**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМ**

**Лабораторна робота №1**

**Тема:**Моделювання кінематики прямолінійного руху

Виконав:

Студент 2-го курсу

факультету інформаційних технологій

Денної форми навчання

Групи ІПЗ – 2

Олексій Станіслав Віталійович

Ужгород 2025

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

**Завдання:**

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 коду завдання зроблено з canvas

Html

<!DOCTYPE html>

<html lang="uk">

  <head>

    <meta charset="UTF-8" />

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

    <title>Моделювання руху матеріальної точки</title>

    <link rel="stylesheet" href="style.css" />

  </head>

  <body>

    <div class="container">

      <div class="inputs">

        <div class="form-group">

          <label for="x0">Початкова координата x₀ (м):</label>

          <input type="number" id="x0" value="0" />

        </div>

        <div class="form-group">

          <label for="y0">Початкова координата y₀ (м):</label>

          <input type="number" id="y0" value="0" />

        </div>

        <div class="form-group">

          <label for="angle">Кут (градуси):</label>

          <input type="number" id="angle" value="45" min="0" max="90" />

        </div>

        <div class="form-group">

          <label for="velocity">Початкова швидкість (м/с):</label>

          <input type="number" id="velocity" value="20" min="0" />

        </div>

        <div class="form-group">

          <label for="acceleration">Прискорення (м/с²):</label>

          <input

            type="number"

            id="acceleration"

            value="9.8"

            min="0"

            step="any"

          />

        </div>

      </div>

      <div class="controls">

        <button id="calculate">Розрахувати траєкторію</button>

        <button id="reset">Скинути</button>

      </div>

      <canvas id="trajectoryCanvas" width="900" height="500"></canvas>

      <div class="info">

        <div>

          <h3>Результати:</h3>

          <div id="results"></div>

        </div>

      </div>

    </div>

    <script src="index.js"></script>

  </body>

</html>

Js

document.addEventListener("DOMContentLoaded", () => {

  const canvas = document.getElementById("trajectoryCanvas");

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

  const calculateBtn = document.getElementById("calculate");

  const resetBtn = document.getElementById("reset");

  const resultsDiv = document.getElementById("results");

  const scale = 10;

  const originX = 50;

  const originY = canvas.height - 50;

  function drawAxes() {

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

    ctx.beginPath();

    ctx.strokeStyle = "#000";

    ctx.lineWidth = 1;

    ctx.moveTo(originX, originY);

    ctx.lineTo(canvas.width - 20, originY);

    ctx.moveTo(canvas.width - 20, originY);

    ctx.lineTo(canvas.width - 30, originY - 5);

    ctx.moveTo(canvas.width - 20, originY);

    ctx.lineTo(canvas.width - 30, originY + 5);

    ctx.moveTo(originX, originY);

    ctx.lineTo(originX, 20);

    ctx.moveTo(originX, 20);

    ctx.lineTo(originX - 5, 30);

    ctx.moveTo(originX, 20);

    ctx.lineTo(originX + 5, 30);

    ctx.stroke();

    ctx.fillStyle = "#000";

    ctx.font = "14px Arial";

    ctx.fillText("x (м)", canvas.width - 30, originY + 20);

    ctx.fillText("y (м)", originX - 30, 30);

    ctx.textAlign = "center";

    for (let x = 5; x <= (canvas.width - originX) / scale; x += 5) {

      const xPixel = originX + x \* scale;

      ctx.beginPath();

      ctx.moveTo(xPixel, originY - 5);

      ctx.lineTo(xPixel, originY + 5);

      ctx.stroke();

      ctx.fillText(x.toString(), xPixel, originY + 20);

    }

    ctx.textAlign = "right";

    for (let y = 5; y <= originY / scale; y += 5) {

      const yPixel = originY - y \* scale;

      ctx.beginPath();

      ctx.moveTo(originX - 5, yPixel);

      ctx.lineTo(originX + 5, yPixel);

      ctx.stroke();

      ctx.fillText(y.toString(), originX - 10, yPixel + 5);

    }

  }

  function calculateTrajectory() {

    const x0 = parseFloat(document.getElementById("x0").value) || 0;

    const y0 = parseFloat(document.getElementById("y0").value) || 0;

    const angle = parseFloat(document.getElementById("angle").value) || 45;

    const v0 = parseFloat(document.getElementById("velocity").value) || 20;

