Державний вищий навчальний заклад

Ужгородський національний університет

Факультет інформаційних технологій

Кафедра програмного забезпечення систем

**ЛАБОРАТОРНА РОБОТА № 1**

**Тема**: Моделювання кінематики прямолінійного руху

Виконав:

студент ІІ курсу

спеціальності: 121 інженерія програмного забезпечення

Вако Сергій Романович

**Ужгород-2024**

**Мета**: отримати уявлення про принципи побудови траєкторії руху тіла.

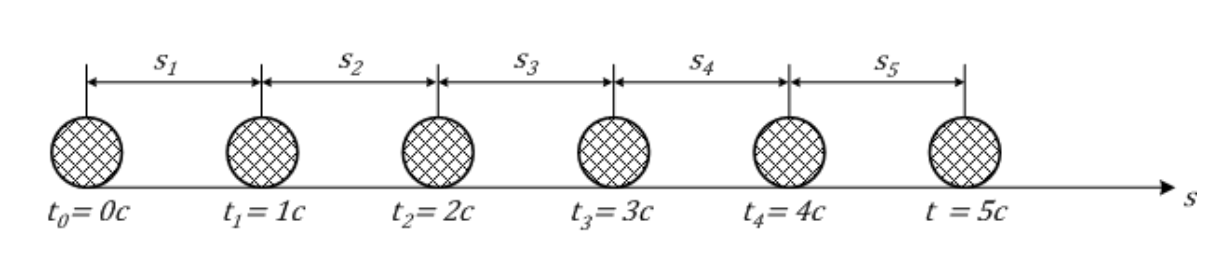
**Короткі теоретичні відомості**

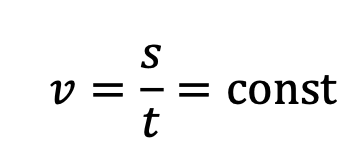
Кінематика (від грец. κινειν — рухатись) у фізиці – розділ механіки, що вивчає способи опису руху матеріальних тіл без урахування їхньої маси, сил, які діють на них і причин виникнення руху.

Залежно від властивостей досліджуваного матеріального тіла, що рухається, розрізняють кінематику:

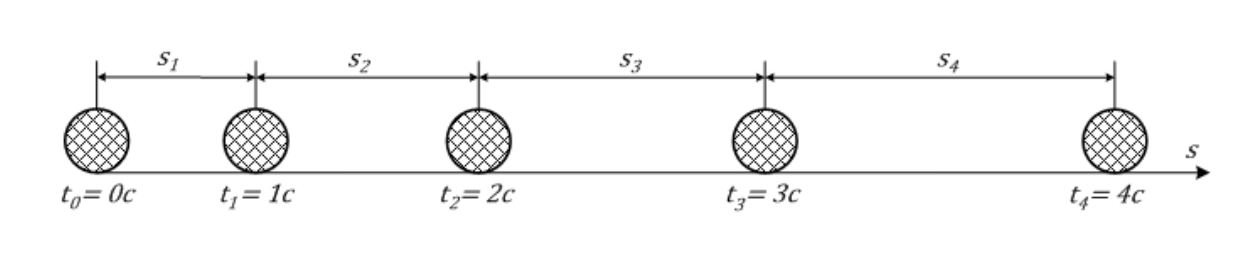
* - матеріальної точки (матеріальне тіло, розмірами якого можна знехтувати порівняно з характерними відстанями між тілами);
* - абсолютно твердого тіла (тіло, відстань між двома будь-якими точками якого не змінюється, тобто воно не деформується);
* - середовища, що деформується (пружно або пластично), та рідин і газів.

**Рівномірний рух** – це такий рух, коли за рівні проміжки часу тіло проходить однакові відстані, тобто швидкість тіла залишається сталою.



