МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Вятский государственный университет»**

**(ФГБОУ ВО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Аффинные преобразования в пространстве

Отчет по лабораторной работе №10 дисциплины

«Компьютерная графика»

Выполнил студент группы ИВТ-21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Седов М.Д. /

Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Клюкин В.Л./

Киров 2018

1 Цель работы: Закрепить лекционный материал по изучению материала одноименной темы, реализовав матрицы переноса, масштабирования, отражения и вращения применительно к координатам описанной в программе объемной фигуры (многогранника) с целью демонстрации движения и преобразования формы этой фигуры в пространстве.

2 Задание на лабораторную работу:

1. Описывающую многогранник (куб) в приборной системе координат.
2. Смещающую его на n пикселов вправо, m - вниз, p - вглубь.
3. Зеркально отражающую относительно плоскостей координат.
4. Растягивающую (сжимающую) его вдоль координатных осей относительно некоторой заданной точки.
5. Вращающую его относительно линии, проходящей через начало координат (относительно координатных осей, диагонали многогранника).

3 Словесное описание

Преобразования осуществляются по следующим формулам:

1.Сдвиг - перенос точки (x,y,z) на m единиц по координате x, на n - по y, на l единиц- по z.: 

2.Повоpот точки (x,y,z) вокpуг оси абсцисс на угол δ: 

вокруг оси ординат на угол λ: 

вокруг оси аппликат на угол μ: 

вокруг оси, проходящей через начало координат на угол ν

.

Здесь n1,n2,n3 - направляющие косинусы вращения. Если ось вращения проходит через начало координат и точку (x1,y1, z1) то



Вращение вокруг произвольной оси осуществляется переносом ее вместе с изображением в начало координат, вращением вокруг перенесенной оси и обратным переносом изображения в исходное положение. Итоговая матрица будет определена умножением соответствующих простых матриц.

3. Симметрия относительно координатных плоскостей осуществляет зеркальное отображение 3D-изображения.

Относительно плоскости XOY 

Плоскости XOZ - -

Плоскости YOZ :  -

4.Масштабирование осуществляется диагональными элементами матрицы преобразования.

Относительно начала координат: ,

где kx, ky, kz - коэффициенты искажения вдоль осей ox,oy,oz, соответственно.

Относительно произвольного центра с координатами (m,n,l):

 -

4 Схема алгоритма



Рисунок 1 – Схема алгоритма

5 Результат работы

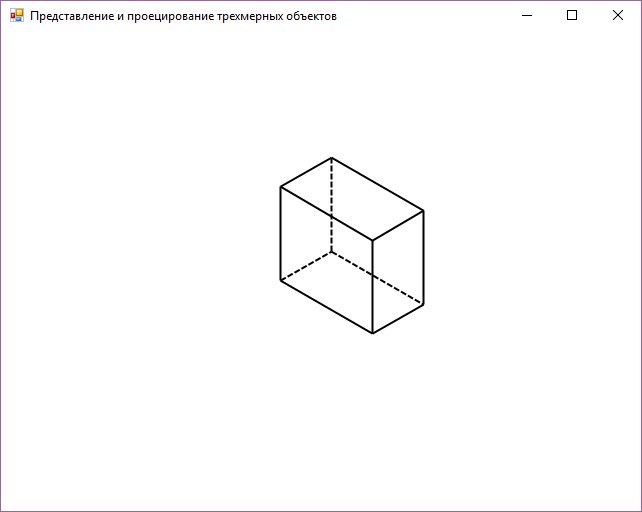


Рисунок 2 – Результат работы программы

6 Выводы: В ходе лабораторной работы было выполнено проецирование трехмерной фигуры на двумерную плоскость.

С развитием компьютерных технологий проблема показа трехмерных изображений на двумерной плоскости стала очень актуальной. Необходимо иметь определенные математические модели, в которых должны учитываться различные факторы, влияющие на визуальное восприятие человеком реальных образов.

Приложение А

(обязательное)

Код программы

**uses GraphABC;**

**const**

**L = 200;**

**sTime = 1;**

**dRot = 60;**

**type**

**pix3D = record**

**X,Y,Z : Real;**

**constructor; begin (X, Y, Z) := (0, 0, 0); end;**

**constructor (X, Y, Z : Real); begin (Self.X, Self.Y, Self.Z) := (X, Y, Z); end;**

**function rotationX(angle : Real) := new pix3D(X, Y\*cos(angle)+Z\*sin(angle), -Y\*sin(angle)+z\*cos(angle));**

**function rotationY(angle : Real) := new pix3D(X\*cos(angle)-z\*sin(angle), Y, X\*sin(angle)+Z\*cos(angle));**

**function rotationZ(angle : Real) := new pix3D(X\*cos(angle)+Y\*sin(angle), -X\*sin(angle)+Y\*cos(angle), Z);**

**function winX := Window.Center.X + Round(X);**

**function winY := Window.Center.Y + Round(Y);**

**end;**

**var pix : array of pix3D = (**

**new pix3D(+L,+L,+L), new pix3D(+L,+L,-L), new pix3D(+L,-L,+L), new pix3D(+L,-L,-L),**

**new pix3D(-L,+L,+L), new pix3D(-L,+L,-L), new pix3D(-L,-L,+L), new pix3D(-L,-L,-L)**

**);**

**var rotX, rotY, rotZ : real;**

**flag:boolean;**

**procedure Li(a, b : pix3D) := Line(a.winX, a.winY, b.winX, b.winY);**

**procedure DrawCube(angleX, angleY, angleZ : Real);**

**begin**

**var r : array of pix3D;**

**SetLength(r, 8);**

**for var i := 0 to 7 do r[i] := pix[i].rotationX(angleX).rotationY(angleY).rotationZ(angleZ);**

**Li(r[0],r[1]); Li(r[0],r[2]); Li(r[1],r[3]); Li(r[2],r[3]); Li(r[0],r[4]); Li(r[1],r[5]);**

**Li(r[4],r[5]); Li(r[2],r[6]); Li(r[4],r[6]); Li(r[3],r[7]); Li(r[5],r[7]); Li(r[6],r[7]);**

**end;**

**procedure KeyDown(Key: integer);**

**var i, t : integer;**

**begin**

**case Key of**

**VK\_Escape : begin**

**Window.Close;**

**end;**

**VK\_X : begin**

**rotX := rotX + pi/180\*3;**

**end;**

**VK\_Y : begin**

**rotY := rotY + pi/180\*3;**

**end;**

**VK\_Z : begin**

**rotZ := rotZ + pi/180\*3;**

**end;**

**VK\_Add : begin**

**for t := 0 to 7 do begin**

**pix[t].x := pix[t].X \* 1.1;**

**pix[t].y := pix[t].y \* 1.1;**

**pix[t].z := pix[t].z \* 1.1;**

**end;**

**end;**

**VK\_Subtract : begin**

**for t := 0 to 7 do begin**

**pix[t].x := pix[t].X \* 0.9;**

**pix[t].y := pix[t].y \* 0.9;**

**pix[t].z := pix[t].z \* 0.9;**

**end;**

**end;**

**VK\_Left : begin**

**for t := 0 to 7 do begin**

**pix[t].X := pix[t].X - 5;**

**pix[t].Y := pix[t].Y - 5;**

**end;**

**end;**

**VK\_Right : begin**

**for t := 0 to 7 do begin**

**pix[t].X := pix[t].X + 5;**

**pix[t].Y := pix[t].Y + 5;**

**end;**

**end;**

**VK\_Up : begin**

**for t := 0 to 7 do begin**

**pix[t].Z := pix[t].Z - 5;**

**end;**

**end;**

**VK\_Down : begin**

**for t := 0 to 7 do begin**

**pix[t].Z := pix[t].Z + 5;**

**end;**

**end;**

**end;**

**FillRect(0, 0, Window.Width, Window.Height);**

**DrawCube(rotX, rotY, rotZ);**

**end;**

**begin**

**Window.Width := 800;**

**Window.Height := 800;**

**Window.CenterOnScreen;**

**OnKeyDown := KeyDown;**

**SetPenColor(clRed);**

**SetPenWidth(3);**

**rotX := pi/6;**

**rotY := pi/6;**

**rotZ := pi/2;**

**DrawCube(rotX,rotY,rotZ);**

**end.**