МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»
Кафедра кібербезпеки та математичного моделювання

КОМП'ЮТЕРНА ГРАФІКА

МЕТОДИЧНІ ВКАЗІВКИ

до виконання розрахунково-графічної роботи
для здобувачів
першого (бакалаврського) рівня вищої освіти
освітньо-професійної програми «Кібербезпека»
спеціальності 125 Кібербезпека та захист інформації

Обговорено і рекомендовано
на засіданні кафедри
кібербезпеки та математичного
моделювання
Протокол №2
від 13 лютого 2024 р.

Чернігів 2024

Комп’ютерна графіка. Методичні вказівки до виконання розрахунково-графічної роботи для здобувачів першого (бакалаврського) рівня вищої освіти освітньо-професійної програми «Кібербезпека» спеціальності 125 Кібербезпека та захист інформації. – Чернігів: НУ «Чернігівська політехніка», 2024 – 22 с.

Укладачі: ТКАЧ ЮЛІЯ МИКОЛАЇВНА, завідувач кафедри кібербезпеки та математичного моделювання, доктор педагогічних наук, професор
МЕХЕД ДМИТРО БОРИСОВИЧ, доцент кафедри кібербезпеки та математичного моделювання, кандидат педагогічних наук, доцент
КАЛЬЧЕНКО ДМИТРО ВОЛОДИМИРОВИЧ, викладач кафедри кібербезпеки та математичного моделювання, доктор філософії
НОРОХА ВЛАДИСЛАВ ОЛЕКСАНДРОВИЧ

Відповідальний за випуск – ТКАЧ ЮЛІЯ МИКОЛАЇВНА,
завідувач кафедри кібербезпеки та математичного моделювання, доктор педагогічних наук, професор

Рецензент – КОРНІЄНКО СВІТЛАНА ПЕТРІВНА,
доцент кафедри кібербезпеки та математичного моделювання, кандидат технічних наук, доцент

ЗМІСТ

ПЕРЕДМОВА

Мета дисципліни *"Комп'ютерна графіка "* - формування у студентів знань теоретичних основ комп'ютерної графіки як складової всіх сучасних комп'ютерних технологій для оволодіння алгоритмами та методами, які планується використовувати при створенні нових реальних систем машинної графіки, а також оволодіння практичними навичками розв'язування типових задач комп'ютерної графіки.

Основні завдання - теоретична та практична підготовка студентів з наступних питань:

- навчання геометричному (графічному) моделюванню об'єктів і формування у студентів певних знань, умінь і навичок маніпуляції комп'ютерними зображеннями цих об'єктів, в тому числі навчання розвитку у студентів алгоритмічного мислення, зокрема, формуванню умінь складати й реалізовувати графічні алгоритми побудови й опрацювання різноманітних зображень;
- вивчення теоретичних основ комп'ютерної графіки, в тому числі її математичного забезпечення для оволодіння існуючими графічними засобами;
- вироблення вміння та навичок для створення шляхів самостійної розробки програмного забезпечення;
- візуалізація інформації, тобто створення зображень різних об'єктів і сцен (у загальному випадку тривимірних) на деякому двовимірному екрані;
- виконання різних дій із зображеннями;
- зберігання та передавання графічної інформації;
- обробка зображень (підвищення контрасту, корекція кольорів, реставрація зображень).

Запропоновані завдання для індивідуальної (розрахунково-графічної) роботи студентів включають методичні вказівки до виконання, завдання для розрахунку, критерії оцінювання. За допомогою розрахунково-графічної роботи та запропонованих завдань досягається більш глибоке опанування теорії, що здійснюється за допомогою розвитку логічного мислення через вирішення задач та дає змогу студентам осмислити нові для них поняття.

Завдання для розрахунково-графічної роботи студентів можуть використовуватися як для аудиторної, так і домашньої роботи. Вони спрямовані на розвиток у студентів організаційних та аналітичних здібностей, а також уміння користуватися теоретичними посиланнями у вирішенні практичних ситуацій та вміння користуватися спеціальною літературою. Завдання для розрахунково-графічної роботи студентів можуть значною мірою полегшити вивчення дисципліни студентами очної форми навчання.

Під час виконання розрахунково-графічної роботи студенти повинні ознайомитися та вивчити лекційний матеріал, запропонований викладачем. Основою для вивчення є літературні джерела, наведені в даній методичній розробці. За наявності незрозумілих питань студентам рекомендується звернутись за консультаціями до викладача з метою отримання всіх необхідних пояснень щодо організації розрахунково-графічної роботи, виконання розрахункових завдань та пошуку додаткових літературних джерел. Викладачем надаються додаткові роз’яснення та індивідуальні консультації для підвищення компетентності студентів та розширення спектру їх знань з даної дисципліни.

