12

CMatrix SpaceToWindow(CRectD& RS, CRect RW)

{

CMatrix matr(3,3);

double Kx = (double)(((double)(RW.right - RW.left))/((double)(RS.right - RS.left)));

double Ky = (double)(((double)(RW.bottom - RW.top))/((double)(RS.bottom - RS.top)));

matr(0,1) = matr(1,0) = matr(2,0) = matr(2,1) = 0;

matr(2,2) = 1;

matr(0,0) = Kx;

matr(0,2) = RW.left - Kx\*RS.left;

matr(1,1) = -Ky;

matr(1,2) = RW.bottom + Ky\*RS.top;

return matr;

};

14.

int SetMyMode(CDC& dc, CRect& RS, CRectD& RW)

{

int dsx = RS.right - RS.left;

int dsy = RS.top - RS.bottom;

int xsl = RS.left;

int ysl = RS.bottom;

int dwx = RW.right - RW.left;

int dwy = RW.bottom - RW.top;

int xwl = RW.left;

int ywl = RW.bottom;

int buf = dc.SetMapMode(MM\_ANISOTROPIC);

dc.SetWindowExt(dsx,dsy);

dc.SetViewportExt(dwx, -dwy);

dc.SetWindowOrg(xsl,ysl);

dc.SetViewportOrg(xwl, ywl);

return buf;

}

22

void CPyramid::Draw1(CDC& dc,CMatrix& PView,CRect& RW)

// Рисует пирамиду С УДАЛЕНИЕМ невидимых ребер

// Самостоятельный пересчет координат из мировых в оконные (MM\_TEXT)

// dc - ссылка на класс CDC MFC

// PView - координаты точки наблюдения в мировой сферической системе координат

// (r,fi(град.), q(град.))

// RW - область в окне для отображения

{

CMatrix ViewCart=SphereToCart(PView); // Декартовы координаты точки наблюдения

//double zz=ViewCart(2);

CMatrix MV=CreateViewCoord(PView(0),PView(1),PView(2)); //Матрица(4x4) //для пересчета в видовую СК

CMatrix ViewVert=MV\*Vertices; // Координаты вершин пирамиды в видовой СК

CRectD RectView;

GetRect(ViewVert,RectView); // Получаем охватывающий прямоугольник

CMatrix MW=SpaceToWindow(RectView,RW); // Матрица (3x3) для пересчета

// в оконную систему координат

// Готовим массив оконных координат для рисования

CPoint MasVert[5]; // Массив оконных координат вершин

CMatrix V(3); //V0(3);

V(2)=1;

//V0(0)=RectView.left;

//V0(1)=RectView.bottom;

// Цикл по количеству вершин - вычисляем оконные координаты вершин

for(int i=0;i<5;i++)

{

V(0)=ViewVert(0,i); // x

V(1)=ViewVert(1,i); // y

//V=V-V0; // x-xL, y-yL, 1

V=MW\*V; // Оконные координаты точки

MasVert[i].x=(int)V(0);

MasVert[i].y=(int)V(1);

}

// Рисуем

CPen Pen(PS\_SOLID, 2, RGB(0, 0, 255));

CPen\* pOldPen =dc.SelectObject(&Pen);

CBrush Brus(RGB(255, 0, 0));

CBrush\* pOldBrush =dc.SelectObject(&Brus);

CMatrix VE=Vertices.GetCol(4,0,2); // Вершина E

CMatrix R1(3),R2(3),VN(3);

double sm;

for(int i=0;i<4;i++)

{

int k;

if(i==3) k=0;

else k=i+1;

R1=Vertices.GetCol(i,0,2);

R2=Vertices.GetCol(k,0,2);

CMatrix V1=R2-R1; // Вектор - ребро в основании

CMatrix V2=VE-R1; // Вектор - ребро к вершине E

VN=VectorMult(V2,V1); // Вектор ВНЕШНЕЙ(!)нормали к грани

sm=ScalarMult(VN,ViewCart); // Скалярное произведение вектора

//нормали к грани и вектора точки наблюдения

if(sm>=0) // Грань видима - рисуем боковую грань

{

dc.MoveTo(MasVert[i]); // Перо

dc.LineTo(MasVert[k]);

dc.LineTo(MasVert[4]);

dc.LineTo(MasVert[i]);

}

}

VN=VectorMult(R1,R2);

sm=ScalarMult(VN,ViewCart);

if(sm>=0)dc.Polygon(MasVert, 4); // Основание

dc.SelectObject(pOldPen);

dc.SelectObject(pOldBrush);

}

23

void CPlot3D::CreateMatrF(double dx,double dy)//////////////

{

double xL = SpaceRect.left;

double xH = SpaceRect.right;

double yL = SpaceRect.bottom;

double yH = SpaceRect.top;

CVecMatrix VecMatrix;

CMatrix V(4);

V(3) = 1;

for(double x = xL;x <= xH;x += dx)

{

VecMatrix.clear();

for(double y = yL; y<= yH;y += dy)

{

V(0) = x;

V(1) = y;

V(2) = pFunc(x,y);

VecMatrix.push\_back(V);

