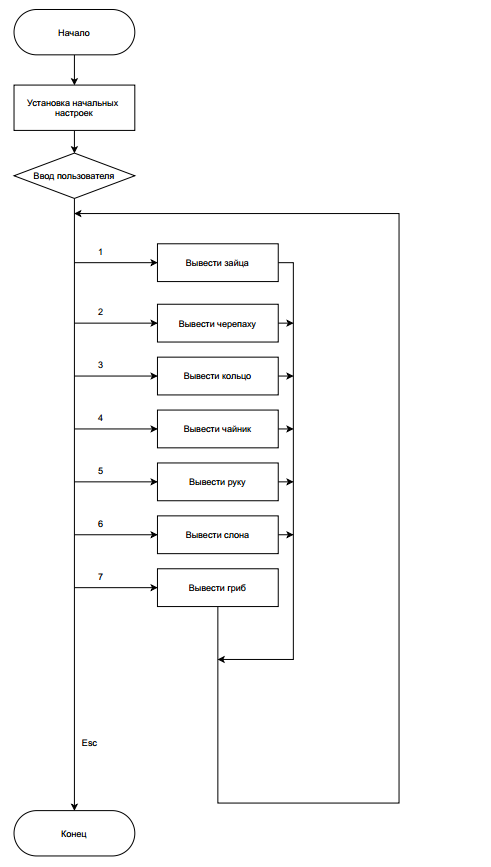
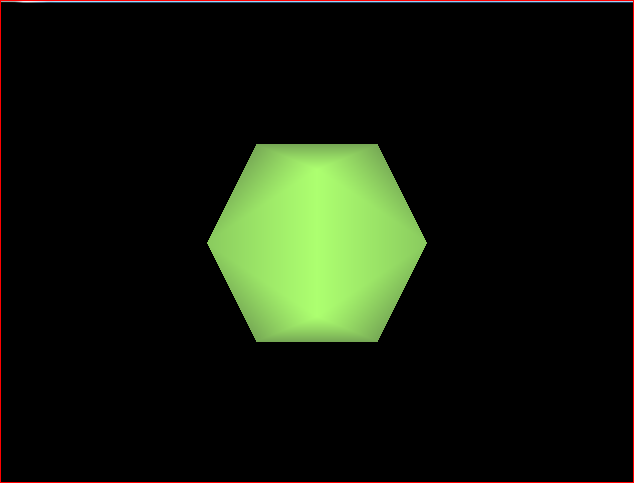
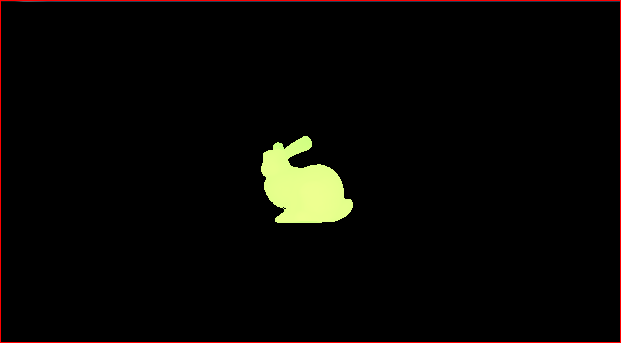
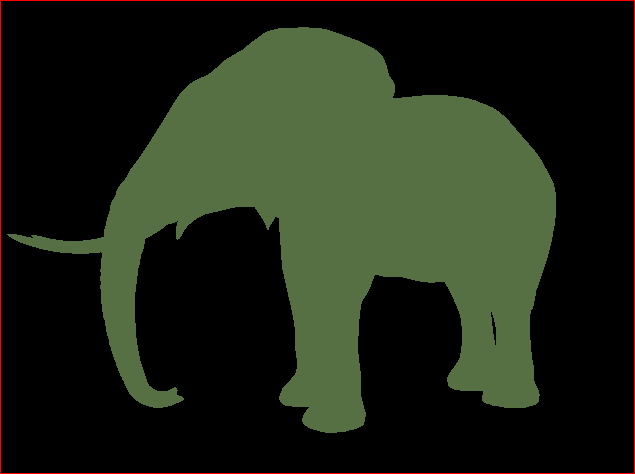