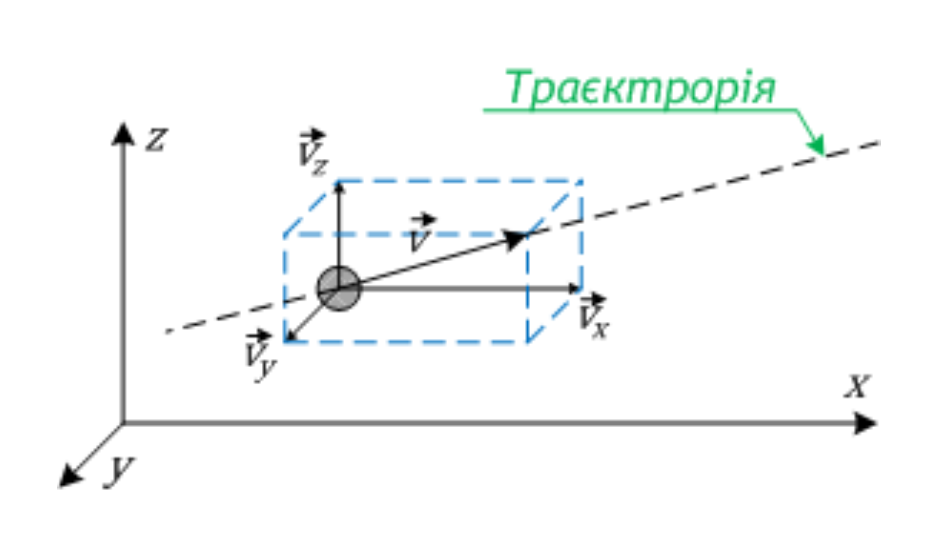


**Прискорений рух** – це такий рух, при якому швидкість тіла змінюється, тобто за однакові проміжки часу тіло проходить різні проміжки шляху.

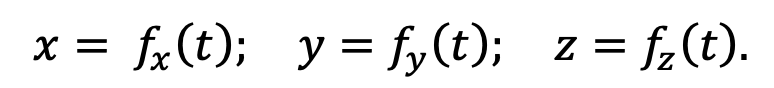


**Прискорення** – це векторна величина, фізичний зміст якої полягає у тому, що це швидкість зміни швидкості. Напрямок прискорення завжди співпадає із напрямком швидкості. Одиниця вимірювання прискорення.

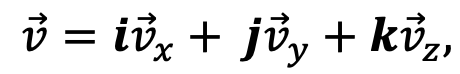
**Прямолінійний рух** – це такий рух, траєкторія якого є прямою лінією. Як правило, прямолінійний рух описується у декартовій (прямокутній) системі координат.



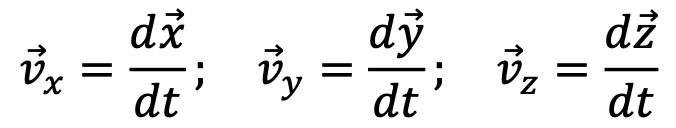
При прямолінійному русі вектор швидкості розкладається на свої складові по осям координат і кінематичне рівняння руху записується у координатній формі:



Миттєва швидкість:

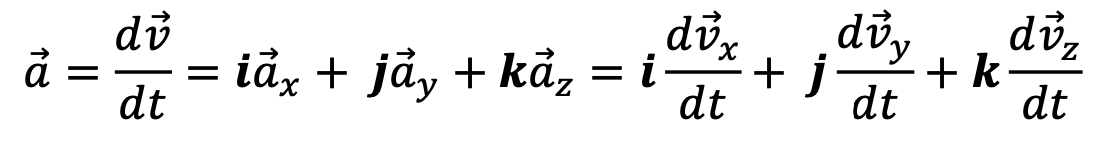


де



проекції швидкості на вісі координат.

Прискорення:

****

**Завдання до лабораторної 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

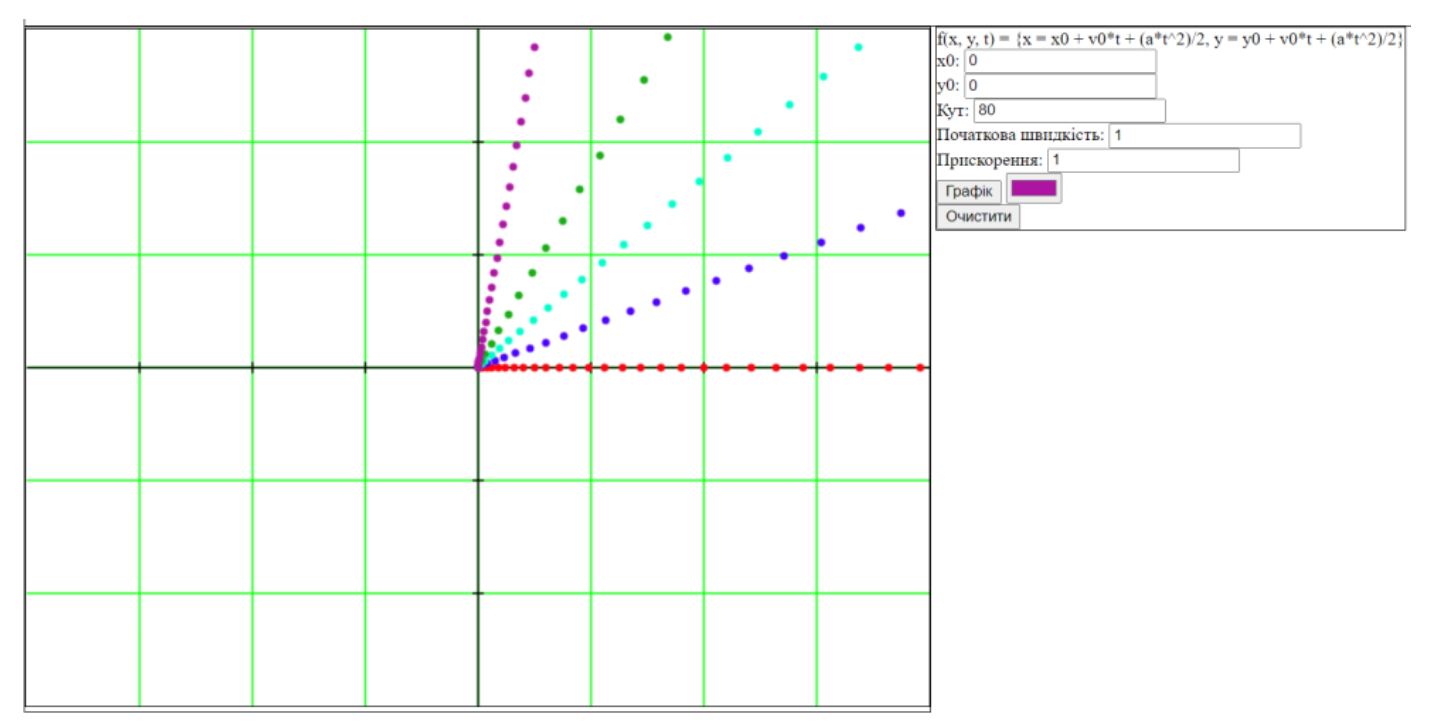
б) у0

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
3. 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.  
     
   Приклад організації папок:  
     
   Методи програмної інженерії/Лаб 1/Бучук/звіт.docx

Хід роботи

**Код:**

import React, { useEffect, useRef, useState } from "react";

import \* as d3 from "d3";

export default function Home() {

const svgRef = useRef<SVGSVGElement>(null);

const [x0, setX0] = useState(50);

const [y0, setY0] = useState(300);

const [angle, setAngle] = useState(45);

const [velocity, setVelocity] = useState(50);

const [acceleration, setAcceleration] = useState(0);

const [color, setColor] = useState("red");

useEffect(() => {

drawTrajectory();

}, [x0, y0, angle, velocity, acceleration, color]);

const drawTrajectory = () => {

const svg = d3.select(svgRef.current);

svg.selectAll("\*").remove();

const width = 600;

const height = 400;

const radianAngle = (angle \* Math.PI) / 180;

let t = 0;

let x = x0;

let y = y0;

const points: [number, number][] = [];

while (y <= height && x <= width) {

x = x0 + velocity \* Math.cos(radianAngle) \* t;

y = y0 - (velocity \* Math.sin(radianAngle) \* t - 0.5 \* acceleration \* t \* t);

if (y >= height) break;

points.push([x, y]);

t += 0.1;

}

const line = d3.line<[number, number]>()

.x((d: any[]) => d[0])

.y((d: any[]) => d[1])

.curve(d3.curveLinear);

svg.append("path")

.datum(points)

.attr("fill", "none")

.attr("stroke", color)

.attr("stroke-width", 2)

.attr("d", line);

};

return (

<div>

<svg ref={svgRef} width={600} height={400} style={{ border: "1px solid black" }} />

<div>

<label>X0: <input type="number" value={x0} onChange={(e) => setX0(+e.target.value)} /></label>

<label>Y0: <input type="number" value={y0} onChange={(e) => setY0(+e.target.value)} /></label>

