# 5. Рисование на html-холсте

## 5.1 Html-холст, введение в API canvas

В главе 2 мы реализовали кнопку «График» для переключения на графическое окно для визуалиции результатов. Рисование осуществляется с помощью объекта WebBrowser, JavaScript и canvas. Это медленнее, чем использование специализированных библиотек, но очень полезно для интеграции с веб-сайтом и обмена данными.

Canvas (холст или канвас) — элемент HTML5, предназначенный для создания растрового двухмерного изображения при помощи скриптов, обычно на языке JavaScript.

Элемент <canvas> является всего лишь контейнером для графики. Вы должны использовать JavaScript, чтобы на самом деле рисовать графику.

В Canvas есть несколько методов рисования контуров, прямоугольников, кругов, текста и добавления изображений.

Canvas - это прямоугольная область на HTML-странице. По умолчанию холст не имеет границ и содержимого.

Разметка выглядит так:

<canvas id= "MyCanvas" width= "200" height= "100"></canvas>

Всегда указывайте атрибут id (на который следует ссылаться в сценарии), а также атрибут width (ширины) и height (высоты) для определения размера холста. Чтобы добавить границу, используйте атрибут style.

<!DOCTYPE html>

<html>

<body>

<h1>Элемент-холст, красный квадрат</h1>

<canvas id="myCanvas">Ваш браузер не поддерживает canvas тэг.</canvas>

<script>

//найти холст

var c = document.getElementById("myCanvas");

//получить контекст (API)

var ctx = c.getContext("2d");

//установить красный цвет для заполнения

ctx.fillStyle = "#FF0000";

//залить прямоугольник

ctx.fillRect(0, 0, 80, 100);

</script>

</body>

</html>

Код выше рисует красный квадрат в браузере. Хороший обзор методов API canvas дан на сайте <https://www.w3schools.com/html/html5_canvas.asp> . Мы же попробуем запускать эти методы через MathPanel.

Добавим в MainWindow.xaml.cs метод

/// <summary>

/// метод для вызова тестов

/// </summary>

public static void GraphExample(string id)

{

if (!bReady || dispObj.HasShutdownStarted) return;

//запустить UI потоке

dispObj.Invoke(delegate

{

webConsole.InvokeScript("ext\_example" + id);

});

}

А в test\_graph.htm методы для тестирования.

//рисовать линию

function ext\_example1(w) {

var c = document.getElementById("canvas2");

var ctx = c.getContext("2d");

ctx.moveTo(0, 0);

ctx.lineTo(1100, 600);

ctx.stroke();

}

//рисовать окружность

function ext\_example2(w) {

var c = document.getElementById("canvas2");

var ctx = c.getContext("2d");

ctx.beginPath();

ctx.arc(95, 50, 40, 0, 2 \* Math.PI);

ctx.stroke();

}

//рисовать текст

function ext\_example3(w) {

var c = document.getElementById("canvas2");

var ctx = c.getContext("2d");

ctx.font = "30px Arial";

ctx.fillText("Hello World", 10, 50);

}

Пишем скрипт

Dynamo.Console("test49\_graph\_example1");

Dynamo.GraphExample("1");

Запускаем, получаем линию (изображение на рис. 5.1).

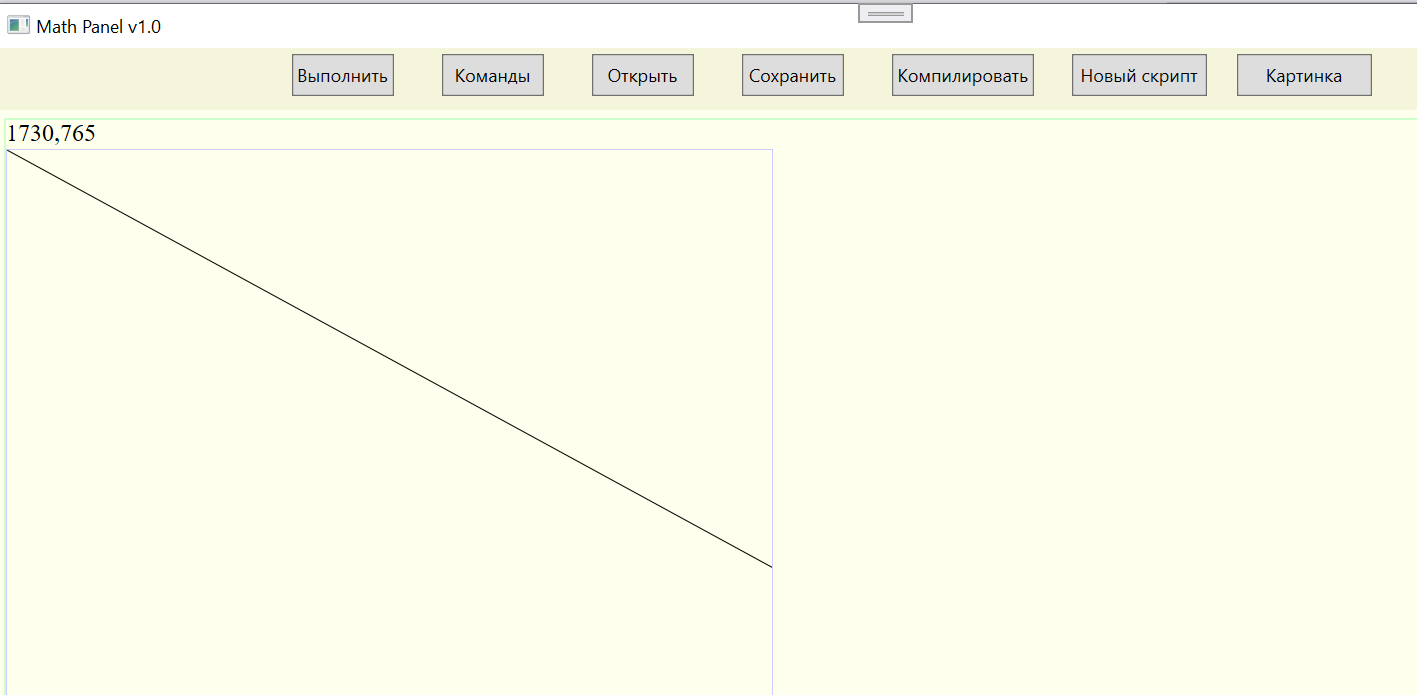


Рис.5.1. Рисование линии.

Пишем скрипт

Dynamo.Console("test49\_graph\_example2");

Dynamo.GraphExample("2");

Запускаем, получаем окружность (изображение на рис. 5.2).

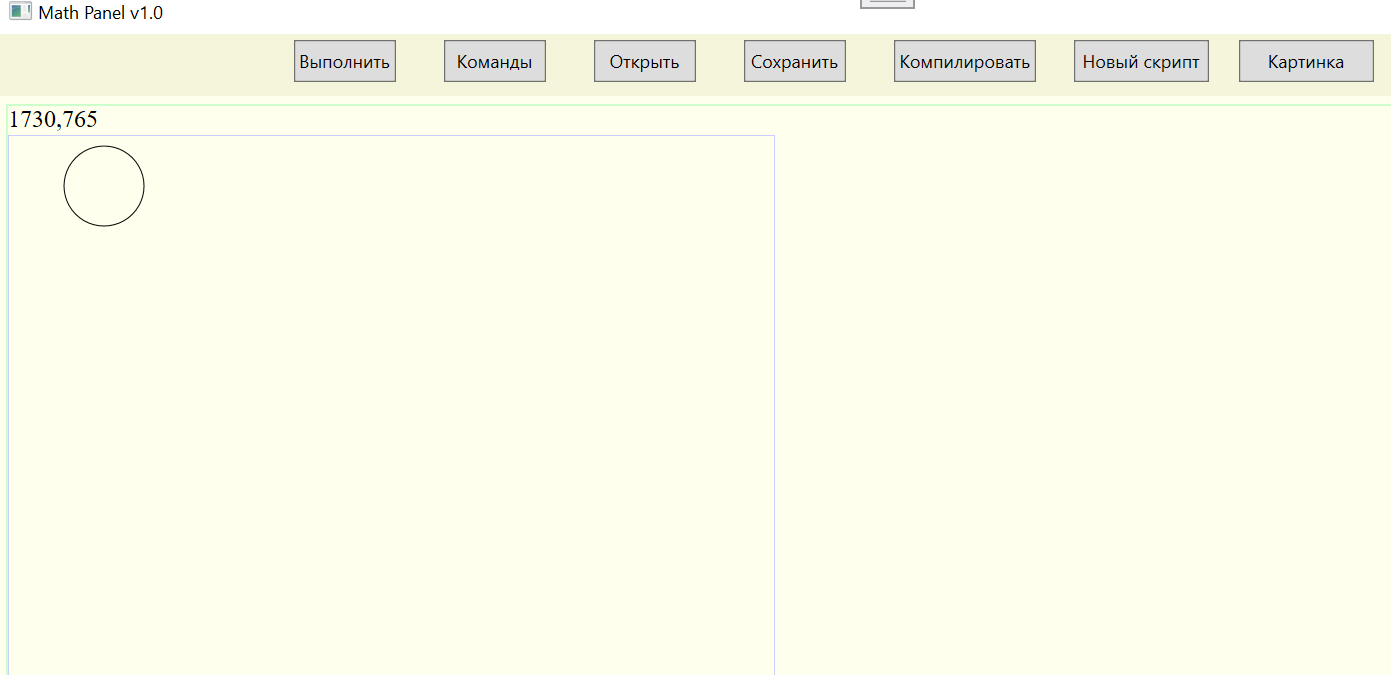


Рис.5.2. Рисование окружности.

