Міністерство освіти і науки України

Дніпропетровський національний університет залізничного транс потру iменi академіка В. Лазаряна

*Кафедра*

*Комп'ютерних інформаційних технологій*

ЗВІТ

з лабораторної роботи №1

з дисципліни "Комп'ютерна графiка"

Виконала

студентка групи 940П

Чегорко Карина

Перевірив

Пюра О.М.

2016

**Цель :получить навыки построения простых графических изображений.**

**Теоретические сведения:**

[<canvas>](https://developer.mozilla.org/ru/docs/ru/HTML/Canvas) это новый элемент [HTML](https://developer.mozilla.org/ru/docs/ru/HTML), который предназначен для создания графики при помощи скриптования (обычно [JavaScript](https://developer.mozilla.org/ru/docs/ru/JavaScript)). Он может быть использован, к примеру, для рисования графиков, создания фото-композиций, или для создания простых ([и не очень](https://developer.mozilla.org/ru/docs/ru/A_Basic_RayCaster)) анимаций.

<canvas> впервые был представлен фирмой Apple для [Mac OS X Dashboard](http://www.apple.com/macosx/features/dashboard/) и позже был реализован в Safari. Браузеры, основанные на [Gecko](https://developer.mozilla.org/ru/docs/ru/Gecko) 1.8, такие, как Firefox 1.5, также поддерживают этот новый элемент. Элемент <canvas> является частью спецификации [WhatWG Web applications 1.0](http://www.whatwg.org/specs/web-apps/current-work/), также известной как HTML 5.

## Возможности

Canvas разместить на холсте: картинку, видео, текст. Залить всё это сплошным цветом, либо обвести контуры или даже добавить градиент. Добавление теней похожих на свойства css3 box-shadow и text-shadow. И, наконец, отрисовка фигур с помощью указания контрольных точек. Причём можно изменять как ширину линий, так и кисть рисовки линий, стиль соединений линий.

## Особенности

* Изменение высоты или ширины холста сотрет всё его содержимое и все настройки, проще говоря он создастся заново;
* Начало отсчёта (точка 0,0) находится в левом верхнем углу. Но её можно сдвигать;
* 3D контекста нет, есть отдельные разработки, но они не стандартизованы;
* Цвет текста можно указывать аналогично CSS, впрочем, как и размер шрифта.

**Задание:** Разработать объектно-ориентированную программу для рисования графика функции (см. индивидуальное задание).

Программа должна предоставлять возможность пользователю:

* задавать диапазон аргумента функции и значения коэффициентов функций;
* цвета начертания и заливки различных элементов графика (фона, осей, заливки и графика функции, маркеров);

**Реализация:**

**index.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Function</title>

<link href="app/style/style.css" rel="stylesheet">

<script src="app/vanilla/main.js" /></script>

</head>

<body>

<div class="card">

<div>

<h3>Background</h3>

</div>

<div class="content">

<span class="myTitle">color:</span>

<input type="color" class="input" id="backgroundColor" />

</div>

<div class="content">

<span class="myTitle">grid color:</span>

<input type="color" class="input" id="gridColor" />

</div>

</div>

<div class="card">

<h3>Axis</h3>

<div class="content">

<span class="myTitle">color:</span>

<input type="color" class="input" id="axisColor" />

</div>

<div class="content">

<span class="myTitle">labels color:</span>

<input type="color" class="input" id="axisTitleColor" />

</div>

</div>

<div class="card">

<h3>Chart</h3>

<div class="content">

<span class="myTitle">color:</span>

<input type="color" class="input" id="chartColor" />

</div>

<div class="content">

<span class="myTitle">attributes:</span>

<input type="number" placeholder="a:" class="input" defaultValue = "5" id="chartAValue" />

</div>

<div class="content">

<input type="number" placeholder="t:" class="input" defaultValue = "-5" id="chartValue" />

</div>

<div class="content">

<label for = "isChartFilled">filled:</label>

<input type="checkbox" class="input-checkbox" id="isChartFilled" />

</div>

</div>

<div class = 'canvases'>

<canvas class='my-canvas' id="background" width="1000" height="800"></canvas>

<canvas class='my-canvas ' id="func" width="1000" height="800"></canvas>

</div>

</body>

</html>

**Drawer.js**

import Base from './Base';

export default class Drawer extends Base {

constructor(options = {

a: 10,

tBegin: -4,

filled: false

}) {

super(options);

this.a = options.a;

this.tBegin = options.tBegin;

this.filled = options.filled;