КРИТЕРІЇ ОЦІНЮВАННЯ РОЗРАХУНКОВО-ГРАФІЧНОЇ РОБОТИ

Розрахунково-графічні завдання виконуються за окремим графіком. Студент самостійно готується до такого заняття за індивідуальним завданням. Обсяг розрахунково-графічної роботи визначається навчальним планом з дисципліни.

З даного курсу розрахунково-графічної робота проводиться у формі виконання індивідуальних завдань з розв'язування різноманітних задач.

Шкала оцінювання знань студентів при виконанні розрахунково-графічної роботи

Рівень виконання розрахункової роботи	**Кількість балів**	
- завдання розв'язані повністю і правильно, містять пояснення до розрахунків; - здійснено посилання на нормативну базу; - показано вміння самостійно формулювати висновки за результатами проведеного дослідження; - присутній творчий підхід та використано новітні інформаційні технології.	9…	10
- завдання виконані повністю, але при розв'язуванні допущені незначні помилки; - не аргументовано викладено матеріал; - у висновках містяться помилки та недоречності	6…	8
- завдання розв'язані, але містять грубі помилки; - завдання розв'язані не у повному обсязі та допущено значні помилки; - не сформульовані висновки за результатами розрахунків	3…	5
- завдання виконані частково і неякісно; - записані тільки формули	0…	2

У зв'язку з тим що, розрахунково-графічна робота містить завдання для розрахунку з різних тем, і може бути виконана після вивчення всіх тем курсу, оцінюється вона після закінчення третього модуля і оцінка за виконання розрахунково-графічної роботи, додається до підсумкової модульної оцінки, переведеної за шкалою ECTS.

ВИМОГИ ДО ОФОРМЛЕННЯ РОЗРАХУНКОВО-ГРАФІЧНОЇ РОБОТИ

Робота оформляється на листах А4 з однієї сторони, поля: з лівого боку – 20 мм, з правого боку – 10 мм, зверху – 20 мм, знизу – 20 мм. Завдання повинні бути виконані акуратно, розбірливим почерком (або надруковані), з детальними поясненнями та всіма проміжними розрахунками. В кінці розрахункового завдання пишеться висновок (відповідь).

Вимоги до комп'ютерного набору розрахункової роботи:

– гарнітура шрифту – Times New Roman;

– кегль шрифту (розмір) – 14;

– міжрядковий інтервал – полуторний;

– абзац – 1,25 см;

– розташування тексту роботи – вирівнювання по ширині;

– міжрядковий інтервал між заголовком (назвою розділу чи підрозділу) і текстом повинна дорівнювати 1 інтервалу.

Приклад оформлення титульної сторінки розрахунково-графічної роботи наведено у Додатку А.

Повністю оформлена і виконана розрахункова робота подається накафедру в термін, що визначений у плані-графіку виконання розрахункової роботи для перевірки її викладачем. Якщо робота виконана не вчасно без поважних причин, то студенту ставиться 0 балів («незадовільно») і він повинен виконати додатково один з варіантів, який вкаже викладач. Розрахункова робота оцінюється після особистої співбесіди з викладачем. В разі зауважень з боку викладача, робота повинна бути доопрацьована в зазначений термін і подана на перевірку. До підсумкового контролю допускаються лише студенти, що вчасно здали і захистили свою роботу.

ЗАВДАННЯ РОЗРАХУНКОВО-ГРАФІЧНОЇ РОБОТИ

Мета роботи

Вивчити прийоми алгоритмічного створення векторних зображень. Зокрема, вивчити логіку і техніку застосування наступних векторних графічних технологій:

а) Процедурне (командне) малювання;

б) Декларативне опис зображення у форматі SVG;

в) Створення векторних SVG-зображень в графічному векторному процесорі.

Теоретичні відомості

Процедурне (командне) малювання

Процедурним (командним, скриптовою) малюванням називає-ся така технологія, яка нагадує малювання олівцем (пером) по аркушу паперу.

У цій технології зображення з'являється внаслідок вико-нання команд, керуючих віртуальним «пером»: наприклад, уста-новка пера (без малювання) в потрібну точку листа; проведення ліній до іншої точки листа і т.п.

Оскільки при такому способі зображення створюється (в про-Стейшн випадку) з відрізків прямих («векторів»), то і такі изоб-ження стали називати «векторні». Кожен намальований таким чином відрізок-вектор відіграє роль примітиву створюваного изоб-ження - він є найпростішим його елементом.

