МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 2
З дисципліни: *“Комп’ютерна графіка”*
На тему: *“Програмування кривої Безьє”*

Лектор:
доц. каф. ПЗ
Левус Є.В.

Виконала:
ст. гр. ПЗ-26
Пелих О.Р.

Прийняв:
доц. каф. ПЗ
Ярема Н.П.

« _____ » _____ 2025 р.
∑=_____ _____

Львів – 2025

Тема роботи: Програмування кривої Безьє
Мета роботи: Навчитися програмувати алгоритми побудови кривої Безьє

ЗАВДАННЯ

Створити редактор кривої Безьє, який має такий функціонал:

1. введення і редагування вершин характеристичної ламаної,
2. побудова кривої за параметричною формулою,
3. додавання нової точки для характеристичної ламаної,
4. контроль коректності введених даних,
5. виведення необхідних підказок, повідомлень,
6. виконання індивідуального варіанту.

ХІД ВИКОНАННЯ

ТЕХНОЛОГІЇ:

HTML – використовується для створення структури веб-сторінки та розміщення елементів, таких як форми введення та полотно (canvas) для малювання графіки.
CSS – відповідає за стилізацію сторінки, зокрема оформлення форм та розташування елементів.
JavaScript (Canvas API) – використовується для малювання графічних об'єктів. Методи на кшталт getContext, beginPath, moveTo, lineTo, stroke та fill допомагають керувати зображенням на полотні.
Переваги обраних технологій:
HTML, CSS та JavaScript є основними інструментами для розробки веб-сторінок, що гарантує їхню підтримку у всіх сучасних браузерах. Canvas API є простим у використанні та дозволяє ефективно працювати з графікою.
Основні функції та методи в коді:
sliderAlignAndCoordinates(event) – Оновлює значення повзунка, його положення та перебудовує координатну сітку.
buildCoordinates(value) – Малює координатну сітку, контрольні точки та криву Безьє.
convertXYtoReal(dot) – Перетворює локальні координати точки у пікселі екрану.
convertRealToLocal(realX, realY) – Перетворює координати пікселів екрану у локальні координати.
casteljauAlgorithm(points, t) – Обчислює точку на кривій Безьє за допомогою алгоритму де Кастельйо.
drawBezierCurve(points) – Побудовує криву Безьє за заданими точками з використанням алгоритму де Кастельйо.
createDot(event) – Додає нову контрольну точку при натисканні на canvas.
clearAll() – Очищає всі точки та оновлює координатну сітку.
getPointsRange() – Виводить координати кривої Безьє з вибраним кроком по X.
calculateBernstein() – Обчислює значення поліномів Бернштейна для заданого t.

ВАРІАНТ 6

6. Візуалізувати Безьє за рекурсивною формулою; відобразити керуючі точки одним кольором, а опорні – іншим; вивести координати точок кривої з заданим користувачем кроком по осі X; обчислити значення перших n-2 поліномів Бернштейна із введеним користувачем кроком для параметру t.

