**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

**ІКНІ**

Кафедра **ПЗ**

### ЗВІТ

до лабораторної роботи № 1

**на тему:** Побудова двовимірних зображень

**з дисципліни:** *“Комп’ютерна графіка”*

**Лектор:**

к.т.н., доцент

Левус Є.В.

**Виконав:**

ст. гр. ПЗ-26

Мінтус С. С.

**Прийняла:**

к.т.н, старший викладач

Ярема Н.П.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2024 р.

∑= \_\_\_\_\_ .

Львів – 2024

**Тема роботи:** Побудова двовимірних зображень.

**Мета роботи:** Навчитись будувати двомірні зображення з допомогою графічних примітивів мови програмування.

**Теоретичні відомості**

Для виконання лабораторної роботи було обрано такі технології: HTML, CSS, JS. Для побудови двовимірних зображень було використано <canvas> - це HTML елемент, що використовується для малювання графіки засобами мови програмування JS. Вибрав саме його через простоту у використанні.

Було розроблено клас Triangle з методами draw() – для побудови трикутника залитого певним кольором (колір обирається при створенні об’єкта) та drawRectAround() – для побудови прямокутника навколо трикутника.

Було розроблено функцію drawFigure() – функція, яка викликається при натиснені на кнопку Create Rectangle. Функція за допомогою DOM бере значення з полів input та використовує їх для створення об’єкту Triangle.

Було використано вже існуючі функції JS, такі як: getContext(), getElementById() – для отримання елементу за його ID, moveTo() – дял переміщення «пензлика», lineTo() – для проведення «пензликом» лінії до конкретної точки, stroke() – для відображення проведених ліній, fillText() – для відображення тексту, beginPath() – позначити початок контуру, closePath() – позначити кінець контуру, fill() – для заливки фігури обмеженої контуром.

**Формулювання завдання**

Написати програму згідно індивідуального варіанту вибраною мовою програмування з використанням її базових графічних примітивів.

Програма має відповідати таким вимогам:

1. Відображення системи координат з початком у центрі області виведення з відповідними підписами та позначками (початок, одиничний відрізок, напрям, назва осей).

2. Задання фігур за введеними координати, що відповідають координатам відповідної побудованої декартової системи, а не координатам області виведення (Canvas).

3. Оптимальний ввід користувачем координат фігури з автоматичним обчисленням за можливості інших координат для уникнення зайвих обчислень користувачем.

4. Передбачити можливість некоректного введення даних.

5. Зручний інтерфейс користувача.

**Варіант 4**.

Побудувати декілька рівнобедрених трикутників за введеними координатами вершин основи та довжини висоти, що опущена на основу. Забезпечити можливість вибору кольору заливки та побудови прямокутників навколо трикутників.

**Текст програми**

**index.html**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

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

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

    <title>LAB 1</title>

  </head>

  <body>

    <main>

      <canvas class="coord" id="canvas"></canvas>

    </main>

    <footer>

      <h1>Lab 1 task</h1>

      <div class="table-container">

        <table>

          <tr>

            <td>

              <span>Point A (</span><input type="text" name="x" id="Ax" /><span

                >, </span

              ><input type="text" name="y" id="Ay" /><span>)</span>

            </td>

          </tr>

          <tr>

            <td>

              <span>Point B (</span><input type="text" name="x" id="Bx" /><span

                >, </span

              ><input type="text" name="y" id="By" /><span>)</span>

            </td>

          </tr>

        </table>

        <table>

          <tr>

            <td><span>Triangle Height = </span></td>

            <td><input type="text" id="height" /></td>

          </tr>

          <tr>

            <td><span>Triangle Color</span></td>

            <td><input type="color" id="triangle-color" /></td>

          </tr>

        </table>

        <table>

          <tr>

            <td><span>Rectangle </span></td>

            <td><input type="checkbox" id="isRectangleEnabled" /></td>

          </tr>

          <tr>

            <td><span>Rectangle Color</span></td>

            <td><input type="color" id="rectangle-color" /></td>

          </tr>

        </table>

        <input type="button" value="Create Triangle" onclick="drawFigure()" />

      </div>

    </footer>

    <script src="script.js"></script>

  </body>

</html>

**style.css**

\* {

  margin: 0;

  padding: 0;

