МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО

ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»

в г. Смоленске

Кафедра ВТ

Отчет  
по лабораторной работе  
Тема: «Создание графического редактора»

по курсу: «Компьютерная графика»

Студент: Покровский А.Н.

Группа: АС-13

Преподаватель: Малашенкова И. В

Смоленск. 2015

**Требования к программному продукту**

Создать программу, которая будет отвечать стандарту CUA (Стандарт интерфейсов пользователя для операционных систем и компьютерных программ). Также должна выполнять следующие действия:

1. Проверять правильность ввода;
2. работать по одному из способов задания полигональных сеток (явное задание многоугольников, список вершин, список ребер).
3. строить три проекции (вид спереди, вид слева, вид сверху) и одну изометрию;
4. осуществлять действия над фигурой: поворот, сдвиг, масштабирование;
5. осуществлять запись и чтение данных из файла.

**Модульная структура программы**



*Рисунок 1. - Модульная структура программы «Графический редактор»*

**Описание функций:**

* Обнуляются коэффициенты нормализации.
* function isCorrectPoint(ed: Tedit): boolean; - проверка Edit на пустоту
* function isCorrectVector(ed: TEdit): boolean; - проверка Edit на пустоту и на существование введенной точки
* function isAlreadyExistPoint(x,y,z: real): boolean; - проверка на существование введенной точки
* procedure SetPoint(number: integer; x,y,z: real); - добавить точку (x,y,z) на позицию numer в массиве Points и в StringGridPoints
* function isAlreadyExistVector(v1,v2: integer): boolean; - проверка на существование введенного отрезка
* procedure SetVector(numer,v1,v2: integer); - добавить отрезок (v1,v2) на позицию numer в массиве Vectors и в StringGridVectors
* procedure ClearImage(Img:TImage); - очистить выбранный Image
* procedure ClearAllImages; - очистка всех image на форме
* Function MultiplyOnMatrix(Point: TPoints; Matrix44: TMatrix44): TPoints; - умножение столбца координат точки на матрицу
* Function T(Dx,Dy,Dz: real): TMatrix44; - формирование матрицы смещения
* Function S(Sx,Sy,Sz: real): TMatrix44; - формирование матрицы масштаба
* Function R(a,b,g: real): TMatrix44; - формирование матрицы поворота
* function normalization: TAoTPoints; - проверка нормализации и нормализация, если необходимо
* procedure PrintPoints; - отрисовка точек
* procedure PrintVectors; - отрисовка отрезков
* procedure PrintAll = ClearAllImages; + PrintPoints; + PrintVectors; - нарисовать все
* procedure BtnCreateClick(Sender: TObject); - создать чистый лист
* procedure BtnOpenClick(Sender: TObject); - загрузить из файла
* procedure BtnSaveClick(Sender: TObject); - записать в файл
* procedure BtnSaveAsClick(Sender: TObject); - записать в выбранный файл
* procedure BtnEditClick(Sender: TObject); - открыть панель редактирования
* procedure NPrintClick(Sender: TObject); - печать
* procedure NFileClick(Sender: TObject); - открыть подменю «файл»
* procedure NMotionClick(Sender: TObject); - открыть панель «действие»
* procedure FormCreate(Sender: TObject) – Происходит заполнение первой строки в таблице для рёбер и вершин. Производится очищение областей рисования.
* procedure BtnAddPointClick(Sender: TObject); - добавить точку
* procedure BtnChangePointClick(Sender: TObject); - изменить точку
* procedure BtnDelPointClick(Sender: TObject); - удалить точку
* procedure BtnAddVectorClick(Sender: TObject); - добавить отрезок
* procedure BtnChangeVectorClick(Sender: TObject); - изменить отрезок
* procedure BtnDelVectorClick(Sender: TObject); - удалить отрезок
* procedure NExitClick(Sender: TObject); - выход
* procedure BtnMotionRoughPlusClick(Sender: TObject); - грубое действие в +
* procedure BtnMotionRoughMinusClick(Sender: TObject); - грубое действие в -
* procedure BtnMotionDueClick(Sender: TObject); - точное действие

**Реализация**

1. **Ввод данных**

Вести данные возможно двумя способами:

1. С помощью открытия раннее записанного файла.
2. С помощью заполнения массивов вручную.

Рассмотрим ввод вручную. Координаты вводятся в два списка: вершин и рёбер. Могут возникать следующие ошибки при вводе значений:

*Для вершин.*

Значение можно добавить в список в том случае, если введены все координаты, иначе появится сообщение ошибки, что «Введите все значения!». Когда добавили координату одной точки и следом добавляем ещё одну, то проверяем на повторение весь список, и если такой не существует, то добавляем новую; если же встречается совпадение, то выдаём сообщение об ошибке «Такая вершина уже существует».

*Для рёбер.*

В список рёбер могут быть введены символы в виде номера добавленной ранее вершины, если у нас будет введено значение 4 и 5, а таких вершин ещё нет, то появиться ошибка «Вершины с таким номером не существует».Также нельзя вводить два раза одно и то же ребро.

1. **Хранение данных**

В программе вершины и рёбра хранятся в динамических массивах Points: array of TPoints(x,y,z:real) и Vectors: array of TVectors(v1,v2: integer) соответственно. Для хранения нормированных координат используем коэффициенты нормировки k, xmin, ymin, zmin. При сохранении массивы пишутся в текстовый файл.

1. **Действия**

**Сдвиг.**

Пользователь вводит коэффициенты сдвига. Данное преобразование выполняется с помощью почленного умножения координат вершин на матрицу:



**Поворот.**

Пользователь вводит коэффициенты поворота. Данное преобразование выполняется с помощью почленного умножения координат вершин на матрицу:

 M(\alpha,\beta,\gamma) = \begin{pmatrix} 
    \cos \alpha \cos \gamma - \sin \alpha \cos \beta \sin \gamma
&  -\cos \alpha \sin \gamma - \sin \alpha \cos \beta \cos \gamma
&   \sin \alpha \sin \beta \\
    \sin \alpha \cos \gamma + \cos \alpha \cos \beta \sin \gamma
&  -\sin \alpha \sin \gamma + \cos \alpha \cos \beta \cos \gamma
&  -\cos \alpha \sin \beta \\
   \sin \beta \sin \gamma
&   \sin \beta \cos \gamma
&   \cos \beta
\end{pmatrix}  

**Масштаб.**

Пользователь вводит коэффициенты масштабирования. Данное преобразование выполняется с помощью почленного умножения координат вершин на матрицу:



После действий над объектами проверяются условия нормировки, и, если необходимо, производится нормирование.

1. **Строка статуса**

После того как выполним как-либо действия, связанные с масштабом, поворотом и сдвигом изменяется значения в строке статуса. В самом начале работы она имеет значения:



*Рисунок 2. Масштаб, поворот, сдвиг до начала действия*

Изменение этих значений нам помогает узнать по какой оси и какое действие мы выполнили с нашим объектом. Изменение в данной строке показано на рисунке:



*Рисунок 3. - Масштаб, поворот, сдвиг после выполнения действия*

1. **Нормирование**

Нормирование - алгоритм, преобразующий введённые координаты точек в нормированные координаты. Нормирование позволяет вписать фигуру в канонический куб. Условия нормирования выглядят следующим образом:

Если условия выполняются, то процедура возвращает исходный массив. Если условия не выполняются, то осуществляется нормирование по следующему алгоритму:

1. Ищем минимальную координату по каждой оси.
2. Выполняем сдвиг на найденные величины (для каждой оси свой коэффициент сдвига).
3. Для преобразованных координат находим максимальную координату – коэффициент нормирования «k».
4. Делим каждую координату на коэффициент нормирования.

