МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



**ОТЧЁТ**

**по лабораторной работе №1**

**«*Метод плавающего горизонта»***

по дисциплине:

**«*Графические системы*»**

|  |  |
| --- | --- |
| Выполнил: *Астафьев В.Е.*  Группа  *АО-71* | Проверил:  Кухто А.В. |

Новосибирск

2018

**Цель работы:**

Целью данной работы является реализация метода плавающего горизонта для удаления невидимых линий

**Ход работы:**

Написать программу, которая:

1. загружает рисунок из файла;
2. строит на основе этого рисунка трёхмерное изображение с использованием метода плавающего горизонта;
3. позволяет вращать полученное изображение по осям X и Y, а по оси Z - опционально.

**Программный код**:

**Progmam.cs:**

using System;

using System.Windows.Forms;

namespace lab1

{

internal sealed class Program

{

[STAThread]

private static void Main(string[] args)

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

**MainForm.cs:**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace lab1

{

public partial class MainForm : Form

{

drawgraph graphik;

public double[,] inmas = new double[67, 67];

int resolutionOfArray = 67;

int ang;

public delegate double functionType(double x, double z);

List<functionType> functions = new List<functionType>();

Bitmap bmp;

Image k;

enum Vis

{

upper,

lower,

invis

}

public struct Point2D

{

public double X, Y;

public Point2D(double x, double y)

{

this.X = x;

this.Y = y;

}

public static implicit operator Point(Point2D pointD)

{

return new Point((int)Math.Round(pointD.X), (int)Math.Round(pointD.Y));

}

}

public struct Point3D

{

public double X, Y, Z;

}

public MainForm()

{

InitializeComponent();

}

public static class Functions

{

public static double f1(double x, double z)

{

return Math.Sin(x + z);

}

public static double f2(double x, double z)

{

return (x\*Math.Cos(2\*z) - z\*Math.Sin(2\*x))/4;

}

public static double f3(double x, double z)

{

return Math.Exp(Math.Sin(Math.Sqrt(x \* x + z \* z)));

}

}

private void button1\_Click(object sender, EventArgs e)

{

graphik.SetFlag(false);

graphik.Draw(pictureBox1.CreateGraphics(), functions[comboBox1.SelectedIndex]);

}

private void Form1\_Load(object sender, EventArgs e)

{

List<String> funcNames = new List<String>();

funcNames.Add("Y = Sin(x + z)");

funcNames.Add("Y = (x\*Cos(2z)-z\*Sin(2x))/4");

funcNames.Add("Y = e^(Sin(Sqrt(x \* x + z \* z )))");

functions.Add(Functions.f1);

functions.Add(Functions.f2);

functions.Add(Functions.f3);

comboBox1.Items.AddRange(funcNames.ToArray());

comboBox1.SelectedIndex = 0;

button3.BackColor = Color.Blue;

button2.BackColor = Color.Black;

graphik = new drawgraph(pictureBox1.Width, pictureBox1.Height);

graphik.granX(Convert.ToDouble(0), Convert.ToDouble(resolutionOfArray-1));

graphik.granZ(Convert.ToDouble(0), Convert.ToDouble(resolutionOfArray-1));

graphik.stepsXZ(Convert.ToDouble(1), Convert.ToDouble(1));

graphik.SetAngleX(trackBar1.Value);

graphik.SetAngleY(trackBar2.Value);

graphik.SetAngleZ(trackBar3.Value);

graphik.SetBackColor(button2.BackColor);

graphik.SetMainColor(button3.BackColor);

}

class drawgraph

{

const double NULL = -1;

int imageWidth, imageHeight;

double[] lowHor, upHor;

public double[,] mas = new double[67,67];

double zMin = NULL, zMax = NULL,

xMin = NULL, xMax = NULL;

double xStep = 0, zStep = 0;

public double[,] inmas1 = new double[67, 67];

double angleX = 0, angleY = 0, angleZ = 0;

bool cur = false;

public int resttt=67;

Graphics graphics;

Bitmap bitmap;

Color backColor, mainColor;

Point3D center;

bool flag=false;

int resTemp = 0;

public Bitmap bmp() { return bitmap; }

public drawgraph(int imageWidth, int imageHeight)

{

this.imageWidth = imageWidth;

this.imageHeight = imageWidth;

bitmap = new Bitmap(imageWidth, imageHeight);

lowHor = new double[imageWidth+100];

upHor = new double[imageWidth+100];

backColor = Color.White;

mainColor = Color.Black;

}

public void SetMasRes(int v)

{

mas = new double[v,v];

inmas1 = new double[v,v];

}

private void ResetHor()

{

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

{

lowHor[i] = imageHeight;

upHor[i] = 0;

}

}

public void SetFlag(bool bb)

{

flag = bb;