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

    a = isNaN(a) ? 9.8 : a;

    const isZeroAcceleration = Math.abs(a) < 1e-10;

    console.log("Чи нульове прискорення:", isZeroAcceleration);

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

    const v0x = v0 \* Math.cos(angleRad);

    const v0y = v0 \* Math.sin(angleRad);

    let flightTime, maxHeight, xMax;

    if (isZeroAcceleration) {

      if (v0y === 0) {

        flightTime = 10;

        maxHeight = y0;

      } else if (v0y > 0) {

        flightTime = 10;

        maxHeight = y0 + v0y \* flightTime;

      } else {

        if (y0 > 0) {

          flightTime = y0 / Math.abs(v0y);

        } else {

          flightTime = 0;

        }

        maxHeight = y0;

      }

      xMax = x0 + v0x \* flightTime;

    } else {

      const A = -0.5 \* a;

      const B = v0y;

      const C = y0;

      const discriminant = B \* B - 4 \* A \* C;

      if (discriminant < 0) {

        flightTime = 10;

      } else {

        const sqrtDisc = Math.sqrt(discriminant);

        const t1 = (-B + sqrtDisc) / (2 \* A);

        const t2 = (-B - sqrtDisc) / (2 \* A);

        if (t1 > 0 && t2 > 0) {

          flightTime = Math.min(t1, t2);

        } else if (t1 > 0) {

          flightTime = t1;

        } else if (t2 > 0) {

          flightTime = t2;

        } else {

          flightTime = 0;

        }

      }

      const tMaxHeight = v0y / a;

      if (tMaxHeight > 0 && tMaxHeight < flightTime) {

        maxHeight = y0 + v0y \* tMaxHeight - 0.5 \* a \* tMaxHeight \* tMaxHeight;

      } else {

        maxHeight = y0;

      }

      xMax = x0 + v0x \* flightTime;

    }

    if (flightTime > 20) {

      flightTime = 20;

    }

    resultsDiv.innerHTML = `

                <p>Час польоту: ${flightTime.toFixed(2)} с</p>

                <p>Максимальна висота: ${maxHeight.toFixed(2)} м</p>

                <p>Дальність польоту: ${xMax.toFixed(2)} м</p>

                <p>Тип руху: ${

                  isZeroAcceleration

                    ? "Рівномірний прямолінійний"

                    : "Рівноприскорений"

                }</p>

            `;

    drawAxes();

    ctx.beginPath();

    ctx.strokeStyle = "#3498db";

    ctx.lineWidth = 2;

    if (flightTime <= 0) {

      console.log("Час польоту <= 0, траєкторія не малюється");

      return;

    }

    const timeStep = flightTime / 100;

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

      let x, y;

      if (isZeroAcceleration) {

        x = x0 + v0x \* t;

        y = y0 + v0y \* t;

      } else {

        x = x0 + v0x \* t;

        y = y0 + v0y \* t - 0.5 \* a \* t \* t;

      }

      const xPixel = originX + x \* scale;

      const yPixel = originY - y \* scale;

      if (t === 0) {

        ctx.moveTo(xPixel, yPixel);

      } else {

        ctx.lineTo(xPixel, yPixel);

      }

    }

    ctx.stroke();

    const startXPixel = originX + x0 \* scale;

    const startYPixel = originY - y0 \* scale;

    ctx.beginPath();

    ctx.arc(startXPixel, startYPixel, 5, 0, 2 \* Math.PI);

    ctx.fill();

    const arrowLength = 30;

    ctx.beginPath();

    ctx.lineWidth = 2;

    ctx.moveTo(startXPixel, startYPixel);

    ctx.lineTo(

      startXPixel + arrowLength \* Math.cos(angleRad),

      startYPixel - arrowLength \* Math.sin(angleRad)

    );

    ctx.stroke();

    console.log("Розрахунок траєкторії завершено");

  }

  drawAxes();

  calculateBtn.addEventListener("click", calculateTrajectory);

  resetBtn.addEventListener("click", () => {

    document.getElementById("x0").value = "0";

    document.getElementById("y0").value = "0";

    document.getElementById("angle").value = "45";

    document.getElementById("velocity").value = "20";

    document.getElementById("acceleration").value = "9.8";