Пишем скрипт

Dynamo.Console("test49\_graph\_example3");

Dynamo.GraphExample("3");

Запускаем, получаем текст (изображение на рис. 5.3).

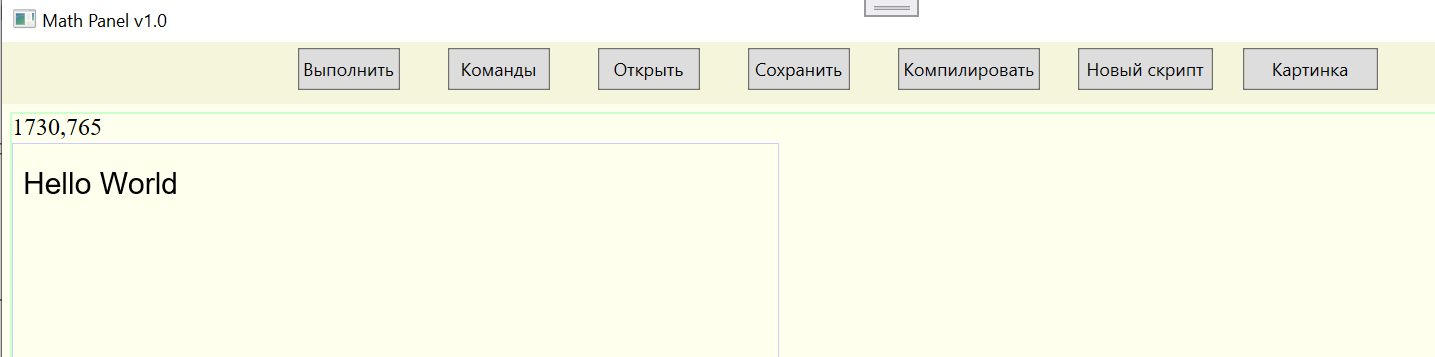


Рис.5.3. Рисование текста.

## 5.2. Обзор «graphix.js» библиотека

Теперь напишем JavaScript псевдо-класс в виде ассоциативного объекта (файл «graphix.js»).

//простой модуль для рисования графиков на "холсте", где левый верхний пиксель (0, 0)

if (typeof GRAPHIX == "undefined") {

GRAPHIX = {

PADDING: 20, //отступ

AX\_BG: "#000000", //фон

AX\_CL: "#00FF00", //цвет осей

borders: 3, //побитовое ИЛИ: 1-горизонтальная граница, 2-вертикальная, 4-горизонтальная сверху, 8-вертикальная сверху

displaySubzero: false, //если размер элемента меньше 1, то при displaySubzero == false, размер будет 0, иначе 1

В коде выше задаем свойства псевдо-класса: отступы, цвет фона, определяем границы и поведение.

//залить фон, нарисовать оси

drawAxes: function(canvas) {

var cnv = document.getElementById(canvas);//найти canvas по id

var w = cnv.width;//ширина холста

var h = cnv.height;//высота холста

var ctx = cnv.getContext('2d');//найти котекст

//залить фон

ctx.fillStyle = this.AX\_BG;

ctx.fillRect(0, 0, w, h);

ctx.strokeStyle = this.AX\_CL;

if (this.borders == 0) return;

//границы

ctx.lineWidth = 1;

ctx.beginPath();

//вертикальная

if( (this.borders & 0x2) > 0 ) {

ctx.moveTo(this.PADDING, this.PADDING);

ctx.lineTo(this.PADDING, h - this.PADDING);

ctx.stroke();

}

//горизонтальная

if( (this.borders & 0x1) > 0 ) {

ctx.moveTo(this.PADDING, h - this.PADDING);

ctx.lineTo(w - this.PADDING, h - this.PADDING);

ctx.stroke();

}

//вертикальная сверху

if( (this.borders & 0x8) > 0 ) {

ctx.moveTo(w - this.PADDING, this.PADDING);

ctx.lineTo(w - this.PADDING, h - this.PADDING);

ctx.stroke();

}

//горизонтальная сверху

if( (this.borders & 0x4) > 0 ) {

ctx.moveTo(this.PADDING, this.PADDING);

ctx.lineTo(w - this.PADDING, this.PADDING);

ctx.stroke();

}

},

В коде выше определяем метод drawAxes для заливки фона и рисования границ.

//отобразить данные

//canvas - id html-элемента, data - двухмерный массив с данными, opt - дополнительные параметры

drawData: function(canvas, data, opt) {

var cnv = document.getElementById(canvas);//найти canvas по id

var w = cnv.width - 2 \* this.PADDING;//ширина без отступа

var h = cnv.height - 2 \* this.PADDING;//высота без отступа

var x0 = opt.x0; //минимальное значение по X-оси

var x1 = opt.x1; //максимальное значение по X-оси

var y0 = opt.y0; //минимальное значение по Y-оси

var y1 = opt.y1; //максимальное значение по Y-оси

var dx = (x1 - x0); //диапазон по X

var dy = (y1 - y0); //диапазон по Y

var sz = "" + opt.size; //размер

if (sz == "undefined") sz = 1;

else sz = sz \* 1;

var clr = "" + opt.clr;//основной цвет

if (clr == "undefined") clr = "#ff0000";

var colorstroke = "" + opt.csk;//цвет для линий

if (colorstroke == "undefined") colorstroke = clr;

var linewidth = "" + opt.lnw;//ширина линий

if (linewidth == "undefined") linewidth = 1;

else linewidth = linewidth \* 1;

var sz\_old;

var style = opt.sty;//стиль

var bLine = false;//признак рисования линии

var height = 10;

var text = "";

var fromzero = "" + opt.fromzero;//1-от оси X

if (fromzero == "undefined") fromzero = 0;

var textonly = "" + opt.textonly;//1-рисовать только текст

if (textonly == "undefined") textonly = 0;

var textdown = "" + opt.textdown;//1-текст под столбиком

if (textdown == "undefined") textdown = 0;

var fontsize = "" + opt.fontsize;//размер шрифта

if (fontsize == "undefined") fontsize = 10;

var textonbottom = "" + opt.textonbottom;//1-текст снизу оси X

if (textonbottom == "undefined") textonbottom = 0;

var ctx = cnv.getContext('2d');//найти котекст

ctx.beginPath();

//заполнить контекст шириной и цветами

ctx.lineWidth = linewidth;

ctx.strokeStyle = colorstroke;

ctx.fillStyle = clr;

ctx.font = fontsize + "px Verdana";

var sqrt2\_2 = Math.sqrt(2) \* 0.3;

var m0;

var bRestoreSize = false, bRestoreColor = false, bRestoreStyle = false, bRestoreLw = false, bRestoreColorstroke = false, bRestoreFont = false;

//проход по данным

for (var i = 0; i < data.length; i++) {

var r = data[i];//каждый элемент тоже массив («точка» с атрибутами)

var x = r[0];

var y = r[1];

if( r.length > 2 && r[2] != null ) {//задан размер

bRestoreSize = true;

sz\_old = sz;

sz = r[2];

}

if (r.length > 3 && r[3] != null) {//задан цвет заполнения

bRestoreColor = true;

ctx.fillStyle = r[3];

}

if (r.length > 4 && r[4] != null) {//задан стиль

bRestoreStyle = true;

style = r[4];

}

if (r.length > 5 && r[5] != null) {//задана высота гистограммы

height = r[5];

}

if (r.length > 6 && r[6] != null) {//задан текст

text = r[6];

}

else text = "";

if (r.length > 7 && r[7] != null) {//задана ширина линий

ctx.lineWidth = r[7];

bRestoreLw = true;

}

if (r.length > 8 && r[8] != null) {//задан цвет линий

ctx.strokeStyle = r[8];

bRestoreColorstroke = true;

}

if (r.length > 9 && r[9] != null) {//задан размер шрифта

ctx.font = r[9] + "px Verdana";

bRestoreFont = true;