Рис. 1. Побудова кривої

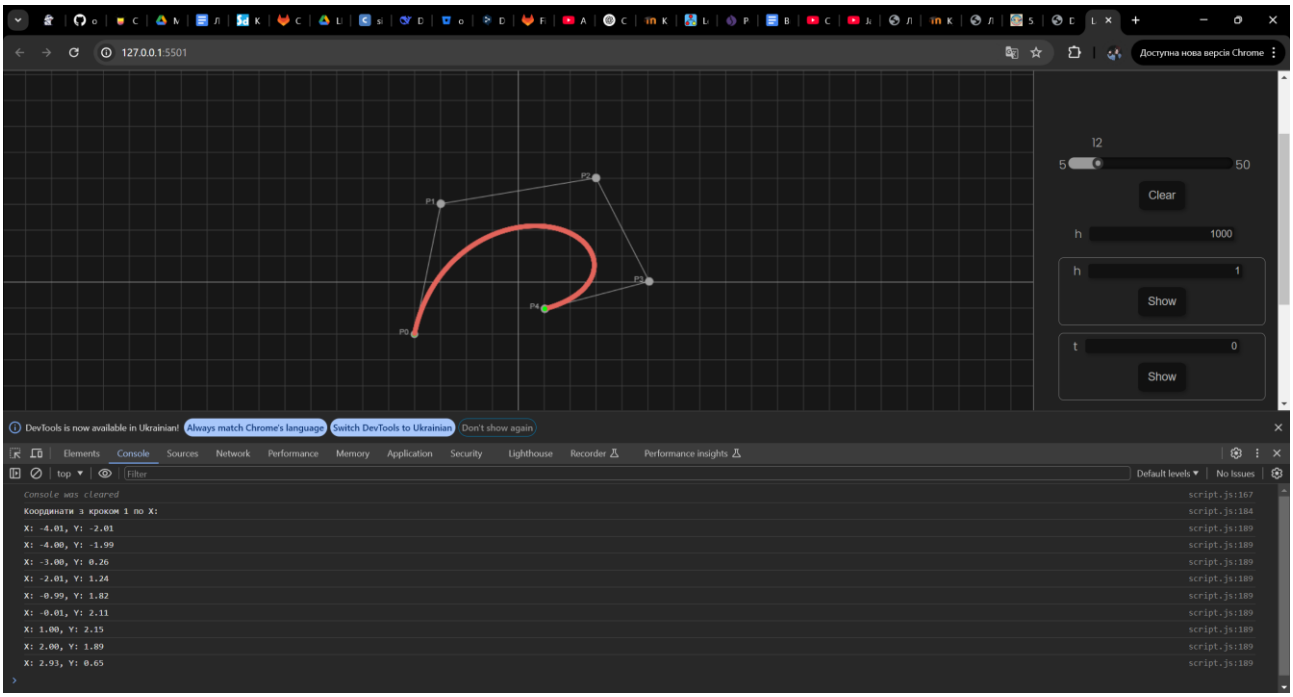

Рис. 2. Виведення координат точок кривої з заданим користувачем кроком по осі X

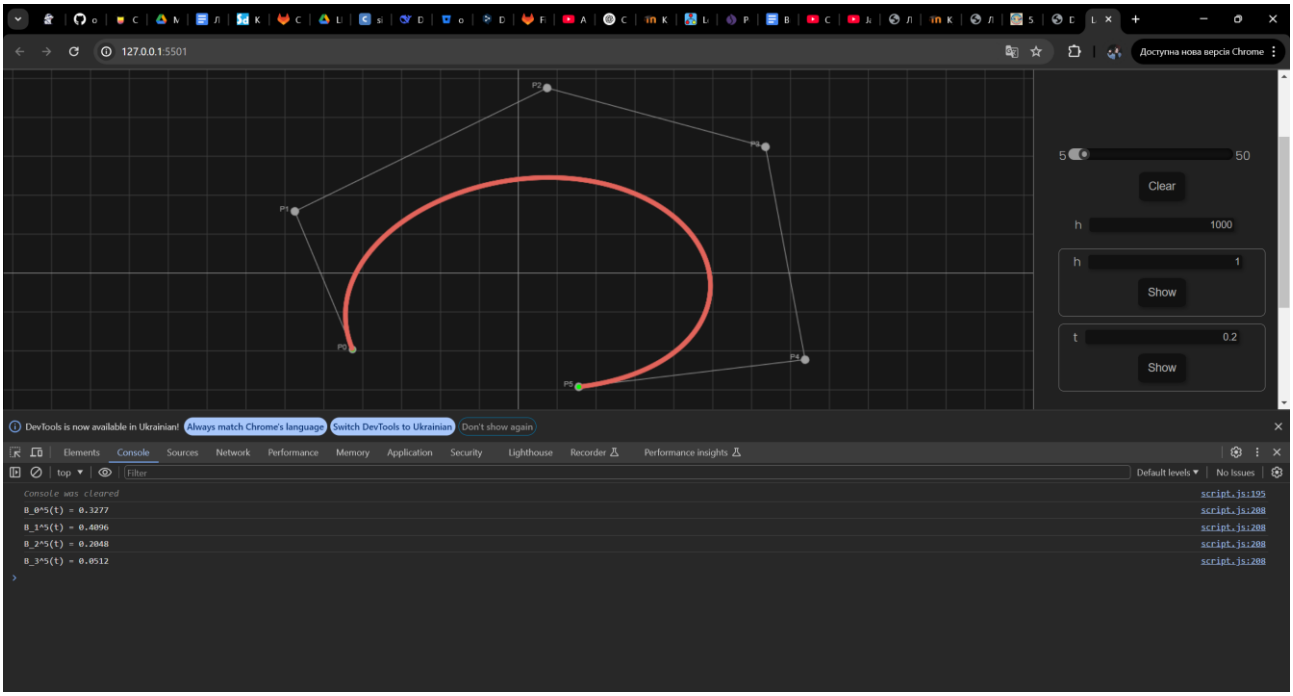

Рис. 3. Обчислити значення перших n-2 поліномів Бернштейна із введеним користувачем кроком для параметру t

