МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

ПРО ЛАБОРАТОРНУ РОБОТУ № 2

ТЕМА: «РОБОТА З ПРИСТРОЯМИ ВВЕДЕННЯ ТА ОРГАНІЗАЦІЯВЗАЄМОДІЇ З КОРИСТУВАЧЕМ»

Виконав:

студент групи ІМ-11

Царик М.М.

Перевірив:

доц.каф.ІПІ

Родіонов.П.Ю.

Київ 2024

Мета: отримати практичні навички щодо створення та взаємодії з об’єктами за допомогою пристроїв введення.

1. Створити комп’ютерну програму, що містить обробник подій, який створює точки за натисканням на комп’ютерну мишу. Врахувати, що точки зсуваються від курсора миші, що є небажаним. Відповідно, потрібно створити обмежувальний прямокутник канвас в клієнтській області за допомогою event.target.getBoundingClientRect() і виправити позицію курсора, використовуючи ліву та верхню координати цього прямокутника.

2. Додати елемент управління «Button», яка очищає канвас. Створити меню вибору кольору, що буде використовуватися після очищення канвас. Додати меню вибору кольору, щоб встановити колір створюваних точок, що вимагає оновлення шейдерних програм.

3. Забезпечити роботу двох режимів рисування. Перший режим має рисувати точки, другий — будувати трикутник. Додати елемент управління «Button» для кожного з режимів. Під час візуалізації створити вершини в масиві індексів точок як точок і нарисувати вершини в масиві індексів трикутника як трикутники. Намагатися викликати gl.drawArrays якомога менше разів.

4. Додати кнопку для режиму рисування кола. Створити масив для індексів кола. При рисуванні в режимі кола програма має додавати точку при першому натисканні на комп’ютерну мишу, відповідно при другому натисканні додавати вершини для окружності, використовуючи положення, задане під час першого натискання на комп’ютерну мишу, щоб встановити радіус кола. Додати індекс центральної вершини до масиву індексів кіл та видалити цей індекс із масиву точкових індексів. Нарисувати кожне коло з використанням режиму gl.TRIANGLE_FAN для візуалізації.

5. Скласти звіт про виконану роботу.

Хід роботи:

index.html:

```
<!DOCTYPE html>
<html>
  <script type="text/javascript" src="libs/webgl-utils.js"></script>
  <script type="text/javascript" src="libs/MV.js"></script>
  <script type="module" src="main.js" module></script>
  <link rel="stylesheet" href="styles.css" />
  <body>
    <canvas id="gl-canvas" width="512" height="512"
      >> Oops ... your browser doesn't support the HTML5 canvas element
    </canvas>
    <div class="inputs">
      <button id="clearButton">Clear Canvas</button>
      <select id="colorSelect">
        <option value="1,1,1">White</option>
        <option value="0,0,0">Black</option>
        <option value="1,0,0">Red</option>
        <option value="0,1,0">Green</option>
        <option value="0,0,1">Blue</option>
      </select>
      <div class="labels">
        <label> <input type="radio" name="drawMode" value="points" checked />
Draw Points </label>
        <label> <input type="radio" name="drawMode" value="triangles" /> Draw
Triangles </label>
        <label> <input type="radio" name="drawMode" value="circles" /> Draw
Circles </label>
      </div>
    </div>
  </body>
</html>
```

styles.css:

```
canvas {
  display: block;
  margin: auto;
}

.inputs {
  display: flex;
  flex-direction: row;
  justify-content: center;
}

button {
  margin: 10px;
```

```
  padding: 10px 20px;
  background-color: #4caf50;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-size: 16px;
  transition: background-color 0.3s;
}

button:hover {
  background-color: #45a049;
}

select {
  margin: 10px;
  padding: 7px;
  font-size: 16px;
  border-radius: 5px;
}

/* Стили для опций выпадающего списка */
select option {
  padding: 7px;
}

.labels {
  border: 1px solid gray;
  border-radius: 10px;
  display: flex;
  align-items: center;
  margin-top: 5px;
}

/* Стили для текста метки */
label {
  padding: 7px;
}
```

