****II. Векторная графика****

**Векторная графика** – это вид компьютерной графики, в котором используется математическое описание геометрических фигур (линий, кругов, прямоугольников, многоугольников и т.д.), в отличие от растровой графики, которая основана на пикселях. **Векторная графика** используется для создания изображений, которые можно масштабировать или преобразовывать без потери качества, так как они не основаны на фиксированной сетке пикселей.

# 

# ****2.1 Особенности векторной графики****

**Независимость от разрешения**: размер векторных изображений можно изменять, сохраняя их четкость и детализацию. Независимо от того, увеличиваются они или уменьшаются, математические уравнения, лежащие в основе векторной графики, будут пересчитаны, поэтому изображения «перерисовываются», потерь в качестве не происходит, что делает их идеальными для печати и для использования в различных устройствах с разным разрешением.

**1. Геометрические компоненты**: векторная графика строится из геометрических объектов, определяемых математическими формулами. Например, круг определяется радиусом и координатами его центра, линия – начальной и конечной точками, а многоугольник – координатами его углов.

**2. Небольшие размеры файлов**: поскольку векторная графика описывается математическими формулами, она занимает меньше места в физических файлах по сравнению с растровыми изображениями (которые состоят из сети пикселей). Это делает их эффективными для хранения и совместного использования.

**3. Редактируемость**: векторные объекты можно редактировать по отдельности, не затрагивая другие части изображения. Например, круг можно изменять, не затрагивая другие элементы графической сцены, что делает работу с векторной графикой более гибкой, чем с растровой.

**4. Сферы применения**: векторная графика используется для создания логотипов, иконок, иллюстраций, в типографии и технических чертежах. Еще одним направлением является **графический дизайн** – создание рекламных материалов, плакатов, флаеров и другой продукции графического дизайна. **Векторная графика** используется для создания анимации, потому что она позволяет легко перемещать и масштабировать графические элементы.

**Преимущества векторной графики**:

* **Масштабируемость**: без потери качества при изменении размера.
* **Редактируемость**: легко изменять и настраивать каждый элемент.
* **Небольшой размер файла**: более эффективное хранение и общий доступ.
* **Стабильное качество**: изображения не становятся пикселироваными независимо от их размера.

### ****Недостатки****:

* **Сложность**: Создание изображений с высокой детализацией может потребовать больше времени.
* **Ограничения в мелкой детализации**: векторная графика лучше подходит для ясных и четких форм, но ее может быть сложнее использовать для изображений с мелкими деталями или сложными световыми и текстурными эффектами.

# 

# 2.2 Формат SVG

Формат SVG – Scalable Vector Graphics используется для описания двухмерных изображений с помощью XML. SVG – это стандарт организации W3C, разработка которой началась в 1999 году. Этот формат позволяет определять изображения тремя методами: текстом, векторной графикой и растровыми изображениями (bitmap – файлы в формате BMP). Хотя существуют специализированные приложения для создания и редактирования SVG, для этой цели можно использовать любой текстовый редактор. Просмотр векторного изображения в формате SVG может быть осуществлен с помощью любого современного браузера.

В настоящее время SVG имеет несколько версий, чтобы лучше адаптироваться к различным требованиям к оборудованию. Таким образом, версии "SVG Tiny" и "SVG Basic" были созданы специально для мобильных устройств с ограниченными ресурсами. В то же время профиль «SVG Print» предназначен для печати документов.

Для анимации изображения SVG W3C рекомендует стандарт "SMIL". Помимо официальной рекомендации, существуют и другие решения, такие как "ECMAScript".

Элемент HTML <svg> является контейнером для векторной графики SVG.

На рисунке 2.1 указаны версии браузеров, которые полностью поддерживают элемент <svg>.

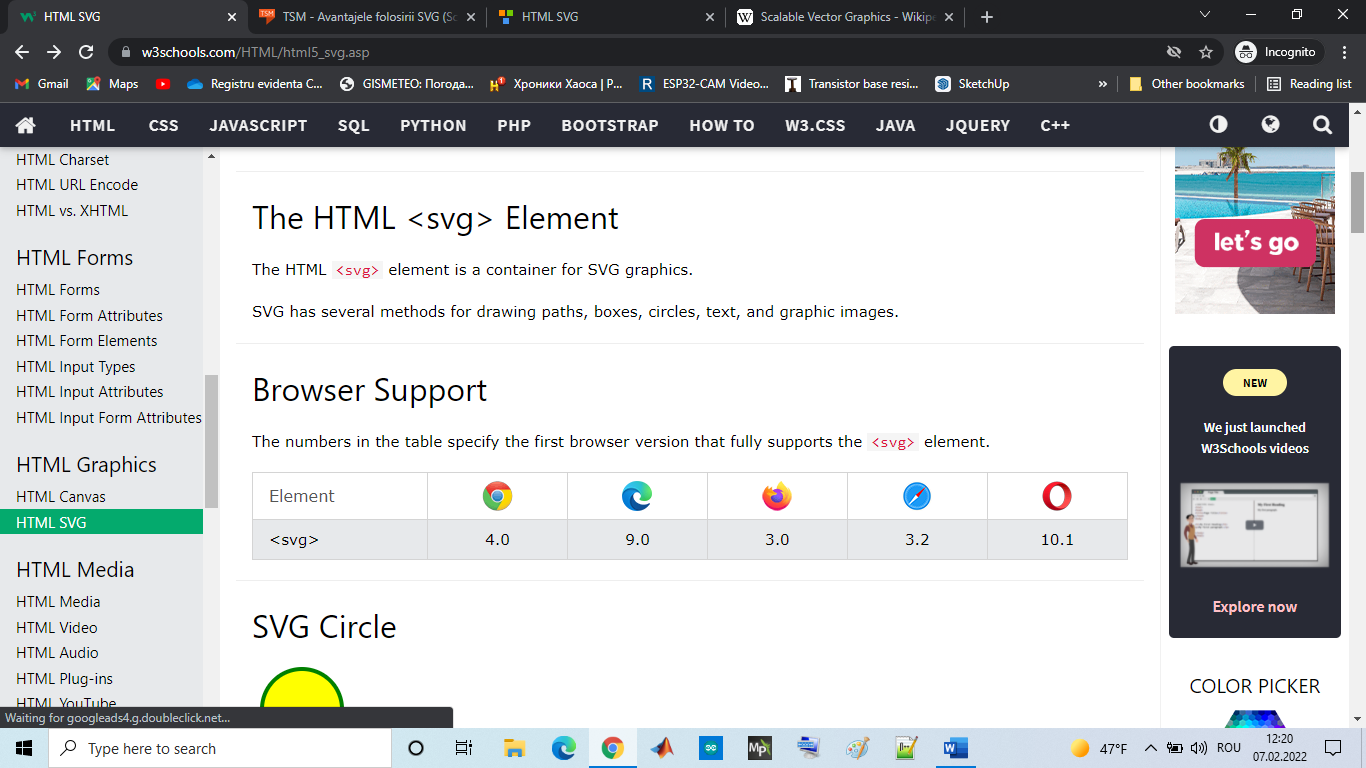


Рисунок 2.1. Версии браузеров, поддерживающие <svg>

# 2.3 Примитивы 2D графики в <svg>

Тег **<svg>** имеет несколько методов для рисования линий, многоугольников, прямоугольников, кругов, текста и графических изображений.

Формат SVG имеет несколько элементов для описания предопределенных форм, которые могут быть использованы разработчиком:

• Прямоугольник <rect>

• Круг <circle>

• Эллипс <ellipse>

• Линия <line>

• Полилиния <polyline>

• Полигон <polygon>

• Траектория <path>

Элемент <line> используется для рисования отрезка прямой:

Атрибуты:

x1 – абсцисса x начальной точки отрезка прямой;

y1 – ординат y начальной точки отрезка прямой;

x2 – абсцисса x конечной точки отрезка прямой;

y2 – ордината y конечной точки отрезка прямой.