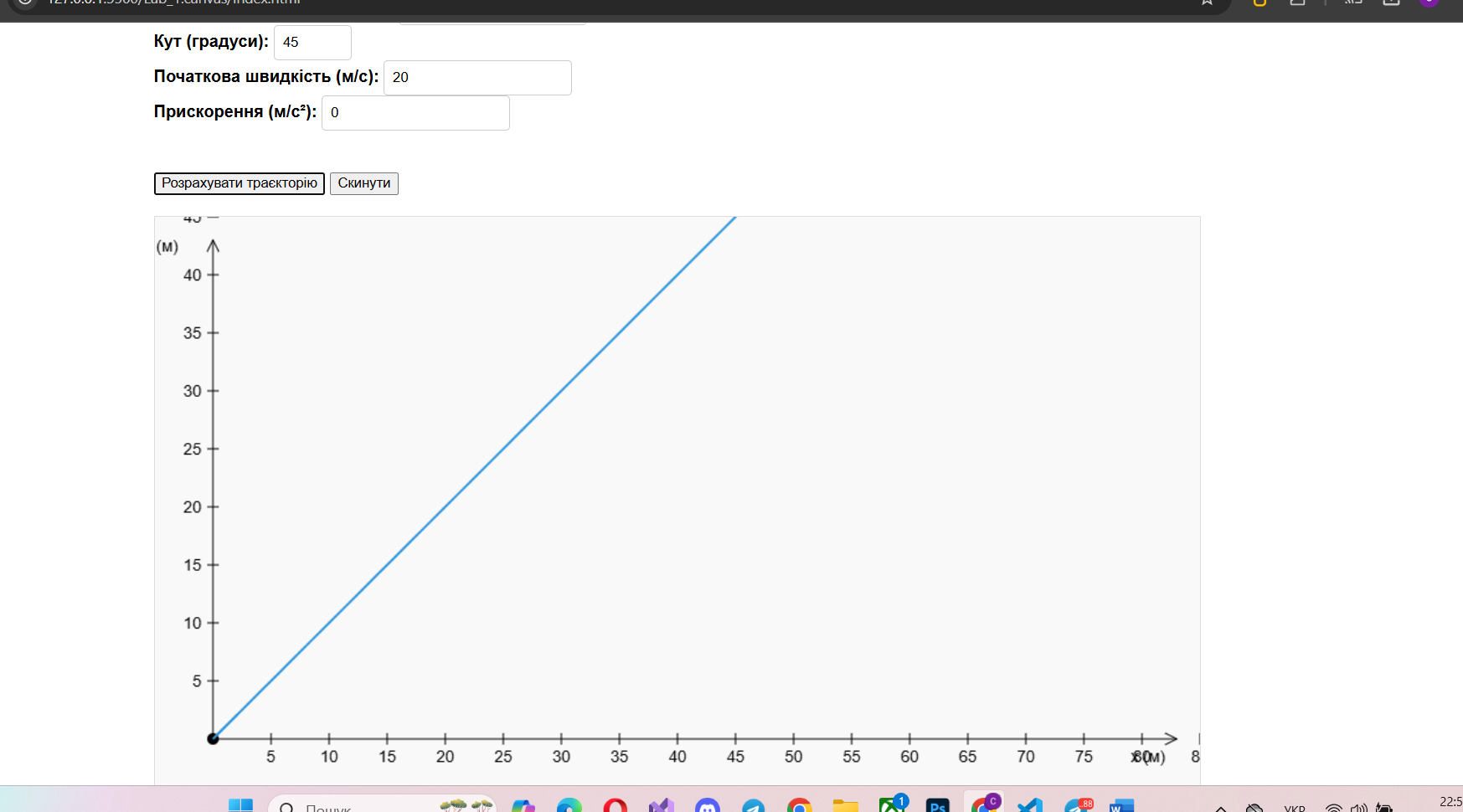
    resultsDiv.innerHTML = "";