main.js:

```
import { fragmentShader, vertexShader, createShaderProgram } from
"./shaders.js";

//Змінні для точки
let pointBuffer;
let pointColorBuffer;
let pointIndex = 0;
const pointVertexNumber = 1;
```

```
//Змінні для трикутника
let triangleBuffer;
let triangleColorBuffer;
let triangleIndex = 0;
let trianglePoints = [];
const triangleVertexNumber = 3;

//Змінні для круга
let circleBuffer;
let circleColorBuffer;
let circleIndex = 0;
let circlePoints = [];
const circleVertexNumber = 38;

//Змінні та налаштування програми
let canvas;
let gl;
let vPosition;
let vColor;
let color = vec4(0.9, 0.9, 0.9, 1.0);
let drawMode = "points";
const maxNumOfFigures = 15;

//Елементи інтерфейсу
const clearButton = document.getElementById("clearButton");
const colorSelect = document.getElementById("colorSelect");
const drawModeOptions = document.getElementsByName("drawMode");

//Ініціалізація програми
window.onload = function init() {
  canvas = document.getElementById("gl-canvas");
  gl = WebGLUtils.setupWebGL(canvas);

  if (!gl) {
    alert("WebGL is not available");
  }

  clearButton.addEventListener("click", clearCanvas);

  colorSelect.addEventListener("change", function (event) {
    color = vec4(event.target.value.split(",").map(parseFloat));
  });

  drawModeOptions.forEach((option) => {
    option.addEventListener("change", function (event) {
      drawMode = event.target.value;
    });
  });

  canvas.addEventListener("click", (event) => {
```

```
    // Перетворимо координати кліка в діапазон [-1, 1]
    const canvasRect = canvas.getBoundingClientRect();
    const x = event.clientX - canvasRect.left;
    const y = event.clientY - canvasRect.top;

    const mouseCoordinates = vec2(
      (2 * x) / canvas.width - 1,
      (2 * (canvas.height - y)) / canvas.height - 1
    );

    switch (drawMode) {
      case "points":
        if (isMaximumReached(pointIndex, maxNumOfFigures, 1)) return;

        gl.bindBuffer(gl.ARRAY_BUFFER, pointBuffer); // Прив'язуємо буфер
вершин точок
        gl.bufferSubData(gl.ARRAY_BUFFER, 8 * pointIndex,
flatten(mouseCoordinates)); // Записуємо координати нової точки у буфер
вершин

        gl.bindBuffer(gl.ARRAY_BUFFER, pointColorBuffer); // Прив'язуємо
буфер кольорів точок
        gl.bufferSubData(gl.ARRAY_BUFFER, 16 * pointIndex, flatten(color));
// Записуємо кольори для нової точки у буфер кольорів

        pointIndex += pointVertexNumber;
        break;
      case "triangles":
        if (isMaximumReached(triangleIndex, maxNumOfFigures, 3)) return;

        trianglePoints.push(mouseCoordinates);

        if (trianglePoints.length === 3) {
          gl.bindBuffer(gl.ARRAY_BUFFER, triangleBuffer);
          gl.bufferSubData(gl.ARRAY_BUFFER, 8 * triangleIndex,
flatten(trianglePoints));

          gl.bindBuffer(gl.ARRAY_BUFFER, triangleColorBuffer);
          for (let i = 0; i < trianglePoints.length; i++) {
            gl.bufferSubData(gl.ARRAY_BUFFER, 16 * (triangleIndex + i),
flatten(color));
          }

          triangleIndex += triangleVertexNumber;
          trianglePoints = [];
        }
        break;
      case "circles":
        if (isMaximumReached(circleIndex, maxNumOfFigures, 39)) return;

        if (circlePoints.length === 0) {
```

```
          circlePoints.push(mouseCoordinates);
        } else {
          const radius = distance(mouseCoordinates, circlePoints[0]);
          for (let i = 0; i <= 360; i += 10) {
            const angle = radians(i);
            const x = circlePoints[0][0] + radius * Math.cos(angle);
            const y = circlePoints[0][1] + radius * Math.sin(angle);
            circlePoints.push(vec2(x, y));
          }

          gl.bindBuffer(gl.ARRAY_BUFFER, circleBuffer);
          gl.bufferSubData(gl.ARRAY_BUFFER, 8 * circleIndex,
flatten(circlePoints));

          gl.bindBuffer(gl.ARRAY_BUFFER, circleColorBuffer);
          for (let i = 0; i < circlePoints.length; i++) {
            gl.bufferSubData(gl.ARRAY_BUFFER, 16 * (circleIndex + i),
flatten(color));
          }

          circleIndex += circleVertexNumber;
          circlePoints = [];
        }
        break;
    }
  });

  gl.viewport(0, 0, canvas.width, canvas.height);
  gl.clearColor(0.2, 0.2, 0.2, 1.0);

  const program = createShaderProgram(gl, vertexShader, fragmentShader);
  gl.useProgram(program);

  pointBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, pointBuffer);
  gl.bufferData(gl.ARRAY_BUFFER, 8 * 1 * maxNumOfFigures, gl.STATIC_DRAW);

  triangleBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, triangleBuffer);
  gl.bufferData(gl.ARRAY_BUFFER, 8 * 3 * maxNumOfFigures, gl.STATIC_DRAW);

  circleBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, circleBuffer);
  gl.bufferData(gl.ARRAY_BUFFER, 8 * 38 * maxNumOfFigures, gl.STATIC_DRAW);

  pointColorBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, pointColorBuffer);
  gl.bufferData(gl.ARRAY_BUFFER, 16 * pointVertexNumber * maxNumOfFigures,
gl.STATIC_DRAW);
  gl.bindBuffer(gl.ARRAY_BUFFER, null);
```

```
  triangleColorBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, triangleColorBuffer);
  gl.bufferData(gl.ARRAY_BUFFER, 16 * triangleVertexNumber * maxNumOfFigures,
gl.STATIC_DRAW);
  gl.bindBuffer(gl.ARRAY_BUFFER, null);

  circleColorBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, circleColorBuffer);
  gl.bufferData(gl.ARRAY_BUFFER, 16 * circleVertexNumber * maxNumOfFigures,
gl.STATIC_DRAW);
  gl.bindBuffer(gl.ARRAY_BUFFER, null);

  vPosition = gl.getAttribLocation(program, "vPosition");
  vColor = gl.getAttribLocation(program, "vColor");

  render();
};

function render() {
  gl.clear(gl.COLOR_BUFFER_BIT);
  drawFigures(pointBuffer, pointColorBuffer, pointIndex, pointVertexNumber,
"point");
  drawFigures(triangleBuffer, triangleColorBuffer, triangleIndex,
triangleVertexNumber, "triangle");
  drawFigures(circleBuffer, circleColorBuffer, circleIndex,
circleVertexNumber, "circle");
  window.requestAnimationFrame(render);
}

function clearCanvas() {
  gl.clear(gl.COLOR_BUFFER_BIT);
  pointIndex = 0;
  triangleIndex = 0;
  circleIndex = 0;
  trianglePoints = [];
  circlePoints = [];
}

function drawFigures(buffer, colorBuffer, index, vertexNumber, type) {
  gl.bindBuffer(gl.ARRAY_BUFFER, buffer); //зв'язуємо буфер з контекстом
WebGL
  gl.vertexAttribPointer(vPosition, 2, gl.FLOAT, false, 0, 0); //ставимо
вказівник атрибута вершини для позиції вершин
  gl.enableVertexAttribArray(vPosition); //Вмикаємо атрибут вершини

  gl.bindBuffer(gl.ARRAY_BUFFER, colorBuffer);
  gl.vertexAttribPointer(vColor, 4, gl.FLOAT, false, 0, 0);
  gl.enableVertexAttribArray(vColor);

  for (let i = 0; i < index; i += vertexNumber) {
    if (type === "point") {
```