**Пример использования:**

**<line x1="0" y1="0" x2="200" y2="200" stroke="black" stroke-width="2" />**

Формат SVG предлагает широкий спектр свойств контура:

**stroke** – цвет линии, контура текста или рисунков;

**stroke-width** – толщина линии, контура текста или контура элемента.

Как использовать различные линии, показано в примере ниже:

<svg height="80" width="300">

<g fill="none">

<line stroke="red" stroke-width="1", x1="0" y1="10" x2="200" y2="10"/>

<line stroke="black" stroke-width="2", x1="0" y1="20" x2="200" y2="20"/>

<line stroke="blue" stroke-width="3", x1="0" y1="30" x2="200" y2="30"/>

</г>

</svg>

Результат выполнения этого кода показан на рисунке 2.2

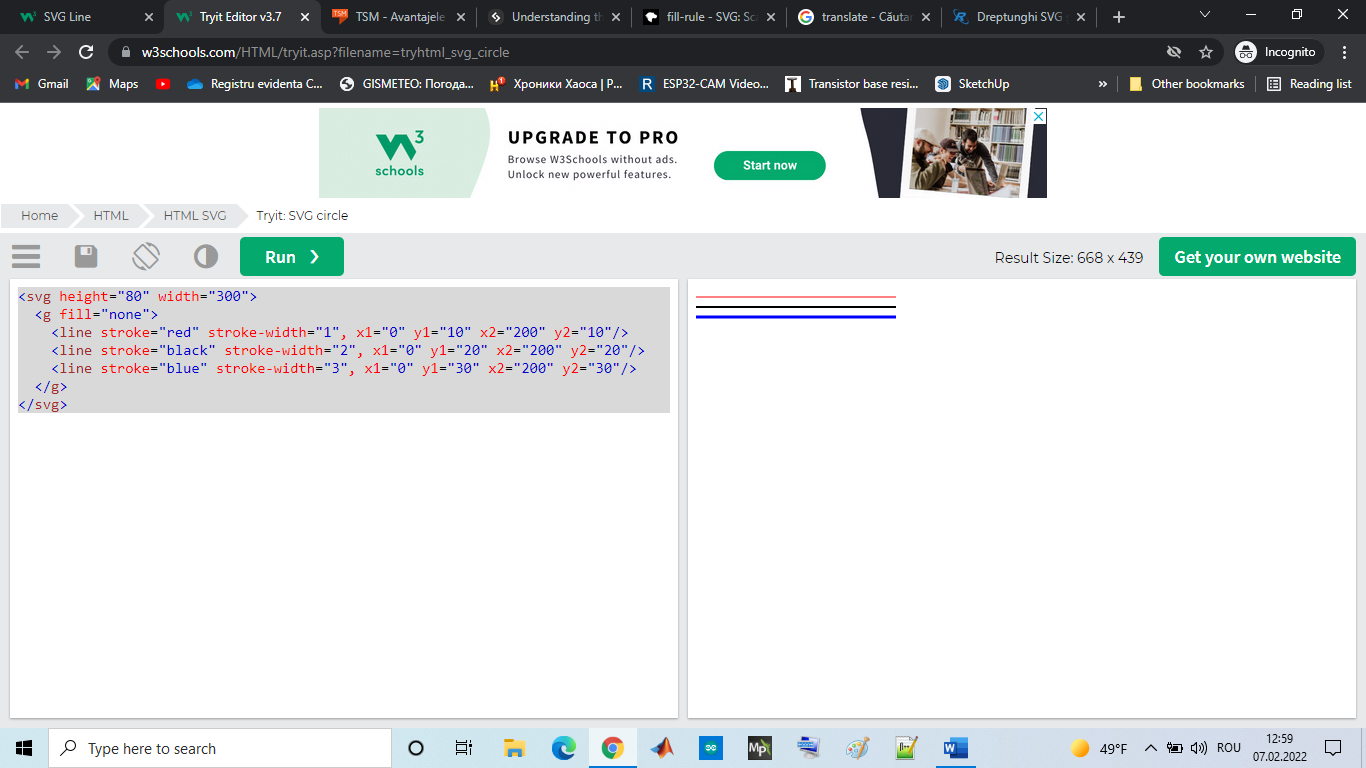


Рисунок 2.2. Линий разного цвета и толщины

**stroke-linecap** – определяет различные типы окончания линий. Существует три основных варианта этого атрибута:

* **butt**: конец линии прямой и заканчивается точно по указанной координате, по умолчанию;
* **round**: конец линии закруглен и удлинен на половину ширины линии;
* **square**: Конец линии прямой, но выходит за пределы координаты, добавляя длину, равную половине толщины линии.

Пример использования различных атрибутов параметра stroke-linecap, результат выполнения этого кода показан на рисунке 2.3:

**Пример:**

<svg height="80" width="300">

<g fill="none" stroke="black" stroke-width="10">

<line stroke-linecap="butt", x1="10" y1="20" x2="200" y2="20"/>

<line stroke-linecap="круглый", x1="10" y1="40" x2="200" y2="40" />

<line stroke-linecap="square", x1="10" y1="60" x2="200" y2="60" />

</г>

</svg>

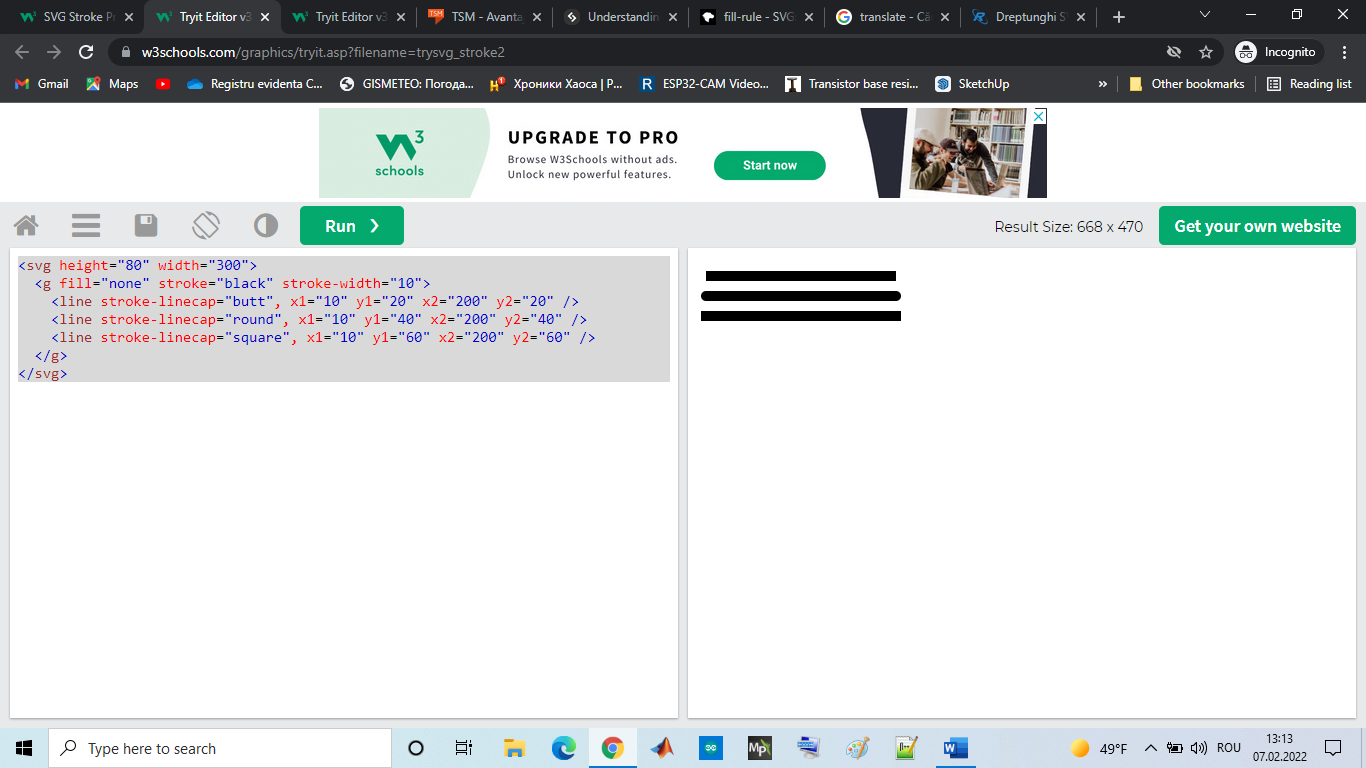


Рисунок 2.3. Типы атрибута stroke-linecap

**Атрибут stroke-dasharray** – используется для создания пунктирных линий, значение состоит из списка цифр, определяющих длину тире и пробелов. Например, **stroke-dasharray="5.5"** создаст линию с отрезками из 5 единиц, за которыми следуют пробелы из 5 единиц.