Подібна схема малювання може бути легко поліпшена шляхом введення різних кольорів ліній, різної їх товщини і т.д. У реально існуючих векторних графічних процесорах (ДП) - про-грамах для комп'ютерного малювання - все це реалізовано найкращим чином.

Але в цих програмах команди відтворення примітивів в явному вигляді нам недоступні. Вони приховані в обробниках подій, метушні-кающих при інтерактивних діях користувача в графічному інтерфейсі програми.

Тому для експериментів з процедурних малюванням ми виберемо в якості робочого середовища те, що присутнє в кожному комп'ютері без жодних додаткових інсталяцій - графічні можливості будь-якого сучасного інтернет-браузера.

Що вміють сучасні інтернет-браузери

Розвиток сучасних інтернет-технологій йде складним шляхом, часто непослідовним і несподіваним. Сьогодні має місце тенденція все більше робіт, що виконуються людьми на комп’ютері, реалізовувати не локальнимм програмними додатками (типу Word або Total Commander), а через інтернет-сервіси (як пошта Google, онлайн-відео, інтернет радіостанції і т. д.); їх вже дуже багато і число їх збільшується з кожним днем.

У цих умовах «воротами» до цих сервісів виступає броузер - комунікаційна програма для «заглядання» в інтернет. Те, що буде розказано далі, відноситься до будь-якого браузеру: і Internet Explorer, і Firefox, і Chrome, і Safari. Ми не ставимо зараз мету докладно розповідати вам про архітектуру браузерів. Ми хочемо лише дати її короткий опис (рис. 1.1), з якого буде зрозуміло, яким чином браузер допоможе нам вирішити наші приватні проблеми з векторним малюванням.

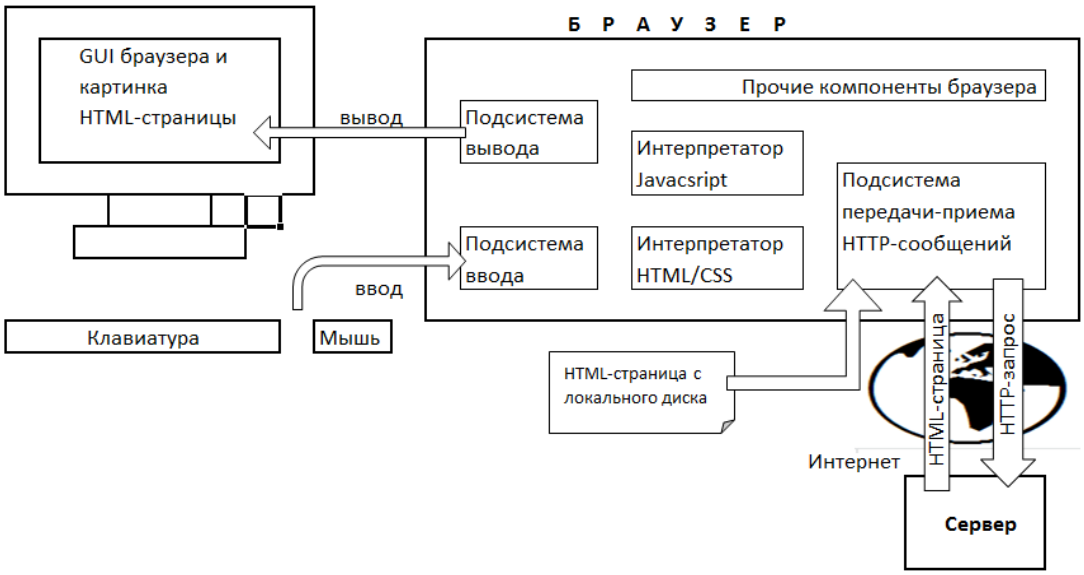

Рисунок 1.1 – Архітектура інтернет-браузера

Як видно з рис.1.1, в складі ПЗ браузера є, принаймні, два інтерпретатора. Якщо в коді сторінки (html-файл) існують команди малювання (тобто, фактично, програма) на мові javascript (JS), то їх відпрацьовує інтерпретатор цієї мови, відповідно до чого на площі оголошеної заздалегідь канви з'являється відмальована цими командами картинка. Цей варіант обговорюється в наступному розділі «Малювання в браузері по канві».

