Министерство образования Республики Беларусь

Министерство образования и науки Российской Федерации

МУВПО “Белорусско-Российский университет”

Кафедра “ПОИТ”

Отчет по

Лабораторной работе №8

Метод Робертса

Выполнил: ст. гр. АСОИ-181

Самусев С.В.

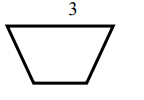
Преподаватель

Шилов А.В.

Могилёв 2020

Цель работы: изучение алгоритма удаления невидимых граней методом Робертса.

Составить алгоритм и программу для поворота объекта № 3



public partial class MainForm : Form {

static Bitmap bitmap;

public MainForm()

{

InitializeComponent();

pictureBox1.Image = null;

bitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);

}

static double gamma = 0;

static int xa, ya, za, xb, yb, zb, xc, yc, zc, xd, yd, zd, xa1, ya1, za1, xb1, yb1, zb1, xc1, yc1, zc1, xd1, yd1, zd1, xs,ys,zs;

private void button3\_Click(object sender, System.EventArgs e)

{

}

private void button2\_Click(object sender, System.EventArgs e)

{

// for (gamma = 0; gamma <= 7; gamma += 0.4)

{

pictureBox1.Image = null;

bitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);

//координата нижней точки A

xa = -50;

ya = -50;

za = -50;

int dlina = 80; //длина

int vusota = 40; //высота

int shirina = 60; //ширина

//определенние координат всех 8 вершин куба

//нижнее основание

xb = xa + dlina;

yb = ya;

zb = za;

xc = xa + dlina;

yc = ya + shirina;

zc = za;

xd = xa;

yd = ya+shirina;

zd = za;

//верхнее

xa1 = xa;

ya1 = ya;

za1 = za + vusota;

xb1 = xb;

yb1 = yb;

zb1 = zb + vusota;

xc1 = xc;

yc1 = yc;

zc1 = zc + vusota;

xd1 = xd;

yd1 = yd;

zd1 = zd + vusota;

xs = xa + dlina / 2;

ys = ya + shirina / 2;

zs = za1 + shirina;

//угол наблюдения за кубиком в радианах от иси х

gammaNabl = 0.985;

поворотZ(gamma, ref xa, ref ya, ref za);

поворотZ(gamma, ref xb, ref yb, ref zb);

поворотZ(gamma, ref xc, ref yc, ref zc);

поворотZ(gamma, ref xd, ref yd, ref zd);

поворотZ(gamma, ref xa1, ref ya1, ref za1);

поворотZ(gamma, ref xb1, ref yb1, ref zb1);

поворотZ(gamma, ref xc1, ref yc1, ref zc1);

поворотZ(gamma, ref xd1, ref yd1, ref zd1);

поворотZ(gamma, ref xs, ref ys, ref zs);

мой\_фигур(xa, ya, za, xb, yb, zb, xc, yc, zc, xd, yd, zd, xa1, ya1, za1, xb1, yb1, zb1, xc1, yc1, zc1, xd1, yd1, zd1, xs,ys,zs);

pictureBox1.Image = bitmap;

//System.Threading.Thread.Sleep(100);

gamma += 0.4;

}

}

static int Min(params int[] x)

{

int min = x[0];

for (int i = 1; i < x.Length; i++)

if (x[i] < min)

min = x[i];

return min;

}

static int Max(params int[] x)

{

int max = x[0];

for (int i = 1; i < x.Length; i++)

if (x[i] > max)

max = x[i];

return max;

}

static void закраска\_треугольника(int xa, int ya, int xb, int yb, int xc, int yc, Color color)

{

//определение ограничивающего прямоугольника

int xmin = Min(xa, xb, xc);

int ymin = Min(ya, yb, yc);

int xmax = Max(xa, xb, xc);

int ymax = Max(ya, yb, yc);

//цикл по оси y

for (int istr = ymin; istr < ymax; istr++)

{

//список активных ребер

bool ab = false;

bool ac = false;

bool bc = false;

int xab = 0, xac = 0, xbc = 0;

//сканирующая строка в верхней половине

if ((ya - istr) \* (yb - istr) <= 0 && ya != yb)

{

ab = true; // ребро активно

//точки пересечения отрезка со строкой istr

xab = X\_Dif\_anal(xa, ya, xb, yb, istr);

}

if ((ya - istr) \* (yc - istr) <= 0 && ya != yc)

{

ac = true; // ребра низа активны

//точки пересечения отрезка со строкой istr

xac = X\_Dif\_anal(xa, ya, xc, yc, istr);

}

if ((yb - istr) \* (yc - istr) <= 0 && yb != yc)