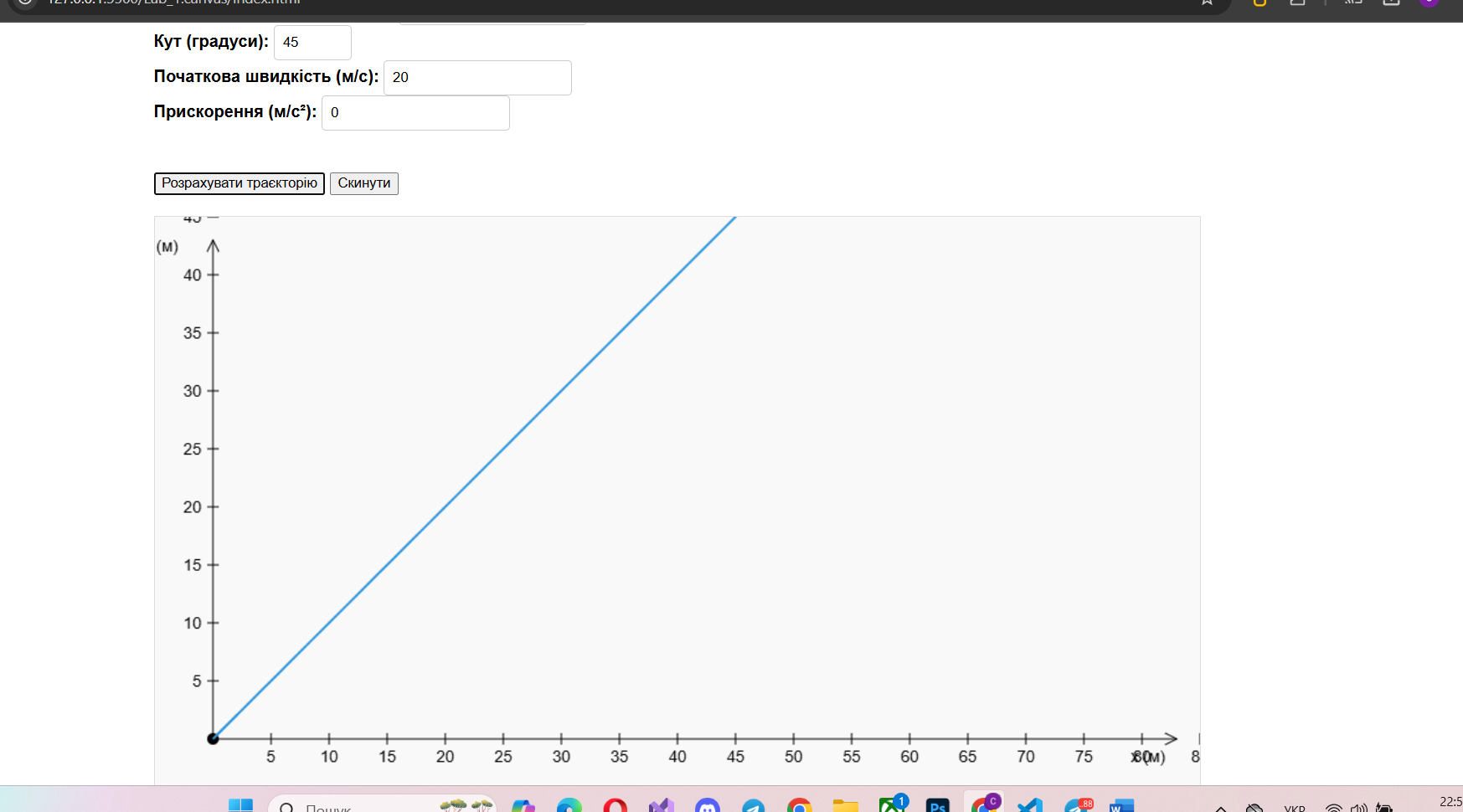
    drawAxes();

  });

});

Виконання коду





Інший варіант з 3d.Js

Приклад коду:

Html

<!DOCTYPE html>

<html lang="uk">

  <head>

    <meta charset="UTF-8" />

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

    <title>Моделювання руху матеріальної точки</title>

    <link rel="stylesheet" href="style.css" />

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

  </head>

  <body>

    <div class="container">

      <div class="inputs">

        <div class="form-group">

          <label for="x0">Початкова координата x₀ (м):</label>

          <input type="number" id="x0" value="0" />

        </div>

        <div class="form-group">

          <label for="y0">Початкова координата y₀ (м):</label>

          <input type="number" id="y0" value="0" />

        </div>

        <div class="form-group">

          <label for="angle">Кут (градуси):</label>

          <input type="number" id="angle" value="45" min="0" max="90" />

        </div>

        <div class="form-group">

          <label for="velocity">Початкова швидкість (м/с):</label>

          <input type="number" id="velocity" value="20" min="0" />

        </div>

        <div class="form-group">

          <label for="acceleration">Прискорення (м/с²):</label>

          <input

            type="number"

            id="acceleration"

            value="9.8"

            min="0"

            step="any"