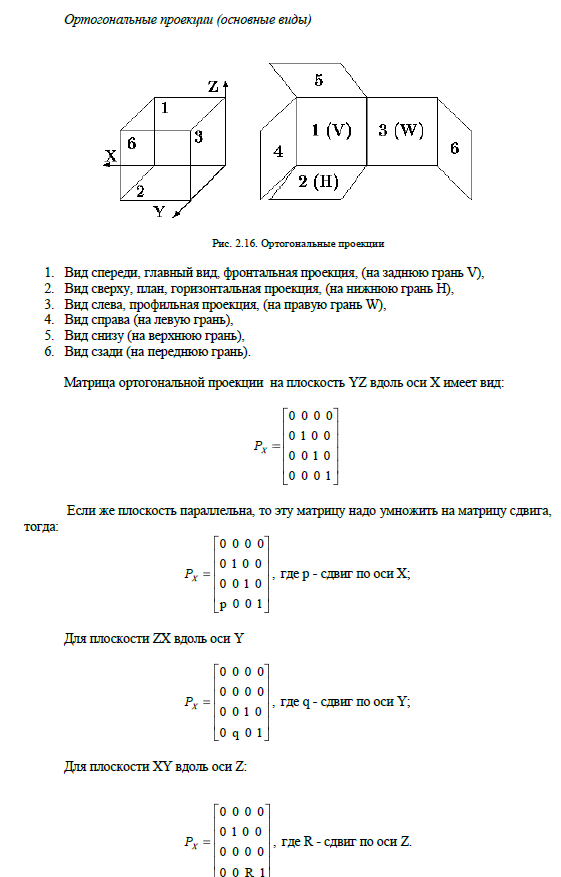
Все действия над координатами выполняются в нормированном виде. После выполнения каждого действия нужно проверять выполнение условий и, если нужно, опять проводить нормирование.

1. **Построение проекций**

Перед выводом проекции, и изометрии происходит нормирование всех координат. Направления осей для ортогональных проекций на плоскости XOY, XOZ, ZOY выбираются следующим образом:



*Рисунок 4. - Положение осей проекции и изометрии на экране*



Нормированные координаты являются промежуточными. Не зависят от устройств. Переход от нормированных координат к экранным выполняется следующим образом:

Xэ = Kэ \* Xн + Dсдв.х

Yэ = Kэ \* Yн + Dсдв.y

Zэ = Kэ \* Zн + Dсдв.z

При этом для каждого поля вывода необходимо учесть соответствие направления осей в геометрии и на экране монитора:

x:=round( normPoints[i].x\*k + d);

y:=round( normPoints[i].y\*k + d);

z:=270 - round( normPoints[i].z\*k);

Form1.ImageYOZ.Canvas.Ellipse( y-5, z-5, y+5, z+5 ); //вывод точки (y,z)

Form1.ImageXOY.Canvas.Ellipse( y-5, x-5, y+5, x+5 ); //вывод точки (y,x)

Form1.ImageXOZ.Canvas.Ellipse( x-5, z-5, x+5, z+5 ); //вывод точки (x,z)

При построении изометрии используем следующее преобразование координат: ;

x:= round( 120 + (normPoints[i].y-normPoints[i].x\*sin(PI/4))\*k/2 + d );

y:= round( 140 - (normPoints[i].z-normPoints[i].x\*sin(PI/4))\*k/2 + d );

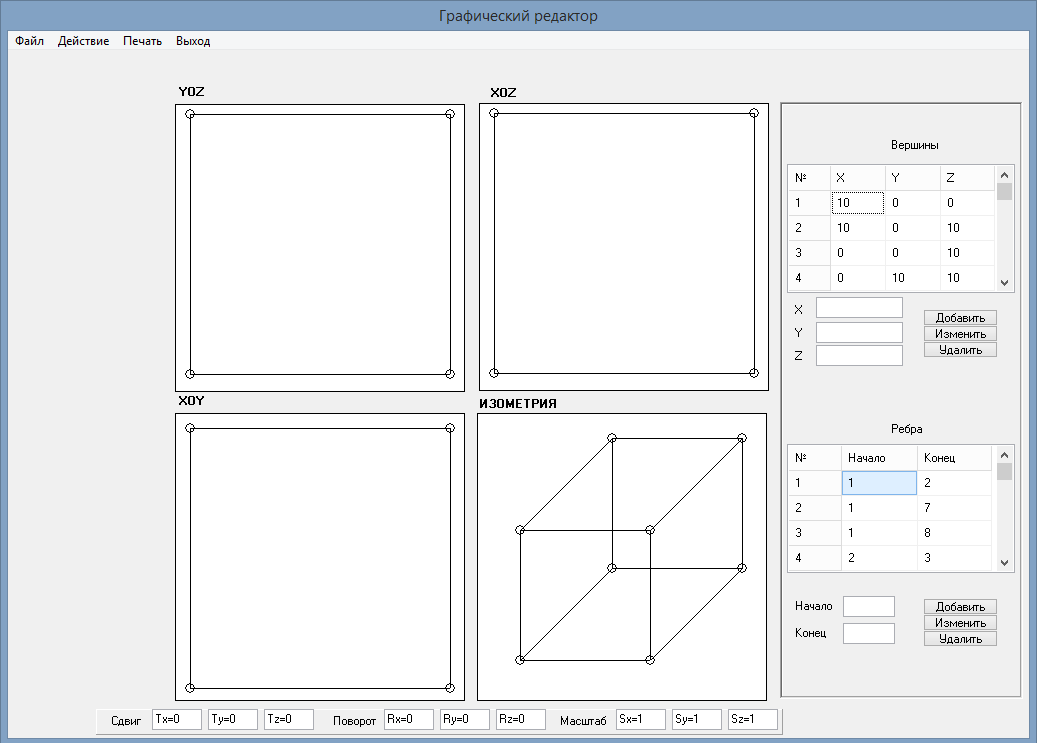
form1.ImageAll.Canvas.Ellipse( x-5, y-5, x+5, y+5 );

1. **Умножение матриц**

Матрицу координат вершины умножаем на матрицу действия (поворот, сдвиг, масштаб). На вход функции подаются 2 матрицы, которые перемножаются по математическим правилам, после чего функция возвращает их произведение.