<label>Angle: <input type="number" value={angle} onChange={(e) => setAngle(+e.target.value)} /></label>

<label>Velocity: <input type="number" value={velocity} onChange={(e) => setVelocity(+e.target.value)} /></label>

<label>Acceleration: <input type="number" value={acceleration} onChange={(e) => setAcceleration(+e.target.value)} /></label>

<label>Color: <input type="color" value={color} onChange={(e) => setColor(e.target.value)} /></label>

</div>

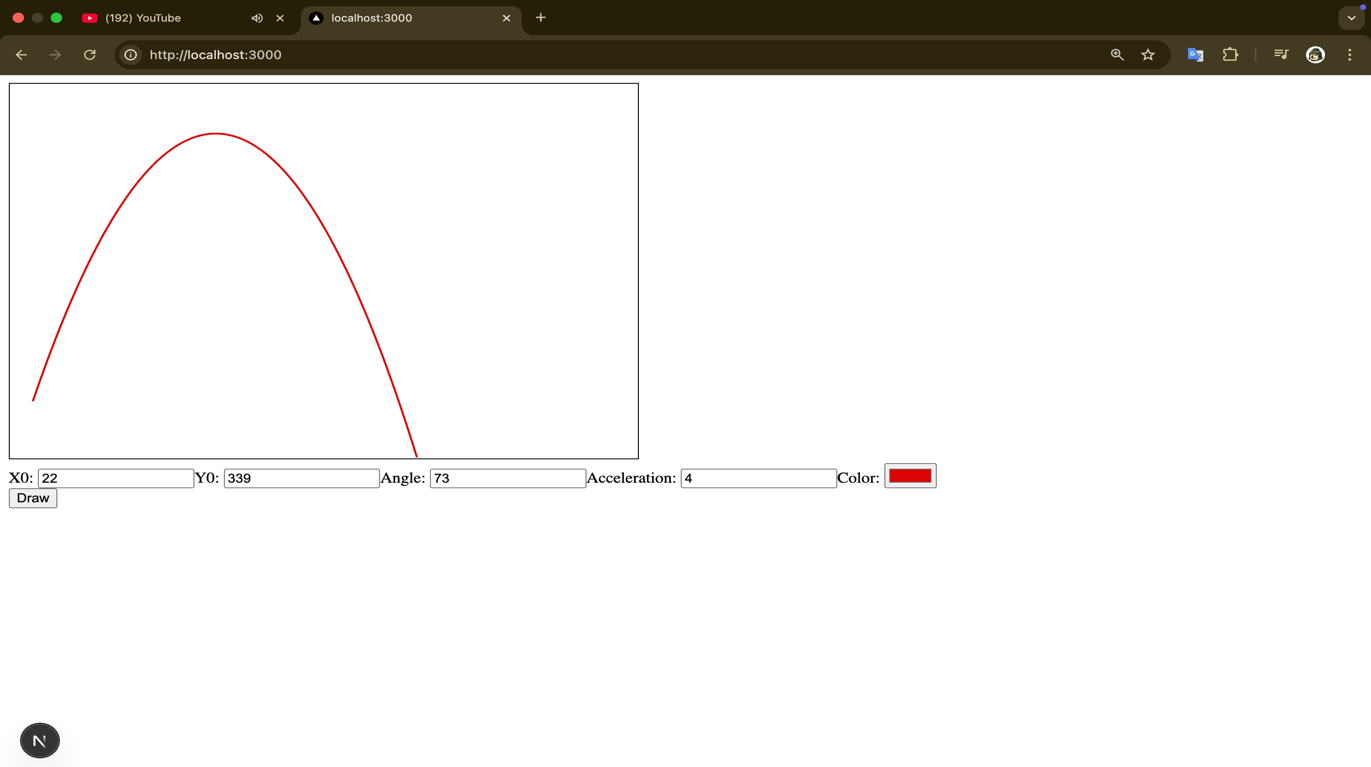
<button onClick={drawTrajectory}>Draw</button>

</div>

);

}

**Приклад роботи:**

****

**Висновки:**

Моделювання прямолінійного руху дозволило зрозуміти вплив початкових параметрів на траєкторію тіла. Використання D3.js замість Canvas покращило візуалізацію та забезпечило більшу гнучкість. Програма стала інтерактивною завдяки можливості змінювати параметри руху в реальному часі.