}

//найти позицию для элемента данных

var l = Math.round((w \* (x - x0)) / dx + this.PADDING);

var m = Math.round((h - (h \* (y - y0)) / dy) + this.PADDING);

//применить стиль

if (style == "dots") {//прямоугольники

ctx.fillRect(l-sz/2, m-sz/2, sz, sz);

if (text != "") ctx.fillText(text, l, m - 10);

}

else if (style == "circle") {//окружности

ctx.beginPath();

ctx.arc(l, m, sz, 0, 2 \* Math.PI);

ctx.stroke();

ctx.fill();

if (text != "") ctx.fillText(text, l, m - 10);

}

else if (style == "hist") {//гистограмма

m0 = h + this.PADDING;

if (fromzero == 0) {

ctx.moveTo(l, m0);

} else {

m0 = Math.round((h - (h \* (0 - y0)) / dy) + this.PADDING);

ctx.moveTo(l, m0);

}

if (m == m0) m = m0 - 1;

ctx.lineTo(l, m);

ctx.stroke();

if (text != "") {

if (textonbottom == 0)

ctx.fillText(text, l, m - 10);

else ctx.fillText(text, l, m0 + fontsize \* 1);

}

}

else if (style == "hist\_3") {//гистограмма 3Д

if (textonly != 1) {

ctx.beginPath();

if (fromzero == 0) {

m0 = h + this.PADDING;

ctx.moveTo(l, m0);

} else {

m0 = Math.round((h - (h \* (0 - y0)) / dy) + this.PADDING);

ctx.moveTo(l, m0);

}

ctx.lineTo(l, m);

ctx.lineTo(l + sz, m);

ctx.lineTo(l + sz, m0);

ctx.closePath();

ctx.fill();

ctx.stroke();

ctx.beginPath();

ctx.moveTo(l, m);

ctx.lineTo(l + sz, m);

ctx.lineTo(l + sz + sz \* sqrt2\_2, m - sz \* sqrt2\_2);

ctx.lineTo(l + sz \* sqrt2\_2, m - sz \* sqrt2\_2);

ctx.closePath();

ctx.fill();

ctx.stroke();

ctx.beginPath();

ctx.moveTo(l + sz, m);

ctx.lineTo(l + sz + sz \* sqrt2\_2, m - sz \* sqrt2\_2);

ctx.lineTo(l + sz + sz \* sqrt2\_2, m0 - sz \* sqrt2\_2);

ctx.lineTo(l + sz, m0);

ctx.closePath();

ctx.fill();

ctx.stroke();

}

if (text != "") ctx.fillText(text, l, textdown != 1 ? m - 10 : m0 + (10 + (i % 4) \* 12));

}

else if (style == " histmap") {//прямоугольник из точки данных, размер height

ctx.moveTo(l, m);

ctx.lineTo(l, m - height);

ctx.stroke();

if( text != "" ) ctx.fillText(text, l, m - height - 10);

}

else if (style == "line" || style == "line\_end" || style == "line\_endf") {//линия или конец линии

if (!bLine) {

ctx.beginPath();

ctx.moveTo(l, m);

bLine = true;

}

else {

ctx.lineTo(l, m);

}

if (style == "line\_end" || style == "line\_endf") {//конец линии

bLine = false;

if( style == "line\_endf") ctx.fill();//залить путь

ctx.stroke();

}

if (text != "") ctx.fillText(text, l, m - 10);

}

else if (style == "text") {

if (text != "") {

if (textonbottom == 0)

ctx.fillText(text, l, m - fontsize \* 1);//текст над точкой

else ctx.fillText(text, l, m + fontsize \* 1);//текст под точкой

}

}

//восстановить значения по умолчанию

if(bRestoreSize) {

bRestoreSize = false;

sz = sz\_old;

}

if(bRestoreColor) {

bRestoreColor = false;

ctx.fillStyle = clr;

}

if(bRestoreStyle) {

bRestoreStyle = false;

style = opt.sty;

}

if(bRestoreLw) {

bRestoreLw = false;

ctx.lineWidth = linewidth;

}

if (bRestoreColorstroke) {

bRestoreColorstroke = false;

ctx.strokeStyle = colorstroke;

}

if (bRestoreFont) {

bRestoreFont = false;

ctx.font = fontsize + "px Verdana";

}

}

if (bLine) ctx.stroke();

},

В коде выше определяем метод drawData для рисования «точек». Параметр data - двухмерный массив с данными, параметр opt – массив дополнительных параметров, используемых по умолчанию. data можно представить в виде массива, каждый элемент («точка») тоже является массивом вида

new Array(x, y, sz, clr, style, height, text, linewidth, colorstroke, fontsize), где

x – горизонтальная позиция на холсте,

y – вертикальная позиция на холсте,

sz – размер рисования,

clr – основной цвет заливки,

style – стиль рисования,

height – высота элементов гистограмм,

text – текст для точки,

linewidth – ширина линии,

colorstroke – цвет линии,

fontsize – размер шрифта.

Стиль рисования точек может иметь следующие значения:

dots – прямоугольники,

circle – кружки,

hist – гистограмма,

hist\_3 - гистограмма объемная (3Д),

histmap - прямоугольник из точки данных, размер height,

line – линия,

line\_end - конец линии,

line\_ endf - конец линии и заливка многоугольника,

text – вывод только текста в местоположении точки.

opt - ассоциативный массив, который может иметь атрибуты:

x0 - минимальное значение по X-оси,

x1 - максимальное значение по X-оси,

y0 - минимальное значение по Y-оси,

y1 - максимальное значение по Y-оси,

size – размер рисования,

clr - основной цвет,

csk - цвет для линий,

lnw - ширина линий,

sty – стиль рисования,

fromzero – гистограмма от оси X,

textonly - рисовать только текст,

textdown - текст под столбиком,

fontsize - размер шрифта,

textonbottom - текст снизу оси X.

.

//отобразить текст

drawText: function(canvas, title, x, y, color, size) {

var cnv = document.getElementById(canvas);

var ctx = cnv.getContext('2d');

ctx.fillStyle = color;

ctx.font = size + "px Verdana";

ctx.fillText(title, x, y);

},

В коде выше определяем метод drawText для рисования текста.

//отобразить картинку

drawImage: function(canvas, imgid, x, y) {

var cnv = document.getElementById(canvas);

var ctx = cnv.getContext('2d');

var img = document.getElementById(imgid);

if( img != null ) ctx.drawImage(img, x, y);

},

В коде выше определяем метод drawImage для рисования картинки (она должна быть в html-документе).

//отобразить json-string

drawJson: function (canvas, w) {

//преобразовать из строки в объект

if ("" + JSON != "undefined") s = JSON.parse(w);

else if ("" + jQuery != "undefined") s = jQuery.parseJSON(w);

else s = eval(w);

var dpa = new Date().getTime();

this.drawJsonObj(canvas, s); //отобразить json-object

return dpa;

},

В коде выше определяем метод drawJson для преобразования строки в JSON-объект и его рисования .

//отобразить json-array

drawJsonArr: function (canvas, sArr) {

//var sArr = JSON.parse(w);//eval(w);

var dpa = new Date().getTime();

for (var i = 0; i < sArr.length; i++) {

if (i > 0) sArr[i]["options"]["second"] = true;

this.drawJsonObj(canvas, sArr[i]); //отобразить json-object

}

return dpa;

},

В коде выше определяем метод drawJsonArr для рисования массива JSON-объектов.

//отобразить json-object

drawJsonObj: function (canvas, s) {

var cnv = document.getElementById(canvas);

var ctx = cnv.getContext('2d');

//alert(s + ",o3=" + s["options"]["x0"] + ",o2=" + s["data"][0]["x"]);

var opt = s["options"];

//установить размеры холста

width\_x = (opt["wid"]);

height\_x = (opt["hei"]);

if (cnv.width != width\_x) {

cnv.width = width\_x;

}

if (cnv.height != height\_x) {

cnv.height = height\_x;

}

//определить сжатие

var scale = cnv.width / (opt["x1"] - opt["x0"]);

//создать пустой массив

var dd = new Array();

for (var i = 0; i < s["data"].length; i++) {

var row = s["data"][i];//это массив

var x = row["x"];//позиция по горизонтали

var y = row["y"];//позиция по вертикали

//размер точки

var sz = ("" + row["rad"]);

if( sz == "undefined" ) sz = null;

else {

sz = sz \* scale;

if (sz < 1) sz = this.displaySubzero ? 1 : 0;

}

//цвет

var clr = ("" + row["clr"]);

if( clr == "undefined" ) clr = null;

//стиль

var style = ("" + row["sty"]);

if (style == "undefined") style = null;

//высота гистограммы

var height = ("" + row["hei"]);

if (height == "undefined") height = null;

//текст

var text = ("" + row["txt"]);

if (text == "undefined") text = null;

//ширина линии

var linewidth = ("" + row["lnw"]);

if (linewidth == "undefined") linewidth = null;

//цвет линии

var colorstroke = ("" + row["csk"]);

if (colorstroke == "undefined") colorstroke = null;

//размер шрифта

var fontsize = ("" + row["fontsize"]);

if (fontsize == "undefined") fontsize = null;

//запихнуть описание точки в массив

dd.push(new Array(x, y, sz, clr, style, height, text, linewidth, colorstroke, fontsize));

}

//фон для холста?

if( ("" + opt["bg"]) != "undefined" ) {

this.AX\_BG = opt["bg"];

}

//рисовать поверх?

if( ("" + opt["second"]) == "undefined" ) {

//рисовать фон и границы

this.AX\_CL = "#ffff00";

this.PADDING = 5;

this.borders = 0;//15;

this.drawAxes(canvas);

}

//задать картинку?

if (("" + opt["img"]) != "undefined") {

var obj = document.getElementById("img1");

if (obj != null) try {

obj.src = opt["img"];

this.drawImage(canvas, "img1", 0, 0);

} catch (e) { };

}

//отрисовать данные

this.drawData(canvas, dd, opt);

},

В коде выше определяем метод drawJsonObj, который преобразует JSON-объектов в двухмерный массив для рисования «точек» через метод drawData.