}

drawFunction() {

var steps = 10,

t = this.tBegin,

x0Value = (this.width - 0.5) / 2 / this.interval,

y0Value = (this.height - 0.5) / 2 / this.interval,

x = this.a \* (t \* t - 1)/(t \* t + 1),

y = this.a \* t \* (t \* t - 1) / (t \* t + 1),

realX = (x + x0Value) \* this.interval,

realY = (y0Value - y) \* this.interval,

grade = 2 \* 3.14 / 360;

this.ctx.beginPath();

this.ctx.moveTo(realX, realY);

while (t < steps) {

x = this.a \* (t \* t - 1)/(t \* t + 1);

y = this.a \* t \* (t \* t - 1) / (t \* t + 1);

console.log(`x:${x} + y:${y}`);

realX = (x + x0Value) \* this.interval;

realY = this.interval \* (y0Value - y);

this.ctx.lineTo(realX, realY);

t += grade;

}

if(this.filled) {

this.ctx.fillStyle = this.color;

this.ctx.fill();

}

this.ctx.strokeStyle = this.color;

this.ctx.stroke();

this.ctx.closePath();

}

clear() {

this.ctx.clearRect(0, 0, this.width, this.height);

}

set chartColor(color) {

this.color = color;

this.drawFunction();

}

get chartColor() {

return this.color;

}

set Avalue(value) {

this.a = value;

this.clear();

this.drawFunction(false);

}

get Avalue() {

return this.a;

}

set beginValue(value) {

this.tBegin = value;

this.clear();

this.drawFunction(false);

}

get beginValue() {

return this.tBegin;

}

set isFilled(value) {

this.filled = value;

this.clear();

this.drawFunction();

}

get isFilled() {

return this.filled;

}

}

**Base.js**

export default class Base {

constructor(options = {

color: '#eee',

width: 500,

height: 500,

interval: 20

}) {

this.ctx = options.ctx;

this.width = options.width;

this.height = options.height;

this.color = options.color;

this.interval = options.interval;

this.begin = 0.5;

}

}

**Axis.js**

import Base from './Base';

export default class Coord extends Base {

constructor(options) {

super(options);

this.font = options.font;

this.labelColor = options.labelColor;

}

drawCoord() {

this.drawXCoord();

this.drawYCoord();

this.drawLabels();

}

drawXCoord() {

var x = this.begin,

y = this.height / 2,

end = this.width,

title = -(this.width / 2 / this.interval) + 1;

this.ctx.beginPath();

this.ctx.fillStyle = this.labelColor;

while (x < end) {

this.ctx.moveTo(x, y);

x += this.interval;

this.ctx.lineTo(x, y);

this.ctx.fillText(title++, x, y - this.interval / 2);

this.ctx.lineTo(x, y - this.interval / 2);

this.ctx.lineTo(x, y + this.interval / 2);

}

this.ctx.moveTo(x, y);

x += this.inteval;

this.ctx.lineTo(x, y);

this.ctx.strokeStyle = this.color;

this.ctx.stroke();

this.ctx.closePath();

}

drawYCoord() {

var y = this.begin,

x = this.width / 2,

end = this.height,

title = this.height / 2 / this.interval - 1;

this.ctx.beginPath();

this.ctx.fillStyle = this.labelColor;

while (y < end) {

this.ctx.moveTo(x, y);

y += this.interval;

this.ctx.lineTo(x, y);

if (title) {

this.ctx.fillText(title--, x - this.interval, y);

} else {

title--;

}

this.ctx.lineTo(x - this.interval / 2, y);

this.ctx.lineTo(x + this.interval / 2, y);

}

this.ctx.moveTo(x, y);

y += this.interval;

this.ctx.lineTo(x, y);

this.ctx.strokeStyle = this.color;

this.ctx.stroke();

this.ctx.closePath();

}

drawLabels() {

this.ctx.fillText('Y', this.width / 2 + this.interval, this.begin + this.interval);

this.ctx.fillText('X', this.width - this.interval, this.height / 2 + this.interval);

}

set axisColor(color) {

this.color = color;

this.drawCoord();

}

get axisColor() {

return this.color;

}

set labelsColor(color) {

this.labelColor = color;

this.drawCoord();

}

get labelsColor() {

return this.labelColor;

}

}

**Coord.js**

import Base from './Base';

export default class Coord extends Base {

constructor(options) {

super(options);

this.font = options.font;

}

drawCoord() {

this.drawXCoord();

this.drawYCoord();

this.drawTitles();