}

public void granX(double xMin, double xMax)

{

this.xMax = xMax;

this.xMin = xMin;

}

public void granZ(double zMin, double zMax)

{

this.zMax = zMax;

this.zMin = zMin;

}

public void stepsXZ(double xStep, double zStep)

{

this.zStep = zStep;

this.xStep = xStep;

}

public void SetAngleX(int angle)

{

angleX = DegreeToRadian(angle);

}

public void SetAngleY(int angle)

{

angleY = DegreeToRadian(angle);

}

public void SetAngleZ(int angle)

{

angleZ = DegreeToRadian(angle);

}

public void SetMainColor(Color mainColor)

{

this.mainColor = mainColor;

}

public void SetBackColor(Color backColor)

{

this.backColor = backColor;

}

private void CalcBodyCenter()

{

center.X = (xMax + xMin) / 2;

center.Z = (zMin + zMax) / 2;

center.Y=mas[(int)Math.Round((center.X + 5) / xStep), (int)Math.Round((center.Z + 5) / xStep)];

}

private void CalcFunction(functionType function)

{

int i=0,j=0;

for (double currentX = xMin; currentX <= xMax; currentX += xStep)

{

for (double currentZ = zMin; currentZ <= zMax; currentZ += zStep) {

mas[i,j]= function(currentX,currentZ);

j++;

}

i++; j = 0;

}

}

public void SetMas(double[,] im)

{

mas = im;

flag = true;

}

public void Draw(Graphics g, functionType function)

{

if (!flag) { CalcFunction(function); }

if(resTemp != resttt){

CalcBodyCenter();

resTemp = resttt;

}

StartDoubleBuffering();

ResetHor();

ResetHor();

Draw(g);

FinishDoubleBuffering(g);

}

private bool CheckPoint(Point2D point)

{

if ((int)Math.Round(point.X) < 0 || (int)Math.Round(point.X) >= imageWidth)

return false;

return true;

}

private void Draw(Graphics imageGraphics)

{

Point2D prevPoint,

leftPoint = new Point2D(NULL, NULL),

rightPoint = new Point2D(NULL, NULL);

for (double currentZ = zMax; currentZ >= zMin; currentZ -= zStep)

{

double y;

y = mas[(int)Math.Round(xMin),(int)Math.Round(currentZ)];

prevPoint = c3Dto2D(xMin, y, currentZ);

DrawifOK(prevPoint, ref leftPoint);

if (!(CheckPoint(prevPoint) && CheckPoint(leftPoint)))

continue;

Vis prevVisibility = CheckVis(prevPoint);

Point2D currentPoint = new Point2D();

for (double currentX = xMin; currentX <= xMax; currentX += xStep)

{

cur = false;

if(currentX > xMax-xStep/2)

{

cur = true;

}

y = mas[(int)Math.Round(currentX ), (int)Math.Round(currentZ)];

currentPoint = c3Dto2D(currentX, y, currentZ);

if (!CheckPoint(currentPoint))

continue;

Vis currentVisibility = CheckVis(currentPoint);

if (currentVisibility == prevVisibility)

{

if (currentVisibility == Vis.lower || currentVisibility == Vis.upper)

{

DrawLine(prevPoint, currentPoint);

Smooth(prevPoint, currentPoint);

}

}

prevVisibility = currentVisibility;

prevPoint = currentPoint;

}

DrawifOK(currentPoint, ref rightPoint);

}

}

private void StartDoubleBuffering()

{

this.graphics = Graphics.FromImage(bitmap);

graphics.Clear(backColor);

graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

}

private void FinishDoubleBuffering(Graphics sourceGraphics)

{

sourceGraphics.DrawImage(bitmap, 0, 0);

}

private Point2D GetPer(Point2D firstPoint, Point2D secondPoint, double[] currentHorizon)

{

if (Math.Round(firstPoint.X) == Math.Round(secondPoint.X))

{

return new Point2D(secondPoint.X, currentHorizon[(int)Math.Round(secondPoint.X)]);

}

if (Math.Round(secondPoint.X) < Math.Round(firstPoint.X))

{

Point2D buf = firstPoint;

firstPoint = secondPoint;

secondPoint = buf;

}

double lineCoef = (secondPoint.Y - firstPoint.Y) / (secondPoint.X - firstPoint.X);

int currentX = (int)Math.Floor(firstPoint.X) + 1;

double currentY = firstPoint.Y + lineCoef;

int prevSignY = Math.Sign(firstPoint.Y + lineCoef - currentHorizon[currentX]);

int currentSignY = prevSignY;

while (prevSignY == currentSignY && currentX <= Math.Floor(secondPoint.X))

{

currentY += lineCoef;

++currentX;

currentSignY = Math.Sign(currentY - currentHorizon[currentX]);