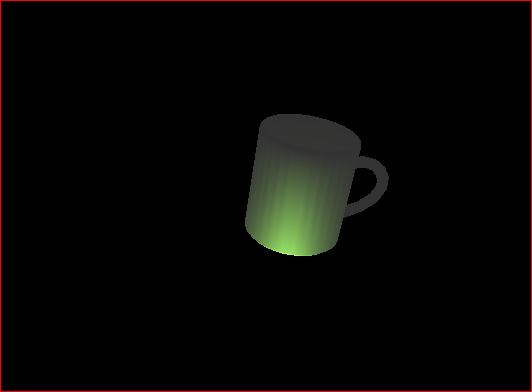
Рисунок 1 – Схема основной программы

**Экранные формы:**









**Исходный код:**

program firstprogram;

{$mode objfpc}{$H+}

uses

gl, glu, glut, Classes, BMPcomn, Windows, StrUtils, SysUtils;

const

CubeSize = 1;

type

TXYZ = record

x, y, z: GLdouble;

end;

TFACES = record

r, g, b: real;

faces:array of integer;

end;

var

// размеры окна

Width : integer = 640;

Height : integer = 480;

// rotationaxisx, rotationaxisy, rotationaxisz - ось, вокруг которой поворачивается объект

// shiftx, shifty, shiftz - смещение (перенос) объекта по каждой оси

// xscale, yscale, zscale - коэффициенты масштабирования по каждой оси

// используется для анимации, равна -1 когда куб уменьшается и 1 когда увеличивается

f:text;

c:string;

ns:integer;

vals :array of string;

i:integer;

numVertex:integer;

numFaces:integer;

numEdges:integer;

vrtx:array of TXYZ;

nv:integer;

arrFaces:array of TFACES;

nfi:integer;

nf:integer;

j:integer;

tvar:TXYZ;

ErrorCode:integer;

Xangle, Yangle, Zangle:integer;

scalex, scaley, scalez :GLfloat;

fid:integer = 1;

fname:string;

translate:GLfloat;

persp:GLfloat;

lz:GLfloat;

lw:GLfloat;

procedure Init(tmp: Boolean);

begin

Xangle := 0;

Yangle := 0;

Zangle := 0;

scalex := 1;

scaley := 1;

scalez := 1;

persp := 0.1;

translate := -5;

// glLightModelf(GL\_LIGHT\_MODEL\_TWO\_SIDE, GL\_TRUE);

// задаём начальные коэффициенты - поворот вокруг оси Х, все коэфф. масштабирования равны 1

// rotationaxisx:=1; xscale:=1; yscale:=1; zscale:=1;

// // цвет очистки

// glClearColor (0.0, 0.0, 0.0, 0.0);

// // установка параметров источника света GL\_LIGHT0

// // установка позиции источника света

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

// // установка интенсивности фонового освещения источника света

// glLightfv(GL\_LIGHT0, GL\_AMBIENT, lmodel\_ambient);

// // включить освещение

// glEnable(GL\_LIGHTING);

// // включить источник света с номером 0

// //glEnable(GL\_LIGHT0);

// // включить проверку на глубину

glEnable(GL\_DEPTH\_TEST);

end;

procedure Display; cdecl;

var

k,l:integer;

DiffuseLight: array[0..2] of GLfloat = (0.4, 0.7, 0.2);

PositionLight: array[0..3] of GLfloat;

begin

PositionLight[0] := 0;

PositionLight[1] := 0;

PositionLight[2] := 1;

PositionLight[3] := 0;

// очистка цветового буфера и буфера глубины

// т.к. включен режим двойной буфферизации, то очищается не отображаемый цветовой буфер

// в буфере глубины координаты z всех пикселей предыддущей сцены, их надо обнулить и посчитать заново

glClear (GL\_COLOR\_BUFFER\_BIT OR GL\_DEPTH\_BUFFER\_BIT OR GL\_STENCIL\_BUFFER\_BIT);

glLoadIdentity;

glTranslatef(0, 0, translate);

glEnable(GL\_LIGHTING);

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

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

// glLightf(GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, 0.0);

//

// glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 0.2);

//

// glLightf(GL\_LIGHT0, GL\_QUADRATIC\_ATTENUATION, 0.4);

glEnable(GL\_LIGHT0);

glEnable(GL\_CULL\_FACE);

glEnable(GL\_COLOR\_MATERIAL);

glScalef(scalex, scaley, scalez);

glRotatef(Xangle, 1, 0, 0);

glRotatef(Yangle, 0, 1, 0);

glRotatef(Zangle, 0, 0, 1);

for k := 0 to numFaces-1 do

begin

glBegin(GL\_LINE\_LOOP);