**Пример:**

<svg height="80" width="300">

<g fill="none" stroke="black" stroke-width="4">

<line stroke-dasharray="5,5", x1="10" y1="20" x2="200" y2="20" />

<line stroke-dasharray="10,10", x1="10" y1="40" x2="200" y2="40" />

<line stroke-dasharray="20,10,5,5,10", x1="10" y1="60" x2="200" y2="60" />

</г>

</svg>

Результат выполнения этой последовательности кода показан на рисунке 2.4.

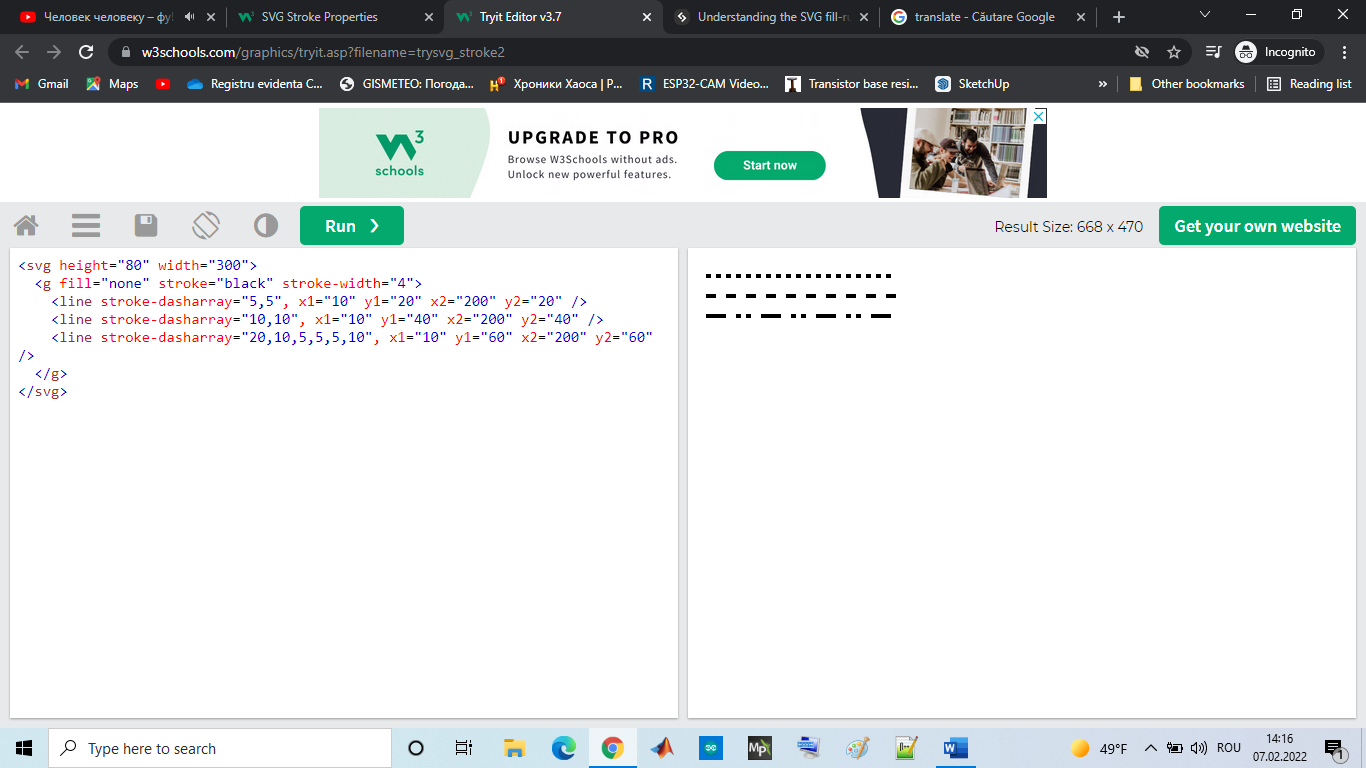


Рисунок 2.4. Виды пунктирных линий

Элемент <**polyline**> используется для создания ломаных линий, построенных из прямых отрезков, соединенных несколькими точками.

Атрибут:

**points** – (список координат *x, y*) содержит список пар координат (*x, y*), необходимых для рисования ломаной линии.

**Пример:**

<svg height="200" width="500">

<полилинейные точки="20.20 40.25 60.40 80.120 120.140 200.180" style="сын: нет; Штрих: черный; ширина штриха:3" />

</svg>

Результат выполнения этой последовательности программы показан на рисунке 2.5.

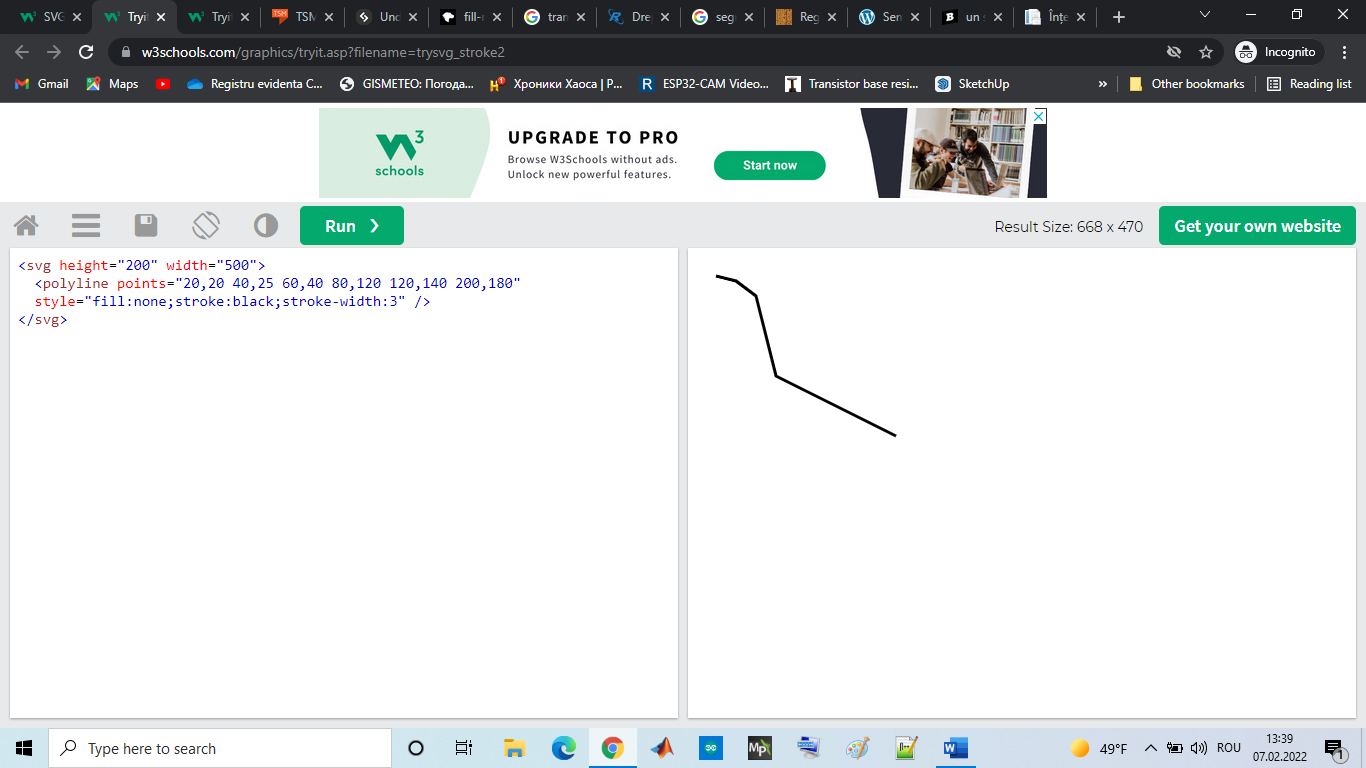


Рисунок 2.5. Ломаная линия

**Элемент** <circle> **–** определяет круг

Атрибуты:

cx – координата по оси x центра окружности;

cy – координата по оси y центра окружности;

r – радиус окружности.

