­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­

Лабораторна робота №2

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 9

Виконав студент гр. КНз-3

Чалий Михайло

­­

Львів 2015

## Мета роботи

Ознайомитись із законами руху геометричних об’єктів на площині та у просторі. Оволодіти математичною мовою опису динаміки та візуалізації на основі закономірностей геометричних перетворень. Набути практичних навиків розробки графічних процедур афінних перетворень у середовищах програмування.

## Теоретичні відомості

Афінні перетворення знаходять широке застосування в задачах машинної графіки. Найбільшого поширення набули часткові випадки афінних перетворень: зсув, поворот, масштабування.

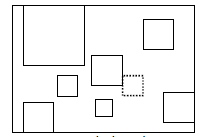
Нехай у площині задана початкова система координат *0ХY* і деяка нова система координат *01X1Y1*. Тоді перетворення, які полягають у тому, щоб у відповідність точці *P* площини ставиться точка *P1*, яка в новій системі має такі самі координати, що й точка*P* у початковій, називаються афінними.

Основні властивості афінних перетворень:

* Множина точок, яка в початковій системі координат задовольняє деяке рівняння, переходить у множину точок, координати яких у новій системі задовольняють таке саме рівняння. Так, пряма переходить у пряму, площина у площину;
* Відношення площ і об’ємів геометричних фігур зберігається;
* Зберігається просте співвідношення трьох точок;
* Існує єдине перетворення площини, що переводить трійку точок, які не належать одній прямій, у нову трійку точок, які також не належать прямій;
* Якщо початкова та нова системи координат є декартовими з однаковими одиничними відрізками по осях, то при перетвореннях зберігаються всі метричні властивості геометричних фігур.

## Завдання

9. Написати імінітаційнк програму з застосування афінних перетворень, яка дозволяє в автоматичному режимі та при ручному керуванні створювати динамічно змінну візуалізацію при якій моделюється набір квадратів випадкового розміру. Потрім розміри квадратів починають зростати. При торканні квадратів мениший знищується. При торканні краю екрана центр квадтрату починає зміщатися.ітш



## Реалізація

Лістінг 1. l2.js

/\* jshint esnext: true \*/

var React = require('react');

var utils = require('../draw-utils');

//var rects = [];

var animation = 0;

var renderCanvas = function(canvas, rects){

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

ctx.resetTransform();

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

ctx.strokeStyle = 'rgb(0,0,0)';

rects.forEach(function(rect){

ctx.setTransform(rect.scale, 0, 0, rect.scale, rect.x, rect.y);

ctx.stroke(rect.path);

});

};

function rand(min, max) {

return Math.floor(Math.random() \* (max - min)) + min;

}

var L2 = React.createClass({

rects: [],

getInitialState: function(){

return {

};

},

genRect: function(x, y){

if(typeof x === 'undefined') {x = rand(25, this.width - 100);}

if(typeof y === 'undefined') {y = rand(25, this.height - 100);}

var size = rand(5, 50);

var path = new Path2D();

path.rect(-size/2, -size/2, size, size);

return {

path: path,

scale: 1,

x: x,

y: y,

size: size

};

},

initRectangles: function(){

this.rects = [];

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

this.rects.push(this.genRect());

}

this.renderCanvas();

},

next: function(){

var width = this.width;

var height = this.height;

// Remove intersecions

var sorted = this.rects.slice(0).sort(function(a, b){

return b.size \* b.scale - a.size \* a.scale;

});

var rects = [];

// TODO Remove stupid O(n^2)

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

var r = sorted[i];

var kill = false;

for(var j = 0; j < rects.length; j++){

var o = rects[j];

var rs = r.size \* r.scale;

var rx = r.x - rs/2;

var ry = r.y - rs/2;

var os = o.size \* o.scale;

var ox = o.x - os/2;

var oy = o.y - os/2;

if (!(rx + rs < ox || ox + os < rx || ry + rs < oy || oy + os < ry)){

kill = true;

break;

}

}

if (!kill){

rects.push(r);

}

}

rects.forEach(function(r){

// Grow all rects

r.scale \*= 1.01;