// glBegin(GL\_POLYGON);

for l := 0 to Length(arrFaces[k].faces)-1 do

begin

tvar := vrtx[arrFaces[k].faces[l]];

glColor3f(1,1,1);

glVertex3d(tvar.x, tvar.y, tvar.z);

end;

glEnd;

end;

glutSwapBuffers;

end;

procedure Reshape(W, H: Integer); cdecl;

begin

// установка области ыидимости внутри окна приложения

glViewport (0, 0, w, h);

// выбор матрицы проекции для выполнения операций над ней

glMatrixMode (GL\_PROJECTION);

// присвоить текущей матрице проекции единичную матрицу

glLoadIdentity ();

// задать коэффициенты перспективной проекции - усеченный конус видимости

gluPerspective(45.0, w/h, persp, 1000.0);

// переключение на модельно-видовую матрицу

glMatrixMode(GL\_MODELVIEW);

// присвоить текущей модельно-видовой матрице проекции единичную матрицу

glLoadIdentity();

// установить параметры камеры наблюдателя (афинные комбинированные преобразования)

// наблюдатель (камера) в точке x=0, y=0, z=5

// центр сцены в точке x=0, y=0, z=0

// верхний вектор (куда направлен верх камеры) x=0, y=1, z=0

gluLookAt (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

end;

procedure rFile();

begin

Xangle := 0;

Yangle := 0;

Zangle := 0;

scalex := 1;

scaley := 1;

scalez := 1;

if fid = 1 then

begin

fname := 'ico';

persp := 0.1;

translate := -5;

lz := 1;

lw := 1;

end

else if fid = 2 then

begin

fname := 'bunny';

persp := 0.5;

translate := -1;

lz := 1;

lw := 1;

end

else if fid = 3 then

begin

fname := 'brain';

persp := 5;

translate := -5;

lz := 1;

lw := 1;

end

else if fid = 4 then

begin

fname := 'elephant';

persp := 0.1;

translate := -50;

lz := 50;

lw := 0;

end

else if fid = 5 then

begin

fname := 'hand';

persp := 25;

translate := -50;

lz := 1;

lw := 1;

end

else if fid = 6 then

begin

fname := 'mug';

persp := 1;

translate := -20;

lz := 1;

lw := 1;

end

else if fid = 7 then

begin

fname := 'mushroom';

persp := 1;

translate := -5;

lz := 1;

lw := 1;

end

else if fid = 8 then

begin

fname := 'teapot';

persp := 1;

translate := -15;

lz := 1;

lw := 1;

end

else if fid = 9 then

begin

fname := 'torus';

persp := 1;

translate := -5;

lz := 1;

lw := 1;

end

else if fid = 0 then

begin

fname := 'turtle';

persp := 10;

translate := -300;

lz := 1;

lw := 0;

end;

assign (f, 'models/'+fname+'.off');

{$i-}

reset (f);

{$i+}

If IOresult<>0 then

begin

writeln ('File doesn''t exist');

halt;

end;

ns := 0;

nv := 0;

nf := 0;

while not eof (f) do begin

readln (f, c);

// https://en.wikipedia.org/wiki/OFF\_(file\_format)

if ns = 1 then

begin

vals := SplitString(c,' ');

numVertex := strtoint(vals[0]);

numFaces := strtoint(vals[1]);

numEdges := strtoint(vals[2]);

setLength(vrtx, numVertex);

setLength(arrFaces, numFaces);

end

else if ((ns > 1) and (ns <= numVertex+1)) then

begin

vals := SplitString(c,' ');

// vrtx[nv].x := strtofloat(vals[0]);

// vrtx[nv].y := strtofloat(vals[1]);

// vrtx[nv].z := strtofloat(vals[2]);

Val(vals[0], vrtx[nv].x, ErrorCode);

Val(vals[1], vrtx[nv].y, ErrorCode);

Val(vals[2], vrtx[nv].z, ErrorCode);

inc(nv);

end

else if ((ns > numVertex+1) and (ns <= numVertex+1+numFaces)) then

begin

vals := SplitString(c,' ');

nfi := strtoint(vals[0]);

setLength(arrFaces[nf].faces, nfi);

for j := 1 to nfi do

begin

arrFaces[nf].faces[j-1] := strtoint(vals[j]);