**Тестирование программы**

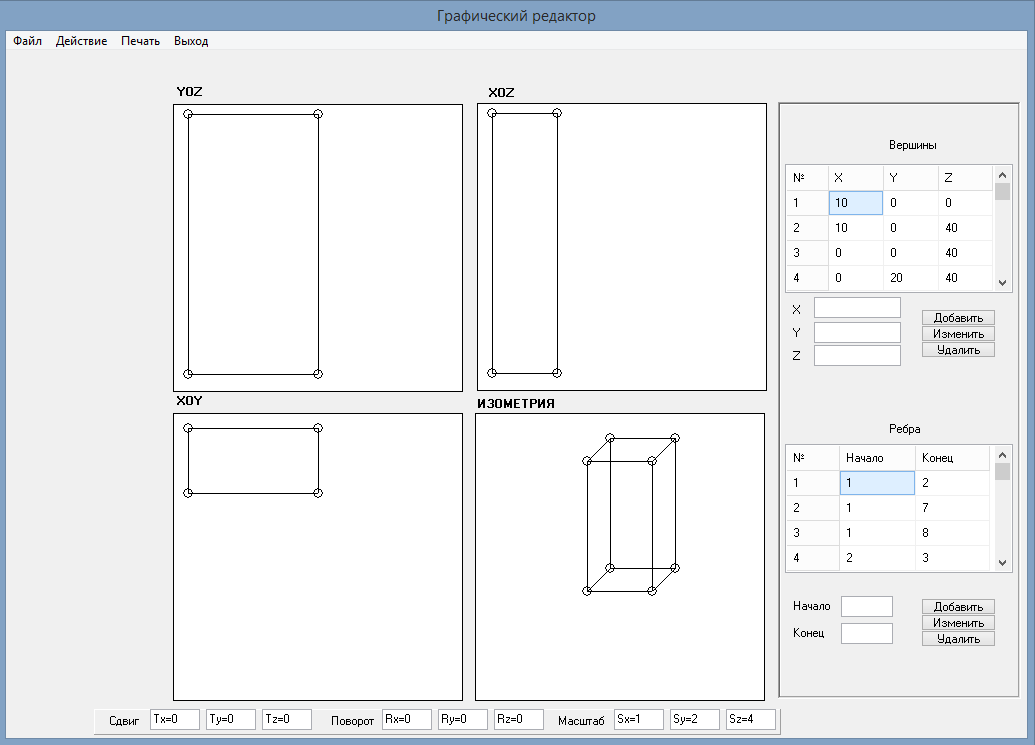
1. **Построение проекций и изометрии.**



*Рисунок 5. - Проекции и изометрия куба*

1. **Применение масштаба.**

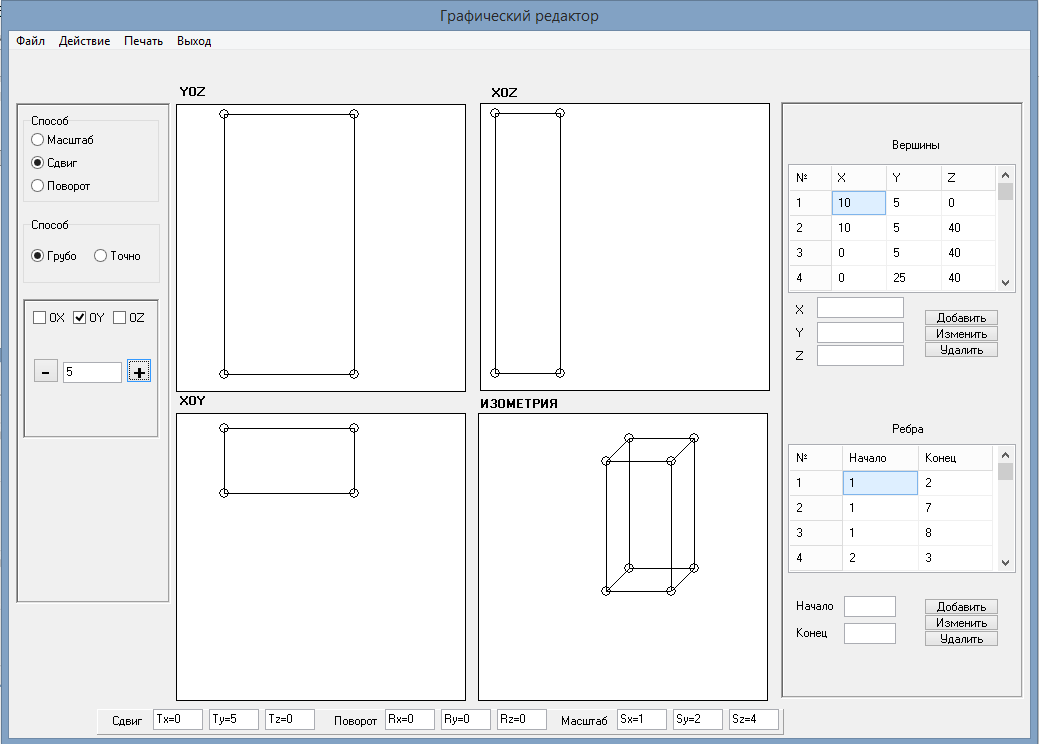
Применим масштаб для нашей фигуры с коэффициентами масштабирования по OX – 1, по OY – 2 и по OZ – 4 в режиме «Точно».



*Рисунок 6. - Применение масштаба*

1. **Сдвиг.**

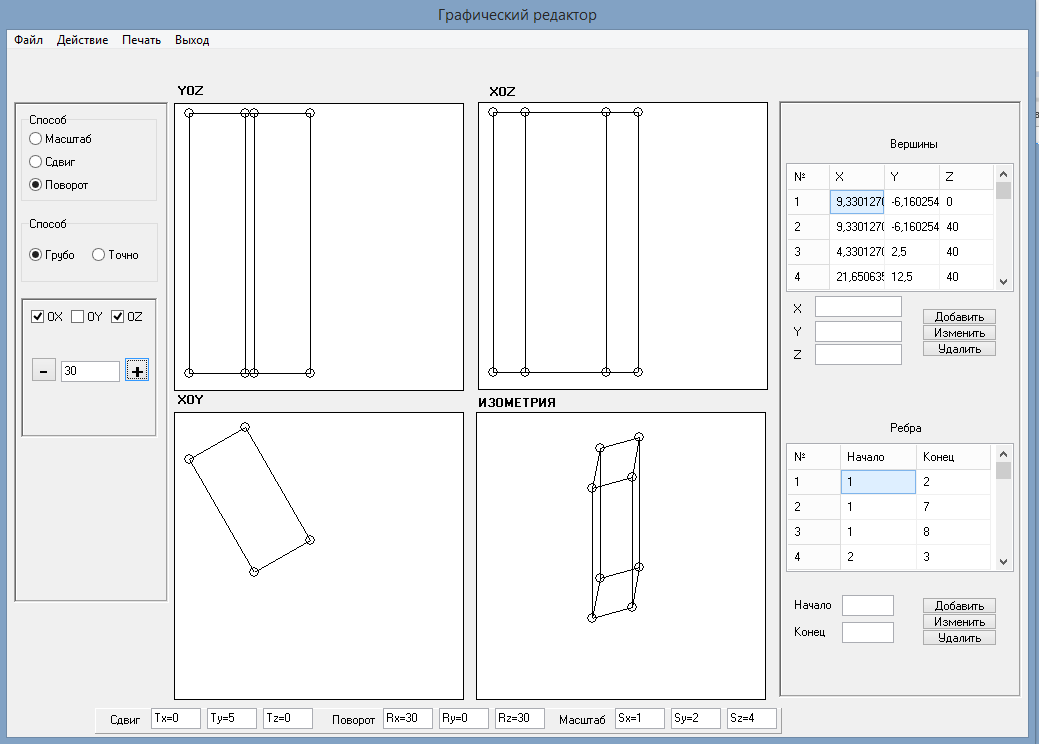
Применим сдвиг нашей фигуры по оси Oу с коэффициентом 5 в режиме «Грубо».



*Рисунок 7. - Сдвиг*

1. **Поворот.**

Сделаем сдвиг на 30 градусов вокруг Ox и Oz



*Рисунок 8. - Поворот*

**Вывод**

В ходе выполнения лабораторных работ по курсу «Компьютерная графика»:

* Изучен стандарт CUA, применяемый в большинстве пакетов САПР, и создан интерфейс, отвечающий требованиям данного международного стандарта.
* Изучен методы описания векторных объектов и реализован один из методов (список ребер).
* Изучено и реализовано нормирование координат.
* Изучены способы построения ортогональных проекций, реализованы наиболее часто используемые в технике частные случаи ортогональных параллельных проекций: вид сверху, вид спереди и вид сбоку.
* Изучены способы построения изометрии, реализован один из них (умножение вектора координат на матрицу изометрии).

**Приложение А**

**Исходный код**

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, Grids, Math,

Menus, ExtCtrls, ComCtrls, Buttons, TabNotBk;

type

TForm1 = class(TForm)

LblYOZ: TLabel;

LblXOZ: TLabel;

LblXOY: TLabel;

PnlEdit: TPanel;

LblPoints: TLabel;

LblPointX: TLabel;

LblPointY: TLabel;

LblPointZ: TLabel;

LblVectors: TLabel;

LblVectorBeg: TLabel;

LblVectorEnd: TLabel;

StringGridPoints: TStringGrid;

BtnAddPoint: TButton;

BtnChangePoint: TButton;

BtnDelPoint: TButton;

EditPointX: TEdit;

EditPointY: TEdit;

EditPointZ: TEdit;

StringGridVectors: TStringGrid;

BtnAddVector: TButton;

BtnChangeVector: TButton;

BtnDelVector: TButton;

EditVectorBeg: TEdit;

EditVectorEnd: TEdit;

PnlFileMenu: TPanel;

BtnOpen: TButton;

BtnSave: TButton;

BtnSaveAs: TButton;

BtnEdit: TButton;

BtnCreate: TButton;

MainMenu1: TMainMenu;

NFile: TMenuItem;

NMotion: TMenuItem;

NPrint: TMenuItem;

NExit: TMenuItem;

PrintDialog1: TPrintDialog;

OpenDialog1: TOpenDialog;

SaveDialog1: TSaveDialog;

PnlFactors: TPanel;

LblShift: TLabel;

LblRotation: TLabel;

LblScale: TLabel;

EditShiftX: TEdit;

EditShiftY: TEdit;

EditShiftZ: TEdit;

EditRotationX: TEdit;

EditRotationY: TEdit;

EditRotationZ: TEdit;

EditScaleX: TEdit;

EditScaleY: TEdit;

EditScaleZ: TEdit;

ImageXOZ: TImage;

ImageXOY: TImage;

ImageAll: TImage;

ImageYOZ: TImage;