  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;

  font-size: 20px;

  background-color: rgb(249, 249, 249);

}

footer {

  padding: 10px;

}

footer .table-container {

  display: flex;

}

main {

  display: flex;

  justify-content: center;

}

.coord {

  width: 100%;

  height: 800px;

  background-color: white;

}

table {

  display: inline-block;

  margin: 0 10px;

}

input {

  width: 50px;

}

input[type="color"] {

  height: 35px;

  width: 100%;

}

input[type="checkbox"] {

  transform: scale(1.5);

}

input[type="button"] {

  width: 150px;

  text-align: center;

}

**script.js**

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

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

let window\_height = 800;

let window\_width = window.innerWidth;

canvas.width = window\_width;

canvas.height = window\_height;

// context.beginPath();

context.moveTo(window\_width / 2, 0);

context.lineTo(window\_width / 2, window\_height);

context.moveTo(0, window\_height / 2);

context.lineTo(window\_width, window\_height / 2);

context.stroke();

context.font = "12px serif";

context.fillText(0, window\_width / 2 - 12, window\_height / 2 + 15);

// context.textAlign = "start";

for (let i = 0; i < window\_width / 2; i += 10) {

  context.moveTo(window\_width / 2 + i, window\_height / 2 - 3);

  context.lineTo(window\_width / 2 + i, window\_height / 2 + 3);

  context.moveTo(window\_width / 2 - i, window\_height / 2 - 3);

  context.lineTo(window\_width / 2 - i, window\_height / 2 + 3);

  context.stroke();

  if (i % 50 == 0 && i != 0) {

    context.fillText(i / 10, window\_width / 2 - 5 + i, window\_height / 2 + 14);

    context.fillText(

      -i / 10,

      window\_width / 2 - 10 - i,

      window\_height / 2 + 14

    );

  }

}

for (let i = 0; i < window\_height / 2; i += 10) {

  context.moveTo(window\_width / 2 - 3, window\_height / 2 + i);

  context.lineTo(window\_width / 2 + 3, window\_height / 2 + i);

  context.moveTo(window\_width / 2 - 3, window\_height / 2 - i);

  context.lineTo(window\_width / 2 + 3, window\_height / 2 - i);

  context.stroke();

  if (i % 50 == 0 && i != 0) {

    context.fillText(-i / 10, window\_width / 2 - 20, window\_height / 2 + i + 5);

    context.fillText(i / 10, window\_width / 2 - 20, window\_height / 2 - i + 5);

  }

}

context.font = "20px serif";

context.moveTo(window\_width, window\_height / 2);

context.lineTo(window\_width - 5, window\_height / 2 + 5);

context.moveTo(window\_width, window\_height / 2);

context.lineTo(window\_width - 5, window\_height / 2 - 5);

context.fillText("x", window\_width - 15, window\_height / 2 + 30);

context.moveTo(window\_width / 2, 0);

context.lineTo(window\_width / 2 - 5, 5);

context.moveTo(window\_width / 2, 0);

context.lineTo(window\_width / 2 + 5, 5);

context.fillText("y", window\_width / 2 - 20, 15);

context.stroke();

// for (let i = 0; i < window\_width / 2; i += 10) {

//   console.log(typeof i);

//   context.moveTo(window\_width / 2 + i, window\_height / 2 - 3);

//   context.lineTo(window\_width / 2 + i, window\_height / 2 + 3);

//   context.stroke();

//   if (i % 50 == 0)

//     context.fillText(i / 10, window\_width / 2 - 10 + i, window\_height / 2 + 14);