          />

        </div>

      </div>

      <div class="controls">

        <button id="calculate">Розрахувати траєкторію</button>

        <button id="reset">Скинути</button>

      </div>

      <div id="trajectoryCanvas"></div>

    </div>

    <script src="index.js"></script>

  </body>

</html>

Js

document.addEventListener("DOMContentLoaded", () => {

  const calculateBtn = document.getElementById("calculate");

  const resetBtn = document.getElementById("reset");

  const margin = { top: 50, right: 50, bottom: 70, left: 70 };

  const width = 900 - margin.left - margin.right;

  const height = 500 - margin.top - margin.bottom;

  const svg = d3

    .select("#trajectoryCanvas")

    .append("svg")

    .attr("width", width + margin.left + margin.right)

    .attr("height", height + margin.top + margin.bottom)

    .append("g")

    .attr("transform", `translate(${margin.left}, ${margin.top})`);

  const xScale = d3.scaleLinear().domain([0, 50]).range([0, width]);

  const yScale = d3.scaleLinear().domain([0, 25]).range([height, 0]);

  function drawAxes() {

    svg.selectAll("\*").remove();

    const xAxis = d3

      .axisBottom(xScale)

      .tickValues(d3.range(0, 55, 5))

      .tickFormat((*d*) => *d*);

    svg

      .append("g")

      .attr("class", "x-axis")

      .attr("transform", `translate(0, ${height})`)

      .call(xAxis);

    const yAxis = d3

      .axisLeft(yScale)

      .tickValues(d3.range(0, 30, 5))

      .tickFormat((*d*) => *d*);

    svg.append("g").attr("class", "y-axis").call(yAxis);

    svg

      .append("text")

      .attr("class", "axis-label")

      .attr("x", width / 2)

      .attr("y", height + 40)

      .style("text-anchor", "middle")

      .text("x (м)");

    svg

      .append("text")

      .attr("class", "axis-label")

      .attr("transform", "rotate(-90)")

      .attr("x", -height / 2)

      .attr("y", -40)

      .style("text-anchor", "middle")

      .text("y (м)");

  }

  function calculateTrajectory() {

    const x0 = parseFloat(document.getElementById("x0").value) || 0;

    const y0 = parseFloat(document.getElementById("y0").value) || 0;

    const angle = parseFloat(document.getElementById("angle").value) || 45;

    const v0 = parseFloat(document.getElementById("velocity").value) || 20;

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

    a = isNaN(a) ? 9.8 : a;

    const isZeroAcceleration = Math.abs(a) < 1e-10;

    console.log("Чи нульове прискорення:", isZeroAcceleration);

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

    const v0x = v0 \* Math.cos(angleRad);

    const v0y = v0 \* Math.sin(angleRad);

    let flightTime, maxHeight, xMax;

    if (isZeroAcceleration) {

      if (v0y === 0) {

        flightTime = 10;

        maxHeight = y0;

      } else if (v0y > 0) {

        flightTime = 10;

        maxHeight = y0 + v0y \* flightTime;

      } else {

        if (y0 > 0) {

          flightTime = y0 / Math.abs(v0y);

        } else {

          flightTime = 0;

        }

        maxHeight = y0;

      }

      xMax = x0 + v0x \* flightTime;

    } else {

      const A = -0.5 \* a;

      const B = v0y;

      const C = y0;

      const discriminant = B \* B - 4 \* A \* C;

      if (discriminant < 0) {

        flightTime = 10;

      } else {

        const sqrtDisc = Math.sqrt(discriminant);

        const t1 = (-B + sqrtDisc) / (2 \* A);

        const t2 = (-B - sqrtDisc) / (2 \* A);

        if (t1 > 0 && t2 > 0) {

          flightTime = Math.min(t1, t2);