LblAll: TLabel;

GrpBoxSave: TGroupBox;

LblSavePermission: TLabel;

BtnSaveYes: TButton;

BtnSaveNo: TButton;

GrpBoxExit: TGroupBox;

LblExitPermission: TLabel;

BtnExitYes: TButton;

BtnExitCancel: TButton;

PnlMotion: TPanel;

PnlRoughMotion: TPanel;

EditMotionRough: TEdit;

ChkBoxMotionX: TCheckBox;

ChkBoxMotionY: TCheckBox;

ChkBoxMotionZ: TCheckBox;

BtnMotionRoughPlus: TButton;

BtnMotionRoughMinus: TButton;

RadioGrpMode: TRadioGroup;

PnlMotionDue: TPanel;

LblMotionDueX: TLabel;

LblMotionDueY: TLabel;

LblMotionDueZ: TLabel;

EditMotionDueX: TEdit;

EditMotionDueY: TEdit;

EditMotionDueZ: TEdit;

BtnMotionDue: TButton;

RadioGrpMethod: TRadioGroup;

procedure BtnCreateClick(Sender: TObject);

procedure BtnOpenClick(Sender: TObject);

procedure BtnSaveClick(Sender: TObject);

procedure BtnSaveAsClick(Sender: TObject);

procedure BtnEditClick(Sender: TObject);

procedure NPrintClick(Sender: TObject);

procedure NFileClick(Sender: TObject);

procedure NMotionClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure StringGridPointsSelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);

procedure StringGridVectorsSelectCell(Sender: TObject; ACol, ARow: Integer; var CanSelect: Boolean);

procedure BtnAddPointClick(Sender: TObject);

procedure BtnChangePointClick(Sender: TObject);

procedure BtnDelPointClick(Sender: TObject);

procedure BtnAddVectorClick(Sender: TObject);

procedure BtnChangeVectorClick(Sender: TObject);

procedure BtnDelVectorClick(Sender: TObject);

procedure BtnSaveNoClick(Sender: TObject);

procedure BtnExitCancelClick(Sender: TObject);

procedure RadioGrpModeClick(Sender: TObject);

procedure BtnMotionRoughPlusClick(Sender: TObject);

procedure BtnMotionRoughMinusClick(Sender: TObject);

procedure NExitClick(Sender: TObject);

procedure BtnExitYesClick(Sender: TObject);

procedure EditMotionDueXKeyPress(Sender: TObject; var Key: Char);

procedure EditMotionDueYKeyPress(Sender: TObject; var Key: Char);

procedure EditMotionDueZKeyPress(Sender: TObject; var Key: Char);

procedure EditPointXKeyPress(Sender: TObject; var Key: Char);

procedure EditPointYKeyPress(Sender: TObject; var Key: Char);

procedure EditPointZKeyPress(Sender: TObject; var Key: Char);

procedure BtnMotionDueClick(Sender: TObject);

procedure RadioGrpMethodClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

type TMatrix44 = array[1..4,1..4] of real;

type TPoints = record

x,y,z: real;

end;

type TVectors = record

V1,V2: integer;

end;

type TAoTPoints = array of TPoints;

//my functions

function isCorrectPoint(ed: Tedit): boolean;

function isCorrectVector(ed: TEdit): boolean;

function isAlreadyExistPoint(x,y,z: real): boolean;

procedure SetPoint(number: integer; x,y,z: real);

function isAlreadyExistVector(v1,v2: integer): boolean;

procedure SetVector(numer,v1,v2: integer);

procedure ClearImage(Img:TImage);

procedure ClearAllImages;

Function MultiplyOnMatrix(Point: TPoints; Matrix44: TMatrix44): TPoints;

Function T(Dx,Dy,Dz: real): TMatrix44;

Function S(Sx,Sy,Sz: real): TMatrix44;

Function R(a,b,g: real): TMatrix44;

function normalization: TAoTPoints;

procedure PrintPoints;

procedure PrintVectors;

procedure PrintAll;

var

Form1: TForm1;

globalTx, globalTy, globalTz,

globalRx, globalRy, globalRz,

globalSx, globalSy, globalSz :real;

Points: TAoTPoints;

Vectors: array of TVectors;

FName:string;

rowOfStringGridPointsSelection, rowOfStringGridVectorsSelection: integer;

xmin,ymin,zmin:real;

kn: real;

const

d = 20;

implementation

{$R \*.dfm}

//--------------------------------------- ИНТЕРФЕЙС --------------------------

procedure TForm1.EditMotionDueXKeyPress(Sender: TObject; var Key: Char);

begin

if Key <> #08 then

try

strtofloat(EditMotionDueX.Text+Key);

except

Key:=#0;

end;

end;

procedure TForm1.EditMotionDueYKeyPress(Sender: TObject; var Key: Char);

begin

if Key <> #08 then

try

strtofloat(EditMotionDueY.Text+Key);

except

Key:=#0;

end;

end;

procedure TForm1.EditPointXKeyPress(Sender: TObject; var Key: Char);

begin

if Key <> #08 then

try

strtofloat(EditPointX.Text+Key);

except

Key:=#0;

end;

end;

procedure TForm1.EditMotionDueZKeyPress(Sender: TObject; var Key: Char);

begin

if Key <> #08 then

try

strtofloat(EditMotionDueZ.Text+Key);

except

Key:=#0;

end;

end;

procedure TForm1.EditPointYKeyPress(Sender: TObject; var Key: Char);

begin

if Key <> #08 then

try

strtofloat(EditPointY.Text+Key);

except

Key:=#0;

end;

end;

procedure TForm1.EditPointZKeyPress(Sender: TObject; var Key: Char);

begin

if Key <> #08 then

try

strtofloat(EditPointZ.Text+Key);

except

Key:=#0;

end;

end;

procedure TForm1.BtnEditClick(Sender: TObject);

begin

PnlEdit.Visible:=True;

PnlFileMenu.Visible:=False;

end;

procedure updatePnlFactors;

begin

Form1.EditShiftX.Text := 'Tx='+floattostr(globalTx);

Form1.EditShiftY.Text := 'Ty='+floattostr(globalTy);

Form1.EditShiftZ.Text := 'Tz='+floattostr(globalTz);

Form1.EditRotationX.Text:= 'Rx='+floattostr(globalRx);

Form1.EditRotationY.Text:= 'Ry='+floattostr(globalRy);

Form1.EditRotationZ.Text:= 'Rz='+floattostr(globalRz);

Form1.EditScaleX.Text := 'Sx='+floattostr(globalSx);

Form1.EditScaleY.Text := 'Sy='+floattostr(globalSy);

Form1.EditScaleZ.Text := 'Sz='+floattostr(globalSz);

end;

procedure TForm1.RadioGrpMethodClick(Sender: TObject);

begin

ChkBoxMotionX.Checked:=False;

ChkBoxMotionY.Checked:=False;

ChkBoxMotionZ.Checked:=False;

case RadioGrpMethod.ItemIndex of

0: begin

EditMotionRough.Text:='1';

editMotionDueX.Text:='1';

editMotionDueY.Text:='1';

editMotionDueZ.Text:='1';

BtnMotionRoughMinus.Visible:=false;

end;

1: begin

EditMotionRough.Text:='0';

editMotionDueX.Text:='0';

editMotionDueY.Text:='0';

editMotionDueZ.Text:='0';

BtnMotionRoughMinus.Visible:=true;

end;

2: begin

EditMotionRough.Text:='0';

editMotionDueX.Text:='0';

editMotionDueY.Text:='0';

editMotionDueZ.Text:='0';

BtnMotionRoughMinus.Visible:=true;

end;

end;

end;

procedure TForm1.RadioGrpModeClick(Sender: TObject);

begin

case RadioGrpMode.ItemIndex of

0: begin

PnlRoughMotion.Visible:=True;

PnlMotionDue.Visible:=False;

end;

1: begin

PnlRoughMotion.Visible:=False;

PnlMotionDue.Visible:=True;

end;

end;

end;

procedure TForm1.NFileClick(Sender: TObject);

begin

PnlFileMenu.Visible:= not PnlFileMenu.Visible;

end;

procedure TForm1.NMotionClick(Sender: TObject);

begin

PnlMotion.Visible:= not PnlMotion.Visible;

end;

procedure TForm1.NPrintClick(Sender: TObject);

begin

PnlMotion.Visible:=false;