// writeln(arrFaces[nf].faces[j-1]);

end;

// arrFaces[nf].r := strtofloat(vals[j+1]);

// arrFaces[nf].g := strtofloat(vals[j+2]);

// arrFaces[nf].b := strtofloat(vals[j+3]);

inc(nf);

end;

inc(ns);

end;

close (f);

end;

procedure Keyboard(Key: Byte; X, Y: Longint); cdecl;

begin

// выбор кода клавиши, которую нажал пользователь

// коды клавиш языка С можно посмотреть, например, тут http://www.expandinghead.net/keycode.html

case Key of

// поворот вниз

115:

begin

Xangle := (Xangle + 10) mod 360;

glutPostRedisplay();

end;

// поворот вверх

119:

begin

Xangle := (Xangle - 10) mod 360;

glutPostRedisplay();

end;

// поворот влево

97:

begin

Yangle := (Yangle - 10) mod 360;

glutPostRedisplay();

end;

// поворот вправо

100:

begin

Yangle := (Yangle + 10) mod 360;

glutPostRedisplay();

end;

101: // zoom out

begin

scalex := scalex \* 1.1;

scaley := scaley \* 1.1;

scalez := scalez \* 1.1;

glutPostRedisplay();

end;

113: // zoom in

begin

scalex := scalex \* 0.9;

scaley := scaley \* 0.9;

scalez := scalez \* 0.9;

glutPostRedisplay();

end;

49: // 1

begin

fid := 1;

rFile();

glutPostRedisplay();

end;

50: // 2

begin

fid := 2;

rFile();

glutPostRedisplay();

end;

51: // 3

begin

fid := 3;

rFile();

glutPostRedisplay();

end;

52: // 4

begin

fid := 4;

rFile();

glutPostRedisplay();

end;

53: // 5

begin

fid := 5;

rFile();

glutPostRedisplay();

end;

54: // 6

begin

fid := 6;

rFile();

glutPostRedisplay();

end;

55: // 7

begin

fid := 7;

rFile();

glutPostRedisplay();

end;

56: // 8

begin

fid := 8;

rFile();

glutPostRedisplay();

end;

57: // 9

begin

fid := 9;

rFile();

glutPostRedisplay();

end;

48: // 0

begin

fid := 0;

rFile();

glutPostRedisplay();

end;

// выход - ECS

27: Halt(0);

end;

end;

procedure glutInitPascal(ParseCmdLine: Boolean);

var

// массив указателей на параметры командной строки

ArgArray: array of PChar;

// ArgCount - число параметров командной строки, которое будет передано в процедуру glutInit

ArgCount, I: Integer;

begin

// функция ParamCount() возвращает число параметров командной строки

// в Delphi нумерация параметров с 0, поэтому увеличиваем число параметров на 1

ArgCount := ParamCount() + 1;

// выделяем для каждого параметра указатель (адрес на этот параметр)

SetLength(ArgArray, ArgCount);

// для каждого параметра получаем адрес функцией PChar() и записываем его в массив ArgArray

for I := 0 to ArgCount - 1 do ArgArray[I] := PChar(ParamStr(I));

// вызываем glutInit, передаём в неё число параметров командной строки и указатель на массив с указателми на параметры

glutInit(@ArgCount, @ArgArray);

end;

begin

glutInitPascal(True);

// установка режима отображения окна

// двойная буферизация, цвет в формате RGB, использовать буфер глубины

glutInitDisplayMode(GLUT\_DOUBLE OR GLUT\_RGB OR GLUT\_DEPTH);

// установить размер окна приложения

glutInitWindowSize(Width, Height);

// создать окно с именем Lab12

glutCreateWindow('Lab8');

// выполнить начальную инициализацию

Init(true);

// указать основные функции OpenGL

// функция вывода изображения - Display

glutDisplayFunc(@Display);

// функция, вызываемая при изменении размера окна - Reshape

glutReshapeFunc(@Reshape);

// функция, вызываемая при нажатии клавиши клавиатуры - Keyboard

glutKeyboardFunc(@Keyboard);

// запуск основного цикла обработки событий

glutMainLoop;

end.

**Вывод:** В ходе данной лабораторной работы была реализована программа, с помощью которой можно выводить каркас векторной полигональной модели. Также познакомились с файловым расширением .off. В файле данного расширения содержится описание 3D-объектов.