// Esnure items does not move outside of the area

var blank = 5;

var rs = r.size \* r.scale;

var rx = r.x - rs/2;

if (rx < 0){

r.x += (0 - rx);

} else if (rx + rs > width) {

r.x -= (rx + rs - width);

}

var ry = r.y - rs/2;

if (ry < 0){

r.y += (0 - ry);

}

if (ry + rs > height) {

r.y -= (ry + rs - height);

}

});

this.rects = rects;

this.renderCanvas();

if (this.rects.length <= 1){

clearInterval(this.nextIntervalId);

}

},

renderCanvas: function(){

if (this.refs.canvas){

var canvas = this.refs.canvas.getDOMNode();

var rects = this.rects;

animation = 0;

requestAnimationFrame(function(){

renderCanvas(canvas, rects);

});

}

},

componentDidMount: function() {

var canvas = this.refs.canvas.getDOMNode();

this.width = canvas.width;

this.height = canvas.height;

console.log(this.height);

canvas.addEventListener("mousedown", this.handleMouseClick, false);

this.initRectangles();

this.nextIntervalId = setInterval(this.next, 100);

},

handleMouseClick: function(e){

var x = e.x;

var y = e.y;

var canvas = this.refs.canvas.getDOMNode();

x -= canvas.offsetLeft;

y -= canvas.offsetTop;

this.rects.push(this.genRect(x, y));

},

componentWillUpdate: function(nextProps, nextState) {

},

render: function() {

return <div>

<canvas ref='canvas' width={600} height={400} style={{borderColor: 'gray', borderThickness: '1', borderStyle: 'solid'}} ></canvas>

</div>;

}

});

exports.L2 = L2;