PnlEdit.Visible:=false;

PnlFileMenu.Visible:=false;

Form1.PrintScale := poPrintToFit;

if PrintDialog1.execute then Form1.Print;

end;

procedure TForm1.NExitClick(Sender: TObject);

begin

PnlMotion.Visible:=false;

PnlEdit.Visible:=false;

PnlFileMenu.Visible:=false;

NFile.Enabled:=false;

NMotion.Enabled:=false;

NPrint.Enabled:=false;

NExit.Enabled:=false;

GrpBoxExit.Visible:=true;

end;

procedure TForm1.BtnSaveNoClick(Sender: TObject);

begin

form1.Close;

end;

procedure TForm1.BtnExitYesClick(Sender: TObject);

begin

GrpBoxSave.Visible:=true;

GrpBoxExit.Visible:=false;

PnlEdit.Visible:=false;

end;

procedure TForm1.BtnExitCancelClick(Sender: TObject);

begin

NFile.Enabled:=true;

NMotion.Enabled:=true;

NPrint.Enabled:=true;

NExit.Enabled:=true;

GrpBoxExit.Visible:=false;

end;

//-------------------------------------ФУНКЦИОНАЛ----------------------------------

//проверка ввода для вершин

function isCorrectPoint(ed: Tedit): boolean;

begin

//проверка на пустую строку

if (ed.Text = '') then begin

showmessage('Введите все значения!');

ed.SetFocus;

isCorrectPoint:=false;

exit;

end;

isCorrectPoint:=true;

end;

//проверка ввода для ребер

function isCorrectVector(ed: TEdit): boolean;

begin

//проверка на пустую строку

if (ed.Text = '') then begin

showmessage('Введите все значение');

ed.SetFocus;

isCorrectVector:=false;

exit;

end;

//если не существует вершин с заданным номером

if (StrToInt(ed.Text) > Length(Points)-1) or (StrToInt(ed.Text) <= 0) then begin

showmessage('Вершины с таким номером не существует');

Ed.SetFocus;

isCorrectVector:=false;

exit;

end;

isCorrectVector:=true;

end;

function isAlreadyExistPoint(x,y,z: real): boolean;

var

i:integer;

NumerOfPoints: integer;

begin

NumerOfPoints:=Length(Points)-1;

for i:=1 to NumerOfPoints do

if (Points[i].x = x) and (Points[i].y = y) and (Points[i].z = z) then begin

showmessage('Вершина уже существует');

isAlreadyExistPoint := true;

exit;//return

end;

isAlreadyExistPoint := false;//return

end;

procedure SetPoint(number: integer; x,y,z: real);

begin

Points[Number].x := x;

Points[Number].y := y;

Points[Number].z := z;

form1.StringGridPoints.Cells[0,Number]:=inttostr(Number);

form1.StringGridPoints.Cells[1,Number]:=floattostr(x);

form1.StringGridPoints.Cells[2,Number]:=floattostr(y);

form1.StringGridPoints.Cells[3,Number]:=floattostr(z);

form1.EditPointX.Clear;

form1.EditPointY.Clear;

form1.EditPointZ.Clear;

end;

function isAlreadyExistVector(v1,v2: integer): boolean;

var

i:integer;

NumerOfVectors: integer;

begin

NumerOfVectors := Length(Vectors)-1;

for i:=1 to NumerOfVectors do

if ( (Vectors[i].V1=v1)and(Vectors[i].V2=v2) ) or ( (Vectors[i].V1=v2)and(Vectors[i].V2=v1) ) then begin

showmessage('Ребро уже существует');

isAlreadyExistVector := true;

exit; //return

end;

isAlreadyExistVector := false; //return

end;

procedure SetVector(numer,v1,v2: integer);

begin

Vectors[numer].v1:=v1;

Vectors[numer].v2:=v2;

form1.StringGridVectors.Cells[0,numer]:=inttostr(numer);

form1.StringGridVectors.Cells[1,numer]:=inttostr(v1);

form1.StringGridVectors.Cells[2,numer]:=inttostr(v2);

form1.EditVectorBeg.Clear;

form1.EditVectorEnd.Clear;

end;

procedure ClearImage(Img:TImage);

begin

with Img do begin

Canvas.Brush.Color:=clWhite;

Canvas.Rectangle(0,0,Width,Height)

end;

end;

procedure ClearAllImages;

begin

ClearImage(form1.ImageXOZ);

ClearImage(form1.ImageXOY);

ClearImage(form1.ImageYOZ);

ClearImage(form1.ImageAll);

end;

//умножение матриц

Function MultiplyOnMatrix(Point: TPoints; Matrix44: TMatrix44): TPoints;

begin

MultiplyOnMatrix.X:= Point.X\*Matrix44[1,1]+Point.Y\*Matrix44[1,2]+Point.Z\*Matrix44[1,3]+Matrix44[1,4];

MultiplyOnMatrix.Y:= Point.X\*Matrix44[2,1]+Point.Y\*Matrix44[2,2]+Point.Z\*Matrix44[2,3]+Matrix44[2,4];

MultiplyOnMatrix.Z:= Point.X\*Matrix44[3,1]+Point.Y\*Matrix44[3,2]+Point.Z\*Matrix44[3,3]+Matrix44[3,4];

end;

//матрица сдвига

Function T(Dx,Dy,Dz: real): TMatrix44;

var

tmp: TMatrix44;

begin

tmp[1,1]:=1; tmp[2,1]:=0; tmp[3,1]:=0; tmp[4,1]:=0;

tmp[1,2]:=0; tmp[2,2]:=1; tmp[3,2]:=0; tmp[4,2]:=0;

tmp[1,3]:=0; tmp[2,3]:=0; tmp[3,3]:=1; tmp[4,3]:=0;

tmp[1,4]:=Dx; tmp[2,4]:=Dy; tmp[3,4]:=Dz; tmp[4,4]:=1;

T:=tmp;

end;

//матрица масштаба

Function S(Sx,Sy,Sz: real): TMatrix44;

var

tmp: TMatrix44;

begin

tmp[1,1]:=Sx; tmp[2,1]:=0; tmp[3,1]:=0; tmp[4,1]:=0;

tmp[1,2]:=0; tmp[2,2]:=Sy; tmp[3,2]:=0; tmp[4,2]:=0;

tmp[1,3]:=0; tmp[2,3]:=0; tmp[3,3]:=Sz; tmp[4,3]:=0;

tmp[1,4]:=0; tmp[2,4]:=0; tmp[3,4]:=0; tmp[4,4]:=1;

S:=tmp;

end;

//матрица поворота

Function R(a,b,g: real): TMatrix44;

var

tmp: TMatrix44;

begin

a:= a/180\*PI;

b:= b/180\*PI;

g:= g/180\*PI;

tmp[1,1]:=cos(a)\*cos(g)-sin(a)\*cos(b)\*sin(g); tmp[2,1]:=-cos(a)\*sin(g)-sin(a)\*cos(b)\*cos(g); tmp[3,1]:=sin(a)\*sin(b); tmp[4,1]:=0;

tmp[1,2]:=sin(a)\*cos(g)+cos(a)\*cos(b)\*sin(g); tmp[2,2]:=-sin(a)\*sin(g)+cos(a)\*cos(b)\*cos(g); tmp[3,2]:=-cos(a)\*sin(b); tmp[4,2]:=0;

tmp[1,3]:=sin(b)\*sin(g); tmp[2,3]:=sin(b)\*cos(g); tmp[3,3]:=cos(b); tmp[4,3]:=0;

tmp[1,4]:=0; tmp[2,4]:=0; tmp[3,4]:=0; tmp[4,4]:=1;

R:=tmp;

end;

//нормирование

function normalization: TAoTPoints;

var

i:integer;

NumerOfPoints: integer;

normTmp: TAoTPoints;

flagNeedNormalization: boolean;

begin

NumerOfPoints:=Length(Points)-1;

SetLength(normTmp,NumerOfPoints+1);

flagNeedNormalization:= false;