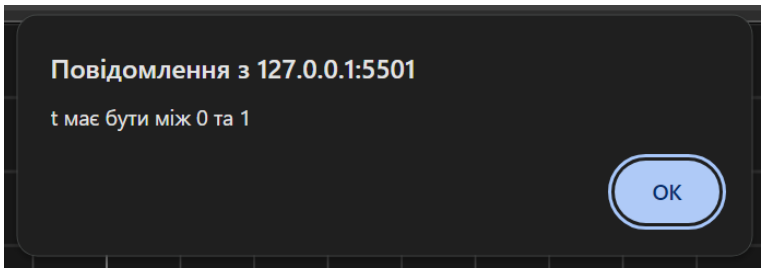

Рис. 4. Виведення повідомлення помилки при некоректно введених даних

КОД ПРОГРАМИ

index.css

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="./css/style.css">
    <meta charset="UTF-8" />
    <meta name="author" content="Anastasiia Feduniak">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Lab2 Feduniak</title>

</head>
<body>
    <canvas id="editor" width="auto" height="auto"></canvas>
    <div id="control-panel">
        <div class="wrapper">
            <label for="slider" id="curr-amount">0</label>
            <label for="slider">5</label>
         <input type="range" id="slider" min="5" max="50" value="5">
             <label for="slider">50</label>

        </div>
```

```
        <button id="clearField">Clear</button>
        <form id="gaps">
            <label for="h-gap">h</label>
            <input type="number" class="inputs" id="h-gap" value="0"></form>
        </form>

        <div class="add1">
            <div>
                <label for="h-num">h</label>
                <input type="number" class="inputs" value="1" id="h-num">
            </div>
            <button id="show-something">Show</button>
        </form>
    </div>
    <div class="add1">
        <div>
            <label for="bernstein-t">t</label>
            <input type="number" class="inputs" value="0" id="bernstein-t">
        </div>
        <button id="bernstein-polynom">Show</button>
    </div>


<script src="./js/script.js"></script>
</body>
</html>
```

style.css

```
@import
url("https://fonts.googleapis.com/css2?family=Poppins:wght@300&display=swap");

#editor {
  border: solid 1px;
  border-right-color: #a1a1a1;
  background-color: #171717;
}

body {
  margin: 0;
  display: flex;
  font-color: #fff;
}

#control-panel {
  background-color: #212121;
  flex-grow: 1;
  border-left-color: #ffffff;
  padding-left: 2%;
  padding-right: 1%;
  margin-top: 0;
```

```
  display: flex;
  flex-direction: column;
  justify-content: center;
}

#slider {
  margin-left: 1%;
  margin-right: 1%;
  width: 80%;
  height: 15px;
  -webkit-appearance: none;
  background: #111;
  outline: none;
  border-radius: 15px;
  overflow: hidden;
  box-shadow: inset 0 0 5px rgba(0, 0, 0, 1);
}
#slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  width: 15px;
  height: 15px;
  border-radius: 50%;
  background: #999999;
  cursor: pointer;
  border: 4px solid #333;
  box-shadow: -407px 0 0 400px #999999;
}

.wrapper {
  display: flex;
}

label {
  font-family: "Poppins", sans-serif;
  color: #b3b3b3;
}

#curr-amount {
  position: absolute;
  transform: translateX(-50%);
  text-align: center;
}

button {
  margin-top: 10px;
  align-self: center;
  background-color: #111;
  border: #111 solid 1px;
  border-radius: 15%;
  box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.3);
  color: #b3b3b3;
  padding: 10px 12px;
  font-size: 15px;
```

```
}

#gaps {
  justify-content: center;
  margin-top: 20px;
  display: flex;
  font-size: 15px;
  margin: 20px 20px;
}

.inputs {
  width: 80%;
  margin: auto 20px;
  background-color: #111;
  border: #111 solid 1px;
  border-radius: 0.2rem;
  box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.3);
  margin-left: 10px;
  color: #b3b3b3;
  text-align: right;
}

input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button {
  -webkit-appearance: none;
  margin: 0;
}

