Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра «САПР»

Лабораторная работа №1

по дисциплине «Компьютерная графика»

Метод Брезенхема

Выполнил: студент гр. −31 qwinmen.

Проверил: С. А.

Тамбов,

***Цели и задачи****.*

Реализовать в коде алгоритм Брезенхема.

***Решение задачи****.*

Алгоритм сводится к задаче построения линии по двум заданным точкам. Разберем пример:

Даны начальные координаты точек [1;1] и [7;12].

Находим dx = 7 – 1 = 6 и dy = 12 – 1 = 11.

if( dx < dy ), то угол больше 45\*;

Меняем координаты местами

Находим dx = 12 – 1 = 11 и dy = 7 – 1 = 6.

(dx >= dy) следовательно d1 = 2\*dy – dx = 2\*6 – 11 = 1 > 0;

if(di >= 0) {

x2 = x1 + 1 = 2;

y2 = y1 + 1 = 2;

d2 = d1 + 2\*(dy - dx) = 1 + (-10) = -9 < 0;

};

if(d2 < 0) {

x3 = x2 + 1 = 3;

y3 = y2 = 2;

d3 = d2 + 2\*dy = -9 + 12 = 3 > 0;

}

x4 = x3 + 1 = 4;

y4 = y3 + 1 = 3;

и так до конечной точки.

Решающим являются два уравнения:

di + 2\*dy – dx, di = плюс.

di + 2\*dy, di = минус.

***Исходный код:***

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

using Tao.FreeGlut;

using Tao.OpenGl;

namespace PointLine

{

public partial class fMain : Form

{

public fMain()

{

InitializeComponent();

OGlControl.InitializeContexts();

}

/// <summary>

/// Инициализация GL

/// </summary>

private void fMain\_Load(object sender, System.EventArgs e)

{

// инициализация Glut

Glut.glutInit();

Glut.glutInitDisplayMode(Glut.GLUT\_RGB | Glut.GLUT\_DOUBLE | Glut.GLUT\_DEPTH);

Gl.glEnable(Gl.GL\_ALPHA\_TEST);//чтобы разрешить обрабатывать четвертый компонент цвета

Gl.glEnable(Gl.GL\_BLEND);//чтобы разрешить наложение цветов

// очитка окна

Gl.glClearColor(255, 255, 255, 1);

// установка порта вывода в соотвествии с размерами элемента anT

Gl.glViewport(0, 0, OGlControl.Width, OGlControl.Height);

// настройка проекции

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

// теперь необходимо корректно настроить 2D ортогональную проекцию

// в зависимости от того, какая сторона больше

// мы немного варьируем то, как будет сконфигурированный настройки проекции

if ((float)OGlControl.Width <= (float)OGlControl.Height)

Glu.gluOrtho2D(0.0, 30.0 \* (float)OGlControl.Height / (float)OGlControl.Width, 0.0, 30.0);

else

Glu.gluOrtho2D(0.0, 30.0 \* (float)OGlControl.Width / (float)OGlControl.Height, 0.0, 30.0);

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

Gl.glLoadIdentity();

//Брезенхем(1, 1, 1, 50);//y

//Брезенхем(0, 0, 40, 2);//x

button1\_Click();

}

private void button1\_Click()

{

//очистка экрана перед визуализацией кадра (Цвет И Глубина)

Gl.glClear(Gl.GL\_COLOR\_BUFFER\_BIT | Gl.GL\_DEPTH\_BUFFER\_BIT);

Gl.glLoadIdentity();

Gl.glColor3f(0, 0, 0);//устанавл цвет отрисовки геометрии

Gl.glPushMatrix();

Gl.glPointSize(5);

// рисуем точки

Gl.glBegin(Gl.GL\_POINTS);

for (double i = 1; i < 50; i += 0.04)

{

Gl.glVertex2d(i, 1);//x

Gl.glVertex2d(1, i);//y

}

Gl.glEnd();

Gl.glPointSize(1);

Брезенхем(20, 20, 300, 21);

Gl.glBegin(Gl.GL\_POINTS);

DrawPoint(0);// [x--------x]

Gl.glEnd();

Gl.glPopMatrix();

Gl.glFlush();

OGlControl.Invalidate();//обновить отображаемый кадр

}

/// <summary>

/// Отрисовка точек из контейнеров

/// </summary>

private void DrawPoint(int key)

{

switch (\_globalFlag)

{

case 0://для Х+45

{

foreach (KeyValuePair<PointF, Color> keyValuePair in PointColor)

{

Gl.glColor3b(keyValuePair.Value.R, keyValuePair.Value.G, keyValuePair.Value.B);

Gl.glVertex2f(keyValuePair.Key.X, keyValuePair.Key.Y);

}

for (double i = 0; i < duplicatePointColor.Count(); i++)

duplicatePointColor[i] = Gradient(i);//пишем новые цвета

double index0 = 0;

foreach (PointF pnt in duplicatePointColor)

{

Gl.glColor3b(duplicatePointColor[index0].R, duplicatePointColor[index0].G, duplicatePointColor[index0].B);

Gl.glVertex2f(pnt.X, pnt.Y);

index0++;

}

}

break;

case 1://Для У-45

{

foreach (KeyValuePair<PointF, Color> keyValuePair in PointColor)

{

Gl.glColor3b(keyValuePair.Value.R, keyValuePair.Value.G, keyValuePair.Value.B);

Gl.glVertex2f(keyValuePair.Key.Y, keyValuePair.Key.X);

}

for (double i = 0; i < duplicatePointColor.Count(); i++)

duplicatePointColor[i] = Gradient(i);//пишем новые цвета

double index1 = 0;

foreach (PointF pnt in duplicatePointColor)