}

}

Применяем закрывающие скобки для ассоциативного массива GRAPHIX.

## 5.3. Простой тестовый контейнер для библиотеки

Теперь проведем тестирование нашей новой библиотеки. Создаем файл “test\_graphix.htm”.

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<title>Test graphix library</title>

<style type="text/css">

body { background-color:#fff; }

#file1 { width: 45%; height:100px; margin-right:10px; float:left; border:1px solid #ccc; }

#result {width: 45%; height: 100px; float:left; border:1px solid #ccc;}

#canvas1 { border:1px solid #cfc; }

h1 {font-size:20px;}

</style>

<script type="text/javascript" src="jquery-1.11.1.min.js"></script>

</head>

<body onload="my\_onload()">

<h1>Test GRAPHIX library</h1>

<canvas id="canvas1" width="800" height="600"></canvas>

<h3>Население стран</h3>

<canvas id="canvas2" width="1200" height="640"></canvas>

<img id="wiz" src="13\_wizard\_small.gif" width="1" />

<img id="map" src="russia.png" width="1" />

<script type="text/javascript" src="graphix.js"></script>

<script type="text/javascript">

function my\_onload() {

//задаем желтые оси, небольшой отступ

GRAPHIX.AX\_CL = "#ffff00";

GRAPHIX.PADDING = 5;

GRAPHIX.borders = 15;

GRAPHIX.drawAxes('canvas1');

//первый график - тонкие синие гистограммы

var data = [

[20, 100],

[100, 160],

[200, 100],

[300, 200],

[390, 300],

];

var opt = { "x0": 0, "x1": 400, "y0": 0, "y1": 400, "clr": "#0000ff", "sty": "hist", "size":"3" };

GRAPHIX.drawData('canvas1', data, opt);

//quadratic - второй график - квадратичная зависимость

var opt2 = { "x0": 0, "x1": 1, "y0": 0, "y1": 1.5, "clr": "#00ffff", "sty": "dots", "size": "3" };

var data2 = new Array();

for (var j = 0; j < 100; j++) {

var t = j \* 0.01;

var t2 = t \* t;

data2.push(new Array(t, t \* (-t2 \* t + 4 \* t2 - 6 \* t + 4)));

}

GRAPHIX.drawData('canvas1', data2, opt2);

//cubic - третий график - кривая 3-го порядка

var opt3 = { "x0": 0, "x1": 1, "y0": 0, "y1": 1.5, "clr": "#ff00ff", "sty": "line" };

var data3 = new Array();

for (var j = 0; j < 100; j++) {

var t = j \* 0.01;

var t2 = t \* t;

data3.push(new Array(t, t \* (4 \* t \* t - 9 \* t + 6)));

}

GRAPHIX.drawData('canvas1', data3, opt3);

//elastic - четвертый график - кривая 4-го порядка

var opt4 = { "x0": 0, "x1": 1, "y0": 0, "y1": 1.5, "clr": "#ffffff", "sty": "line" };

var data4 = new Array();

for (var j = 0; j < 100; j++) {

var t = j \* 0.01;

var t2 = t \* t;

data4.push(new Array(t, t \* (33 \* t2 \* t2 - 106 \* t2 \* t + 126 \* t2 - 67 \* t + 15)));

}

GRAPHIX.drawData('canvas1', data4, opt4);

//выводим текст и картинку

GRAPHIX.drawText('canvas1', "quadratic", 40, 40, "#00ffff", 20);

GRAPHIX.drawText('canvas1', "cubic", 40, 60, "#ff00ff", 20);

GRAPHIX.drawText('canvas1', "elastic", 40, 80, "#ffffff", 20);

GRAPHIX.drawText('canvas1', "random linear", 40, 100, "#00ff00", 20);

GRAPHIX.drawImage('canvas1', "wiz", 716, 6);

//random linear - случайная линия

var opt5 = { "x0": 0, "x1": 1, "y0": 0, "y1": 2, "csk": "#00ff00", "clr": "#ff0000", "sty": "circle", "size":"4" };

var data5 = new Array();

for (var j = 0; j <= 20; j++) {

var t = j \* 0.05;

var t2 = t + Math.random() \* 0.15;

if( j >= 10 && j <= 15)

data5.push(new Array(t, t2, null, null, j == 15 ? "line\_end" : "line"));

else {

if( j >= 4 && j <= 6)

data5.push(new Array(t, t2, null, null, j == 6 ? "line\_endf" : "line"));

else

data5.push(new Array(t, t2));

}

}

GRAPHIX.drawData('canvas1', data5, opt5);

//canvas2 - гистограмма с населением стран

GRAPHIX.drawAxes('canvas2');

GRAPHIX.drawImage('canvas2', "map", 0, 0);

var data3d = [

[20, 300, null, null, null, 10, "10 млн."],//Финляндия

[50, 180, null, null, null, 10, "9.5 млн."],//Беларусь

[105, 125, null, null, null, 44, "44 млн."],//Украина

[275, 115, null, null, null, 35, "35 млн."],//Казахстан

[540, 90, null, null, null, 5, "5 млн."],//Монголия

[520, 285, null, null, null, 145, "146 млн."],//Россия

];

var opt3d = { "x0": 0, "x1": 1200, "y0": 0, "y1": 640, "clr": "#0099cc", "sty": "histmap", "lnw":"10" };

GRAPHIX.drawData('canvas2', data3d, opt3d);

return false;

}

</script>

</body>

</html>

В нем определяются два холста (canvas). В первом строим синюю гистограмму, кривую 2-го порядка сине-зеленым, кривую 3-го порядка фиолетовым, кривую 4-го порядка белым,

смесь кружков и линий с распределением близким к линейному (рис. 5.4).

На втором холсте загружаем карту России с ближайшими соседями и строим гистограмму населения (рис.5.5).