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

## Результат

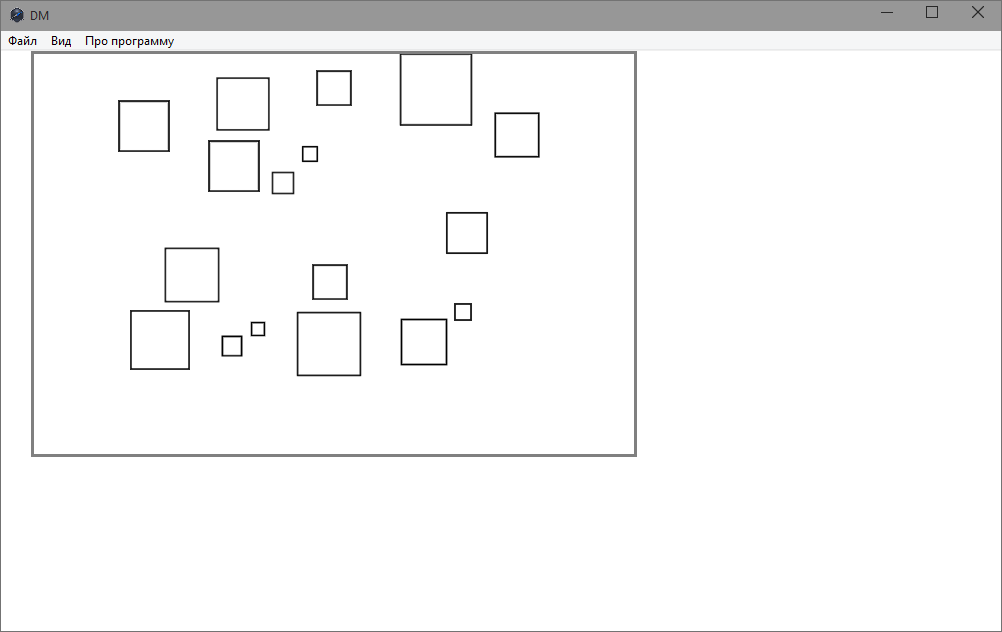


Рис . Перший етап

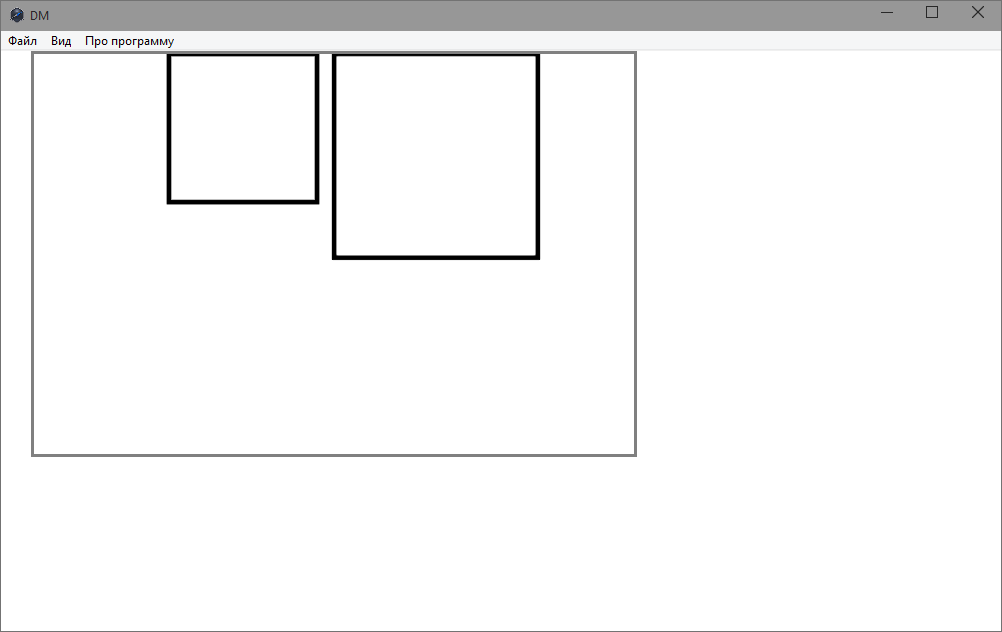


Рис . Другий етап

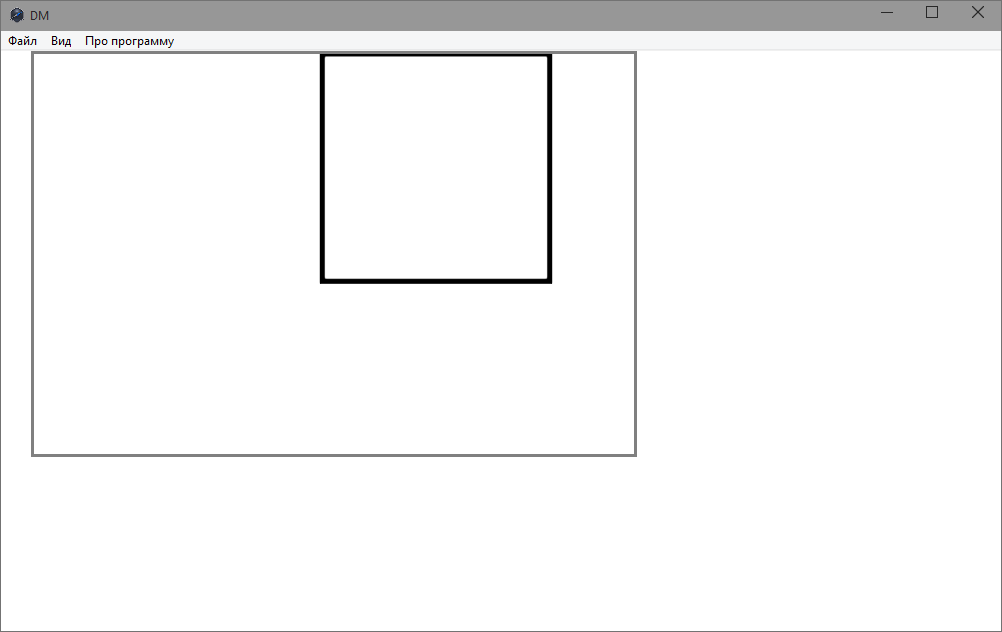


Рис . Третій етап

## Висновки

Ознайомився із законами руху геометричних об’єктів на площині та у просторі. Оволодів математичною мовою опису динаміки та візуалізації на основі закономірностей геометричних перетворень. Набув практичних навиків розробки графічних процедур афінних перетворень у середовищах програмування.