}

if (Math.Abs(currentY - lineCoef - currentHorizon[currentX - 1]) <= Math.Abs(currentY - currentHorizon[currentX]))

{

currentY -= lineCoef;

--currentX;

}

return new Point2D(currentX, currentY);

}

private Vis CheckVis(Point2D currentPoint)

{

if (currentPoint.Y < upHor[(int)Math.Round(currentPoint.X)] && currentPoint.Y > lowHor[(int)Math.Round(currentPoint.X)])

return Vis.invis;

else

if (currentPoint.Y >= upHor[(int)Math.Round(currentPoint.X)])

return Vis.upper;

else

return Vis.lower;

}

private void DrawifOK(Point2D prevPoint, ref Point2D currentPoint)

{

currentPoint = prevPoint;

}

private void Smooth(Point2D firstPoint, Point2D secondPoint)

{

if (Math.Round(firstPoint.X) == Math.Round(secondPoint.X))

{

upHor[(int)Math.Round(secondPoint.X)] = Math.Max(upHor[(int)Math.Round(secondPoint.X)], Math.Max(secondPoint.Y, firstPoint.Y));

lowHor[(int)Math.Round(secondPoint.X)] = Math.Min(lowHor[(int)Math.Round(secondPoint.X)], Math.Min(secondPoint.Y, firstPoint.Y));

}

else

{

if (secondPoint.X < firstPoint.X)

{

Point2D buf = firstPoint;

firstPoint = secondPoint;

secondPoint = buf;

}

double lineCoef = (secondPoint.Y - firstPoint.Y) / (secondPoint.X - firstPoint.X);

double currentY;

for (int currentX = (int)Math.Round(firstPoint.X); currentX <= Math.Round(secondPoint.X); ++currentX)

{

currentY = lineCoef \* (currentX - firstPoint.X) + firstPoint.Y;

upHor[currentX] = Math.Max(upHor[currentX], currentY);

lowHor[currentX] = Math.Min(lowHor[currentX], currentY);

}

for (int currentX = (int)Math.Round(secondPoint.X); currentX <= Math.Round(firstPoint.X); ++currentX)

{

currentY = lineCoef \* (currentX - secondPoint.X) + secondPoint.Y;

upHor[currentX] = Math.Max(upHor[currentX], currentY);

lowHor[currentX] = Math.Min(lowHor[currentX], currentY);

}

}

}

private Point2D c3Dto2D(double x, double y, double z)

{

double sc = 1.5;

double scale = Math.Min(imageWidth, imageHeight) / (Math.Max(xMax - xMin, zMax - zMin) \* sc);

x = center.X + (x - center.X) \* scale;

y = -(center.Y + (y - center.Y) \* scale);

z = center.Z + (z - center.Z) \* scale;

// y

double temp = z \* Math.Cos(angleY) + x \* Math.Sin(angleY);

x = -z \* Math.Sin(angleY) + x \* Math.Cos(angleY);

z = temp;

// x

temp = y \* Math.Cos(angleX) + z \* Math.Sin(angleX);

z = -y \* Math.Sin(angleX) + z \* Math.Cos(angleX);

y = temp;

// z

temp = x \* Math.Cos(angleZ) + y \* Math.Sin(angleZ);

y = -x \* Math.Sin(angleZ) + y \* Math.Cos(angleZ);

x = temp;

x += imageWidth / 2 - center.X;

y += imageHeight / 3 - center.Y;

return new Point2D(x, y);

}

private double DegreeToRadian(int angleInDegrees)

{

return Math.PI \* angleInDegrees / 180;

}

private void DrawLine(Point2D firstPoint, Point2D secondPoint)

{

graphics.DrawLine(new Pen(mainColor, 1), firstPoint, secondPoint);

}

}

private void trackBar1\_ValueChanged(object sender, EventArgs e)

{

graphik.SetAngleX(trackBar1.Value);

graphik.Draw(pictureBox1.CreateGraphics(), functions[comboBox1.SelectedIndex]);

}

private void trackBar2\_ValueChanged(object sender, EventArgs e)

{

ang = trackBar2.Value;

Draw1();

}

private void trackBar3\_ValueChanged(object sender, EventArgs e)

{

graphik.SetAngleZ(trackBar3.Value);

graphik.Draw(pictureBox1.CreateGraphics(), functions[comboBox1.SelectedIndex]);

}

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

{

colorDialog1.ShowDialog();

graphik.SetBackColor(colorDialog1.Color);

button2.BackColor = colorDialog1.Color;

}

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

{

colorDialog1.ShowDialog();

graphik.SetMainColor(colorDialog1.Color);

button3.BackColor = colorDialog1.Color;

}

private void Draw1()

{

int kr = 0;