**Пример:**

<svg width="100" height="100">

<circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />

</svg>

Результат выполнения этой последовательности программы показан на рисунке 2.6.

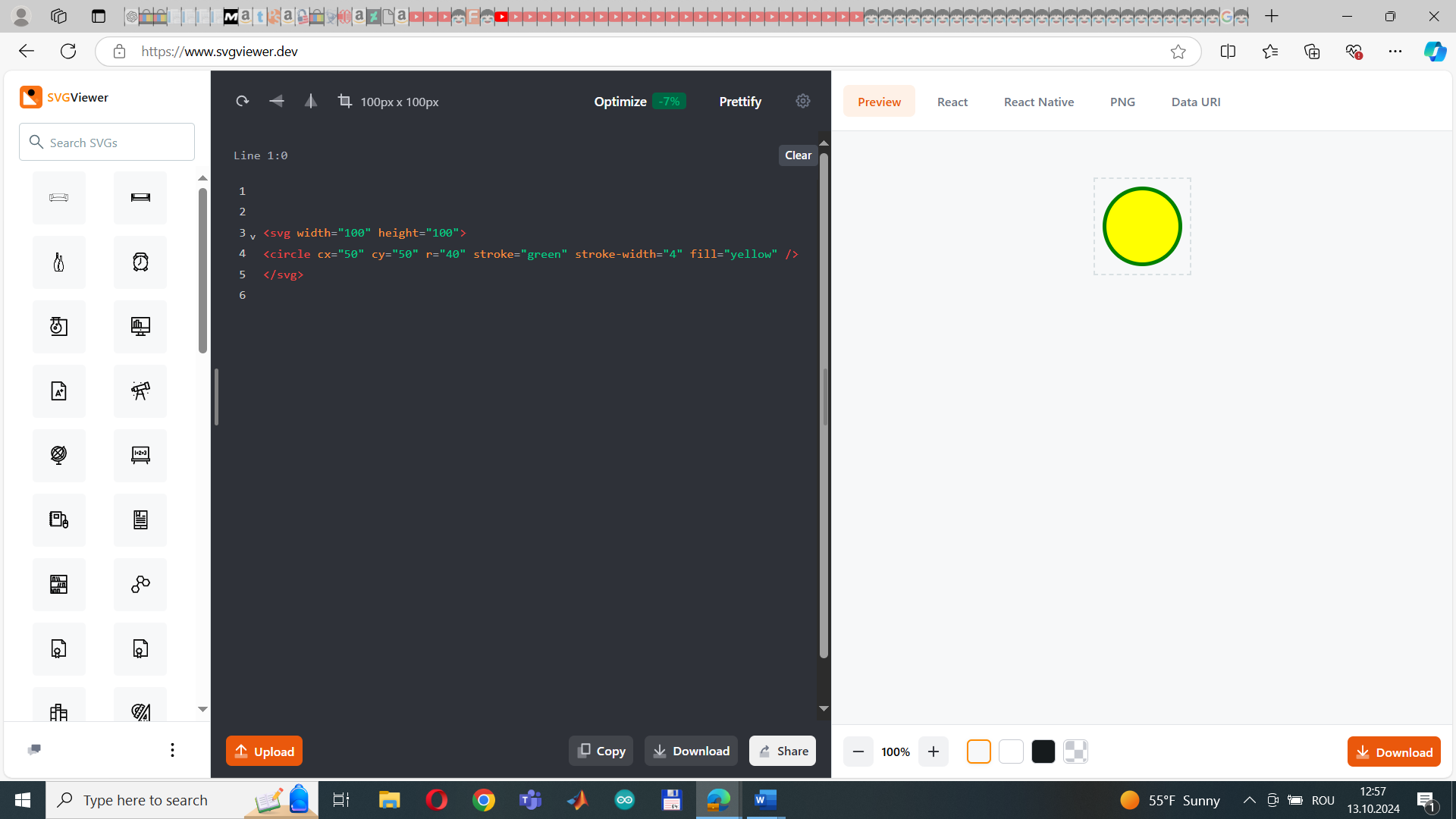


Рисунок 2.6. Вид круга в <svg>

**Элемент <ellipse>** – используется для рисования эллипса

Атрибуты:

cx – координата по оси x центра эллипса;

cy – координата по оси y центра окружности;

rx – длина радиуса эллипса по оси x (горизонтальный радиус – ширина);

ry – длина радиуса эллипса по оси Y (vertical radius – высота).

Эллипс является обобщенным случаем круга. Разница между этими фигурами заключается в том, что эллипс имеет разную длину радиусов по осям x и y, а в случае с окружностью они равны.

**Пример:**

<svg height="140" width="500">

<эллипсис cx="200" cy="80" rx="100" ry="50" tyle="fill:yellow; штрих:фиолетовый; ширина штриха: 2"/>

</svg>

Результат выполнения этой последовательности программы показан на рисунке 2.7.

