­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­

Лабораторна робота №3

з курсу «Геометричне моделювання у конструюванні інженерних об'єктів та систем»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 26

Виконав студент гр. КНз-3

Чалий Михайло

­­

Львів 2015

## Мета роботи

Ознайомитись там практично засвоїти технології й основи роботи з графічними можливостями Win API операційного середовища Windows, що можуть викликатися прикладними програмами. Вивчити основні графічні функції та процедури WinAPI засобами програмних середовищ. Оволодіти принципами створення графічних Windows-програм

## Теоретичні відомості

**Graphics Device Interface**, **GDI** — один з трьох основних компонентів або «підсистем», що разом з ядром і Windows API складають користувацький інтерфейс (віконний менеджер GDI) Microsoft Windows.

GDI — це інтерфейс Microsoft Windows для представлення графічних об'єктів і передачі їх на пристрої відображення, таких як монітори і принтери.

GDI відповідає за растеризацію ліній і кривих, відображення шрифтів і обробку палітри. Він не відповідає за растеризацію вікон, меню тощо. Це завдання закріплене за користувацькою підсистемою, що розташоване в user32.dll і грунтуються на GDI.

Одна з найбільших переваг використання GDI замість методів прямого доступу до обладнання — це можливість розширення функцій над різними пристроями. Використовуючи GDI, можна легко малювати на кількох різних пристроях, таких як екран або принтер, і досягти практично однакового відображення на них. Ця можливість лежить у центрі всіх WYSIWYG застосунків для Microsoft Windows.

Прості ігри, які не потребують швидкої графіки, використовують GDI (отже, GDI - повільний задля швидкої графіки). Однак GDI не може забезпечити якісну анімацію (немає можливості синхронізації з кадровим буфером). Також GDI не вистачає растеризації для відмальовування 3D-графіки. Сучасні ігри використовують DirectX чи OpenGL, що дає програмістам доступ до більшої кількості апаратних можливостей.

## GDI +

Після виходу Windows XP використання GDI стало небажаним через вихід у світ його нащадка, підсистеми GDI+, заснованої на C++. GDI+ є поліпшеним середовищем для 2D графіки, в яке крім іншого додали такі можливості:

* Згладжування ліній (antialiasing).
* Використання координат з плаваючою точкою.
* Градієнтний заливка.
* Вбудована підтримка сучасних графічних форматів (JPEG і PNG).
* Перетворення 2D матриць тощо.

GDI+ використовує ARGB кольори. Ці можливості використовуються в інтерфейсі користувача Windows XP, а їхня присутність у базовому графічному шарі полегшує використання систем векторної графіки, таких як Flash або SVG. Динамічні бібліотеки GDI+ можуть поширюватися разом з застосунками для використання в попередніх версіях Windows.

В .NET можна використовувати функції GDI+ через простір імен System.Drawing.

## Завдання

26. Написати програму із застосуванням графічних функцій Win API, яка дозвояляє візуалізувати графіки функцій з можливостями маштабування у вікні на певному інтервалі. Застосувати фіункції зміни кольору, штриховки, зафарбування замкунтих обламтей, текстових підписів, виводу координатної сітки. Оріганізувати вивід графіка у зовнішній метафайл.

Побудувати в одній області екрана графіки функцій:



## Реалізація

Лістінг 1. l3.js

/\* jshint esnext: true \*/

var React = require('react');

var u = require('../u');

var cm = require('../coord-map');

var assign = Object.assign || require('object-assign');

var ReactBootstrap = require('react-bootstrap');

var Input = ReactBootstrap.Input;

var STEP = 25;

var L3 = React.createClass({

getInitialState: function() {

return {

xv1: -2,

xv2: 1.5,

yv1: -1,

yv2: 1,

f1: '(x\*x - 1)/(x\*x\*x\*x + 1)',

f2: '-(x\*x\*x\*x\*x) + 2\*x\*x\*x - 1'

};

},

componentDidMount: function() {

},

componentWillUnmount: function() {

},

getWorld: function(){

return assign(cm.CoordSys(), {

top: this.state.yv2,

bottom: this.state.yv1,

left: this.state.xv1,

right: this.state.xv2

});

},

getScreen: function(){

return assign(cm.CoordSys(), {

top: 0,

bottom: this.height,

left: 0,

right: this.width

});

},

worldToScreen: function(x, y){

var world = this.getWorld();

var screen = this.getScreen();

return cm.worldToScreen(x, y, world, screen);