Image image = k; //Загружаем в image изображение из выбранного файла

//bmp = new Bitmap(image, image.Width, image.Height); //создаем и загружаем из image изображение в формате bmp

if (k.Height>k.Width){

kr = (int)Math.Round(resolutionOfArray/Math.Sqrt(Math.Pow(k.Height/k.Width,2)+1));

}

else{

kr = (int)Math.Round(resolutionOfArray/Math.Sqrt(Math.Pow(k.Width/k.Height,2)+1));

}

Bitmap bmp\_temp;

if(k.Height>k.Width){

bmp = new Bitmap(image, (int)resolutionOfArray\*k.Width/k.Height, resolutionOfArray);

bmp\_temp = new Bitmap(resolutionOfArray,resolutionOfArray);

}

else{

bmp = new Bitmap(image, kr,(int)kr\*k.Height/k.Width);

bmp\_temp = new Bitmap(resolutionOfArray,resolutionOfArray);

}

Graphics g = Graphics.FromImage(bmp\_temp);

g.TranslateTransform((float)bmp\_temp.Width/2, (float)bmp\_temp.Height / 2);

g.RotateTransform(ang);

g.TranslateTransform(-(float)bmp\_temp.Width/2,-(float)bmp\_temp.Height / 2);

g.DrawImage( bmp, new Point( (int)(resolutionOfArray-bmp.Width)/2 , (int)(resolutionOfArray-bmp.Height)/2 ) );

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

for (int j = 0; j < resolutionOfArray; j++)

{

double B = bmp\_temp.GetPixel(i, j).GetBrightness();

inmas[i, j] = B \* 50;

}

graphik.SetMas(inmas);

graphik.Draw(pictureBox1.CreateGraphics(), functions[comboBox1.SelectedIndex]);

}

private void button5\_Click(object sender, EventArgs e)

{

OpenFileDialog dialog = new OpenFileDialog();

dialog.Filter = "Image files (\*.BMP, \*.JPG, \*.GIF, \*.TIF, \*.PNG, \*.ICO, \*.EMF, \*.WMF)|\*.bmp;\*.jpg;\*.gif; \*.tif; \*.png; \*.ico; \*.emf; \*.wmf";

Bitmap bmp\_temp;

int kr = 0;

if (dialog.ShowDialog() == DialogResult.OK)

{

Image image = Image.FromFile(dialog.FileName);

k = image;

//bmp = new Bitmap(image, image.Width, image.Height); //создаем и загружаем из image изображение в формате bmp

if(k.Height>k.Width){

kr = (int)Math.Round(resolutionOfArray/Math.Sqrt(Math.Pow(k.Height/k.Width,2)+1));

}

else{

kr = (int)Math.Round(resolutionOfArray/Math.Sqrt(Math.Pow(k.Width/k.Height,2)+1));

}

if(k.Height>k.Width){

//koef = k.Width/k.Height;

bmp = new Bitmap(image, (int)kr\*k.Width/k.Height, kr);

bmp\_temp = new Bitmap(resolutionOfArray,resolutionOfArray);

}

else{

bmp = new Bitmap(image, kr,(int)kr\*k.Height/k.Width);

bmp\_temp = new Bitmap(resolutionOfArray,resolutionOfArray);

}

Graphics g = Graphics.FromImage(bmp\_temp);

g.TranslateTransform((float)bmp.Width/2, (float)bmp.Height / 2);

g.RotateTransform(ang);

g.TranslateTransform(-(float)bmp.Width/2,-(float)bmp.Height / 2);

g.DrawImage( bmp, new Point( (int)(resolutionOfArray-bmp.Width)/2 , (int)(resolutionOfArray-bmp.Height)/2 ) );

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

for (int j = 0; j < resolutionOfArray; j++)

{

double B = bmp\_temp.GetPixel(i, j).GetBrightness();

inmas[i, j] = B \* 50;

}

graphik.SetMas(inmas);

graphik.Draw(pictureBox1.CreateGraphics(), functions[comboBox1.SelectedIndex]);

}

}

void Button6Click(object sender, EventArgs e)

{

graphik.SetMasRes(Convert.ToInt32(textBox1.Text));

inmas = new double[Convert.ToInt32(textBox1.Text),Convert.ToInt32(textBox1.Text)];

resolutionOfArray = Convert.ToInt32(textBox1.Text);

graphik.granX(Convert.ToDouble(0), Convert.ToDouble(resolutionOfArray-1));

graphik.granZ(Convert.ToDouble(0), Convert.ToDouble(resolutionOfArray-1));

graphik.resttt = Convert.ToInt32(textBox1.Text);

//ang = 40;

Draw1();

}

private void button4\_Click(object sender, EventArgs e)

{

}

private void label5\_Click(object sender, EventArgs e)

{

}

private void label8\_Click(object sender, EventArgs e)

{

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

}

}

**Результат работы программы**:



