**Державний вищий навчальний заклад**

**Ужгородський національний університет**

**Факультет інформаційних технологій**

**ЛАБОРАТОРНА РОБОТА № 1**

**Тема:** Моделювання кінематики прямолінійного руху

Виконала студентка ІI курсу

спеціальності «Інженерія

програмного забезпечення»

Петрус Альбіна

**Ужгород 2025**

***Мета:*** Моделювання кінематики прямолінійного руху

***Завдання до роботи:***

1. Використовуючи базові навички програмування на мові **JavaScript**, та за допомогою **HTML** розмітки і каскадних таблиць стилів **CSS** побудувати траєкторію руху матеріальної точки в залежності від вхідних параметрів, які будуть задані користувачем:

а ) х0

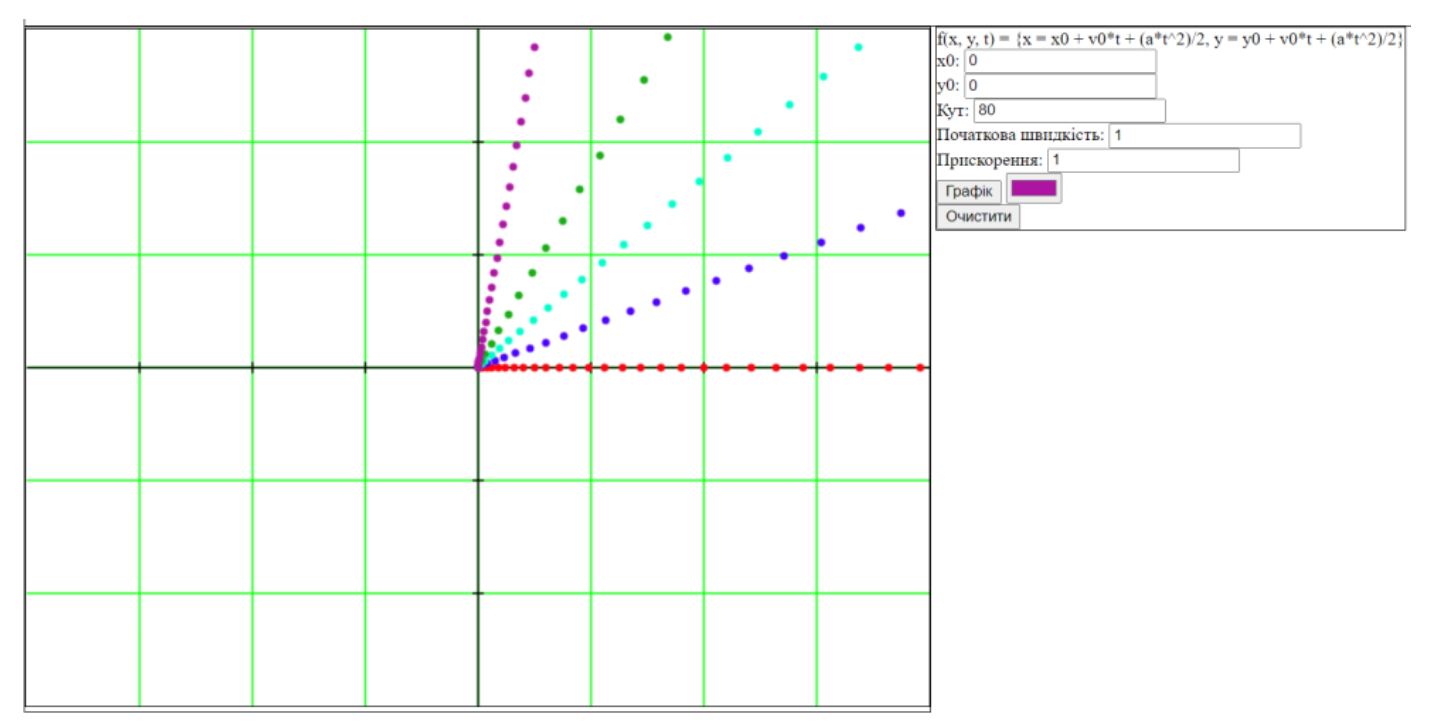
б) у0

в) кут

г) початкова швидкiсть

д) прискорення

Приклад отриманого результату:



Як видно з графіку, користувач повинен мати змогу задавати колір траєкторії, а також мати можливість очистити графік.

Для побудови графіка доцільно буде використати HTML таг Canvas

1. Використовуючи попередній результата зробити рефакторинг коду, та замість використання Canvas використати бібліотеку для побудови діаграм та графіків в JavaScript [D3.JS](https://d3js.org)
2. Оформити звіт виконаної роботи, згідно взірця

Отримані результати (**лістинг коду** завантажити на **репозиторій**, або у вигляді **архіву**, та **звіт,** завантажити у *папку курсу*, *підпапку* лабораторної роботи та *підпаку* із назвою, що відповідає вашому прізвищу). Якщо код завантажено на репозиторій, то у звіті потрібно вказати посилання на репозиторій.  
  