},

/\*

\* Генерація координатної сітки

\*/

coordNet: function(){

var width = this.width;

var height = this.height;

var self = this;

var world = this.getWorld();

var screen = this.getScreen();

var step = ((world.getWidth()) / (screen.getWidth())) \* 50;

var vert = function\*(){

for(let x = 0; x < world.right; x += step){

yield x;

}

for(let x = 0; x > world.left; x -= step){

yield x;

}

};

var horis = function\*(){

for(let y = 0; y < world.top; y += step){

yield y;

}

for(let y = 0; y > world.bottom; y -= step){

yield y;

}

};

var net = function\*(){

for(var x of [...vert()]){

var vc = self.worldToScreen(x, 0);

yield 'M ' + vc.x + ',' + screen.top;

yield 'L ' + vc.x + ',' + screen.bottom;

}

for(var y of [...horis()]){

var hc = self.worldToScreen(0, y);

yield 'M ' + screen.left + ',' + hc.y;

yield 'L ' + screen.right + ',' + hc.y;

}

};

var labels = function\*(){

var i = 0;

for(var x of [...vert()]){

var vc = self.worldToScreen(x, 0).move(2, 20);

var label = Math.round(x \* 10) / 10;

yield <text key={i++} x={vc.x} y={vc.y}>{label}</text>;

}

for(var y of [...horis()]){

var hc = self.worldToScreen(0, y).move(-30, -2);

var label = Math.round(y \* 10) / 10;

yield <text key={i++} x={hc.x} y={hc.y}>{label}</text>;

}

};

var zero = this.worldToScreen(0, 0);

var axis = function\*(){

yield 'M ' + 0 + ',' + zero.y;

yield 'L ' + width + ',' + zero.y;

yield 'M ' + zero.x + ',' + 0;

yield 'L ' + zero.x + ',' + height;

};

return <g>

<path d={u.concate(net, ' ')} style={{stroke: 'gray'}} />

<path d={u.concate(axis, ' ')} style={{stroke: 'black'}} />

<g>{[...labels()]}</g>

</g>

},

func: function(f, color){

var self = this;

var world = this.getWorld();

var screen = this.getScreen();

var xv1 = world.left;

var xv2 = world.right;

// Each pixel on screen

var step = ((world.getWidth()) / (screen.getWidth()));

var gr = function\*(){

var c = self.worldToScreen(xv1, f(xv1));

yield 'M ' + c.x + ',' + c.y;

yield 'L';

for(var xv = xv1; xv <= xv2; xv += step){

var c = self.worldToScreen(xv, f(xv));

yield c.x + ',' + c.y;

}

};

//console.log(u.concate(gr, ' '));

return <path d={u.concate(gr, ' ')} style={{stroke: color, fill: 'none'}} />

},

render: function() {

this.width = 600;

this.height = 400;

var self = this;

return (

<div className='container'>

<div className='row'>

<div className='col-md-8'>

<svg width={this.width} height={this.height } >

<rect width={this.width} height={this.height } style={{stroke: 'green', fill: 'none'}} />

{this.coordNet()}

{this.func(x => eval(self.state.f1), 'red')}

{this.func(x => eval(self.state.f2), 'blue')}

</svg>

</div>

<div className='col-md-4'>

<form>

<Input name='xv1' type='number' step='any' label='Від X' defaultValue={this.state.xv1} onChange={this.\_handleChange} />

<Input name='xv2' type='number' step='any' label='До X' defaultValue={this.state.xv2} onChange={this.\_handleChange} />

<Input name='yv1' type='number' step='any' label='Від Y' defaultValue={this.state.yv1} onChange={this.\_handleChange} />

<Input name='yv2' type='number' step='any' label='До Y' defaultValue={this.state.yv2} onChange={this.\_handleChange} />

<Input name='f1' type='text' label='Формула #1' defaultValue={this.state.f1} onChange={this.\_handleChange} />

<Input name='f2' type='text' label='Формула #2' defaultValue={this.state.f2} onChange={this.\_handleChange} />

</form>

</div>

</div>

</div>

);

},

\_handleChange: function(e){

var patch = {};

patch[e.target.name] = e.target.value;

this.setState(patch);

}

});

exports.L3 = L3;

Повна версія коду доступна на https://github.com/chaliy/studies-octo-adventure/tree/master/lp/c3\_2/gm/src/