Другий варіант - використовувати в якості виконавця команд малювання інтерпретатор HTML / CSS. Він, зокрема, розуміє опису («декларації») малюнків у форматі SVG і здатний відтворювати їх на канві SVG-малюнка. В цьому випадку канву заздалегідь оголошувати не треба, вона формується в самому SVG-описі. Цей варіант ми теж обговорюємо нижче в підрозділі «Векторний графічний формат SVG».

Малювання в браузері по канві

«Канва» («полотно») - це спеціальний растровий об'єкт веб-сторінки, що володіє здатністю сприймати команди малювання і відображати результат їх виконання на своїй площі. Він формується тегом <canvas> </ canvas>.

Як працювати з канвою, спочатку покажемо на простому прикладі

```
1  <html>
2  <head>
3    <title>Контур</title>
4  </head>
5  <body>
6    <canvas id='i1'>Ваш браузер
7                    не работает с канвой.</canvas>
8    <script>
9      var example = document.getElementById("i1");
10     var ctx = example.getContext('2d');
11     example.height = 160;
12     example.width = 160;
13     example.style='border:1px solid'
14     ctx.beginPath(); //          *1
15     ctx.moveTo(40, 140);
16     ctx.lineTo(20, 40);
17     ctx.lineTo(60, 100);
18     ctx.stroke(); //             *2
19   </script>
20 </body>
21 </html>
```

Рисунок 1.2 – Скріншот вихідного тексту веб-сторінки.

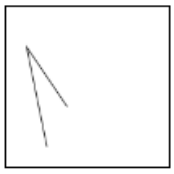

Рисунок 1.3 – Контур, намальований браузером по канві сторінки.

Пояснення до рис.1.2, що означають рядки тексту:

1, 21 - відкриває і закриває теги <html>, вони вико-ють роль контейнера всього вмісту сторінки.

2, 4 - теги <head>, контейнер заголовка веб-сторінки.

3 - тег <title>, задає текст в заголовку вкладки браузера (рис.1.2).

5,20 - тег <body>, це контейнер тіла веб-сторінки. Браузер показує на екрані те, що описано в тілі.

6,7 - тег <canvas>, оголошує об'єкт веб-сторінки класу «кан-ва» з ідентифікатором «i1». Якщо браузер підтримує цей тег, то в пам'яті браузера породжується такий об'єкт і відбувається перехід до обробки подальшого контенту сторінки. Але старі браузери «не розуміють» цього тега. Якщо браузер не розпізнає якийсь тег, він його просто пропускає без обробки, як би «робить вигляд», що цього тега немає зовсім. Тоді на екрані буде показаний текст «Ваш браузер не працює з канвою» як будь-який текст, надрукований в тілі веб-сторінки.

8, 19 - тег <script>, це контейнер скрипта, тобто програми, ко-торую виконуватиме вбудований в браузер інтерпретатор мови javascript. Скрипти можна писати на різних мовах, але по промовчу-нію (якщо не вказано інше) мається на увазі саме javascript.

9 - Методом getElementById змінної-посилання example при-привласнюється значення покажчика на об'єкт веб-сторінки (ім'я docu-ment означає за змістом «дана веб-сторінка»)), що має иденти-фікатор 'i1'.

10 - Методом getContext ('2d') змінної-посилання ctx присвоюють-ється значення покажчика на двовимірний графічний контекст об'єк-єкта, на який вказує посилання example. Далі все графічні дії будуть проводитися через посилання ctx. Згаданий кон-текст - це список графічних об'єктів, що розміщуються командами відтворення на канві.

11, 12 - канві встановлюється висота 160 пікселів і ширина 160 пікселів.

13 - у канви для зручності задається межа шириною 1 піксель, суцільна (solid). Якщо властивість style не ставити, канва залишиться там, де вона і була, але буде на веб-сторінці ніяк не помітні. З кордоном тут зручніше.

14 - Методом beginPath () створюється початок ланцюжка (групи) графічних дій на контексті ctx.

15 - Методом moveTo (40, 140) контексту ctx віртуальне перо встановлюється в точку з координатами x = 40 пікселів, y = 140 пік-селів. Координати відраховуються від верхнього лівого кута канви: x - зліва направо, y - зверху вниз.

16, 17 - викликами методу lineTo (x, y) «проводяться» лінії від поточного положення пера до точки (x, y). Слово «проводяться» напи-сано в лапках тому, що саме по собі виконання малюють ме-тодов lineTo () і подібних йому (буде розказано нижче) до появи видимих ліній, дуг, прямокутників і т.п. не приводить. Браузер просто запам'ятовує, що ці лінії в цих місцях повинні бути, не більше (лінії як би «намічаються», але не «малюються»). Щоб стало ясно, як лінії все-таки з'являються, читайте опис рядка 17.