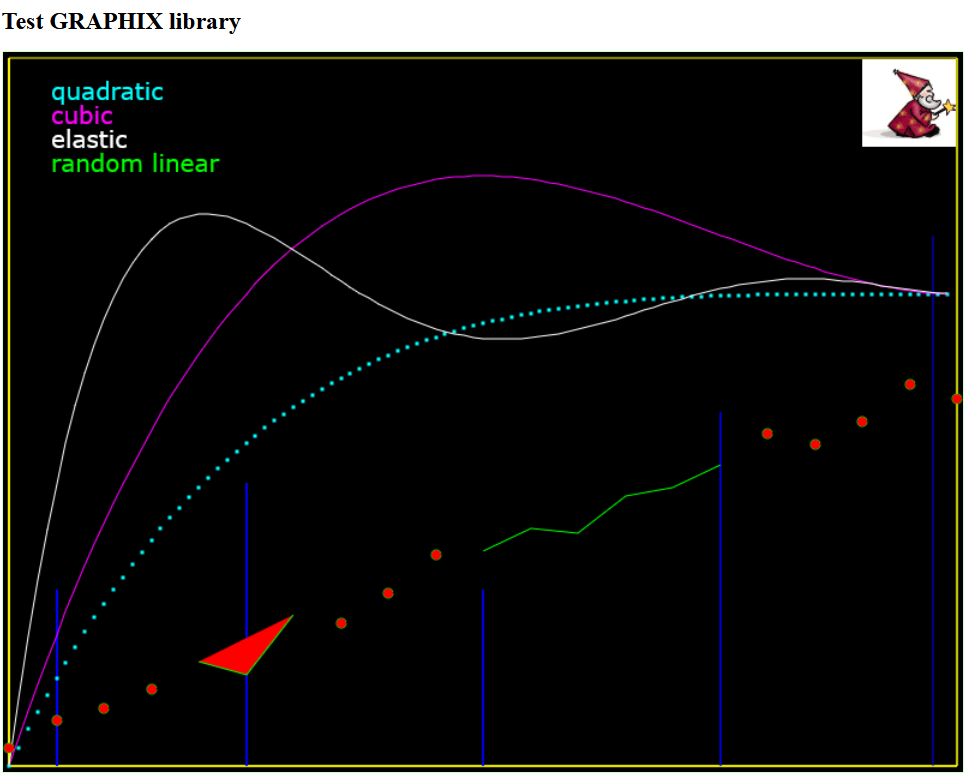


Рис. 5.4. Графики на первом холсте

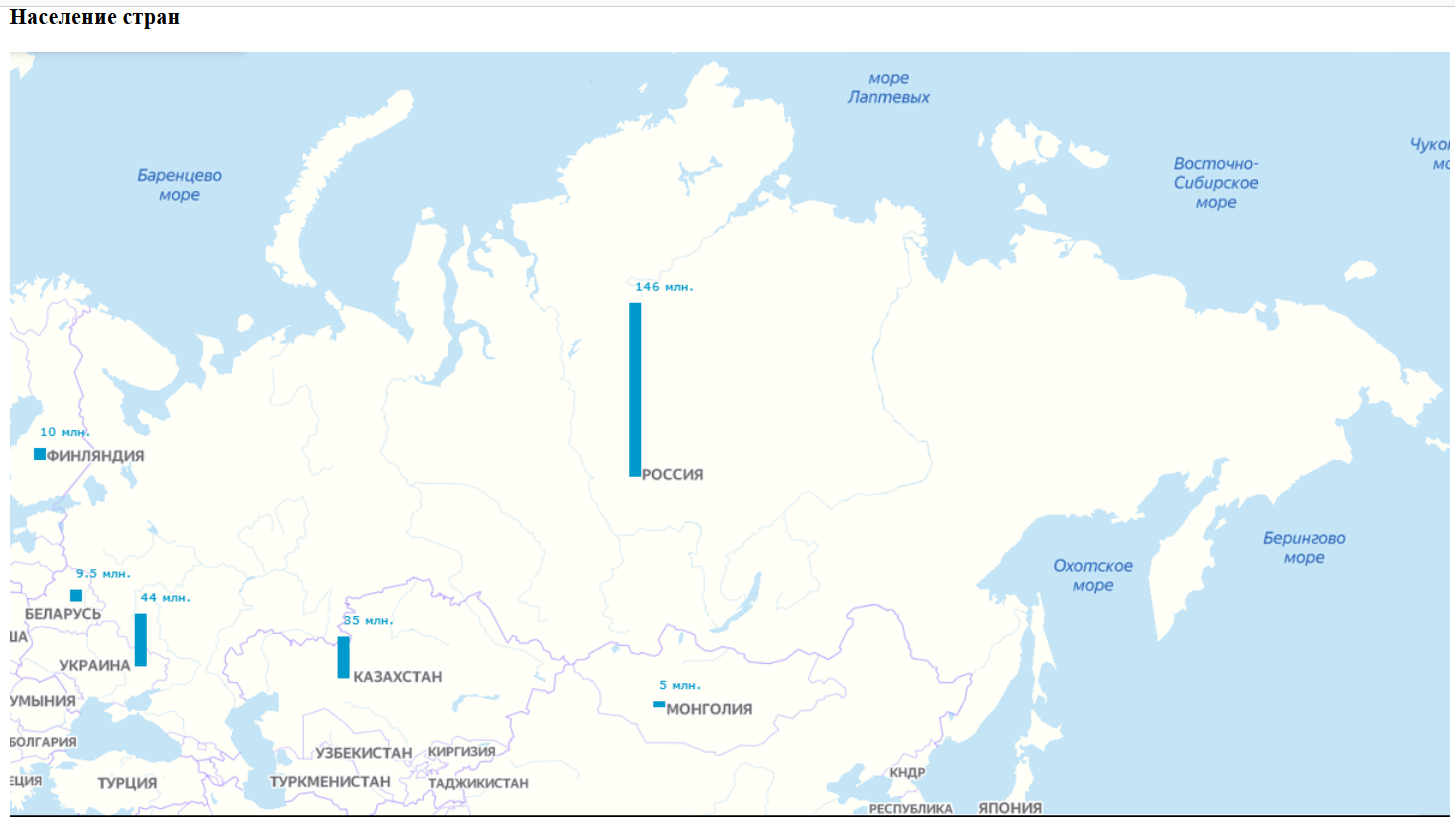


Рис.5.5. Население России и соседей.

## 5.4. Класс QuadroEqu для упрощения подготовки данных

Создадим новый класс QuadroEqu для решения уравнений (пока квадратного) и рисования примитивов (на основе подготовки данных для вывода через JSON в GRAPHIX).

using System;

using MathPanel;

namespace MathPanelExt

{

/// <summary>

/// класс для решения квадратного уравнения и рисования примитивов (на основе подготовки данных в вывода через JSON в GRAPHIX

/// </summary>

public class QuadroEqu

{

public QuadroEqu()

{

}

/// <summary>

/// корни квадратного уравнения

/// </summary>

public static void Solve(double a, double b, double c, out double x1, out double x2)

{

double discr = Math.Sqrt(b \* b - 4 \* a \* c);

x1 = (-b - discr) / (2 \* a);

x2 = (-b + discr) / (2 \* a);

}

//форматы вывода

static readonly string sXY = "{{\"x\":{0},\"y\":{1}}}";

static readonly string sClr = "{{\"x\":{0},\"y\":{1},\"clr\":\"{2}\"}}";

static readonly string sTxt = "{{\"x\":{0},\"y\":{1}, \"txt\":\"{2}\", \"sty\":\"{3}\"}}";

static readonly string sFull =

"{{\"x\":{0},\"y\":{1}, \"txt\":\"{2}\", \"sty\":\"{3}\", \"clr\":\"{4}\", \"rad\":\"{5}\", \"fontsize\":\"{6}\"}}";

/// <summary>

/// подготовка данных для квадратного уравнения

/// </summary>

public static string DrawRange(double a, double b, double c, double x0, double x1, int n)

{

System.Text.StringBuilder sb = new System.Text.StringBuilder();

double step = (x1 - x0) / n;

double x = x0;

for (int i = 0; i <= n; i++, x += step)

{

var y = x \* (a \* x + b) + c;

if (i != 0) sb.Append(",");

sb.AppendFormat(sXY, Dynamo.D2S(x), Dynamo.D2S(y));

}

return sb.ToString();

}

}}

Здесь мы определяем метод Solve для поиска корней квадратного уравнения и метод DrawRange для подготовки данных для визуализации квадратного уравнения.

Напишем простой скрипт для вывода этих данных.

double x1 = 0, x2 = 0;

MathPanelExt.QuadroEqu.Solve(1, 0, -1, out x1, out x2);

Dynamo.Console(string.Format("x1 {0}, x2 {1}", x1, x2));

var s1 = MathPanelExt.QuadroEqu.DrawRange(1, 0, -1, -3, 3, 10);

Dynamo.Console(s1);

На выходе получаем

x1 -1, x2 1

{"x":-3,"y":8},{"x":-2.4,"y":4.76},{"x":-1.8,"y":2.24},{"x":-1.2,"y":0.44},{"x":-0.6,"y":-0.64},{"x":2.22E-16,"y":-1},{"x":0.6,"y":-0.64},{"x":1.2,"y":0.44},{"x":1.8,"y":2.24},{"x":2.4,"y":4.76},{"x":3,"y":8}

Первая строка – значение корней уравнения X\*X – 1 = 0.

Вторая строка – JSON данные для значений этого уравнения в диапазоне X от -3 до 3.