.inputs:focus {
  outline: none;
  border-color: #a1a1a1;
}

.add1 {
  padding: 2%;
  border: 1px #808080 solid;
  display: flex;
  flex-direction: column;
  border-radius: 0.4rem;
  justify-content: center;
  padding-bottom: 4%;
  margin-bottom: 10px;
}

.add1 div {
  display: flex;
  margin-left: 5%;
}
```

script.js

```
const canvas = document.getElementById("editor");
```

```
const slider = document.getElementById("slider");
canvas.height = window.innerHeight;
canvas.width = window.innerWidth * 0.8;
let selectedDotIndex = -1;
var gap = 0;
let dots = [];
let curveDots = []; //Масив точок у реальній системі координат (пікселі екрана)
window.addEventListener(
  "load",
  sliderAlignAndCoordinates(document.getElementById("slider"))
);
slider.addEventListener("input", sliderAlignAndCoordinates);

function sliderAlignAndCoordinates(event) {
  const sliderValue = document.getElementById("curr-amount");
  sliderValue.textContent = slider.value;
  const thumbWidth =
    (parseInt(window.getComputedStyle(slider).getPropertyValue("width")) *
      this.value) /
    (parseInt(this.max) - parseInt(this.min));
  sliderValue.style.left = `${
    slider.getBoundingClientRect().left + thumbWidth - 20
  }px`;
  sliderValue.style.top = `${
    slider.getBoundingClientRect().top - slider.offsetHeight * 2
  }px`;
  buildCoordinates(slider.value);
}

function buildCoordinates(value) {
  var context = canvas.getContext("2d");
  context.clearRect(0, 0, canvas.width, canvas.height);

  // Сітка
  var ct = canvas.getContext("2d");
  ct.strokeStyle = "#414141";
  ct.beginPath();
  gap = canvas.height / (parseInt(value) * 2);
  for (var i = 0; i < parseInt(value) * 2; i++) {
    ct.moveTo(0, i * gap);
    ct.lineTo(canvas.width, i * gap);
  }

  var n = parseInt(canvas.width / gap);
  var start = (canvas.width - n * gap) / 2;
  if (n % 2 !== 0) {
    start += gap / 2;
  }
  for (var i = 0; i < n * 2; i++) {
    ct.moveTo(start + i * gap, 0);
    ct.lineTo(start + i * gap, canvas.height);
  }
  ct.stroke();
```

```
  ct.beginPath();
  ct.strokeStyle = "#A1A1A1";
  ct.moveTo(0, canvas.height / 2);
  ct.lineTo(canvas.width, canvas.height / 2);
  ct.moveTo(canvas.width / 2, 0);
  ct.lineTo(canvas.width / 2, canvas.height);
  ct.stroke();

  // Керуючі точки та лінії
  ct.strokeStyle = "#A1A1A1";
  ct.fillStyle = "#A1A1A1";
  var prevX = 0,
    prevY = 0;
  for (var i = 0; i < dots.length; i++) {
    var dot = dots[i];
    ct.beginPath();
    var newX = canvas.width / 2 + dot[0] * gap;
    var newY = canvas.height / 2 - dot[1] * gap;
    ct.arc(newX, newY, 5, 0, 2 * Math.PI);
    ct.fillText("P" + i, newX - 20, newY);
    ct.fill();
    ct.closePath();
    if (i === 0) {
      prevX = newX;
      prevY = newY;
      continue;
    }
    ct.moveTo(prevX, prevY);
    ct.lineTo(newX, newY);
    ct.stroke();
    prevX = newX;
    prevY = newY;
  }

  // Крива Безьє
  if (dots.length >= 3) {
    drawBezierCurve(dots);
  }
}

function convertXYtoReal(dot) {
  return [canvas.width / 2 + dot[0] * gap, canvas.height / 2 - dot[1] * gap];
}

function convertRealToLocal(realX, realY) {
  return [(realX - canvas.width / 2) / gap, (canvas.height / 2 - realY) / gap];
}

function casteljauAlgorithm(points, t) {
  var ct = canvas.getContext("2d");
  let p = points.map((dot) => convertXYtoReal(dot));
```