**Посилання: https://github.com/albinapetrus/MetodyProgramnoi**

**Код програми:**

1)

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Траєкторія руху</title>

    <script>

        function plotTrajectory() {

            let x0 = parseFloat(document.getElementById("x0").value);//parseFloat-рядок в число з .

            let y0 = parseFloat(document.getElementById("y0").value);

            let ang = parseFloat(document.getElementById("ang").value);

            let v0 = parseFloat(document.getElementById("v0").value);

            let a = parseFloat(document.getElementById("pr").value);

            let color = document.getElementById("gr").value;

            let canvas = document.getElementById("myCanvas");

            let ctx = canvas.getContext("2d"); //getContext("2d")-метод для малювання

            //  Pi- властивість об'єкта Math, яка повертає значення числа π (Пі).

            let angleInRadians = ang \* Math.PI / 180;

            let timeStep = 1;

            let maxTime = 1000;

            let x = x0;

            let y = y0;

            ctx.beginPath();

            ctx.moveTo(x, canvas.height - y); // moveTo-переміщення  поточної точки малювання в зазначену точку

            ctx.fillStyle = color; // fillStyle-визначення стилю(колір, градієнт, шаблон)

            for (let t = 0; t <= maxTime; t += timeStep) {

                x = x0 + v0 \* Math.cos(angleInRadians) \* t + 0.5 \* a \* Math.cos(angleInRadians) \* t \* t;

                y = y0 + v0 \* Math.sin(angleInRadians) \* t + 0.5 \* a \* Math.sin(angleInRadians) \* t \* t;

                ctx.beginPath();

                ctx.arc(x, canvas.height - y, 2, 0, Math.PI \* 1); //arc- використовується для малювання кола чи дуги (x, y, radius, startAngle, endAngle)

                ctx.fill();//fill- метод в API , який використовується для заповнення поточної форми або шляху

            }

        }

        function clearCanvas() {

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

            let ctx = canvas.getContext("2d");

            ctx.clearRect(0, 0, c.width, c.height); //clearRect-метод очищення

        }

    </script>

    <style>

        body{

            display: grid;

            justify-content: center;

        }

       #myCanvas{

           margin-left: 5em;

            margin-top: 30px;

            border-style: dotted;

        }

        .form {

            max-width: 350px;

            width: 100%;

            margin: auto;

            margin-top: 20%;

            display: grid;

            justify-content: center;

            background-color: gainsboro;

            padding: 30px;

        }

    </style>

</head>

<body>

    <form class="form">

        <div>

            <label for="x0">Координата x0:</label>

            <input type="number" id="x0" name="x0" min="-100" max="100" step="0.0001" />

        </div><br/>

        <div>

            <label for="y0">Координата y0:</label>

            <input type="number" id="y0" name="y0" min="-100" max="100" step="0.0001" />

        </div><br/>

        <div>

            <label for="ang">Кут:</label>

            <input type="number" id="ang" name="ang" min="0" max="360" step="0.0001" />

        </div><br/>

        <div>

            <label for="v0">Початкова швидкість:</label>

            <input type="number" id="v0" name="v0" min="0" max="1000" step="0.0001" />

        </div><br/>

        <div>

            <label for="pr">Прискорення:</label>

            <input type="number" id="pr" name="pr" min="-1000" max="1000" step="0.0001" />

        </div><br/>

        <div>

            <label for="gr">Графік:</label>

            <input type="color" id="gr" name="gr"  />

        </div><br/>

        <button type="button" onclick="plotTrajectory()">Побудувати траєкторію</button><br/>

        <button type="button" onclick="clearCanvas()">Очистити</button>

    </form>

    <canvas id="myCanvas" width="200" height="200"  style="border:1px dotted #942e2e;"></canvas>

</body>

</html>

2)

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Траєкторія руху</title>

    <script src="https://d3js.org/d3.v7.min.js">

        //підключення бібліотеки через cdn

    </script>

    <script>

        function plotTrajectory() {

            let x0 = parseFloat(document.getElementById("x0").value);

            let y0 = parseFloat(document.getElementById("y0").value);

            let ang = parseFloat(document.getElementById("ang").value);

            let v0 = parseFloat(document.getElementById("v0").value);

            let a = parseFloat(document.getElementById("pr").value);

            let color = document.getElementById("gr").value;

            let width = 600;

            let height = 400;

            let svg = d3.select("#graphContainer").append("svg")//d3-глобальна змінна, select-повертає елемент по id, append-додавання нового елементу

                .attr("width", width) //attr-метод d3 для встановлення атрибутів

                .attr("height", height)

                .style("border", "1px solid #942e2e");//метод для встановлення стилю

            let angleInRadians = ang \* Math.PI / 180;

            // Параметри часу

            let timeStep = 1;

            let maxTime = 1000;

            let points = [];