Теперь добавим в MainWindow.xaml.cs метод

/// <summary>

/// передать JSON данные для визуализации в canvas

/// </summary>

public static void SceneJson(string s\_json, bool bSecond = false)

{

if (!bReady || dispObj.HasShutdownStarted) return;

screenJson = s\_json;

//мы запускаем код в UI потоке

dispObj.Invoke(delegate

{

webConsole.InvokeScript("ext\_json", screenJson, bSecond);

});

}

А в «test\_graph.htm» добавим метод

function ext\_json(w) {

var d1 = new Date().getTime();

var dpa = GRAPHIX.drawJson("canvas2", w);

var d2 = new Date().getTime();

return false;

}

Теперь мы сможем передавать в холст правильно подготовленные данные, т.е. рисовать из скрипта. Дополним наш скрипт.

string s2 = "{\"options\":{\"x0\": -3.5, \"x1\": 3.5, \"y0\": -3, \"y1\": 10, \"clr\": \"#ff0000\", \"sty\": \"circle\", \"size\":3, \"lnw\": 3, \"wid\": 800, \"hei\": 600 }";

s2 += ", \"data\":[" + s1 + "]}";

Dynamo.SceneJson(s2);

Запускаем скрипт на выполнение, получаем в окне сообщений

{"options":{"x0": -3.5, "x1": 3.5, "y0": -3, "y1": 10, "clr": "#ff0000", "sty": "circle", "size":3, "lnw": 3, "wid": 800, "hei": 600 }, "data":[{"x":-3,"y":8},{"x":-2.4,"y":4.76},{"x":-1.8,"y":2.24},{"x":-1.2,"y":0.44},{"x":-0.6,"y":-0.64},{"x":2.22E-16,"y":-1},{"x":0.6,"y":-0.64},{"x":1.2,"y":0.44},{"x":1.8,"y":2.24},{"x":2.4,"y":4.76},{"x":3,"y":8}]}

Переключаемся на «График» - видим часть уравнения (рис.5.6)

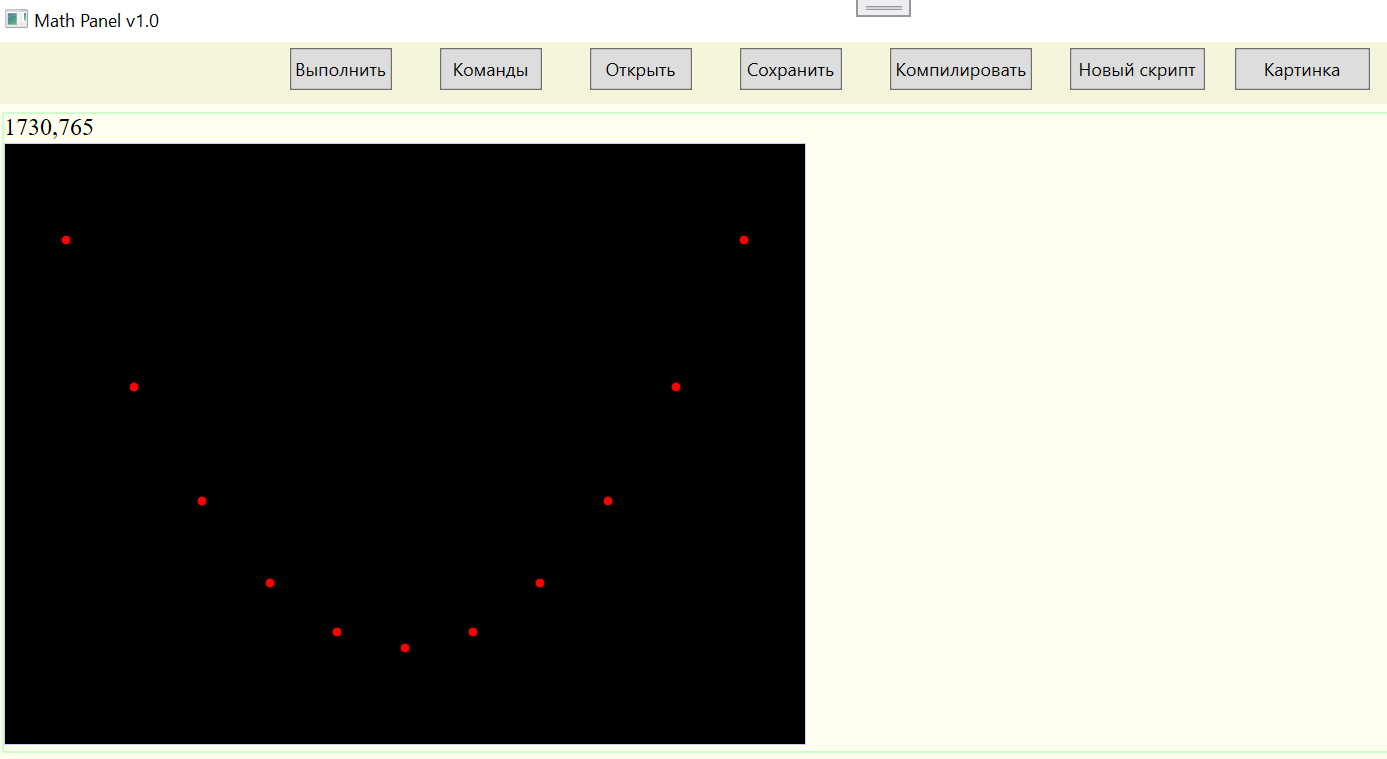


Рис.5.6. Дискретные значения уравнения Y = X\*X – 1.

Формат данных для визуализации – ассоциативный массив с двумя атрибутами. "options" описывает как надо показывать данные. "data" описывает данные как массив элементов, каждый из которых есть ассоциативный массив.

Атрибуты options:

x0 - минимальное значение по X-оси,

x1 - максимальное значение по X-оси,

y0 - минимальное значение по Y-оси,

y1 - максимальное значение по Y-оси,

size – размер рисования,

clr - основной цвет,

csk - цвет для линий,

lnw - ширина линий,

sty – стиль рисования,

wid – размер канваса по горизонтали,

hei – размер канваса по вертикали.

Атрибуты элемента внутри data («точки»):

x – физическое значение по X-оси (отличается от положения в холсте),

y – физическое значение по Y-оси (отличается от положения в холсте),

txt – текст,

sty – стиль точки,

clr – цвет,

rad – физический размер,

fontsize – размер шрифта.

Добавим новые методы в QuadroEqu.

/// <summary>

/// подготовка данных для эллипса

/// </summary>

public static string DrawEllipse(double a, double b, double x0, double y0, double fi0, double fi1, int n)

{

System.Text.StringBuilder sb = new System.Text.StringBuilder();

double step = (fi1 - fi0) / n;

double fi = fi0;

for (int i = 0; i <= n; i++, fi += step)

{

var x = x0 + a \* Math.Cos(fi);

var y = y0 + b \* Math.Sin(fi);

if (i != 0) sb.Append(",");

sb.AppendFormat(sXY, Dynamo.D2S(x), Dynamo.D2S(y));

}

return sb.ToString();

}

/// <summary>

/// подготовка данных для линии

/// </summary>

public static string DrawLine(double x0, double y0, double x1, double y1)

{

System.Text.StringBuilder sb = new System.Text.StringBuilder();

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

{

var x = i == 0 ? x0 : x1;

var y = i == 0 ? y0 : y1;

if (i != 0) sb.Append(",");

sb.AppendFormat(sXY, Dynamo.D2S(x), Dynamo.D2S(y));

}

return sb.ToString();

}

/// <summary>

/// вычиcлить точку Bezier 1-го уровня для заданного шага

/// </summary>

static double Bezier1(double x0, double x1, double step)

{

return x0 + (x1 - x0) \* step;

}

/// <summary>

/// вычиcлить точку Bezier 2-го уровня для заданного шага

/// </summary>

static double Bezier2(double x0, double x1, double x2, double step)

{

double p0 = Bezier1(x0, x1, step);

double p1 = Bezier1(x1, x2, step);

return Bezier1(p0, p1, step);

}

/// <summary>

/// вычиcлить точку Bezier 3-го уровня для заданного шага

/// </summary>

static double Bezier3(double x0, double x1, double x2, double x3, double step)

{

double p0 = Bezier2(x0, x1, x1, step);

double p1 = Bezier2(x1, x2, x3, step);

return Bezier1(p0, p1, step);

}

/// <summary>

/// подготовка данных Bezier 2-го уровня

/// </summary>

public static string DrawBezier2(double x0, double y0, double x1, double y1, double x2, double y2, int n)

{

System.Text.StringBuilder sb = new System.Text.StringBuilder();

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

{

double step = ((double)i) / n;

var x = Bezier2(x0, x1, x2, step);

var y = Bezier2(y0, y1, y2, step);

if (i != 0) sb.Append(",");

sb.AppendFormat(sXY, Dynamo.D2S(x), Dynamo.D2S(y));

}

return sb.ToString();

}

/// <summary>

/// подготовка данных Bezier 3-го уровня

/// </summary>

public static string DrawBezier3(double x0, double y0, double x1, double y1, double x2, double y2, double x3, double y3, int n)

{

System.Text.StringBuilder sb = new System.Text.StringBuilder();

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

{

double step = ((double)i) / n;

var x = Bezier3(x0, x1, x2, x3, step);

var y = Bezier3(y0, y1, y2, y3, step);

if (i != 0) sb.Append(",");

sb.AppendFormat(sXY, Dynamo.D2S(x), Dynamo.D2S(y));

}

return sb.ToString();

}

/// <summary>

/// заполняет цветами ячейки таблицы типа битмап

/// </summary>

public static string DrawBitmap(int rows, int cols, System.Drawing.Color[] clrs,

int xShift = 0, int yShift = 0, bool bFromBottom = true)

{

System.Text.StringBuilder sb = new System.Text.StringBuilder();

for (int i = 0; i < clrs.Length && i < rows \* cols; i++)