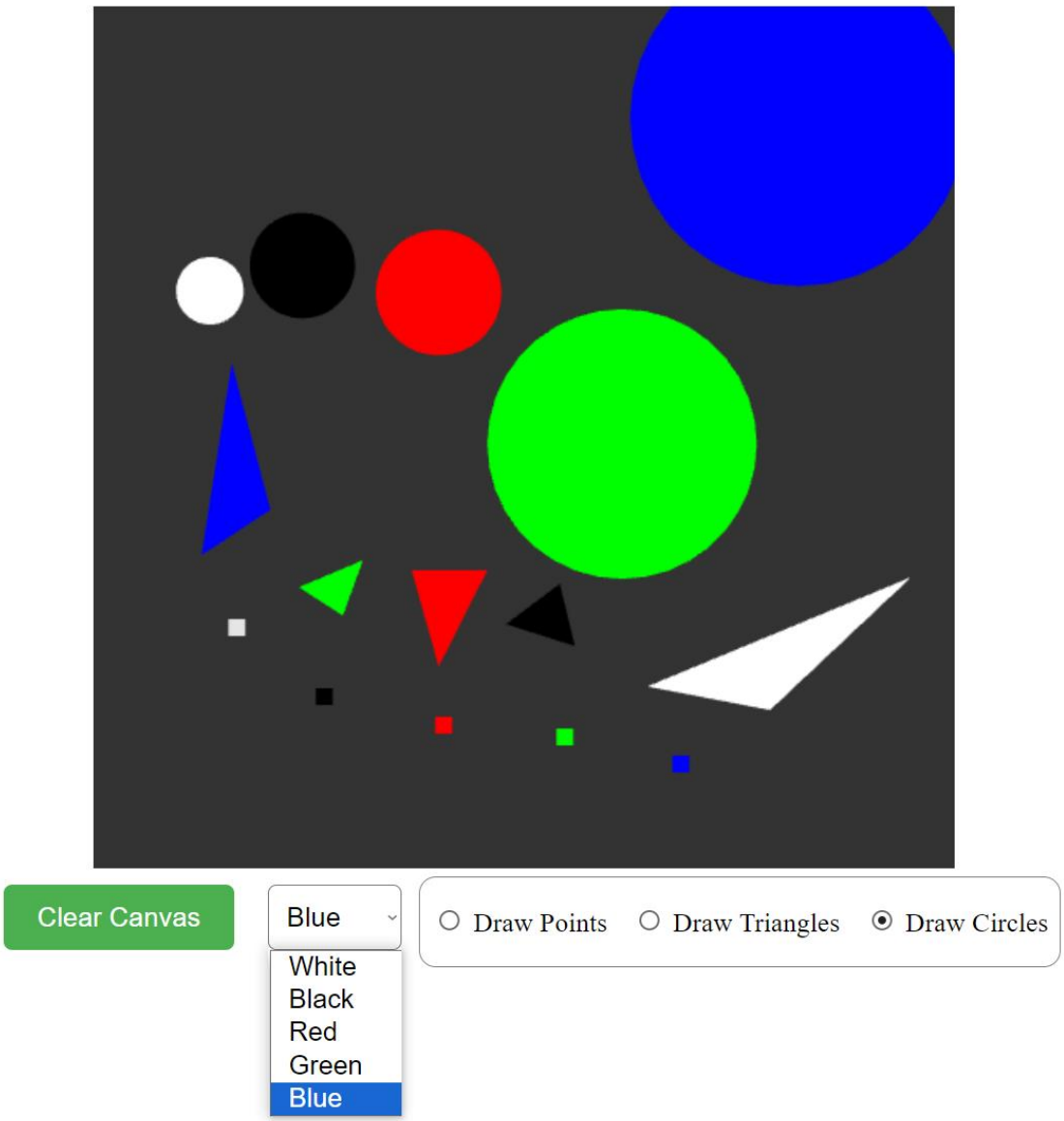
```
      gl.drawArrays(gl.POINTS, i, vertexNumber);
    } else if (type === "triangle") {
      gl.drawArrays(gl.TRIANGLES, i, vertexNumber);
    } else {
      gl.drawArrays(gl.TRIANGLE_FAN, i, vertexNumber);
    }
  }
}

function isMaximumReached(figureIndex, maxNumberOfFigures,
numberOfFigureVertex) {
  if (figureIndex / numberOfFigureVertex >= maxNumberOfFigures) {
    alert("The maximum number of figures of this type has been reached ");
    return true;
  } else {
    return false;
  }
}

function distance(point1, point2) {
  const dx = point2[0] - point1[0];
  const dy = point2[1] - point1[1];
  return Math.sqrt(dx * dx + dy * dy);
}
```