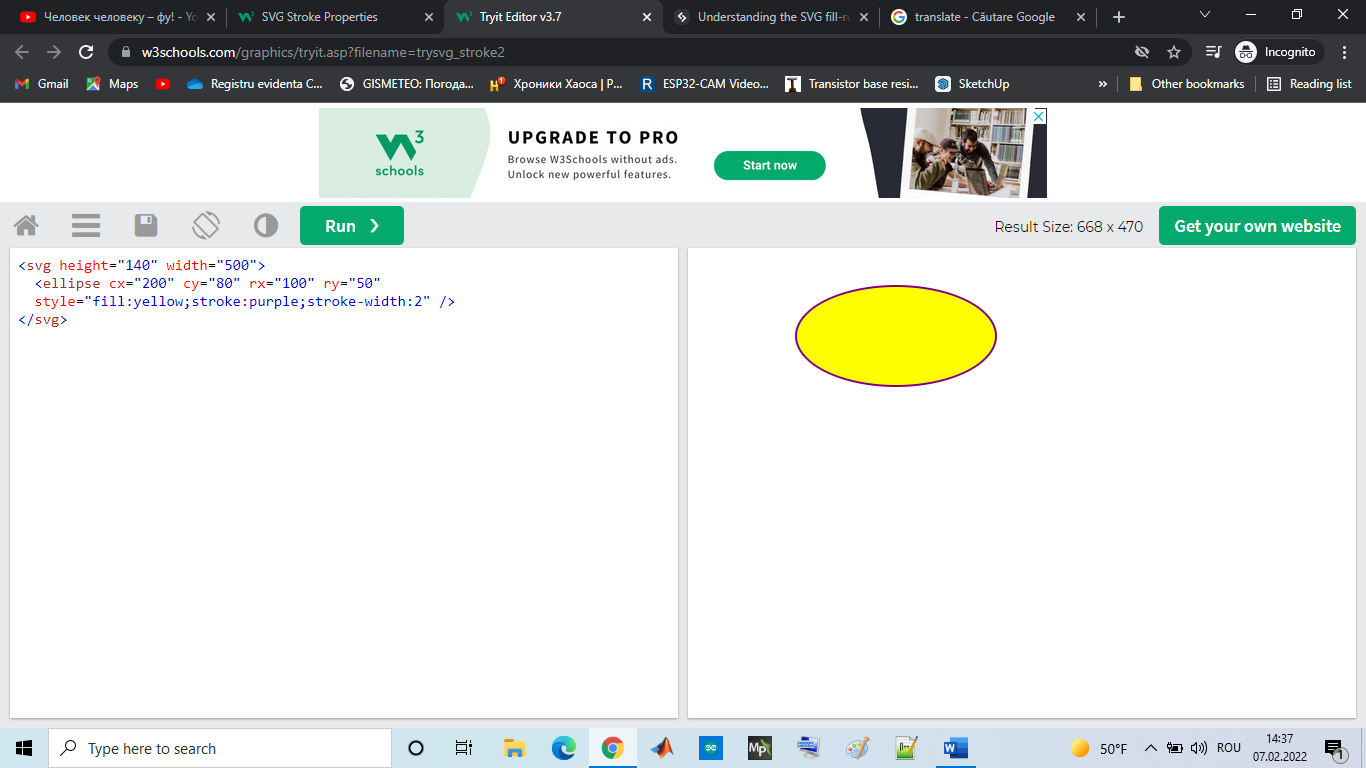


Рисунок 2.7. Эллипс

**Элемент <rect>** – позволяет создать прямоугольник и производные от него фигуры:

Атрибуты:

x – координата по оси x верхнего левого угла прямоугольника;

y – координата по оси y угла в левом верхнем углу прямоугольника;

rx – радиус по оси x (для округления вершин элемента);

ry – радиус по оси Y (для округления вершин элемента);

width – ширина прямоугольника;

height – высота прямоугольника.

**Пример:**

<svg width="400" height="180">

<rect x="50" y="20" rx="20" ry="20" width="150 height="150"

style="сын:красный; штрих: черный; ширина штриха:5; Непрозрачность:0,5"/>

</svg>

Результат выполнения этой последовательности программы показан на рисунке 2.8.

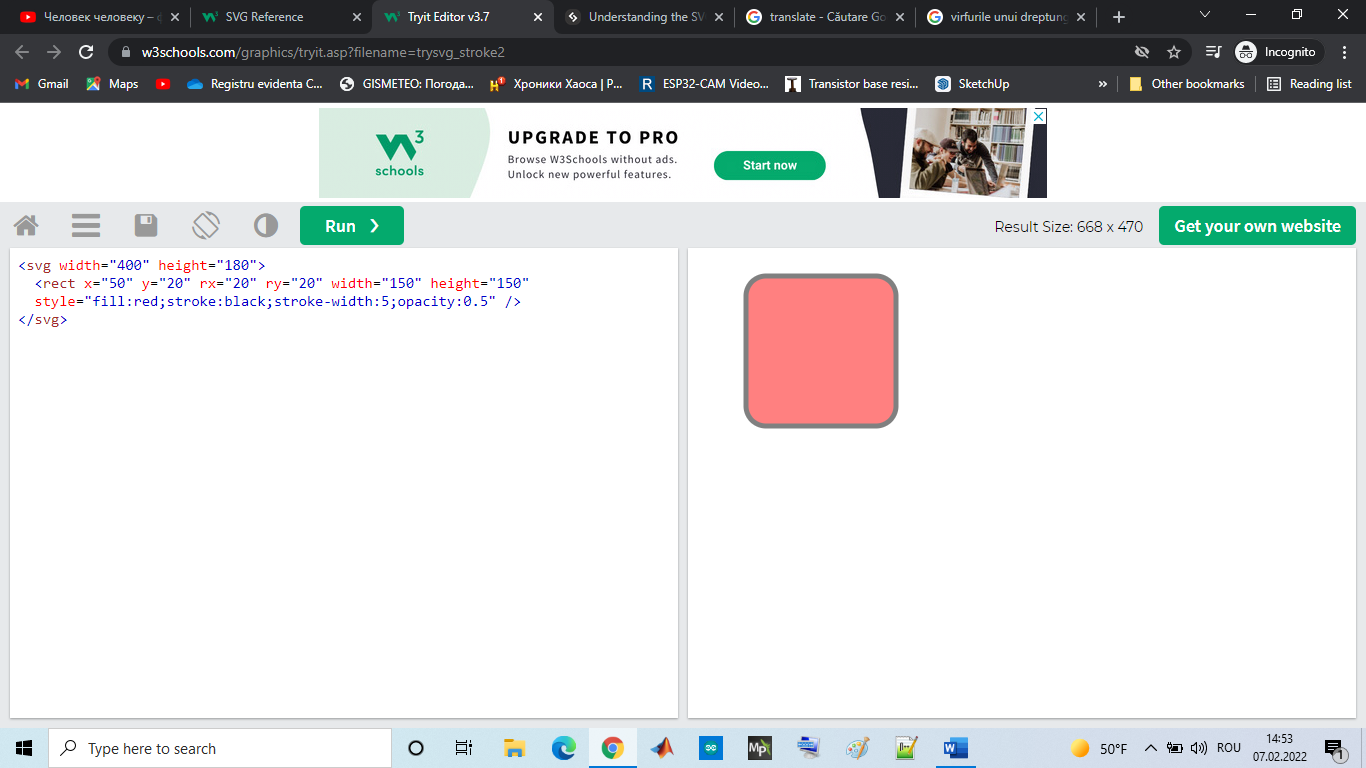


Рисунок 2.8. Прямоугольник

**Элемент <polygon>** – используется для создания геометрических фигур, которые должны содержать не менее 3 сторон. Многоугольники состоят из прямых линий, а фигура представляет собой замкнутый контур.

Атрибуты:

points – содержит координаты x и y каждой точки в наборе точек, составляющих вершины многоугольника; Общее количество точек должно быть четным.

fill-rule – представляет собой атрибут, который определяет алгоритм, используемый для представления внутренней части фигуры (метод заливки/раскрашивания) и может принимать значения evenodd или nonzero.

**Пример:**

<svg height="250" width="500">

<polygon points="220.10 300.210 170.250 123.234" style="fill:lime; штрих:фиолетовый; ширина штриха:1" />

</svg>

Результат выполнения этой последовательности программы показан на рисунке 2.9.