}

drawXCoord() {

var x = this.begin,

y = this.height / 2,

end = this.width,

title = -(this.width / 2 / this.interval) + 1;

this.ctx.beginPath();

while (x < end) {

this.ctx.moveTo(x, y);

x += this.interval;

this.ctx.lineTo(x, y);

this.ctx.fillText(title++, x, y - this.interval / 2);

this.ctx.lineTo(x, y - this.interval / 2);

this.ctx.lineTo(x, y + this.interval / 2);

}

this.ctx.font = this.font;

this.ctx.fillText('X', x - this.interval, y + this.interval);

this.ctx.moveTo(x, y);

x += this.inteval;

this.ctx.lineTo(x, y);

this.ctx.strokeStyle = this.color;

this.ctx.stroke();

this.ctx.closePath();

}

drawYCoord() {

var y = this.begin,

x = this.width / 2,

end = this.height,

title = this.height / 2 / this.interval - 1;

this.ctx.beginPath();

while (y < end) {

this.ctx.moveTo(x, y);

y += this.interval;

this.ctx.lineTo(x, y);

if (title) {

this.ctx.fillText(title--, x - this.interval, y);

} else {

title--;

}

this.ctx.lineTo(x - this.interval / 2, y);

this.ctx.lineTo(x + this.interval / 2, y);

}

this.ctx.font = this.font;

this.ctx.fillText('Y', x + this.interval, this.begin + this.interval);

this.ctx.moveTo(x, y);

y += this.interval;

this.ctx.lineTo(x, y);

this.ctx.strokeStyle = '#000000';

this.ctx.stroke();

this.ctx.closePath();

}

drawTitles() {

}

}

**Field.js**

import Base from './Base';

export default class Field extends Base {

drawGrid() {

var x, y;

this.ctx.beginPath();

this.ctx.moveTo(0, 0);

for (x = this.begin; x < this.width; x += this.interval) {

this.ctx.moveTo(x, 0);

this.ctx.lineTo(x, this.height);

}

for (y = this.begin; y < this.height; y += this.interval) {

this.ctx.moveTo(0, y);

this.ctx.lineTo(this.width, y);

}

this.ctx.strokeStyle = this.color;

this.ctx.stroke();

this.ctx.closePath();

}

set gridColor(color) {

this.color = color;

this.drawGrid();

}

get gridColor() {

return this.color;

}

}

**main.js**

import Drawer from './modules/Drawer';

import Field from './modules/Field';

import Axis from './modules/Axis';

document.addEventListener("DOMContentLoaded", () => {

let mainEl = document.getElementById('background'),

ctx = mainEl.getContext('2d'),

funcEl = document.getElementById('func'),

funcCtx = funcEl.getContext('2d'),

interval = 20,

width = 1000,

height = 800,

field = new Field({

interval,

width,

height,

ctx,

color: '#eee'

}),

axis = new Axis({

interval,

width,

height,

ctx,

font: '10px Arial',

color: '#000000'

}),

drawer = new Drawer({

color: "#3E2723",

interval,

width,

height,

ctx: funcCtx,

a: 5,

tBegin: -5,

filled: false

});

field.drawGrid();

axis.drawCoord();

drawer.drawFunction(false);

addListeners({

mainEl,

funcEl,

field,

axis,

drawer

});

});

const addListeners = ({

mainEl,

field,

axis,

drawer

}) => {

var bcolor = document.getElementById('backgroundColor'),

gridColor = document.getElementById('gridColor'),

axisColor = document.getElementById('axisColor'),

labelsColor = document.getElementById('axisTitleColor'),

chartColor = document.getElementById('chartColor'),

chartTBalue = document.getElementById('chartValue'),

chartAValue = document.getElementById('chartAValue'),

isChartFilled = document.getElementById('isChartFilled'),

map = new Map();

map.set(bcolor, color => changeBgColor(mainEl, color))

.set(gridColor, color => field.gridColor = color)

.set(axisColor, color => axis.axisColor = color)

.set(chartColor, color => drawer.chartColor = color)

.set(chartAValue, value => drawer.Avalue = +value)

.set(chartTBalue, value => drawer.beginValue = +value)

.set(labelsColor, color => axis.labelsColor = color);

map.forEach((value, key) => {

key.addEventListener('change', (e) => {

e.preventDefault();

value(e.target.value);

});

});

isChartFilled.addEventListener('change', e => {

e.preventDefault();

drawer.clear();

drawer.isFilled = e.target.checked;

});

};

const changeBgColor = (el, color) => {

el.style.backgroundColor = color;