// }

// context.stroke();

class Triangle {

  constructor(Ax, Ay, Bx, By, height, color, rectangle\_color) {

    this.Ax = Ax \* 10 + window\_width / 2;

    this.Ay = -Ay \* 10 + window\_height / 2;

    this.Bx = Bx \* 10 + window\_width / 2;

    this.By = -By \* 10 + window\_height / 2;

    this.height = height \* 10;

    this.color = color;

    this.rectangle\_color = rectangle\_color;

  }

  #determine(Ax, Ay, Bx, By, length) {

    let px = this.Ax - this.Bx;

    let py = this.Ay - this.By;

    const len = length / Math.hypot(px, py);

    px \*= len;

    py \*= len;

    return [-py, px];

  }

  draw() {

    // let region = new Path2D();

    context.beginPath();

    context.moveTo(this.Ax, this.Ay);

    context.lineTo(this.Bx, this.By);

    context.moveTo(this.Bx, this.By);

    let coords = this.#determine(

      this.Ax,

      this.Ay,

      this.Bx,

      this.By,

      this.height

    );

    context.lineTo(

      (this.Bx + this.Ax) / 2 + coords[0],

      (this.By + this.Ay) / 2 + coords[1]

    );

    context.lineTo(this.Ax, this.Ay);

    context.closePath();

    context.fillStyle = this.color;

    context.fill();

    // context.strokeStyle = "blue";

    context.stroke();

  }

  drawRectAround() {

    let coords = this.#determine(

      this.Ax,

      this.Ay,

      this.Bx,

      this.By,

      this.height

    );

    context.beginPath();

    context.moveTo(this.Ax, this.Ay);

    context.lineTo(this.Bx, this.By);

    context.lineTo(this.Bx + coords[0], this.By + coords[1]);

    context.lineTo(this.Ax + coords[0], this.Ay + coords[1]);

    context.lineTo(this.Ax, this.Ay);

    context.fillStyle = this.rectangle\_color;

    context.fill();

    context.stroke();

    context.closePath();

  }

}

function drawFigure() {

  const Ax = document.getElementById("Ax").value;

  const Bx = document.getElementById("Bx").value;

  const Ay = document.getElementById("Ay").value;

  const By = document.getElementById("By").value;

  const height = document.getElementById("height").value;

  //   console.log(!isNaN(Ax.value));

  if (

    isNaN(Ax) ||

    isNaN(Bx) ||

    isNaN(Ay) ||

    isNaN(By) ||

    isNaN(height) ||

    Ax === "" ||

    Bx === "" ||

    Ay === "" ||

    By === "" ||

    height === "" ||

    Math.abs(Ax) > 95 ||

    Math.abs(Bx) > 95 ||

    Math.abs(Ay) > 40 ||

    Math.abs(By) > 40

    // Math.abs(height) >  ||

  ) {

    alert("Non valid input data!");

    return;

  }

  const checkbox = document.getElementById("isRectangleEnabled").checked;

  const triangle\_color = document.getElementById("triangle-color").value;

  const rectangle\_color = document.getElementById("rectangle-color").value;

  let triangle = new Triangle(

    Ax,

    Ay,

    Bx,

    By,

    height,

    triangle\_color,

    rectangle\_color

  );