{

var x = i % cols + xShift;

var y = bFromBottom ? (i / cols + yShift) : (rows - 1 - i / cols + yShift);

if (i != 0) sb.Append(",");

sb.AppendFormat(sClr, Dynamo.D2S(x), Dynamo.D2S(y), MathPanel.Facet3.ColorHtml(clrs[i]));

}

return sb.ToString();

}

/// <summary>

/// подготовка данных рисования текста

/// </summary>

public static string DrawText(double x, double y, string text)

{

return string.Format(sTxt, Dynamo.D2S(x), Dynamo.D2S(y), text, "circle");

}

/// <summary>

/// подготовка данных для точки

/// </summary>

public static string DrawPoint(double x, double y, string text, string style)

{

return string.Format(sTxt, Dynamo.D2S(x), Dynamo.D2S(y), text, style);

}

/// <summary>

/// подготовка данных для точки, все параметры

/// </summary>

public static string DrawPoint(double x, double y, string text, string style, string clr, string pointsize, string fontsize)

{

return string.Format(sFull, Dynamo.D2S(x), Dynamo.D2S(y), text, style, clr, pointsize, fontsize);

}

/// <summary>

/// подготовка данных для графика

/// </summary>

public static string DrawGraphic(double x0, double x1, double[] yArr, int n)

{

System.Text.StringBuilder sb = new System.Text.StringBuilder();

double step = (x1 - x0) / (n > 1 ? n - 1 : 1);

double x = x0;

for (int i = 0; i < n; i++, x += step)

{

var y = yArr[i];

if (i != 0) sb.Append(",");

sb.AppendFormat(sXY, Dynamo.D2S(x), Dynamo.D2S(y));

}

return sb.ToString();

}

/// <summary>

/// подготовка данных для прямоугольника

/// </summary>

public static string DrawRect(double x0, double y0, double x1, double y1, bool bFill)

{

System.Text.StringBuilder sb = new System.Text.StringBuilder();

sb.AppendFormat(sXY, Dynamo.D2S(x0), Dynamo.D2S(y0));

sb.Append(",");

sb.AppendFormat(sXY, Dynamo.D2S(x1), Dynamo.D2S(y0));

sb.Append(",");

sb.AppendFormat(sXY, Dynamo.D2S(x1), Dynamo.D2S(y1));

sb.Append(",");

if (bFill) sb.AppendFormat(sTxt, Dynamo.D2S(x0), Dynamo.D2S(y1), "", "line\_endf");

else

{

sb.AppendFormat(sXY, Dynamo.D2S(x0), Dynamo.D2S(y1));

sb.Append(",");

sb.AppendFormat(sTxt, Dynamo.D2S(x0), Dynamo.D2S(y0), "", "line\_end");

}

return sb.ToString();

}

Добавим в MainWindow.xaml.cs метод

/// <summary>

/// преобразовать вещественное число в строку с точностью 4 знака после запятой

/// </summary>

/// <param name="d">число</param>

public static string D2S(double d)

{

string s = d.ToString("G4", CultureInfo.InvariantCulture.NumberFormat);

return s;

}

## 5.5. Рисование с помощью примеров сценариев

Напишем скрипт с использованием новых методов.

//test17\_use\_quadro

//красная кривая 2-го порядка - подбородок

var s1 = MathPanelExt.QuadroEqu.DrawRange(1, 0, -1, -2, 2, 24 );

string s2 = "{\"options\":{\"x0\": -3, \"x1\": 3, \"y0\": -3, \"y1\": 9, \"clr\": \"#ff0000\", \"sty\": \"circle\", \"size\":10, \"lnw\": 3, \"wid\": 800, \"hei\": 600 }";

s2 += ", \"data\":[" + s1 + "]}";

Dynamo.SceneJson(s2);

//зеленая кривая 2-го порядка - усы

var s3 = MathPanelExt.QuadroEqu.DrawRange(-2, 0, 1, -1.8, 2, 24);

string s4 = "{\"options\":{\"x0\": -3, \"x1\": 3, \"y0\": -3, \"y1\": 9, \"clr\": \"#00ff00\", \"sty\": \"line\", \"size\":5, \"lnw\": 3, \"wid\": 800, \"hei\": 600, \"second\":1 }";

s4 += ", \"data\":[" + s3 + "]}";

Dynamo.SceneJson(s4, true);

//синий эллипс - глаз

var s5 = MathPanelExt.QuadroEqu.DrawEllipse(1.1, 1, -1.7, 4, 0, Math.PI \* 2, 24);

string s6 = "{\"options\":{\"x0\": -3, \"x1\": 3, \"y0\": -3, \"y1\": 9, \"clr\": \"#0000ff\", \"sty\": \"line\", \"size\":5, \"lnw\": 3, \"wid\": 800, \"hei\": 600, \"second\":1 }";

s6 += ", \"data\":[" + s5 + "]}";

Dynamo.SceneJson(s6, true);

//оранжевый эллипс - глаз

var s7 = MathPanelExt.QuadroEqu.DrawEllipse(1, 2.5, 1.5, 4, 0, Math.PI \* 2, 24);

string s8 = "{\"options\":{\"x0\": -3, \"x1\": 3, \"y0\": -3, \"y1\": 9, \"clr\": \"#ffaa00\", \"sty\": \"line\", \"size\":5, \"lnw\": 3, \"wid\": 800, \"hei\": 600, \"second\":1 }";

s8 += ", \"data\":[" + s7 + "]}";

Dynamo.SceneJson(s8, true);

//желтая линия - пробор

var s9 = MathPanelExt.QuadroEqu.DrawLine(-2.5, 6, 1.5, 7);

string s10 = "{\"options\":{\"x0\": -3, \"x1\": 3, \"y0\": -3, \"y1\": 9, \"clr\": \"#ffff00\", \"sty\": \"line\", \"size\":5, \"lnw\": 3, \"wid\": 800, \"hei\": 600, \"second\":1 }";

s10 += ", \"data\":[" + s9 + "]}";

Dynamo.SceneJson(s10, true);

//фиолетовая кривая Безье 2-го порядка - бровь

var s11 = MathPanelExt.QuadroEqu.DrawBezier2(0, 4, 1.5, 7, 3, 5, 10);

string s12 = "{\"options\":{\"x0\": -3, \"x1\": 3, \"y0\": -3, \"y1\": 9, \"clr\": \"#ff00ff\", \"sty\": \"line\", \"size\":5, \"lnw\": 3, \"wid\": 800, \"hei\": 600, \"second\":1 }";

s12 += ", \"data\":[" + s11 + "]}";

Dynamo.SceneJson(s12, true);

//белая кривая Безье 3-го порядка - шляпа

var s13 = MathPanelExt.QuadroEqu.DrawBezier3(-3, 6, -1, 10, 1, 5, 3, 7, 20);

string s14 = "{\"options\":{\"x0\": -3, \"x1\": 3, \"y0\": -3, \"y1\": 9, \"clr\": \"#ffffff\", \"sty\": \"line\", \"size\":5, \"lnw\": 3, \"wid\": 800, \"hei\": 600, \"second\":1 }";

s14 += ", \"data\":[" + s13 + "]}";

Dynamo.SceneJson(s14, true);

//сине-зеленая линия - нос

var s15 = MathPanelExt.QuadroEqu.DrawLine(-0.5, 2, 0.3, 6);

string s16 = "{\"options\":{\"x0\": -3, \"x1\": 3, \"y0\": -3, \"y1\": 9, \"clr\": \"#00ffff\", \"sty\": \"line\", \"size\":5, \"lnw\": 3, \"wid\": 800, \"hei\": 600, \"second\":1 }";

s16 += ", \"data\":[" + s15 + "]}";

Dynamo.SceneJson(s16, true);

Запустим и получим рожицу как на рис.5.7.