        } else if (t1 > 0) {

          flightTime = t1;

        } else if (t2 > 0) {

          flightTime = t2;

        } else {

          flightTime = 0;

        }

      }

      const tMaxHeight = v0y / a;

      if (tMaxHeight > 0 && tMaxHeight < flightTime) {

        maxHeight = y0 + v0y \* tMaxHeight - 0.5 \* a \* tMaxHeight \* tMaxHeight;

      } else {

        maxHeight = y0;

      }

      xMax = x0 + v0x \* flightTime;

    }

    if (flightTime > 20) {

      flightTime = 20;

    }

    xScale.domain([0, Math.ceil(xMax / 5) \* 5 + 5]);

    const maxYValue = Math.max(maxHeight, y0 + 5);

    yScale.domain([0, Math.ceil(maxYValue / 5) \* 5]);

    drawAxes();

    const trajectoryData = [];

    if (flightTime <= 0) {

      return;

    }

    const timeStep = flightTime / 100;

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

      let x, y;

      if (isZeroAcceleration) {

        x = x0 + v0x \* t;

        y = y0 + v0y \* t;

      } else {

        x = x0 + v0x \* t;

        y = y0 + v0y \* t - 0.5 \* a \* t \* t;

      }

      trajectoryData.push({ x, y });

    }

    const line = d3

      .line()

      .x((*d*) => xScale(*d*.x))

      .y((*d*) => yScale(*d*.y))

      .curve(d3.curveMonotoneX);

    svg

      .append("path")

      .datum(trajectoryData)

      .attr("class", "trajectory")

      .attr("fill", "none")

      .attr("stroke", "#3498db")

      .attr("stroke-width", 2)

      .attr("d", line);

    svg

      .append("circle")

      .attr("cx", xScale(x0))

      .attr("cy", yScale(y0))

      .attr("r", 5)

      .attr("fill", "#000");

    const arrowLength = 30;

    const endX = x0 + (arrowLength / 50) \* Math.cos(angleRad);

    const endY = y0 + (arrowLength / 50) \* Math.sin(angleRad);

    svg

      .append("line")

      .attr("x1", xScale(x0))

      .attr("y1", yScale(y0))

      .attr("x2", xScale(endX))

      .attr("y2", yScale(endY))

      .attr("stroke", "#000")

      .attr("stroke-width", 2);

    const arrowHeadSize = 5;

    const angle1 = angleRad + (Math.PI \* 3) / 4;

    const angle2 = angleRad - (Math.PI \* 3) / 4;

    svg

      .append("polygon")

      .attr(

        "points",

        `

          ${xScale(endX)},${yScale(endY)}

          ${xScale(endX - (arrowHeadSize / 50) \* Math.cos(angle1))},${yScale(

          endY - (arrowHeadSize / 50) \* Math.sin(angle1)

        )}

          ${xScale(endX - (arrowHeadSize / 50) \* Math.cos(angle2))},${yScale(

          endY - (arrowHeadSize / 50) \* Math.sin(angle2)

        )}

        `

      )

      .attr("fill", "#000");

    svg

      .append("text")

      .attr("x", 10)

      .attr("y", 20)

      .attr("class", "info-text")

      .text(`Максимальна висота: ${maxHeight.toFixed(2)} м`);

    svg

      .append("text")

      .attr("x", 10)

      .attr("y", 40)

      .attr("class", "info-text")

      .text(`Дальність польоту: ${xMax.toFixed(2)} м`);

    svg

      .append("text")

      .attr("x", 10)

      .attr("y", 60)

      .attr("class", "info-text")

      .text(`Час польоту: ${flightTime.toFixed(2)} с`);

  }

  drawAxes();

  calculateBtn.addEventListener("click", calculateTrajectory);

  resetBtn.addEventListener("click", () => {

    document.getElementById("x0").value = "0";

    document.getElementById("y0").value = "0";

    document.getElementById("angle").value = "45";

    document.getElementById("velocity").value = "20";

    document.getElementById("acceleration").value = "9.8";

    xScale.domain([0, 50]);