shader.js:

```
const vertexShader = `
attribute vec4 vPosition; // Вхід.атр., позиція вершини
attribute vec4 vColor;    // Вхід.атр., колір вершини

varying vec4 fColor;      // Змінна, що буде передана до фрагментного шейдера

void main() {
    gl_Position = vPosition; // Встановлюємо позицію вершини у просторі
    fColor = vColor;         // Передаємо колір вершини у фрагментний шейдер
    gl_PointSize = 10.0;     // Встановлюємо розмір точки для вершини
}
`;

const fragmentShader = `
precision mediump float;

varying vec4 fColor;  // Вхід. змінна, яка містить колір вершини

void main() {
    gl_FragColor = fColor; // Встановлюємо кінцевий колір пікселя
}
`;
```

```
function createShaderProgram(gl, VSHADER_SOURCE, FSHADER_SOURCE) {
  const vertexShader = loadShader(gl, gl.VERTEX_SHADER, VSHADER_SOURCE);
  const fragmentShader = loadShader(gl, gl.FRAGMENT_SHADER, FSHADER_SOURCE);

  const shaderProgram = gl.createProgram();
  gl.attachShader(shaderProgram, vertexShader);
  gl.attachShader(shaderProgram, fragmentShader);
  gl.linkProgram(shaderProgram);

  // Перевірка на успішність лінкування
  if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)) {
    console.error("An error occurred while initializing the shader program");
    return null;
  }

  return shaderProgram;
}

function loadShader(gl, type, source) {
  const shader = gl.createShader(type);
  gl.shaderSource(shader, source); //Вказуємо джерело
  gl.compileShader(shader); // Компілюємо

  // Перевірка на успішність компіляції
  if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)) {
    console.error("An error occurred while compiling a shader");
    gl.deleteShader(shader);
    return null;
  }

  return shader;
}

export { createShaderProgram, fragmentShader, vertexShader };
```