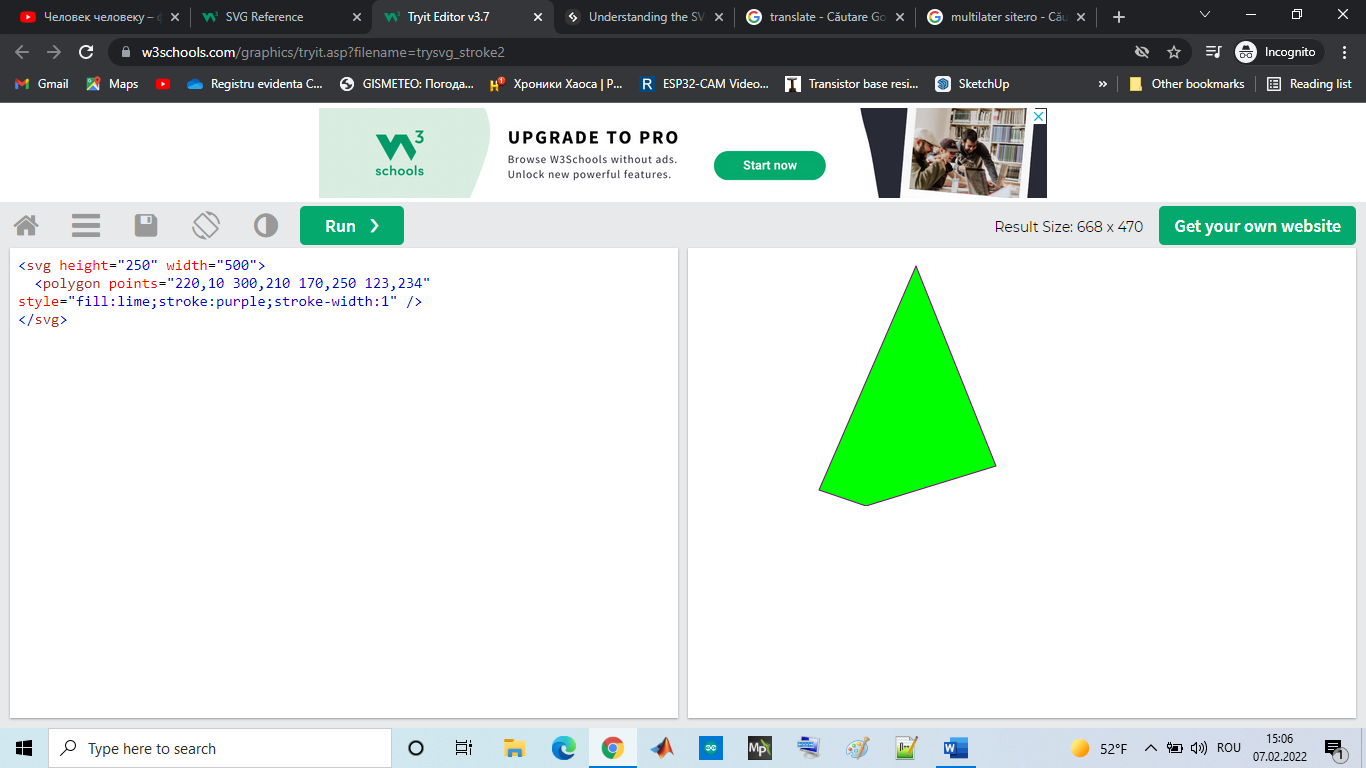


Рисунок 2.9. Четырехсторонний многоугольник

**Элемент <path>** – используется для определения траектории.

Атрибуты:

d – параметр path data – это набор команд, определяющий траекторию;

pathLength – если присутствует, то объект масштабируется таким образом, чтобы рассчитанная длина траектории, проходящего через точки, была равна этому значению;

transform – список преобразований.

Параметр path data может принимать значения:

M = moveto

L = lineto

H = horizontal lineto

V = vertical lineto

C = curveto

S = smooth curveto

Q = quadratic Bézier curve

T = smooth quadratic Bézier curveto

A = elliptical Arc

Z = closepath

Все вышеперечисленные команды также могут быть написаны строчными буквами. Использование заглавных букв означает абсолютноe позиционированный, использование строчных букв означает относительное позиционирование.

Кривые Безье используются для формирования плавных линий, которые можно масштабировать бесконечно без потери качества. Для представления кривой Безье пользователь устанавливает две точки, определяющие концы кривой и одну или две контрольные точки. Кривая Безье с контрольной точкой называется квадратичной кривой Безье, а кривая с двумя контрольными точками — кубической кривой Безье.

**Элемент <text>** – используется для определения текста.

Атрибуты:

x – список координат по оси x символов.

x – список координат по оси x символов. Элемент *n* в списке координат по оси x соответствует символу с порядковым номером *n* в тексте. Если есть дополнительные символы, которые не соответствуют значениям координат оси x, они помещаются после последнего символа. Значение по умолчанию – 0.

y – список координат по оси Y символов.

dx – список значений по оси x, с которыми символы перемещаются относительно абсолютного положения последнего нарисованного символа;

dy – список значений по оси y, с которыми символы перемещаются относительно абсолютного положения последнего нарисованного символа;

rotate – список углов поворота. Значение *n* в списке поворота применяется к символу *n* в отображаемом тексте. Символы дополнительные не вращаются.

# 2.4 Интеграция векторной графики в веб-редактор p5.js

Чтобы интегрировать SVG-код в язык HTML с помощью веб-редактора p5.js вам необходимо авторизоваться под своей учетной записью Google, Account или GitHub на соответствующей платформе. Для просмотра файлов в компоненте проекта необходимо нажать кнопку со стрелкой в левой части окна, как показано на рисунке 2.10. После выполнения данного действия открывается окно Sketch Files, в котором вы можете увидеть файлы компонентов проекта: index.html, sketch.js и style.css. Файл index.html содержит HTML-код для разметки интерфейса компонентов веб-страницы проекта. Добавление SVG-кода в проект осуществляется путем редактирования файла index.html, а именно замены тега par **<script src="sketch.js"></script> внутри тега <body></body>** на тег par **<svg></svg>** required, рисунок 2.10.

В приведенном ниже примере показано создание квадратичной кривой Безье, где точки **A** и **C** представляют вершины, а точка **B** представляет собой контрольную точку:

Визуализацию результата выполнения программы можно увидеть в правой части веб-редактора p5.js в окне предварительного просмотра, как показано на рисунке 2.10. Если удаленное изображение будет увеличено или уменьшено, вы можете заметить, что его качество не изменится.

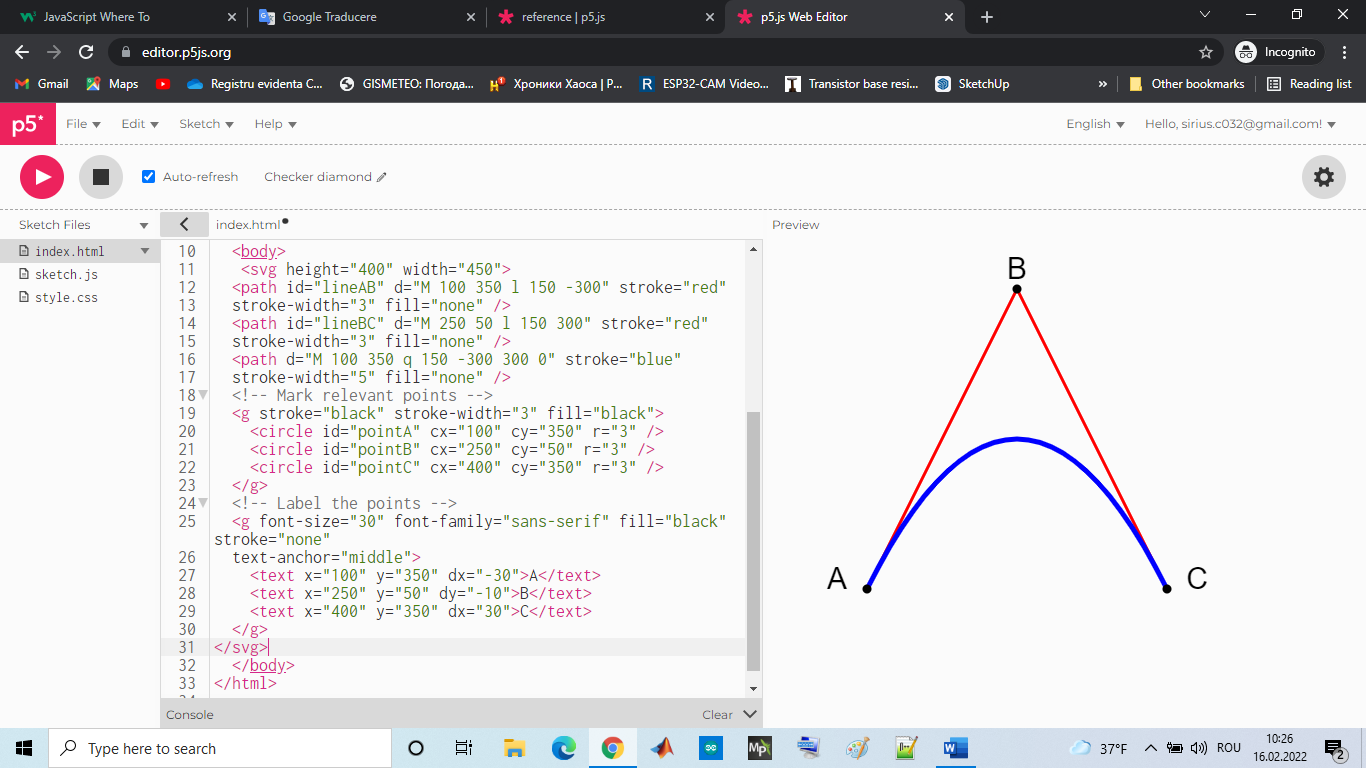


Рисунок 2.10. Рабочее окно редактора p5.js

# 2.5 Бесплатное программное обеспечение для векторной графики

**Inkscape** - это программное обеспечение для редактирования графики для Windows, Mac OS X и Linux, оно бесплатно и с открытым исходным кодом.

**Gravit Designer Online** - это бесплатное онлайн-приложение для векторного дизайна и редактирования для Mac, Win, Linux, Chrome OS, браузера. Он позволяет редактировать векторы в режиме онлайн прямо в браузере.

**SVG Viewer** – это программный инструмент, который позволяет просматривать, редактировать или манипулировать файлами SVG (масштабируемая векторная графика) в режиме онлайн. SVG Viewer доступен по адресу <https://www.svgviewer.dev>, это бесплатное программное обеспечение для редактирования графики, используемое для простого и интуитивно понятного создания векторной графики. Он прост и подходит как для веб, так и для настольных платформ. Это бесплатный векторный редактор, его можно запустить на персональном компьютере или браузере, он используется для создания векторов и других графических элементов. Это простая в использовании кроссплатформенность на Mac, Win, Linux, Chrome OS, браузере.

**Vectr** – это широко используемый инструмент для создания, редактирования изображений, а также рисования макетов и диаграмм. Программа позволяет создавать масштабируемые векторные дизайны с помощью простых и понятных функций. Он прост в освоении, идеально подходит для тех, кто только начинает знакомиться с векторной графикой и имеет достаточно базовых иллюстративных возможностей.

Программное обеспечение Vectr имеет большое количество интерактивных уроков. Vectr позволяет создавать и редактировать векторную графику с помощью инструментов находящихся на рабочей области. На Youtube есть много видеоуроков https: //youtu.be/CnSRzM91FYY.

Рабочее окно приложения Vectr показано на рисунке 2.11.

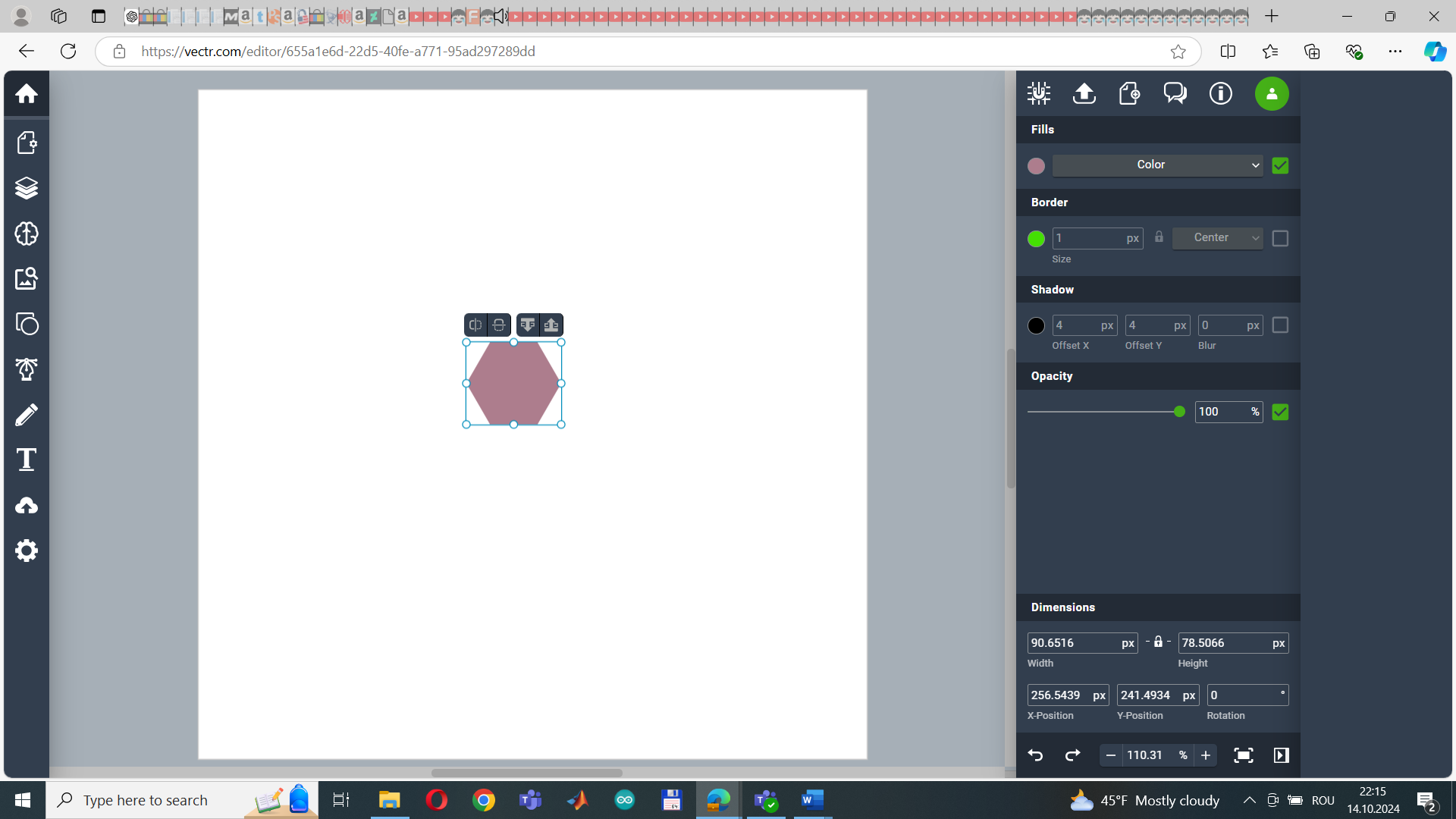


Рисунок 2.11. Рабочее окно редактора Vectr

Чтобы начать работу с онлайн-редактором Vectr, в Vectr загружается изображение. Разрешен импорт файлов в форматах EPS, AI, SVG, PNG или JPEG. Графический интерфейс пользователя удобно организован. Вкладки «Страницы» и «Слои» в верхнем левом углу дают контроль над процессом создания проекта. Панель фильтров расположена в правой части интерфейса. Приложение предлагает возможность изменить угол, добавить тени, границы или различные эффекты.

После завершения этапа проектирования программа предлагает возможность экспортировать чертежи в формате PNG, JPEG или SVG. SVG – единственный масштабируемый векторный формат, который можно открыть в других приложениях, в то время как растровые изображения PNG и JPEG больше подходят для Интернета. Приложение предоставляет полезную информацию и практические советы в разделе обучения. Разработчики редактора Vectr предоставили пользователям ряд инструкций и учебных пособий для решения самых популярных дизайнерских задач: создание иконок, логотипов, инфографики, меню, коллажей, графиков и многого другого.

### Лабораторная работа No2

### Тема: Генерация векторных изображений

**Цель работы:**

**1. Понимание понятия векторной графики**: ознакомление с основными принципами векторной графики и отличиями векторной графики от растровой.

**2. Использование примитивов графа SVG**: применение базовых примитивов графики SVG, таких как: <rect>, <circle>, <ellipse>, <line>, <polyline>, <polygon>, <path> и т.д.

**3. Использование атрибутов SVG**: понимание и применение соответствующих атрибутов для настройки внешнего вида графических элементов, таких как – fill, stroke, stroke-width, opacity, transform, и другие.