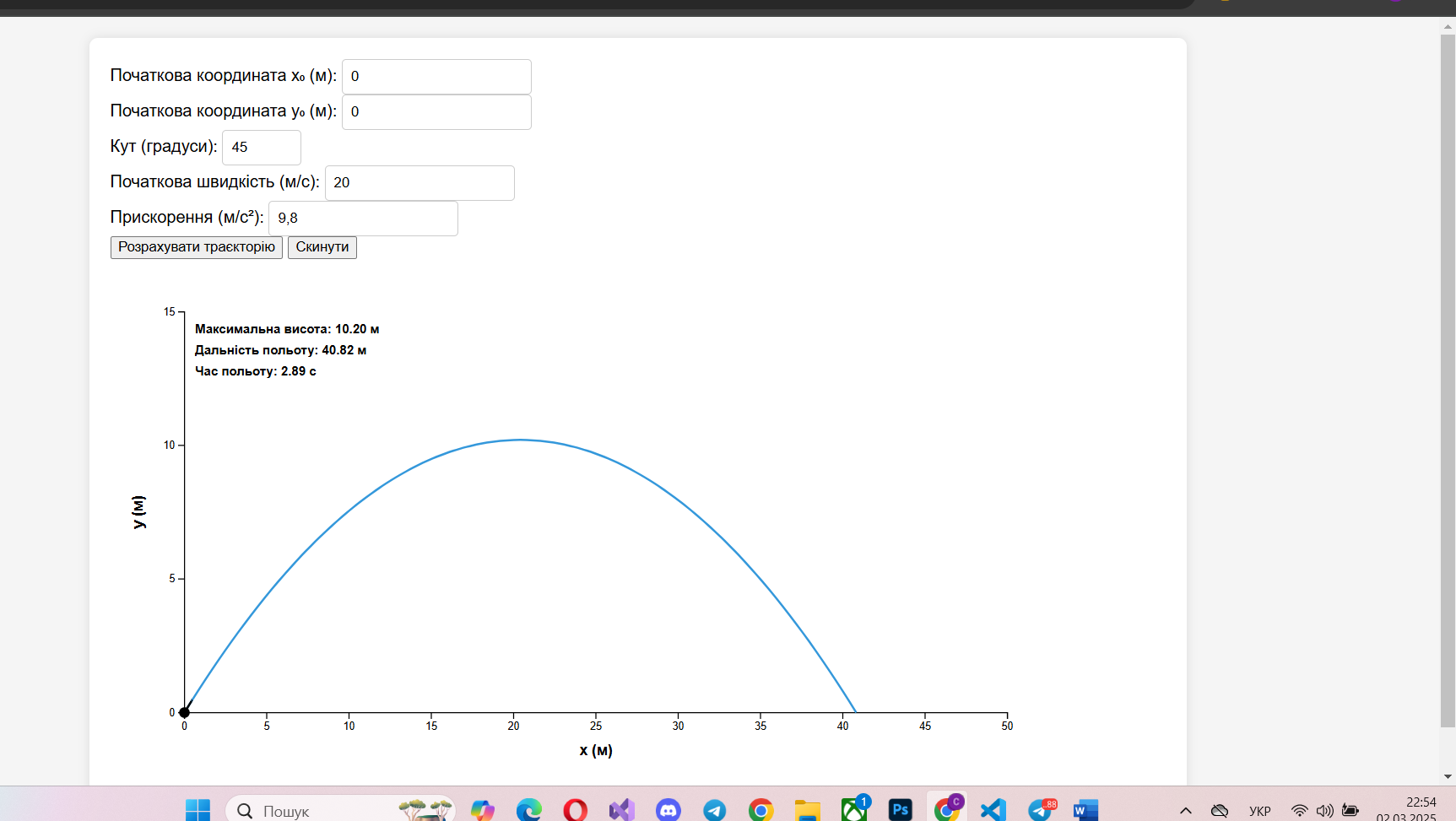
    yScale.domain([0, 25]);

    drawAxes();

  });

});

Результат подібний



Також я створив репозиторій де запушив ці два варіанти :

<https://github.com/Stas-collab/Lab_1/tree/master>

Висновки: У цій лабораторній роботі я дослідив різницю між canvas і 3D.JS.

Якщо потрібно ефективно обробляти велику кількість точок (наприклад, фізичні симуляції), **Canvas** буде кращим вибором. Якщо потрібна інтерактивна аналітика та масштабованість, **D3.js** — оптимальний варіант.