            // Моделюємо траєкторію

            for (let t = 0; t <= maxTime; t += timeStep) {

                let x = x0 + v0 \* Math.cos(angleInRadians) \* t + 0.5 \* a \* Math.cos(angleInRadians) \* t \* t;

                let y = y0 + v0 \* Math.sin(angleInRadians) \* t + 0.5 \* a \* Math.sin(angleInRadians) \* t \* t;

                points.push([x, height - y]); // push-додавання елементів до кінця масиву

            }

            // Створюємо круги для кожної точки траєкторії

            svg.selectAll("circle") //selectAll-вибирає всі елементи

                .data(points)//метод привязує дані

                .enter().append("circle")//enter-створює нові елементи для тих в кого нема елемента в Dom

                .attr("cx", d => d[0])

                .attr("cy", d => d[1])

                .attr("r", 3)

                .attr("fill", color);

        }

        function clearGraph() {

            d3.select("#graphContainer").selectAll("svg").remove();//remove()-видалення елементів з графіка чи діаграми

        }

    </script>

    <style>

        body {

            display: grid;

            justify-content: center;

        }

        .form {

            max-width: 350px;

            width: 100%;

            margin: auto;

            margin-top: 20%;

            display: grid;

            justify-content: center;

            background-color: gainsboro;

            padding: 30px;

        }

        #graphContainer {

            margin-top: 30px;

        }

    </style>

</head>

<body>

    <form class="form">

        <div>

            <label for="x0">Координата x0:</label>

            <input type="number" id="x0" name="x0" min="-100" max="100" step="0.0001" />

        </div><br/>

        <div>

            <label for="y0">Координата y0:</label>

            <input type="number" id="y0" name="y0" min="-100" max="100" step="0.0001" />

        </div><br/>

        <div>

            <label for="ang">Кут:</label>

            <input type="number" id="ang" name="ang" min="0" max="360" step="0.0001" />

        </div><br/>

        <div>

            <label for="v0">Початкова швидкість:</label>

            <input type="number" id="v0" name="v0" min="0" max="1000" step="0.0001" />

        </div><br/>

        <div>

            <label for="pr">Прискорення:</label>

            <input type="number" id="pr" name="pr" min="-1000" max="1000" step="0.0001" />

        </div><br/>

        <div>

            <label for="gr">Графік:</label>

            <input type="color" id="gr" name="gr"  />

        </div><br/>

        <button type="button" onclick="plotTrajectory()">Побудувати траєкторію</button><br/>

        <button type="button" onclick="clearGraph()">Очистити</button>

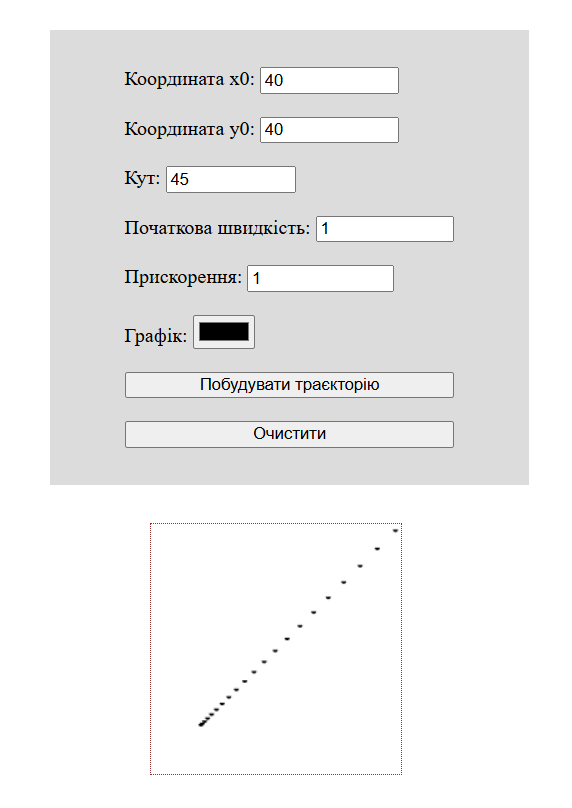
    </form>

    <div id="graphContainer"></div>

</body>

</html>

**Скріншот роботи програми:**

****

**Висновок:** В результаті виконання лабораторної роботи було розглянуто дві реалізації моделювання кінематики прямолінійного руху матеріальної точки:

Перша частина роботи полягає у побудові траєкторії руху з використанням HTML Canvas. В цій реалізації користувач може задавати початкові умови, такі як координати, кут, початкова швидкість і прискорення, після чого будується траєкторія. Графік відображається на HTML-канвасі, що дозволяє візуалізувати рух. Користувач також має можливість вибору кольору траєкторії та очищення графіку.

Друга частина роботи полягає у покращенні попереднього рішення з використанням бібліотеки D3.js. Використання D3 дозволяє створювати більш гнучкі та масштабовані візуалізації з можливістю роботи з даними на рівні SVG елементів. Замість Canvas для побудови графіка використовується SVG, що дає більше контролю за маніпулюванням елементами графіку. Цей підхід дозволяє легше інтегрувати різні графічні елементи та забезпечує кращу підтримку інтерактивності.

Таким чином, було виконано успішний перехід від використання Canvas до D3.js, що дозволяє значно покращити гнучкість і масштабованість створюваних графіків, зберігаючи основну функціональність і можливості взаємодії з користувачем.