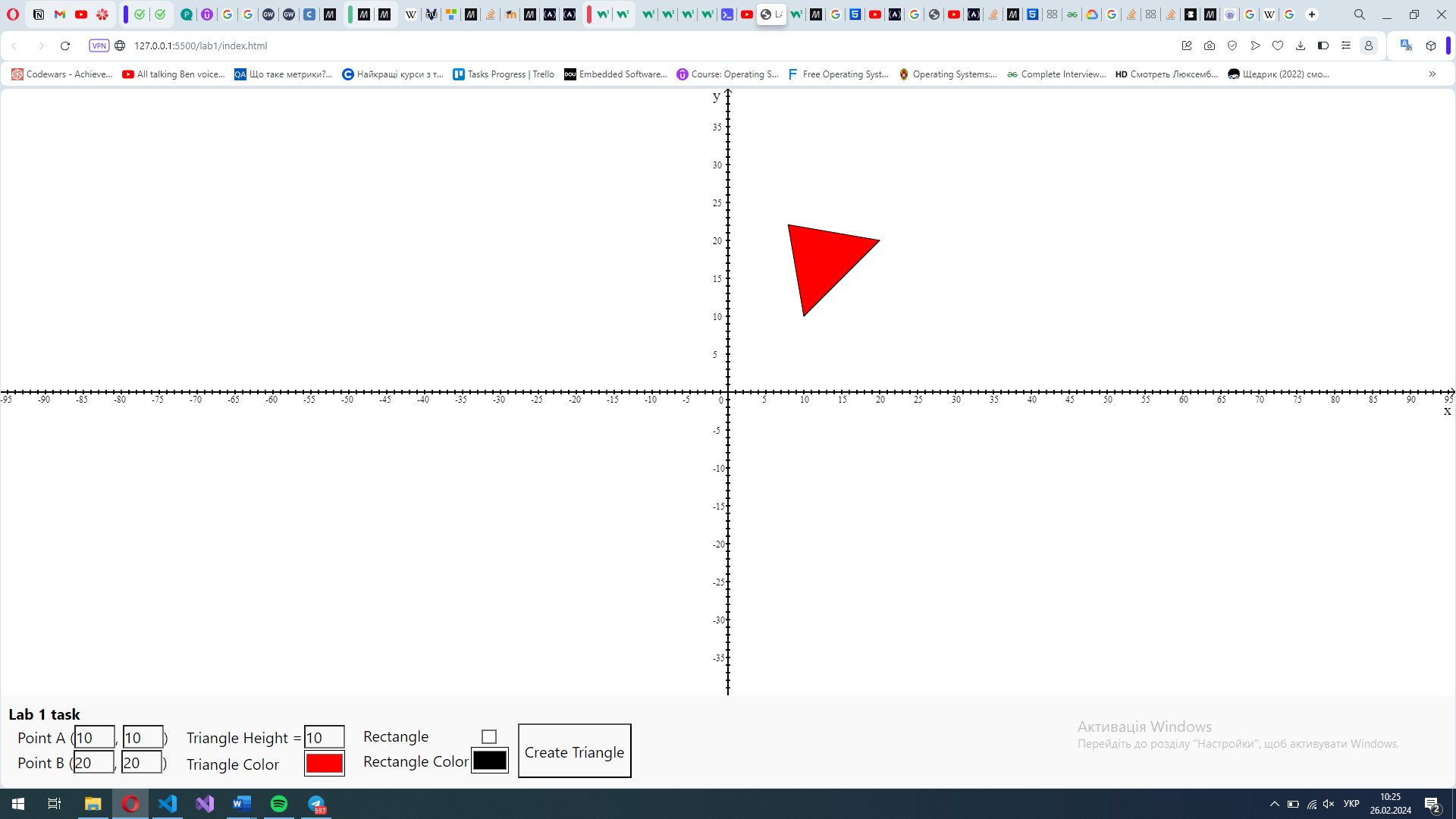
  if (checkbox) triangle.drawRectAround();

  triangle.draw();

