­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­

Лабораторна робота №1

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 10

Виконав студент гр. КНз-3

Чалий Михайло

­­

Львів 2015

## Мета роботи

Ознайомитись із графічними можливостями програмних модулів, їх функціональним складом, параметрами, властивостями. Набути практичних навиків розробки програм з використанням графічних процедур та функцій у текстовому та графічному режимах.

## Теоретичні відомості

Для виконання лабораторної роботи було обрано HTML Canvas - елемент HTML5, який можна застосовувати для малювання графіки використовуючи скрипти (переважно JavaScript). Наприклад його можна застосувати для малювання графів, створення фотокомпозицій а також анімації.

Елемент <canvas> є частиною специфікації WHATWG HTML та W3C HTML Canvas 2D Context.

<canvas> вперше було втілено Apple в Mac OS X Dashboard та Safari 3.1. У Gecko підтримка <canvas> з'явилася у версії Firefox 1.5, у Presto з версії 9.0 веб-браузера Opera, а Internet Explorer підтримує <canvas> починаючи з 9-ї версії.

Щоб відобразити <canvas> в html-документі, слід використати наступний код:

<canvas id="tutorial" width="150" height="150">

</canvas>

Елемент <canvas> дуже схожий на тег <img>, з тією лише різницею, що не містить атрибутів src і alt. Елемент <canvas> має всього два атрибути — width і height. Обидва вони не є обов'язковими, і можуть бути задані через властивості DOM. Якщо ширина і висота не визначені, <canvas> буде створений шириною в 300 пікселів і 150 пікселів заввишки. Розмір елемента може бути довільним і задаватися через CSS, але при промальовуванні картинка масштабується відповідно до компонування.

Атрибут id не є специфічним для елемента <canvas>, але є одним з атрибутів HTML за замовчуванням, і може бути застосований майже до всіх елементів HTML (також як class, наприклад). Завжди визначати id елемента — гарна ідея, тому що це значно спрощує ідентифікацію його в за допомогою скриптів.

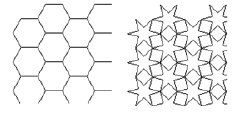
Стиль елемента <canvas> може налаштовуватися також, як і звичайне зображення через CSS (margin, border, background, і т.п.). Ці правила, проте, не впливають на саме малювання в <canvas>. Якщо ніякі налаштування стилю не задані, <canvas> буде створений повністю прозорим.

<canvas> створює поверхню для малювання, яка надає один або більше контекстів для відтворення, який використовується для створення відображуваного контенту і маніпуляцій з ним. Ми сфокусуємо на 2D (двомірному) контексті відтворення, який в наш час є єдиним певним контекстом. У майбутньому інші контексти зможуть підтримувати інші види відтворення: наприклад, цілком ймовірно, що буде додано 3D контекст, заснований на OpenGL ES.

<canvas> спочатку порожній, і для того, щоб що-небудь відобразити, скрипту необхідно отримати контекст відтворення і малювати вже на ньому. Елемент <canvas> має DOM-метод getContext і призначений для отримання контексту відтворення разом з його функціями малювання. getContext () приймає один параметр — тип контексту

## Завдання

21. Скласти програму заповнення прямокутних областей оранментом з використанням процедур рисування правильних багатокутників.



## Реалізація

Лістінг 1. L1.js

var drawLines = function(ctx, points){

if (points.length > 1) {

var start = fix(points[0]);

ctx.moveTo(start.x, start.y);

for(var i = 1; i < points.length; i++){

var next = fix(points[i]);

ctx.lineTo(next.x, next.y);

}

}

};

var buildPoligone = function(start, size, type){

var x = start.x;

var y = start.y;

var points = [];

for (var i = 0; i <= VERTICLES\_NUM; i++) {

var ang = i \* 2 \* Math.PI / VERTICLES\_NUM;

points.push({

x: x + size \* Math.cos(ang),

y: y + size \* Math.sin(ang)

});

if (type === 'star'){

var ang2 = ((i \* 2) + 1) \* Math.PI / VERTICLES\_NUM;

points.push({

x: x + (size / 2) \* Math.cos(ang2),

y: y + (size / 2) \* Math.sin(ang2)

});

}

}

return points;

};

var poligoneThirdWidth = function(size){

return 3 \* size \* Math.cos(2 \* Math.PI / VERTICLES\_NUM);

};

var poligoneHalfHeight = function(size){

return size \* Math.sin(2 \* Math.PI / VERTICLES\_NUM);

};

var renderCanvas = function(canvas, type){

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

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

var size = 20; // Segment length

var dx = poligoneThirdWidth(size);

var dy = poligoneHalfHeight(size);

var nx = canvas.width / dx;

var ny = canvas.height / (2 \* dy);

ctx.beginPath();

for(var x = 0; x < nx; x++){

for(var y = 0; y < ny; y++){

var lx = x \* dx;

var ly = y \* (2 \* dy);

if (x % 2 === 0){

ly += dy;

}

var points = buildPoligone({x: lx, y: ly}, size, type);

drawLines(ctx, points);

}

}

ctx.strokeStyle = '#000';

ctx.stroke();

};

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

## Результат

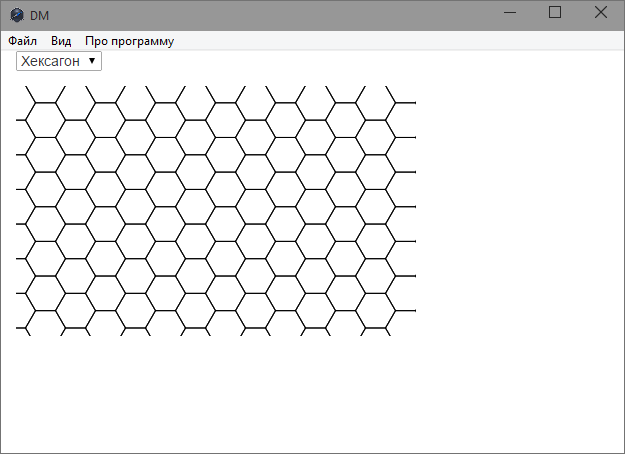


Рис 1. Перший варіант

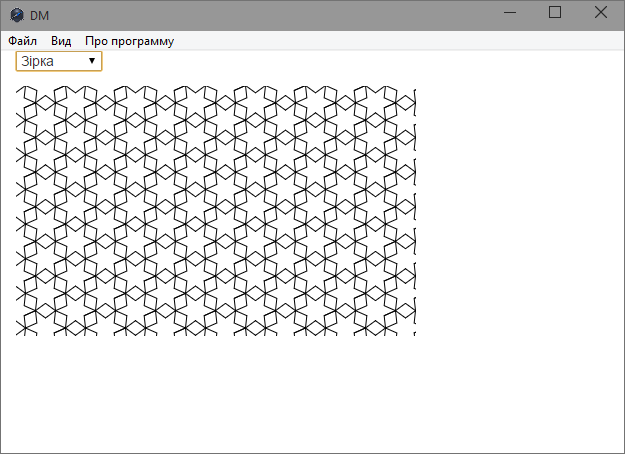


Рис 2. Другий варіант

## Висновки

Ознайомився із графічними можливостями програмних модулів, їх функціональним складом, параметрами, властивостями. Набув практичних навиків розробки програм з використанням графічних процедур та функцій у текстовому та графічному режимах.