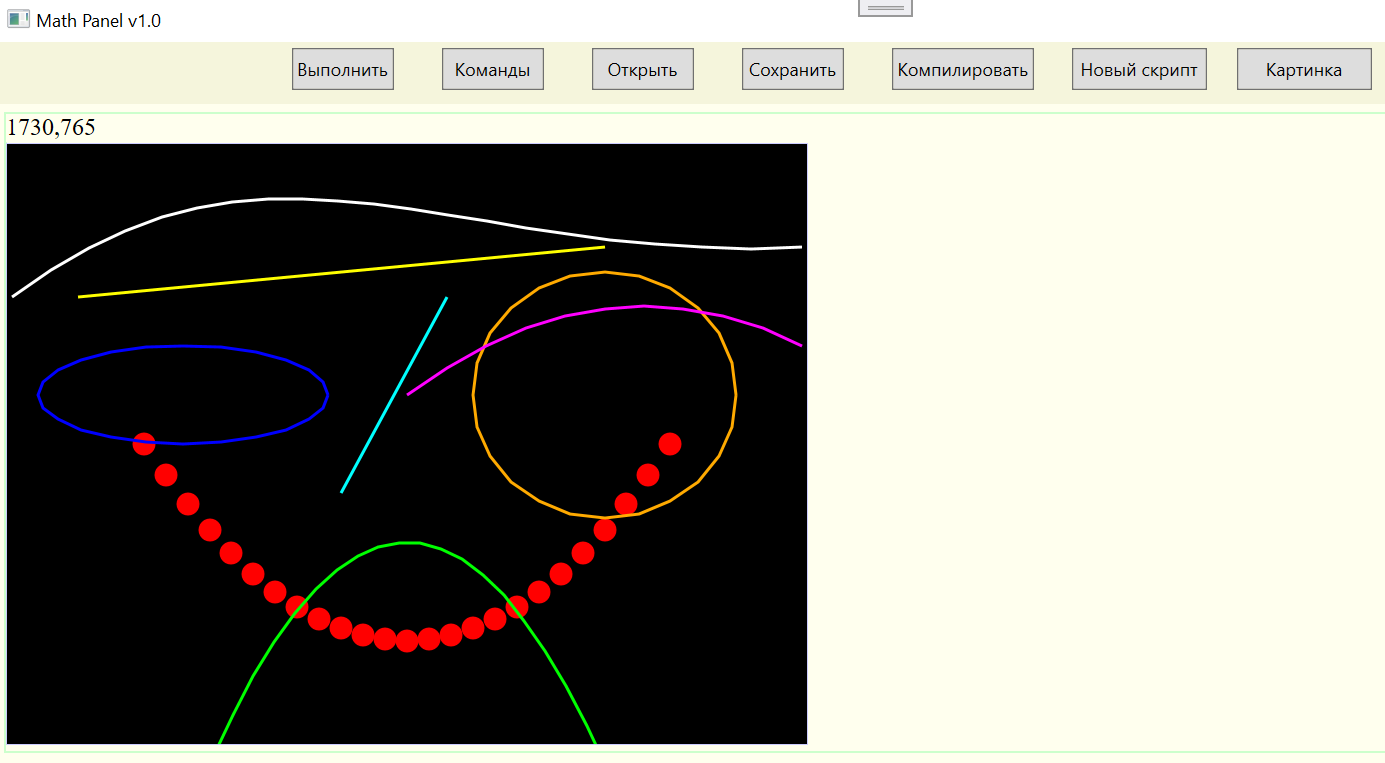


Рис.5.7. Результат рисования из скрипта.

## 5.6. Цветок жизни

В заключение нарисуем красивую фигуру – «цветок жизни» (относят к сакральной геометрии). Напишем скрипт. В нем рисуем большую окружность, потом 6 малых с вращением вокруг центра и малую окружность по центру.

//рисуем цветок

string[] clr =

{

"#ff0000", "#ffaa00", "#ffff00", "#00ff00", "#0000ff", "#ff00ff"

};

double rad = 4;

//большая окружность

string s = MathPanelExt.QuadroEqu.DrawEllipse(rad \* 2, rad \* 2, 0, 0, 0, Math.PI \* 2, 64);

string s10 = "{\"options\":{\"x0\": -10, \"x1\": 10, \"y0\": -10, \"y1\": 10, \"clr\": \"#ffffff\", \"sty\": \"line\", \"size\":0, \"lnw\": 3, \"fontsize\":24, \"wid\": 800, \"hei\": 800, \"\_second\":1 }";

s10 += ", \"data\":[" + s + "]}";

Dynamo.SceneJson(s10, true);

//6 малых окружностей вращаем

for (i = 0; i < 6; i++)

{

double x = rad \* Math.Cos((Math.PI \* i) / 3);

double y = rad \* Math.Sin((Math.PI \* i) / 3);

s = MathPanelExt.QuadroEqu.DrawEllipse(rad, rad, x, y, 0, Math.PI \* 2, 64);

s10 = "{\"options\":{\"x0\": -10, \"x1\": 10, \"y0\": -10, \"y1\": 10, \"clr\": \"" + clr[i] + "\", \"sty\": \"line\", \"size\":0, \"lnw\": 3, \"fontsize\":24, \"wid\": 800, \"hei\": 800, \"second\":1 }";

s10 += ", \"data\":[" + s + "]}";

Dynamo.SceneJson(s10, true);

}

//малая окружность

s = MathPanelExt.QuadroEqu.DrawEllipse(rad, rad, 0, 0, 0, Math.PI \* 2, 64);

s10 = "{\"options\":{\"x0\": -10, \"x1\": 10, \"y0\": -10, \"y1\": 10, \"clr\": \"#ffffff\", \"sty\": \"line\", \"size\":0, \"lnw\": 3, \"fontsize\":24, \"wid\": 800, \"hei\": 800, \"second\":1 }";

s10 += ", \"data\":[" + s + "]}";

Dynamo.SceneJson(s10, true);

Запустим и получим наш цветок как на рис.5.8.

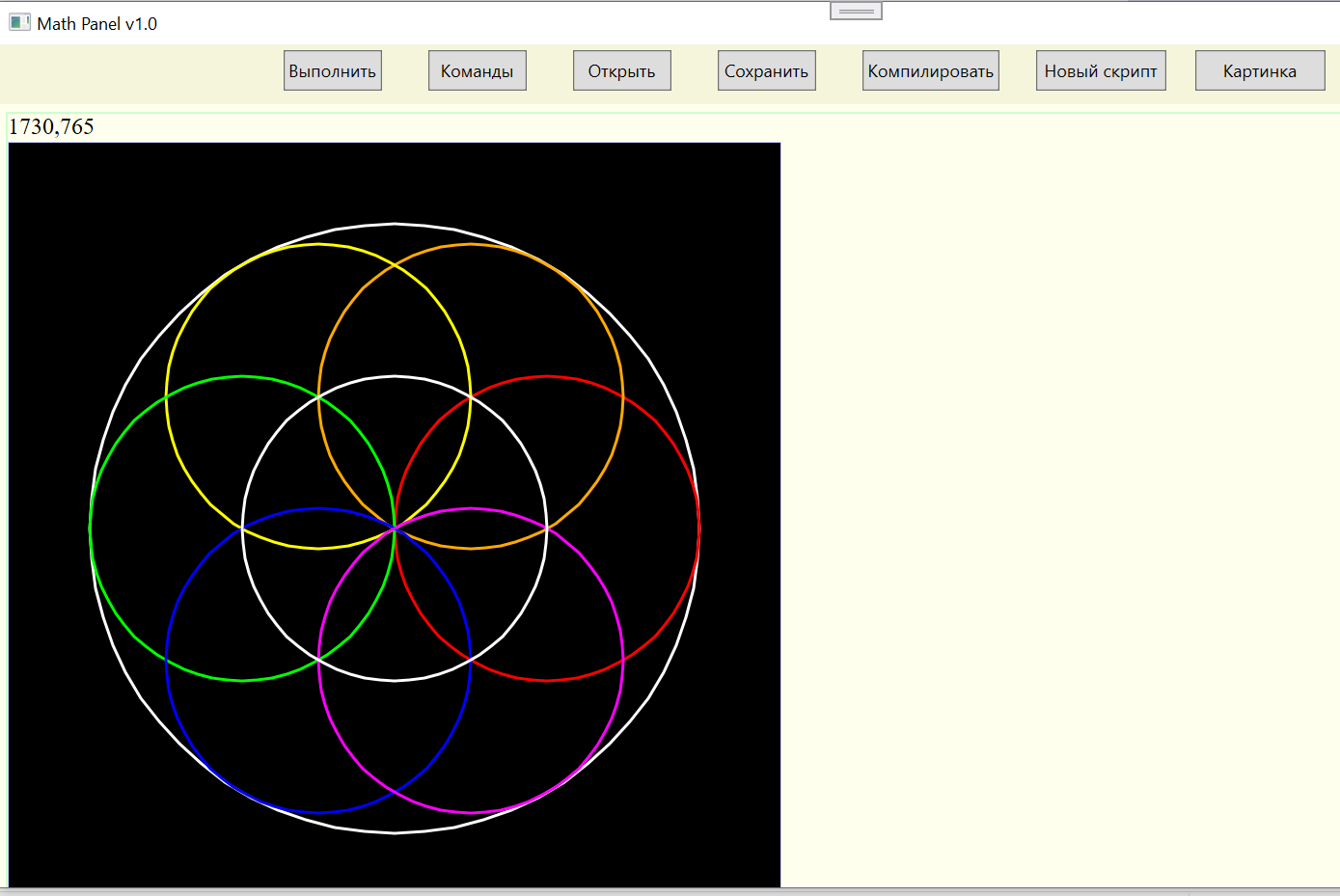


Рис.5.8. «Цветок жизни»