// проверка необходимости нормализации

for i := 1 to NumerOfPoints do begin

normTmp[i]:= MultiplyOnMatrix( MultiplyOnMatrix(Points[i], T(-xmin,-ymin,-zmin)), S(kn,kn,kn) );

if not ( (0<=normTmp[i].x)and(normTmp[i].x<=1) ) or not ( (0<=normTmp[i].y)and(normTmp[i].y<=1) ) or not ( (0<=normTmp[i].z)and(normTmp[i].z<=1) ) then begin

flagNeedNormalization:= true;

break;

end;

end;

// если нормализация нужна

if flagNeedNormalization then begin

kn:=-1;

xmin:=Points[1].x;

ymin:=Points[1].y;

zmin:=Points[1].z;

//находим мин. - нижний левый угол

for i:=2 to NumerOfPoints do begin

if Points[i].x<xmin then xmin:=Points[i].x;

if Points[i].y<ymin then ymin:=Points[i].y;

if Points[i].z<zmin then zmin:=Points[i].z;

end;

//находим макс. - коэффициент нормировки

for i:=1 to NumerOfPoints do begin

if Points[i].x-xmin>kn then kn:=Points[i].x-xmin;

if Points[i].y-ymin>kn then kn:=Points[i].y-ymin;

if Points[i].z-zmin>kn then kn:=Points[i].z-zmin;

end;

if kn<>0 then kn:=1/kn;

for i := 1 to NumerOfPoints do begin

normTmp[i]:= MultiplyOnMatrix( MultiplyOnMatrix(Points[i], T(-xmin,-ymin,-zmin)), S(kn,kn,kn) );

end;

end;

normalization:=normTmp;

end;

/// вывод точек

procedure PrintPoints;

const

k = 260;

d = 15;

var

NumerOfPoints: integer;

normPoints: TAoTPoints;

i: Integer;

x,y,z: integer;

begin

NumerOfPoints:=Length(Points)-1;

normPoints:=normalization;

for i := 1 to NumerOfPoints do begin

x:=round( normPoints[i].x\*k + d);

y:=round( normPoints[i].y\*k + d);

z:=270 - round( normPoints[i].z\*k);

Form1.ImageYOZ.Canvas.Ellipse( y-5, z-5, y+5, z+5 );

Form1.ImageXOY.Canvas.Ellipse( y-5, x-5, y+5, x+5 );

Form1.ImageXOZ.Canvas.Ellipse( x-5, z-5, x+5, z+5 );

//изометрия

{tmp:= MultiplyOnMatrix(normPoints[i],R(35,45,0));

x:= round(tmp.x\*k/1.6 + d);

y:= 215 - round(tmp.y\*k/1.6);}

x:= round( 120 + (normPoints[i].y-normPoints[i].x\*sin(PI/4))\*k/2 + d );

y:= round( 140 - (normPoints[i].z-normPoints[i].x\*sin(PI/4))\*k/2 + d );

form1.ImageAll.Canvas.Ellipse( x-5, y-5, x+5, y+5 );

end;

end;

//вывод отрезков

procedure PrintVectors;

const

k = 260;

d = 15;

var

NumerOfVectors: integer;

normPoints: TAoTPoints;

i: Integer;

x1,y1,z1,x2,y2,z2: integer;

begin

NumerOfVectors:=Length(Vectors)-1;

normPoints:=normalization;

for i := 1 to NumerOfVectors do begin

x1:=round( normPoints[Vectors[i].V1].x\*k + d);

y1:=round( normPoints[Vectors[i].V1].y\*k + d);

z1:=270 - round( normPoints[Vectors[i].V1].z\*k);

x2:=round( normPoints[Vectors[i].V2].x\*k + d);

y2:=round( normPoints[Vectors[i].V2].y\*k + d);

z2:=270 - round( normPoints[Vectors[i].V2].z\*k);

form1.ImageYOZ.Canvas.MoveTo( y1, z1 );

Form1.ImageYOZ.Canvas.LineTo( y2, z2 );

form1.ImageXOY.Canvas.MoveTo( y1, x1 );

Form1.ImageXOY.Canvas.LineTo( y2, x2 );

form1.ImageXOZ.Canvas.MoveTo( x1, z1 );

Form1.ImageXOZ.Canvas.LineTo( x2, z2 );

//изометрия

{tmp:= MultiplyOnMatrix(normPoints[Vectors[i].V1],R(35,45,0));

x1:= round(tmp.x\*k/1.6 + d);

y1:= 215 - round(tmp.y\*k/1.6);}

x1:= round( 120 + (normPoints[Vectors[i].V1].y-normPoints[Vectors[i].V1].x\*sin(PI/4))\*k/2 + d );

y1:= round( 140 - (normPoints[Vectors[i].V1].z-normPoints[Vectors[i].V1].x\*sin(PI/4))\*k/2 + d );

form1.ImageAll.Canvas.MoveTo( x1, y1 );

{tmp:= MultiplyOnMatrix(normPoints[Vectors[i].V2],R(35,45,0));

x2:= round(tmp.x\*k/1.6 + d);

y2:= 215 - round(tmp.y\*k/1.6); }

x2:= round( 120 + (normPoints[Vectors[i].V2].y-normPoints[Vectors[i].V2].x\*sin(PI/4))\*k/2 + d );

y2:= round( 140 - (normPoints[Vectors[i].V2].z-normPoints[Vectors[i].V2].x\*sin(PI/4))\*k/2 + d );

form1.ImageAll.Canvas.LineTo ( x2, y2);

end;

end;

procedure PrintAll;

begin

ClearAllImages;

PrintPoints;

PrintVectors;

end;

//------------------------------------------------------------------------------

//----------------------------------------ДОБАВЛЕНИЕ/УДАЛЕНИЕ--------------------------------

procedure TForm1.StringGridPointsSelectCell(Sender: TObject; ACol,

ARow: Integer; var CanSelect: Boolean);

begin

rowOfStringGridPointsSelection:=ARow;

EditPointX.Text:=StringGridPoints.Cells[1,Arow];

EditPointY.Text:=StringGridPoints.Cells[2,Arow];

EditPointZ.Text:=StringGridPoints.Cells[3,Arow];

end;

procedure TForm1.StringGridVectorsSelectCell(Sender: TObject; ACol,

ARow: Integer; var CanSelect: Boolean);

begin

rowOfStringGridVectorsSelection:=ARow;

EditVectorBeg.Text:=StringGridVectors.Cells[1,Arow];

EditVectorEnd.Text:=StringGridVectors.Cells[2,Arow];

end;

procedure TForm1.BtnAddPointClick(Sender: TObject); {Добавить вершину}

var

x,y,z:real;

NumerOfPoints: integer;

begin

if (isCorrectPoint(EditPointX))and(isCorrectPoint(EditPointY))and(isCorrectPoint(EditPointZ))then begin

x:=strtofloat(EditPointX.Text);

y:=strtofloat(EditPointY.Text);

z:=strtofloat(EditPointZ.Text);

if isAlreadyExistPoint(x,y,z) then exit;

//добавление вершины

NumerOfPoints:=Length(Points)-1;

NumerOfPoints := NumerOfPoints + 1;

SetLength(Points,NumerOfPoints+1);

StringGridPoints.RowCount:=StringGridPoints.RowCount+1;

SetPoint(NumerOfPoints,x,y,z);

//рисование

PrintAll;

end;

end;

procedure TForm1.BtnChangePointClick(Sender: TObject);

var

x,y,z:real;

n: integer;

NumerOfPoints: integer;

begin

NumerOfPoints:=Length(Points)-1;

n:=rowOfStringGridPointsSelection;

if (n=0) or (n>NumerOfPoints) then begin

showmessage('Сначала выберете вершину!');

exit;

end;

if (isCorrectPoint(EditPointX))and(isCorrectPoint(EditPointY))and(isCorrectPoint(EditPointZ))then begin

x:=strtofloat(EditPointX.Text);

y:=strtofloat(EditPointY.Text);

z:=strtofloat(EditPointZ.Text);

if isAlreadyExistPoint(x,y,z) then exit;

SetPoint(rowOfStringGridPointsSelection,x,y,z);

end;

//рисование

PrintAll;

end;

//удаление вершины

procedure TForm1.BtnDelPointClick(Sender: TObject);

var

n,i,tmp:integer;