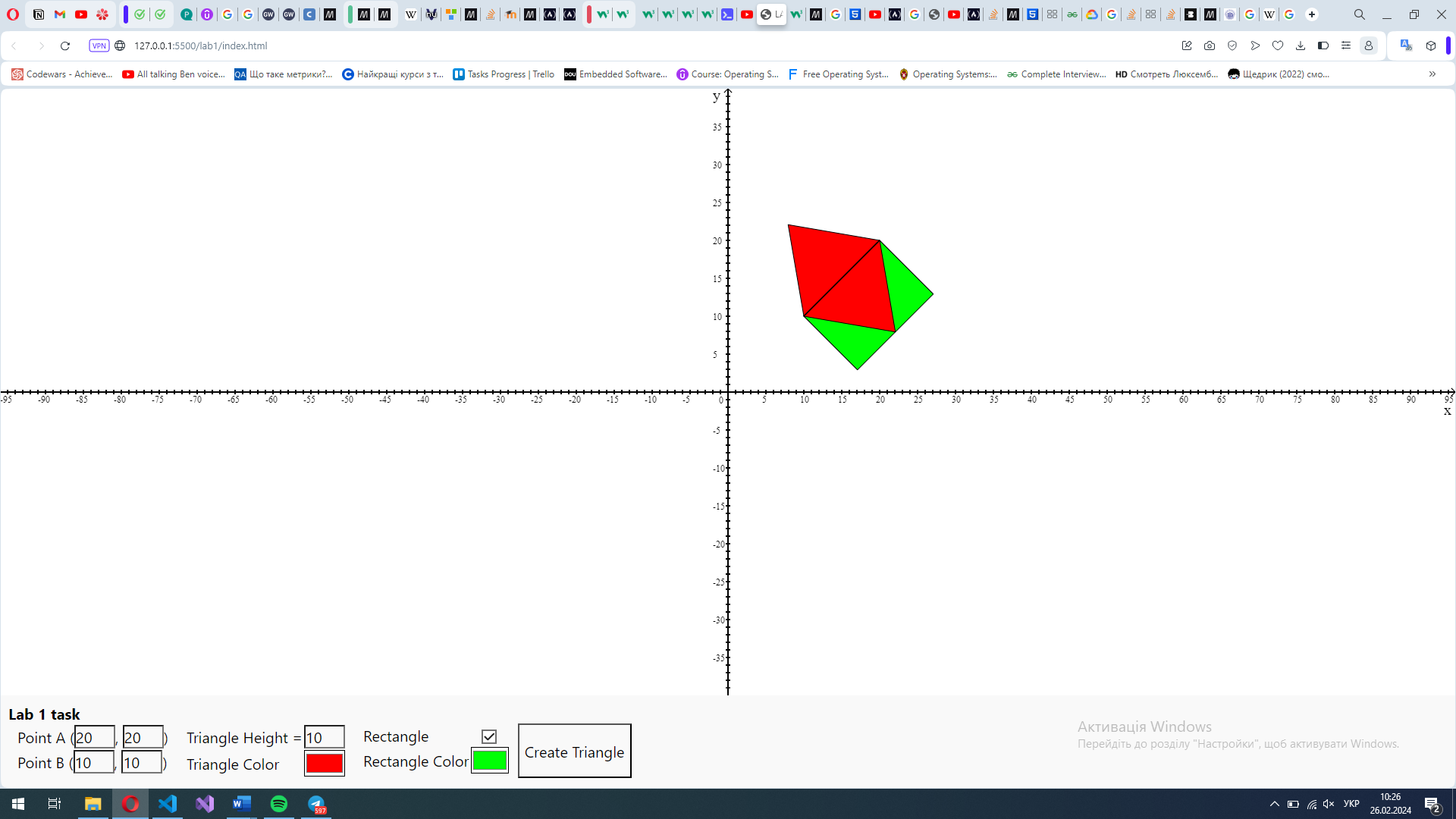
  return;

}

**Результати виконання програми**



**Рис. 1. Трикутник, основа AB якого має координати A(10, 10), B(20, 20), з висотою 10 та кольором заливки червоним.**



**Рис. 2. Трикутник, основа AB якого має координати A(20, 20), B(10, 10), з висотою 10 та кольором заливки трикутника червоним, а заливка прямокутника описаного навколо - зелена.**

**Висновки**

Індивідуальне завдання виконано: створено веб-сторінку з координатною площиною, на якій можна побудувати трикутник за координатами його основи та заданою висотою, також є можливість обрати колір трикутника та можливість побудувати прямокутник навколо трикутника з вибраним для нього кольором.

Під час виконання лабораторної роботи вивчив canvas, та опанував базові навички створення функцій та класів в мові JS.

Стикнувся з такою проблемою - реалізувати функцію для побудови трикутника за координатами основи та висотою. Для її реалізації використав знання з аналітичної геометрії – для знаходження координат перпендикулярного вектора.