```
  // Алгоритм де Кастельйо
  while (p.length > 1) {
    let pNew = [];
    ct.beginPath();
    for (let j = 0; j < p.length - 1; j++) {
      let x = (1 - t) * p[j][0] + t * p[j + 1][0];
      let y = (1 - t) * p[j][1] + t * p[j + 1][1];
      pNew.push([x, y]);
    }
    p = pNew;
  }

  const epsilon = 1e-6; // Допустима похибка для порівняння
  if (Math.abs(t - 0) < epsilon) {
    // Початкова точка (t = 0)
    ct.fillStyle = "#00FF00"; // Зелений
  } else if (Math.abs(t - 1) < epsilon) {
    // Кінцева точка (t = 1)
    ct.fillStyle = "#00FF00"; // Зелений
  } else {
    // Решта точок
    ct.fillStyle = "#e3645b"; // Червоний
  }

  // Малювання точки
  ct.beginPath();
  ct.arc(p[0][0], p[0][1], 3, 0, 2 * Math.PI);
  ct.fill();

  curveDots.push([p[0][0], p[0][1]]);
}

function drawBezierCurve(points) {
  curveDots = [];
  let h = 1 / Math.max(parseInt(document.getElementById("h-gap").value), 100);
  for (let t = 0; t <= 1 + h / 2; t += h) {
    casteljauAlgorithm(points, Math.min(t, 1)); // t не перевищує 1
  }
}

canvas.addEventListener("click", createDot);

function createDot(event) {
  var rect = canvas.getBoundingClientRect();
  var x = event.clientX - rect.left;
  var y = event.clientY - rect.top;
  var local = convertRealToLocal(x, y);
  dots.push([local[0], local[1]]);
  buildCoordinates(slider.value);
}

function clearAll() {
  dots = [];
```

```
  curveDots = [];
  buildCoordinates(slider.value);
}

// Виведення координат з кроком по X
function getPointsRange() {
  console.clear();
  let step = parseFloat(document.getElementById("h-num").value);
  if (step <= 0) {
    alert("Крок має бути додатнім");
    return;
  }

  let localPoints = curveDots.map((dot) => {
    let local = convertRealToLocal(dot[0], dot[1]);
    return { x: local[0], y: local[1] };
  }); //Масив localPoints з координатами {x, y} у локальній системі

  localPoints.sort((a, b) => a.x - b.x);

  let minX = Math.floor(localPoints[0].x / step) * step;
  let maxX = Math.ceil(localPoints[localPoints.length - 1].x / step) * step;

  console.log("Координати з кроком " + step + " по X:");
  for (let x = minX; x <= maxX; x += step) {
    let closest = localPoints.reduce((prev, curr) =>
      Math.abs(curr.x - x) < Math.abs(prev.x - x) ? curr : prev
    );
    console.log(`X: ${closest.x.toFixed(2)}, Y: ${closest.y.toFixed(2)}`);
  }
}

// Обчислення поліномів Бернштейна
function calculateBernstein() {
  console.clear();
  let t = parseFloat(document.getElementById("bernstein-t").value);
  if (t < 0 || t > 1) {
    alert("t має бути між 0 та 1");
    return;
  }
  let m = dots.length - 1; // Ступінь кривої
  if (m < 2) {
    alert("Потрібно щонайменше 3 точки");
    return;
  }
  for (let i = 0; i <= m - 2; i++) {
    let bern = combination(m, i) * Math.pow(t, i) * Math.pow(1 - t, m - i);
    console.log(`B_${i}^${m}(t) = ${bern.toFixed(4)}`);
  }
}

// Допоміжні функції
function combination(n, k) {
```

```
  return factorial(n) / (factorial(k) * factorial(n - k));
}

function factorial(num) {
  if (num === 0) return 1;
  return num * factorial(num - 1);
}

document.getElementById("clearField").addEventListener("click", clearAll);
document
  .getElementById("show-something")
  .addEventListener("click", getPointsRange);
document
  .getElementById("bernstein-polynom")
  .addEventListener("click", calculateBernstein);
```

ВИСНОВКИ

У цій лабораторній роботі було реалізовано візуалізацію кривих Безьє за допомогою JavaScript та Canvas API. Було створено веб-додаток, який дозволяє будувати криві, використовуючи алгоритм де Кастельйо, та відображати координатну сітку з контрольними точками.

Користувач може додавати точки, очищувати полотно та отримувати координати точок кривої з вибраним кроком по осі X. Також було реалізовано обчислення поліномів Бернштейна. Найважчим було коректно запрограмувати формулу для рекурсивної реалізації кривої Безьє.