## Результат

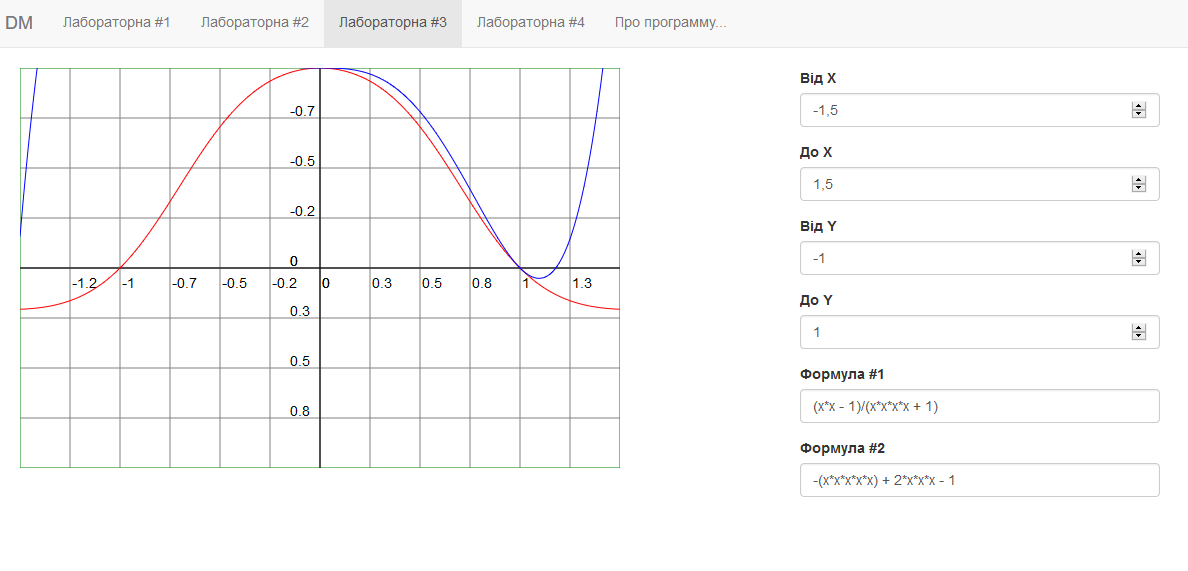


Рис . Перший етап

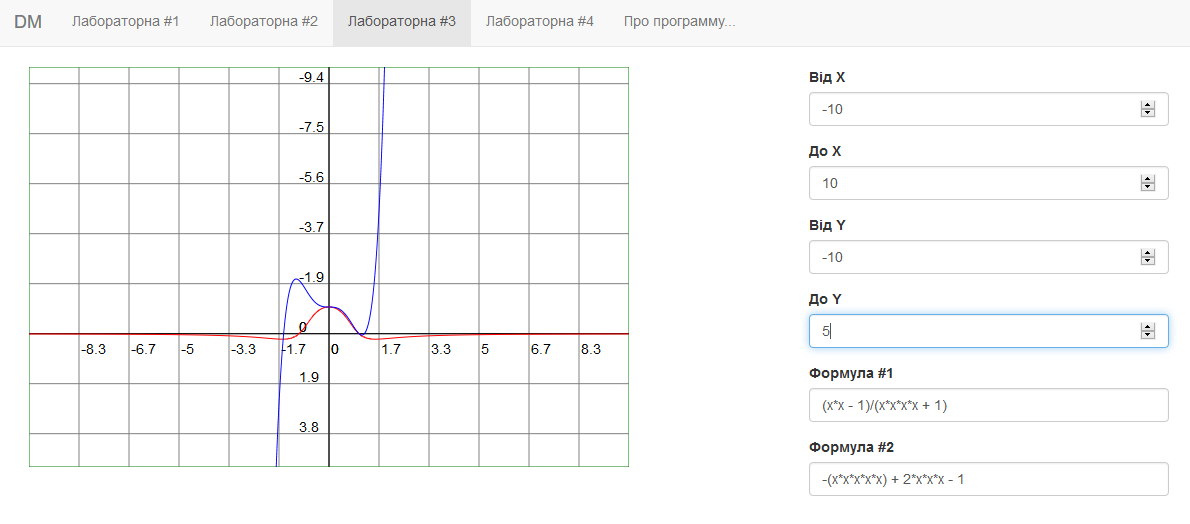


Рис . Другий етап

Програма також доступна на http://chaliy.name/play/lp/c3\_2/gm/#/l3

## Висновки

Ознайомився та практично засвоїв технології й основи роботи з графічними можливостями Win API операційного середовища Windows, що можуть викликатися прикладними програмами. Вивчив основні графічні функції та процедури WinAPI засобами програмних середовищ. Оволодіти принципами створення графічних Windows-програм