}

MatrF.push\_back(VecMatrix);

}

}

void CPlot3D:: CreateMatrView()////////////////

{

CMatrix MV = CreateViewCoord(ViewPoint(0),ViewPoint(1),ViewPoint(2));

CVecMatrix VecMatrix;

CMatrix VX(4),V(3);

V(2) = 1;

double xmin = DBL\_MAX;

double xmax = DBL\_MIN;

double ymin = DBL\_MAX;

double ymax = DBL\_MIN;

for(int i = 0;i < MatrF.size();i++)

{

VecMatrix.clear();

for(int j = 0;j < MatrF[i].size();j++)

{

VX = MatrF[i][j];

VX = MV\*VX;

V(0) = VX(0);

V(1) = VX(1);

VecMatrix.push\_back(V);

double x = V(0);

double y = V(1);

if(x<xmin)xmin = x;

if(x>xmax)xmax = x;

if(y<ymin)ymin = y;

if(y>ymax)ymax = y;

}

MatrView.push\_back(VecMatrix);

}

ViewRect.SetRectD(xmin,ymax,xmax,ymin);

}

void CPlot3D:: CreateMatrWindow()/////////////

{

CMatrix MW = SpaceToWindow(ViewRect,WinRect);

CVecPoint VecPoint;

CMatrix V(3);

for(int i = 0; i < MatrView.size();i++)

{

VecPoint.clear();

for(int j = 0;j < MatrView[i].size();j++)

{

V = MatrView[i][j];

V = MW\*V;

CPoint P((int)V(0),(int)V(1));

VecPoint.push\_back(P);

}

MatrWindow.push\_back(VecPoint);

}

}

void CPlot3D::Draw(CDC& dc)//////

{

if(MatrWindow.empty())

{

TCHAR\* error = \_T("Массив данных для рисования в окне пуст! ");

MessageBox(NULL,error,\_T("Ошибка"),MB\_ICONSTOP);

return;

}

CPoint pt[4];

int k,d;

int nRows = MatrWindow.size();

int nCols = MatrWindow[0].size();

for(int i = 0; i < nRows-1;i++)

for(int j = 0; j < nCols-1;j++)

{

pt[0] = MatrWindow[i][j];

pt[1] = MatrWindow[i][j+1];

pt[2] = MatrWindow[i+1][j+1];

pt[3] = MatrWindow[i+1][j];

dc.Polygon(pt,4);

}

}

25

void CPyramid::ColorDraw(CDC& dc,CMatrix& PView,CRect& RW,COLORREF Color)

{

BYTE red=GetRValue(Color);

BYTE green=GetGValue(Color);

BYTE blue=GetBValue(Color);

CMatrix ViewCart=SphereToCart(PView);

CMatrix MV=CreateViewCoord(PView(0),PView(1),PView(2));

CMatrix ViewVert=MV\*Vertices;

CRectD RectView;

GetRect(ViewVert,RectView);

CMatrix MW=SpaceToWindow(RectView,RW);

CPoint MasVert[5];

CMatrix V(3);

V(2)=1;

for(int i=0;i<5;i++)

{

V(0)=ViewVert(0,i); // x

V(1)=ViewVert(1,i); // y

//V=V-V0; // x-xL, y-yL, 1

V=MW\*V;

MasVert[i].x=(int)V(0);

MasVert[i].y=(int)V(1);

}

CMatrix VE=Vertices.GetCol(4,0,2);

CMatrix R1(3),R2(3),VN(3);

double sm;

for(int i=0;i<4;i++)

{

int k;

if(i==3) k=0;

else k=i+1;

R1=Vertices.GetCol(i,0,2);

R2=Vertices.GetCol(k,0,2);

CMatrix V1=R2-R1;

CMatrix V2=VE-R1;

VN=VectorMult(V2,V1);

sm=CosV1V2(VN,ViewCart);

if(sm>=0)

{

CPen Pen(PS\_SOLID, 2, RGB(sm\*sm\*red, sm\*sm\*green, sm\*sm\*blue));

CPen\* pOldPen =dc.SelectObject(&Pen);

CBrush Brus(RGB(sm\*sm\*red, sm\*sm\*green, sm\*sm\*blue));

CBrush\* pOldBrush =dc.SelectObject(&Brus);

CPoint MasVertR[3]={MasVert[i],MasVert[k],MasVert[4]};

dc.Polygon(MasVertR, 3); // Ребро

dc.SelectObject(pOldBrush);

dc.SelectObject(pOldPen);

}

}

VN=VectorMult(R1,R2);

sm=CosV1V2(VN,ViewCart);

if(sm>=0)

{

CPen Pen(PS\_SOLID, 2, RGB(sm\*sm\*red, sm\*sm\*green, sm\*sm\*blue));

CPen\* pOldPen =dc.SelectObject(&Pen);

CBrush Brus(RGB(sm\*sm\*red, sm\*sm\*green, sm\*sm\*blue));

CBrush\* pOldBrush =dc.SelectObject(&Brus);

dc.Polygon(MasVert, 4); // Основание

dc.SelectObject(pOldBrush);

dc.SelectObject(pOldPen);

}

}