NumerOfPoints, NumerOfVectors: integer;

begin

NumerOfPoints:=Length(Points)-1;

n:=rowOfStringGridPointsSelection;

if NumerOfPoints=0 then begin

showmessage('Нечего удалять!');

exit;

end;

if (n=0) or (n>NumerOfPoints) then begin

showmessage('Сначала выберете вершину!');

exit;

end;

for i:=n+1 to NumerOfPoints do

SetPoint(i-1,Points[i].x,Points[i].y,Points[i].z);

NumerOfPoints:=NumerOfPoints-1;

SetLength(Points, NumerOfPoints+1);

StringGridPoints.RowCount:=StringGridPoints.RowCount-1;

StringGridPoints.Rows[StringGridPoints.RowCount-1].Clear;

EditPointX.Clear; EditPointY.Clear; EditPointZ.Clear;

rowOfStringGridPointsSelection:=0;

//удаление ребер, содержавших точку

//и -1 для тех, что содержали номера выше n

//(в n - номер удаленного ребра)

tmp:=rowOfStringGridVectorsSelection;

NumerOfVectors:=Length(Vectors)-1;

i:=1;

while i<=NumerOfVectors do begin

if (Vectors[i].V1=n)or(Vectors[i].V2=n) then begin

// имитируем нажатие "удалить вектор"

rowOfStringGridVectorsSelection:=i;

BtnDelVectorClick(Sender);

NumerOfVectors:=NumerOfVectors-1;

continue;

end;

if (Vectors[i].V1>n) then Vectors[i].V1:=Vectors[i].V1-1;

if (Vectors[i].V2>n) then Vectors[i].V2:=Vectors[i].V2-1;

i:=i+1;

end;

rowOfStringGridVectorsSelection:=tmp;

//рисование

PrintAll;

end;

//ребра

procedure TForm1.BtnAddVectorClick(Sender: TObject);

var

v1,v2:integer;

NumerOfVectors: integer;

begin

NumerOfVectors:=Length(Vectors)-1;

if (isCorrectVector(EditVectorBeg))and(isCorrectVector(EditVectorEnd)) then begin

v1:=strtoint(EditVectorBeg.Text);

v2:=strtoint(EditVectorEnd.Text);

if v1=v2 then begin

showmessage('Начало и конец ребра не могут совпадать');

exit;

end;

if isAlreadyExistVector(v1,v2) then exit;

//добавление ребра

NumerOfVectors:=NumerOfVectors+1;

SetLength(Vectors,NumerOfVectors+1);

StringGridVectors.RowCount:=StringGridVectors.RowCount+1;

SetVector(NumerOfVectors,v1,v2);

end;

//рисование

PrintAll;

end;

procedure TForm1.BtnChangeVectorClick(Sender: TObject); //изменение ребра

var

v1,v2: integer;

NumerOfVectors: integer;

n: integer;

begin

n:=rowOfStringGridVectorsSelection;

NumerOfVectors:=Length(Vectors)-1;

if (n=0) or (n>NumerOfVectors) then begin

showmessage('Сначала выберете отрезок!');

exit;

end;

if (isCorrectVector(EditVectorBeg))and(isCorrectVector(EditVectorEnd)) then begin

v1:=strtoint(EditVectorBeg.Text);

v2:=strtoint(EditVectorEnd.Text);

//если начало и конец ребра равны

if v1=v2 then begin

showmessage('Начало и конец ребра не могут совпадать');

exit;

end;

if isAlreadyExistVector(v1,v2) then exit;

SetVector(n,v1,v2);

//рисование

PrintAll;

end;

end;

procedure TForm1.BtnDelVectorClick(Sender: TObject); //удаление ребра

var

i:integer;

NumerOfVectors: integer;

n: integer;

begin

n:=rowOfStringGridVectorsSelection;

NumerOfVectors:=Length(Vectors)-1;

if NumerOfVectors = 0 then begin

showmessage('Нет ребер!');

exit;

end;

if (n=0) or (n>NumerOfVectors) then begin

showmessage('Сначала выберете отрезок!');

exit;

end;

for i:=n+1 to NumerOfVectors do

SetVector(i-1,Vectors[i].V1,Vectors[i].V2);

NumerOfVectors:=NumerOfVectors-1;

SetLength(Vectors,NumerOfVectors+1);

StringGridVectors.RowCount:=StringGridVectors.RowCount-1;

StringGridVectors.Rows[StringGridVectors.RowCount-1].Clear;

EditVectorBeg.Clear; EditVectorEnd.Clear;

rowOfStringGridVectorsSelection:=0;

//рисование

PrintAll;

end;

//---------------------------------ИНИЦИАЛИЗАЦИЯ-------------------

procedure TForm1.FormCreate(Sender: TObject);

begin

with StringGridPoints do begin

Cells[0,0]:='№'; Cells[1,0]:='X'; Cells[2,0]:='Y'; Cells[3,0]:='Z';

end;

with StringGridVectors do begin

Cells[0,0]:='№'; Cells[1,0]:='Начало'; Cells[2,0]:='Конец';

end;

BtnCreateClick(Sender);

end;

procedure TForm1.BtnCreateClick(Sender: TObject);

begin

//коэффициенты нормализации

kn:=1; xmin:=0; ymin:=0; zmin:=0;

globalTx:=0; globalTy:=0; globalTz:=0;

globalRx:=0; globalRy:=0; globalRz:=0;

globalSx:=1; globalSy:=1; globalSz:=1;

updatePnlFactors;

rowOfStringGridVectorsSelection:=0;

rowOfStringGridPointsSelection:=0;

SetLength(Vectors,1);

SetLength(Points,1);

FName:='';

ClearAllImages;

StringGridPoints.RowCount:=2;

StringGridPoints.Rows[1].Clear;

StringGridVectors.RowCount:=2;

StringGridVectors.Rows[1].Clear;

PnlEdit.Visible:=True;

PnlFileMenu.Visible:=False;

PnlMotion.Visible:=false;

end;

//-----------------------------SAVE/LOAD----------------------------------------

procedure TForm1.BtnSaveClick(Sender: TObject);

var

i,n:integer;

fin: textfile;

begin

if FName='' then BtnSaveAsClick(Sender)

else begin

AssignFile(fin,FName);

Rewrite(fin);

//вершинs

n:=Length(Points)-1;

writeln(fin,inttostr(n));

for i := 1 to n do writeln(fin,floattostr(Points[i].x),' ',floattostr(Points[i].y),' ',floattostr(Points[i].z));

//vectors

n:=Length(Vectors)-1;

writeln(fin,inttostr(n));

for i := 1 to n do writeln(fin,inttostr(Vectors[i].v1),' ',inttostr(Vectors[i].V2));

CloseFile(fin);

PnlFileMenu.Visible:=False;

end;

end;

procedure TForm1.BtnSaveAsClick(Sender: TObject);

begin

if SaveDialog1.Execute then begin

FName:=SaveDialog1.FileName;

BtnSaveClick(Sender);

end;

end;

procedure TForm1.BtnOpenClick(Sender: TObject); //открыть

var

i,j: integer;

fin: textfile;

n: integer;

x,y,z: real;

strx,stry,strz,tmp: string;

v1,v2: integer;

begin

PnlEdit.Visible:=True;

PnlFileMenu.Visible:=False;

PnlMotion.Visible:=false;

if OpenDialog1.Execute then begin

BtnCreateClick(Sender);

FName:=OpenDialog1.FileName;

AssignFile(fin,FName);

Reset(fin);

//points

readln(fin,n);

SetLength(Points, n+1);

StringGridPoints.RowCount:=n+2;

for i := 1 to n do begin

readln(fin,tmp);

j:=1;

strx:='';

repeat

strx:=strx+tmp[j];

j:=j+1;

until tmp[j]=' ';

j:=j+1;

stry:='';

repeat

stry:=stry+tmp[j];

j:=j+1;

until tmp[j]=' ';

j:=j+1;

strz:='';

repeat

strz:=strz+tmp[j];

j:=j+1;

until j=length(tmp)+1;

x:=strtofloat(strx);

y:=strtofloat(stry);

z:=strtofloat(strz);

SetPoint(i, x,y,z);

end;

//vectors

readln(fin,n);

SetLength(Vectors, n+1);

StringGridVectors.RowCount:=n+2;

for i := 1 to n do begin

readln(fin,v1,v2);

SetVector(i, v1,v2);

end;