{

Gl.glColor3b(duplicatePointColor[index1].R, duplicatePointColor[index1].G, duplicatePointColor[index1].B);

Gl.glVertex2f(pnt.Y, pnt.X);

index1++;

}

\_globalFlag = 0;

}

break;

}

}

/// <summary>

/// Цвет основной линии

/// </summary>

private static Color \_globalColor = Color.FromArgb(127, 0, 0);

private static byte index = 0;

private static int tmp;

/// <summary>

/// Расчитать цвет опорного пиксела от позиции его в ступеньке

/// </summary>

private static Color Gradient(double step)

{

if(step == 0) tmp = \_stupeniList[0];

//от step до tmp

//tmp = tmp + \_stupeniList[1]

if(step<=tmp)

{

int Rgradient = \_globalColor.R-27;// \_stupeniList[index];

int Ggradient = \_globalColor.G;// \_stupeniList[index];

int Bgradient = \_globalColor.B;// \_stupeniList[index];

return Color.FromArgb(Rgradient, Ggradient, Bgradient);

}

else

{

index++;

//tmp = tmp + \_stupeniList[index];

int Rgradient = \_globalColor.R-27;//\_stupeniList[index];

int Ggradient = \_globalColor.G;// \_stupeniList[index];

int Bgradient = \_globalColor.B;// \_stupeniList[index];

return Color.FromArgb(Rgradient, Ggradient, Bgradient);

}

}

private static int \_globalFlag = 0;//координатная четверть

//Алгоритм построения\высчитывания линии\координат по двум точкам

private static void Брезенхем(int a, int b, int c, int d)

{

int xStart = a, yStart = b, xEnd = c, yEnd = d;

//Находим dx dy

int dx = (xEnd - xStart);

int dy = (yEnd - yStart);

\_globalFlag = 0;//reset

if (dx < dy)//поменять местами координаты точек

{

int tmp = xStart;//2

xStart = yStart;//1

yStart = tmp;//2

tmp = xEnd;//7

xEnd = yEnd;//4

yEnd = tmp;//7

dx = (xEnd - xStart);

dy = (yEnd - yStart);

\_globalFlag = 1;

}

int d1 = (dy << 1) - dx;//2\*dy - dx;

PointColor.Add(new Point(xStart, yStart), \_globalColor);//начальная точа

duplicatePointColor.Add(new PointF(xStart, yStart + 0.04f), \_globalColor);//Для точек всей ступеньки

for (int i = xStart; i < xEnd; i++)//т.к по Х постоянное приращение на 1

d1 = Coordinats(d1 >= 0, dx, dy, d1, i == xEnd - 1);

}

/// <summary>

/// di = di + 2\*dy - dx

/// </summary>

private static int ControlFull(int dx, int dy, int di) { return di + ((dy - dx) << 1); }//di + 2\*(dy - dx)

/// <summary>

/// di = di + 2\*dy

/// </summary>

private static int ControlLite(int dy, int di) { return di + (dy << 1); }//di + 2\*(dy)

/// <summary>

/// находим прирост [x;y]

/// </summary>

private static int Coordinats(bool flag, int dx, int dy, int d1, bool stop)

{

if(!flag && stop) \_stupeniList.Add(\_index+1);//1 - конец

if(flag)

{//flag=true начало новой ступеньки

float x2 = PointColor.Last().Key.X + 0.04f;//xStart + 1;

float y2 = PointColor.Last().Key.Y + 0.04f;//yStart + 1;

PointColor.Add(new PointF(x2, y2), \_globalColor);

//duplicatePointColor.Add(new PointF(x2+0.04f, y2+0.04f), \_globalColor);//Для точек в углах лесинки

duplicatePointColor.Add(new PointF(x2, y2 + 0.04f), \_globalColor);//Для точек всей ступеньки

\_stupeniList.Add(\_index+1);

\_index = 0;

return ControlFull(dx, dy, d1);//d1 = d1 + 2 \* (dy - dx);

}

else

{//flag=false продолжение старой ступени

float x2 = PointColor.Last().Key.X + 0.04f;//xStart + 1;

float y2 = PointColor.Last().Key.Y;//yStart;

\_index++;

PointColor.Add(new PointF(x2, y2), \_globalColor);

//duplicatePointColor.Add(new PointF(x2+0.04f, y2), \_globalColor);//Для точек в углах лесинки

duplicatePointColor.Add(new PointF(x2, y2 + 0.04f), \_globalColor);

return ControlLite(dy, d1);//d1 = d1 + 2 \* (dy);

}

}

private static int \_index = 0;//счетчик вызовов flag = false

/// <summary>

/// Основной контейнер x;y;color

/// </summary>

private static Dictionary<PointF, Color> PointColor = new Dictionary<PointF, Color>();

/// <summary>

/// Сколько пикселей на одной ступеньке (для градиента цветов)

/// </summary>

private static List<int> \_stupeniList = new List<int>();

/// <summary>

/// Вспомогательный контейнер x;y;color

/// </summary>

private static clArrayList duplicatePointColor = new clArrayList();

private int a, b, c, d;

private void OGlControl\_MouseDown(object sender, MouseEventArgs e)

{

if(e.Button == MouseButtons.Left)

{

a = e.X;

b = OGlControl.Height - e.Y;//768 - e.Y;

}

}

private void OGlControl\_MouseUp(object sender, MouseEventArgs e)

{

if(e.Button == MouseButtons.Left)

{

c = e.X;

d = OGlControl.Height - e.Y;//768 - e.Y;

//Брезенхем(2, 2, c, d);

//button1\_Click();

}

PointColor.Clear();

duplicatePointColor.Clear();

}

}

}

***Конечный результат:***