{

bc = true; // ребра низа активны

//точки пересечения отрезка со строкой istr

xbc = X\_Dif\_anal(xb, yb, xc, yc, istr);

}

if (ab && ac == true)

{

for (int col = (int)xab; col <= (int)xac; col++)

bitmap.SetPixel((int)col, (int)istr, color);

if (xac < xab)

{

for (int col = (int)xab; col >= (int)xac; col--)

bitmap.SetPixel((int)col, (int)istr, color);

}

}

if (ab && bc == true)

{

for (int col = (int)xab; col <= (int)xbc; col++)

bitmap.SetPixel((int)col, (int)istr, color);

if (xab > xbc)

{

for (int col = (int)xab; col >= (int)xbc; col--)

bitmap.SetPixel((int)col, (int)istr, color);

}

}

if (bc && ac == true)

{

for (int col = (int)xac; col <= (int)xbc; col++)

bitmap.SetPixel((int)col, (int)istr, color);

if (xac > xbc)

{

for (int col = (int)xac; col >= (int)xbc; col--)

bitmap.SetPixel((int)col, (int)istr, color);

}

}

}

}

static int X\_Dif\_anal(int xx1, int yy1, int xx2, int yy2, int yanal)

{

double k = (double)(xx2 - xx1) / (yy2 - yy1);

int Xanal = xx1 + (int)((yanal - yy1) \* k);

return Xanal;

}

static double gammaNabl = 0.985;

static void XYZ(int wx, int vy, int uz, ref int qX, ref int qY)

{

double al = 3 \* Math.PI / 4;

поворот(al, 0, 0, vy, 0, ref qX, ref qY);

сдвиг(100 + wx, 0, qX, qY, ref qX, ref qY);

сдвиг(0, 100 - uz, qX, qY, ref qX, ref qY);

}

static double[,] M\_umnog(double[,] m1, double[,] m2)

{

double[,] rez = new double[m1.GetLength(1), m2.GetLength(0)];

for (int i = 0; i < rez.GetLength(0); i++)

for (int j = 0; j < rez.GetLength(1); j++)

{

rez[i, j] = 0;

for (int k = 0; k < m1.GetLength(1); k++)

rez[i, j] += m1[i, k] \* m2[k, j];

}

return rez;

}

static double[] M\_umnog(double[,] m1, double[] m2)

{

double[] rez = new double[m1.GetLength(0)];

for (int i = 0; i < rez.Length; i++)

{

rez[i] = 0;

for (int j = 0; j < m1.GetLength(1); j++)

{

rez[i] += m1[i, j] \* m2[j];

}

}

return rez;

}

static void поворот(double ugol, int x0, int y0, int x, int y, ref int XX, ref int YY)

{

double[,] сдвиг\_системы\_координат = { { 1,0,x0},

{ 0,1,y0},

{ 0,0, 1} };

double[,] поворот = { { Math.Cos(ugol),-Math.Sin(ugol),0},

{ Math.Sin(ugol),Math.Cos(ugol),0},

{ 0,0, 1} };

double[,] сдвиг\_системы\_координат1 = { { 1,0,-x0},

{ 0,1,-y0},

{ 0,0, 1} };

double[] XY = M\_umnog(M\_umnog(M\_umnog(сдвиг\_системы\_координат, поворот), сдвиг\_системы\_координат1), new double[] { x, y, 1 });

XX = (int)XY[0];

YY = (int)XY[1];

}

static void поворотZ(double ugol, ref int X, ref int Y, ref int Z)

{

double[,] матрица\_поворота = { { Math.Cos(ugol),-Math.Sin(ugol),0,0},

{ Math.Sin(ugol),Math.Cos(ugol),0,0},

{ 0,0, 1,0},

{ 0,0, 0,1 } };

double[] XYZ = M\_umnog(матрица\_поворота, new double[] { X, Y, Z, 1 });

X = (int)XYZ[0];

Y = (int)XYZ[1];

Z = (int)XYZ[2];

}

static void сдвиг(int x0, int y0, int x, int y, ref int XX, ref int YY)

{

double[,] сдвиг\_системы\_координат = { { 1,0,x0},

{ 0,1,y0},

{ 0,0, 1} };

double[] XY = M\_umnog(сдвиг\_системы\_координат, new double[] { x, y, 1 });

XX = (int)XY[0];

YY = (int)XY[1];

}

static double[] видимые\_грани()

{

int[,] матрица\_кубаV = { { -2,2,0,0,0,0},

{ 0,0,-2,2,0,0},

{0,0,0,0,-2,2 },

{1,1,1,1,1,1}};

double[] вектор\_наблюдателя\_Е = { -Math.Cos(gammaNabl - gamma), -1, -Math.Sin(gammaNabl - gamma), 0 };

double[] rez = new double[матрица\_кубаV.GetLength(1)];

for (int i = 0; i < rez.Length; i++)

for (int j = 0; j < матрица\_кубаV.GetLength(0); j++)

rez[i] += вектор\_наблюдателя\_Е[j] \* матрица\_кубаV[j, i];

return rez;

}

static void мой\_фигур(int xa, int ya, int za, int xb, int yb, int zb, int xc, int yc, int zc, int xd, int yd, int zd, int xa1, int ya1, int za1, int xb1, int yb1, int zb1, int xc1, int yc1, int zc1, int xd1, int yd1, int zd1, int xs, int ys, int zs)

{

bool[] gran = { false, false, false, false, false, false };

double[] v = видимые\_грани();

for (int i = 0; i < gran.Length; i++)

{

if (v[i] > 0)

gran[i] = true;

}

if (gran[0] == true)

четырехугольникXYZ(xb1, yb1, zb1, xb, yb, zb, xc, yc, zc,xc1,yc1,zc1 ,Color.Orange);

if (gran[1] == true)

четырехугольникXYZ(xa, ya, za, xa1, ya1, za1, xd1, yd1, zd1, xd, yd, zd, Color.Blue);

if (gran[3] == true)

четырехугольникXYZ(xa, ya, za, xb, yb, zb, xc, yc, zc, xd, yd, zd, Color.Beige);

if (gran[4] == true)

четырехугольникXYZ(xc1, yc1, zc1, xc, yc, zc, xd, yd, zd, xd1, yd1, zd1, Color.Red);

if (gran[5] == true )

четырехугольникXYZ(xa, ya, za, xa1, ya1, za1, xb1, yb1, zb1, xb, yb, zb, Color.Pink);

if (gran[2] == true || gran[1] == true)

трехугольникXYZ(xa1,ya1,za1,xb1,yb1,zb1,xs,ys,zs, Color.Green);

if (gran[2] == true || gran[3] == true)

трехугольникXYZ(xb1, yb1, zb1, xc1, yc1, zc1, xs, ys, zs, Color.Green);

if (gran[2] == true || gran[2] == true)

трехугольникXYZ(xc1, yc1, zc1, xd1, yd1, zd1, xs, ys, zs, Color.Green);

if (gran[2] == true || gran[3] == true)

трехугольникXYZ(xa1, ya1, za1, xd1, yd1, zd1, xs, ys, zs, Color.Green);

}

static void четырехугольникXYZ(int xa, int ya, int za, int xb, int yb, int zb, int xc, int yc, int zc, int xd, int yd, int zd, Color color)

{

int xxa = 0, yya = 0, xxb = 0, yyb = 0, xxc = 0, yyc = 0, xxd = 0, yyd = 0;

XYZ(xa, ya, za, ref xxa, ref yya);

XYZ(xb, yb, zb, ref xxb, ref yyb);

XYZ(xc, yc, zc, ref xxc, ref yyc);

XYZ(xd, yd, zd, ref xxd, ref yyd);

закраска\_треугольника(xxa, yya, xxb, yyb, xxc, yyc, color);

закраска\_треугольника(xxa, yya, xxc, yyc, xxd, yyd, color);

}

static void трехугольникXYZ(int xa, int ya, int za, int xb, int yb, int zb, int xc, int yc, int zc, Color color)

{

int xxa = 0, yya = 0, xxb = 0, yyb = 0, xxc = 0, yyc = 0;

XYZ(xa, ya, za, ref xxa, ref yya);

XYZ(xb, yb, zb, ref xxb, ref yyb);

XYZ(xc, yc, zc, ref xxc, ref yyc);

закраска\_треугольника(xxa, yya, xxb, yyb, xxc, yyc, color);

}

static void пятиугольникXYZ(int xa, int ya, int za, int xb, int yb, int zb, int xc, int yc, int zc, int xd, int yd, int zd, int xe, int ye, int ze, Color color)

{

int xxa = 0, yya = 0, xxb = 0, yyb = 0, xxc = 0, yyc = 0, xxd = 0, yyd = 0, xxe = 0, yye = 0;

XYZ(xa, ya, za, ref xxa, ref yya);

XYZ(xb, yb, zb, ref xxb, ref yyb);

XYZ(xc, yc, zc, ref xxc, ref yyc);

XYZ(xd, yd, zd, ref xxd, ref yyd);

XYZ(xe, ye, ze, ref xxe, ref yye);

закраска\_треугольника(xxa, yya, xxb, yyb, xxc, yyc, color);

закраска\_треугольника(xxa, yya, xxc, yyc, xxd, yyd, color);

закраска\_треугольника(xxa, yya, xxe, yye, xxd, yyd, color);

}

}