CloseFile(fin);

//рисование

PrintAll;

end;

end;

// точное действие

procedure TForm1.BtnMotionDueClick(Sender: TObject);

var

NumerOfPoints: integer;

i: integer;

rx,ry,rz: real;

tmp: TPoints;

begin

// проверка на пустоту edit

if not isCorrectPoint(EditMotionDueX) or not isCorrectPoint(EditMotionDueY) or not isCorrectPoint(EditMotionDueZ) then exit;

rx:= strtofloat(EditMotionDueX.Text);

ry:= strtofloat(EditMotionDueY.Text);

rz:= strtofloat(EditMotionDueZ.Text);

NumerOfPoints:=Length(Points)-1;

case RadioGrpMethod.ItemIndex of

0://масштаб

begin

globalSx:=globalSx\*rx;

globalSy:=globalSy\*ry;

globalSz:=globalSz\*rz;

if (rx\*ry\*rz = 0) then begin

ShowMessage('0 - недопустимый коэффициент масштабирования');

exit;

end;

for i := 1 to NumerOfPoints do begin

tmp:= MultiplyOnMatrix( Points[i], S(rx,ry,rz) );

SetPoint(i, tmp.x, tmp.y, tmp.z);

end;

//перерисовка

PrintAll;

//прочистка edit

EditMotionDueX.Text:='1';

EditMotionDueY.Text:='1';

EditMotionDueZ.Text:='1';

EditMotionRough.Text:='1';

end;

1://сдвиг

begin

globalTx:=globalTx+rx;

globalTy:=globalTy+ry;

globalTz:=globalTz+rz;

for i := 1 to NumerOfPoints do begin

tmp:= MultiplyOnMatrix( Points[i], T(rx,ry,rz) );

SetPoint(i, tmp.x, tmp.y, tmp.z);

end;

//перерисовка

PrintAll;

//прочистка edit

EditMotionDueX.Text:='0';

EditMotionDueY.Text:='0';

EditMotionDueZ.Text:='0';

EditMotionRough.Text:='0';

end;

2://поворот

begin

globalRx:=globalRx+rx;

globalRy:=globalRy+ry;

globalRz:=globalRz+rz;

for i := 1 to NumerOfPoints do begin

tmp:= MultiplyOnMatrix( Points[i], R(rx,ry,rz) );

SetPoint(i, tmp.x, tmp.y, tmp.z);

end;

//перерисовка

PrintAll;

//прочистка edit

EditMotionDueX.Text:='0';

EditMotionDueY.Text:='0';

EditMotionDueZ.Text:='0';

EditMotionRough.Text:='0';

end;

end;

updatePnlFactors;

PnlMotion.Visible:=false;

end;

//грубый +

procedure TForm1.BtnMotionRoughPlusClick(Sender: TObject);

var

NumerOfPoints: integer;

i: integer;

rx,ry,rz: real;

tmp: TPoints;

begin

// проверка на пустоту edit

if not isCorrectPoint(EditMotionRough) then exit;

NumerOfPoints:=Length(Points)-1;

case RadioGrpMethod.ItemIndex of

0://масштаб

begin

if ChkBoxMotionX.Checked then rx:= strtofloat(EditMotionRough.Text)

else rx:=1;

if ChkBoxMotionY.Checked then ry:= strtofloat(EditMotionRough.Text)

else ry:=1;

if ChkBoxMotionZ.Checked then rz:= strtofloat(EditMotionRough.Text)

else rz:=1;

if (rx\*ry\*rz = 0) then begin

ShowMessage('0 - недопустимый коэффициент масштабирования');

exit;

end;

globalSx:=globalSx\*rx;

globalSy:=globalSy\*ry;

globalSz:=globalSz\*rz;

for i := 1 to NumerOfPoints do begin

tmp:= MultiplyOnMatrix( Points[i], S(rx,ry,rz) );

SetPoint(i, tmp.x, tmp.y, tmp.z);

end;

//перерисовка

PrintAll;

//прочистка edit

EditMotionDueX.Text:='1';

EditMotionDueY.Text:='1';

EditMotionDueZ.Text:='1';

//EditMotionRough.Text:='1';

end;

1://сдвиг

begin

if ChkBoxMotionX.Checked then rx:= strtofloat(EditMotionRough.Text)

else rx:=0;

if ChkBoxMotionY.Checked then ry:= strtofloat(EditMotionRough.Text)

else ry:=0;

if ChkBoxMotionZ.Checked then rz:= strtofloat(EditMotionRough.Text)

else rz:=0;

globalTx:=globalTx+rx;

globalTy:=globalTy+ry;

globalTz:=globalTz+rz;

for i := 1 to NumerOfPoints do begin

tmp:= MultiplyOnMatrix( Points[i], T(rx,ry,rz) );

SetPoint(i, tmp.x, tmp.y, tmp.z);

end;

//перерисовка

PrintAll;

//прочистка edit

EditMotionDueX.Text:='0';

EditMotionDueY.Text:='0';

EditMotionDueZ.Text:='0';

//EditMotionRough.Text:='0';

end;

2://поворот

begin

if ChkBoxMotionX.Checked then rx:= strtofloat(EditMotionRough.Text)

else rx:=0;

if ChkBoxMotionY.Checked then ry:= strtofloat(EditMotionRough.Text)

else ry:=0;

if ChkBoxMotionZ.Checked then rz:= strtofloat(EditMotionRough.Text)

else rz:=0;

globalRx:=globalRx+rx;

globalRy:=globalRy+ry;

globalRz:=globalRz+rz;

for i := 1 to NumerOfPoints do begin

tmp:= MultiplyOnMatrix( Points[i], R(rx,ry,rz) );

SetPoint(i, tmp.x, tmp.y, tmp.z);

end;

//перерисовка

PrintAll;

//прочистка edit

EditMotionDueX.Text:='0';

EditMotionDueY.Text:='0';

EditMotionDueZ.Text:='0';

//EditMotionRough.Text:='0';

end;

end;

updatePnlFactors;

end;

//грубый -

procedure TForm1.BtnMotionRoughMinusClick(Sender: TObject);

var

NumerOfPoints: integer;

i: integer;

rx,ry,rz: real;

tmp: TPoints;

begin

// проверка на пустоту edit

if not isCorrectPoint(EditMotionRough) then exit;

NumerOfPoints:=Length(Points)-1;

case RadioGrpMethod.ItemIndex of

1://сдвиг

begin

if ChkBoxMotionX.Checked then rx:= strtofloat(EditMotionRough.Text)

else rx:=0;

if ChkBoxMotionY.Checked then ry:= strtofloat(EditMotionRough.Text)

else ry:=0;

if ChkBoxMotionZ.Checked then rz:= strtofloat(EditMotionRough.Text)

else rz:=0;

globalTx:=globalTx-rx;

globalTy:=globalTy-ry;

globalTz:=globalTz-rz;

for i := 1 to NumerOfPoints do begin

tmp:= MultiplyOnMatrix( Points[i], T(-rx,-ry,-rz) );

SetPoint(i, tmp.x, tmp.y, tmp.z);

end;

//перерисовка

PrintAll;

//прочистка edit

EditMotionDueX.Text:='0';

EditMotionDueY.Text:='0';

EditMotionDueZ.Text:='0';

//EditMotionRough.Text:='0';

end;

2://поворот

begin

if ChkBoxMotionX.Checked then rx:= strtofloat(EditMotionRough.Text)

else rx:=0;

if ChkBoxMotionY.Checked then ry:= strtofloat(EditMotionRough.Text)

else ry:=0;

if ChkBoxMotionZ.Checked then rz:= strtofloat(EditMotionRough.Text)

else rz:=0;

globalRx:=globalRx-rx;

globalRy:=globalRy-ry;

globalRz:=globalRz-rz;

for i := 1 to NumerOfPoints do begin

tmp:= MultiplyOnMatrix( Points[i], R(-rx,-ry,-rz) );

SetPoint(i, tmp.x, tmp.y, tmp.z);

end;

//перерисовка

PrintAll;

//прочистка edit

EditMotionDueX.Text:='0';

EditMotionDueY.Text:='0';

EditMotionDueZ.Text:='0';

//EditMotionRough.Text:='0';

end;

end;

updatePnlFactors;

end;

end.