Результати виконання:



Висновки:

У результаті успішно виконаної лабораторної роботи було створено комп'ютерну програму з обробником подій, яка дозволяє створювати точки, трикутники та кола на канвасі з використанням різних режимів рисування. Реалізація відповідає вимогам завдання та забезпечує зручний інтерфейс користувача для взаємодії з програмою. З висновків можна виділлити:

1) Робота допомогла навчитися поєднувати webGL та JavaScript, комбінуючи роботу з шейдерами та додавання прослуховувачів на канвас та кнопки.

2) Виникли труднощі з буферами, коли для всіх кольорів використовувся один буфер і при зміні кольору всі попередні фігури міняли колір. Проблема вирішилася створенням окремих буферів під кожен тип елементу.
3) Була використана техніка сбору даних, де під час створення трикутнику программа очікує коли користувач зробить три кліки і тільки після цього малює фігуру, така ж техніка була використана для малювання кругу.

Список використаних джерел:

1. WebGL 2D Rotation by WebGLFundamentals: https://webglfundamentals.org/webgl/lessons/webgl-2d-rotation.html
2. ProgrammingTIL WebGL by David Parker: https://www.youtube.com/watch?v=vcCb2XEkZ-U&t=7s
3. WebGL Assignment Book using by The University of New Mexico https://www.youtube.com/playlist?list=PLjcVFFANLS5zH_PeKC6I8p0Pt1hzph_rt