};

**style.css**

body {

background-color: #fafaf7;

}

.card {

width: 300px;

margin: 20px;

padding: 3px 20px 20px 20px;

box-shadow: 0 0 5px grey;

background-color: ghostwhite;

}

.input {

width: 150px;

height: 30px;

font-size: 18px;

font-family: Helvetica, Arial, sans-serif;

font-weight: 300;

padding: 5px;

}

.input-checkbox {

height: 30px;

padding: 5px;

width: 20px;

display: inline-block;

vertical-align: bottom;

margin: 0px;

}

.myTitle {

display: block;

color: #003d4d;

font-size: 20px;

}

.content {

margin: 10px;

}

.my-canvas {

position: absolute;

top: 50px;

right: 150px;

}

h3 {

color: #003d4d;

font-size: 24px;

font-weight: 200;

display: block;

}

.my-canvas {

box-shadow: 0 0 5px grey;

}

label {

color: #003d4d;

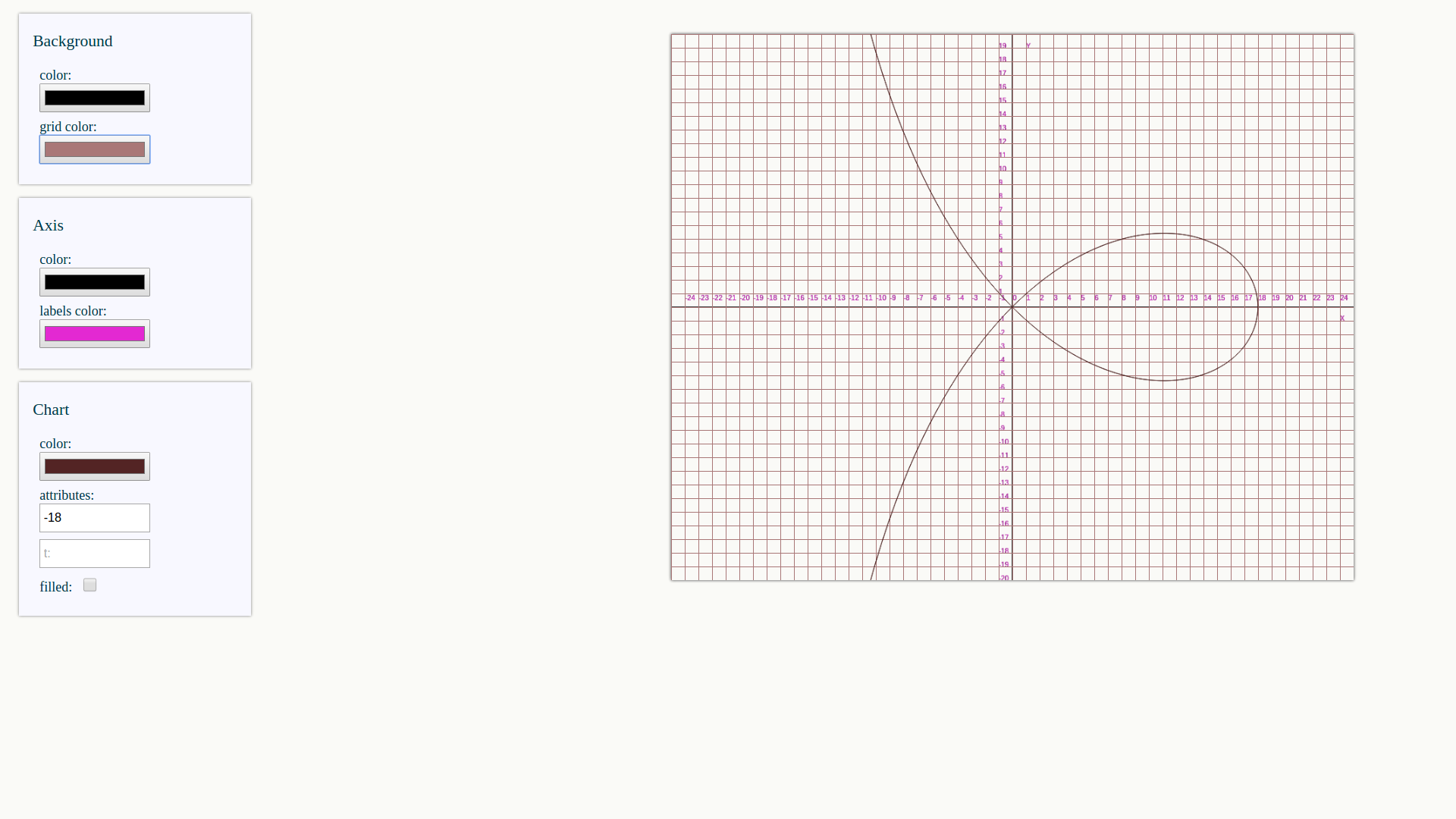
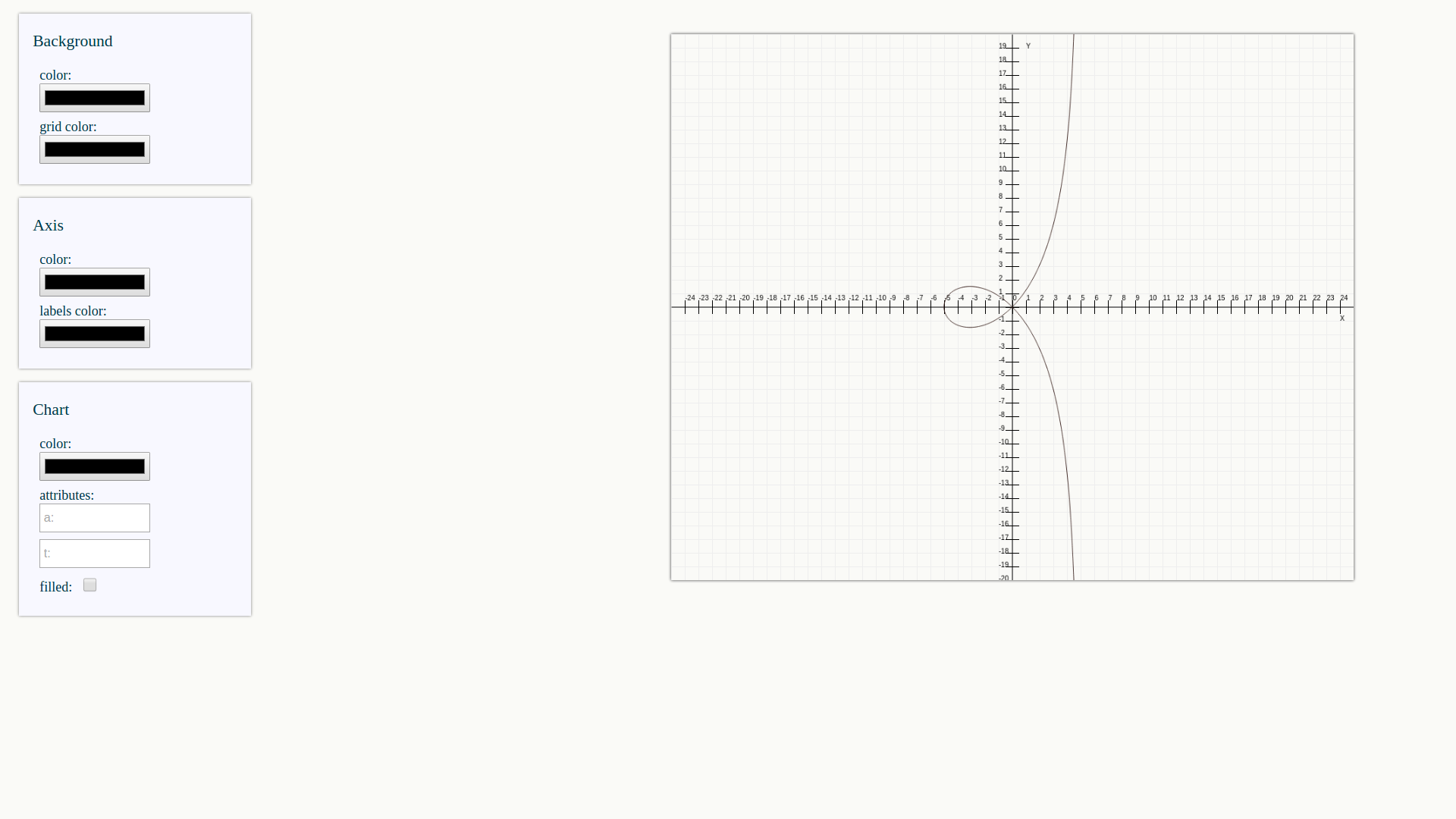
font-size: 20px;

display: inline-block;

width: 60px;

}

Результаты работы программы:



Выводы: во время выполнения работы научилась строить простые графические компоненты. Освоила работу с элементом canvas в javascript