**4. Создание 2D векторных графических сцен:** проектирование и создание сложной 2D векторной сцены с использованием упомянутых примитивов с целью организации их в целостную сцену.

**5. Экспорт и просмотр изображений SVG**: понимание процесса экспорта изображений SVG и просмотр их в различных приложениях или браузерах для проверки точности и качества результатов.

**6. Получение практических навыков интеграции графики**: объединение нескольких форм и графики для построения целостной сцены, развитие навыков создания более сложных и привлекательных векторных изображений.

**Количество часов, необходимое для прохождения – 4 академических часа.**

**Цель работы:** получить практические знания по синтезу и генерации 2D векторных графических сцен с использованием простых графических примитивов в формате SVG. Изучение основных концепций, таких как определение геометрических форм и использование атрибутов для их персонализации.

***Задача работы:***

Создание 2D векторной графической сцены с использованием необходимых графических примитивов, таких как: *<rect>, <circle>, <ellipse>, <line>, <polyline>, <polygon>, <path>,* а также соответствующих атрибутов. Сцена должна содержать элемент *<текст>* размещенный в нижнем углу, в правой части экрана, который указывал бы на имя, фамилию и группу ученика. Разработать векторный вариант персонажа, нарисованного в лабораторной работе No 1. Варианты представлены в **таблице 1.3**.

Для воспроизведения сложных фигур или кривых допускается использование векторных графических редакторов.

**Пример:** Создание двухмерного векторного изображения, на котором изображена божья коровка.

<!DOCTYPE html>

<html lang="en">

<head>

<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.9.1/p5.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.9.1/addons/p5.sound.min.js"></script>

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

<meta charset="utf-8" />

<style>

svg {

**filter: drop-shadow(20px 10px 5px rgba(0,0,0,0.66));**

}

</style>

</head>

<body>

<svg width="288" height="321" viewBox="0 0 288 321" fill="none" >

<**linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">**

<stop offset="0%" style="stop-color:rgb(255,255,255);stop-opacity:1" />

<stop offset="100%" style="stop-color:rgb(201,16,16);stop-opacity:5" />

**</linearGradient>**

<path d="M235.806 221.399C199.733 279.997 132.807 303.604 86.4958 275.096C40.185 246.587 31.1209 176.2 67.1938 117.602C103.267 59.0032 170.193 35.3956 216.504 63.9043C262.815 92.413 271.879 162.8 235.806 221.399Z" fill="url(#grad1)" stroke="black" stroke-width="3"/>

<ellipse cx="158" cy="250.5" rx="18" ry="15.5" fill="black"/>

<ellipse cx="97.5" cy="213.5" rx="13.5" ry="11.5" fill="black"/>

<circle cx="213" cy="181" r="18" fill="black"/>

<circle cx="213" cy="181" r="18" fill="black"/>

<circle cx="134.5" cy="133.5" r="22.5" fill="black"/>

<circle cx="134.5" cy="133.5" r="22.5" fill="black"/>

<line x1="167.328" y1="280.303" x2="188.328" y2="320.303" stroke="black" stroke-width="3"/>

<line x1="44.6949" y1="203.329" x2="0.694881" y2="226.329" stroke="black" stroke-width="3"/>

<line x1="50.6949" y1="150.329" x2="6.69488" y2="173.329" stroke="black" stroke-width="3"/>

<line x1="84.2833" y1="93.473" x2="6.28327" y2="108.473" stroke="black" stroke-width="3"/>

<line x1="254.222" y1="179.131" x2="286.222" y2="224.131" stroke="black" stroke-width="3"/>

<line x1="223.328" y1="234.303" x2="244.328" y2="274.303" stroke="black" stroke-width="3"/>

<ellipse cx="212.5" cy="62.5" rx="54.5" ry="41.5" fill="#1E1E1E"/>

<ellipse cx="191" cy="50.5" rx="10" ry="11.5" fill="white"/>

<ellipse cx="228" cy="73.5" rx="10" ry="11.5" fill="white"/>

<ellipse cx="191.5" cy="50.5" rx="3.5" ry="5.5" fill="black"/>

<ellipse cx="227.5" cy="73.5" rx="3.5" ry="5.5" fill="black"/>

<ellipse cx="235.5" cy="26" rx="17.5" ry="13" fill="#1E1E1E"/>

<line x1="227" y1="16" x2="227" stroke="black" stroke-width="2"/>

<line x1="245.211" y1="25.3861" x2="259.211" y2="7.38606" stroke="black" stroke-width="2"/>

<line x1="91.5791" y1="274.73" x2="175.579" y2="143.73" stroke="#1E1E1E"/>

</svg>

</body>

</html>

Результат работы программы показан на рисунке 2.12, для раскрашивания персонажа был использован градиент, а для создания более сложного изображения используется затенение.

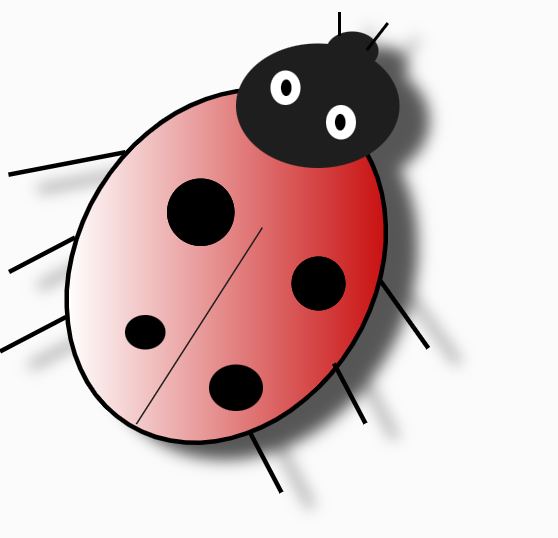
****

Рисунок 2.12. Результат выполнения программы

**Критерии оценки:**

**1. Правильность кода и реализации (30%):** Оценка правильности использования графических примитивов SVG, таких как <rect>, <circle>, <ellipse>, <line>, <polyline>, <polygon> и <path>. Проверяем, что все необходимые примитивы были использованы правильно, полностью и в соответствии с требованиями работы. Если графика правильно нарисована и скомпонована для формирования нужной сцены.

**2. Правильное использование атрибутов SVG (20%**): оценивается правильное применение атрибутов SVG, таких как заливка, обводка, ширина обводки, непрозрачность, трансформация и другие, для настройки графических элементов. Убедитесь, что стилистические изменения применены к каждой форме в соответствии с требованиями дизайна.

**3. Креативность и сложность композиции (15%)** – мера степени креативности и сложности создаваемого изображения, в том числе то, как примитивы сочетаются для создания интересной композиции. Оценивается степень оригинальности и сложности работы, т.е. то, как студент добавляет дополнительные детали для создания более интересной сцены.

**4. Эстетика и организация сцены (10%):** оценивается визуальный аспект сцены, включая расположение и композицию графических элементов. Визуальные детали соответствуют предназначению и демонстрируют понимание принципов рисования.

**5. Оценка знаний (20%):** пояснения о процессе выполнения работы, которые могут включать описание основных функций и используемой логики.

**6. Соблюдение срока сдачи (5%):** будет учитываться соблюдение установленного преподавателем срока сдачи работы.

**Вопросы для самооценки знаний:**

1. Что такое SVG и чем векторная графика отличается от растровой графики?

2. Перечислите графические примитивы, используемые в SVG, и объясните, для чего используется каждая из них.

3. Как определяется прямоугольник в SVG и какие основные атрибуты необходимы?

4. Объясните роль атрибутов fill, stroke и stroke-width. Как они влияют на внешний вид примитива?

5. Какое влияние атрибут opacity оказывает на элемент SVG?

6. Как можно изменить размеры примитива SVG, такого как круг или прямоугольник, с помощью доступных атрибутов?

7. В чем преимущества и недостатки использования векторной графики по сравнению с растровой?

8. Как можно использовать функции polyline и polygon для создания сложных фигур в SVG?