18 - оператор ctx.stroke () виконають відразу два дії. Перше - він візуалізує всі лінії, намальовані в даній групі гра-чних дій (група була розпочата в рядку 13 методом begin-Path ()). Тобто «намічені» в рядках 16 і 17 лінії стають яв-но промальованим і видимими. Друге, що робить метод stroke () (і подібні до нього за логікою роботи інші методи, наприклад, fillRect ()) - він закриває групу графічних дій (рядки 14-18).

Ось, загалом, і вся премудрість. Попутно ви швидко познако-мілись з азами програмування на мові javascript. Більш основа-тельное знайомство з ним дає курс «Веб-технології», але він изуча-ється пізніше.

Синтаксис javascript дуже близький до синтаксису C / C ++. До при-міру, цикли пишуться точно так же. Є деякі концептуальні відмінності: немає строгого контролю типів, присутній динамічна типізація змінних, об'єктна орієнтованість в JS трактує-ся не так, як в Сі, ... Якщо цікаво, почитайте відмінний веб-ресурс

https://learn.javascript.ru/tutorial

Однак, повернемося до наших «ноукам», чи то пак до векторного малювання. Якими методами (функціями) можна користуватися для малювання по канві?

Функції малювання по канві

Швидкий старт в малюванні по канві можна знайти в багатьох місцях в інтернеті. Наприклад, чому б не скористатися цим:

http://htmlbook.ru/html5/canvas

Ми не ставимо перед собою мету перетворювати цю методичку в підручник з малювання на канві. Все одно інтернет-ресурси і пів-неї, і доступніше. Тут же обмежимося тим, що наведемо для справ-ки перелік методів контексту для відтворення зображень, взятий з:

https://ru.wikipedia.org/wiki/Canvas_%28HTML%29#.D0.A0.D0.B0.D0.B1.D0.BE.D1.82.D0.B0_.D1.81_context

Таблиця 1.1 – Методи відмальовування примітивів на канві

Метод (функция)	**Описание**	**Пример**
1	2	3
.moveTo(x, y)	Перемещает перо в указанное место	context.moveTo(20, 20)
.beginPath()	Начать строить фигуру (группу)	context.beginPath()
.closePath()	Замкуть контур фигуры	context.closePath()
.lineTo(x, y)	Связывает две точки между собой	context.lineTo(20, 20)
.rect(x, y, width, height)	Рисует прямоугольник	context.rect(20, 20, 300, 400)
.arc(x, y, radius, startAngle, endAngle [, anticlockwise])	Рисует окружностьи (углы в радианах)	context.arc(20, 20, 500, 0, 50)
.quadraticCurveTo(cpx, cpy, x, y) 	Создает квадратическую кривую Безье от текущей точки (в вызове не указывается), с участием одной контрольной точки, к конечной точке (всего участвуют 3 точки)	context.quadraticCurveTo(20, 20, 40, 50)
.ellipse(x, y, radiusX, radiusY, rotation, startAngle, endAngle [, anticlockwise])	Рисует эллипс	context.ellipse(20, 20, 40, 90, 9, 0, 50)
.bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)	Создает кубическую кривую Безье от текущей точки (в вызове не указывается), с	context.bezierCurveTo(20, 20, 90, 90, 50, 50)

	участием двух контрольных точек, к конечной точке (всего участвуют 4 точки)	
.arcTo(x1, y1, x2, y2, radiusX [, radiusY, rotation])	Рисует дуги по контрольным точкам	context.arcTo(20, 20, 20, 20, 20)
.fillStyle [= value]	Указывает стиль заливки контура фигуры	context.fillStyle = "black";
.strokeStyle [= value]	Указывает стиль контура фигуры	context.strokeStyle = "black";
.clearRect(x, y, width, height)	Очищает прямоугольник (делает белым)	context.clearRect(20, 20, 90, 80)
.fillRect()	Рисует закрашенный прямоугольник	context.fillRect()
.strokeRect()	Рисует контур прямоугольника	context.strokeRect()
.fill()	Закрасить все фигуры	context.fill();
.stroke()	Обвести все фигуры	context.stroke()
.drawImage(img, x, y, width, height [, scaleWidth, scaleHeight])	Нарисовать изображение	context.drawImage(img, 20, 20, 90, 90);

Векторний графічний формат SVG

Растрові картинки мають неприємну властивість «розмиватися» при збільшенні - вони втрачають різкість через те, що кожен піксель вихідного зображення перетворюється в групу пікселів збільшеної картинки (рис. 1.4).

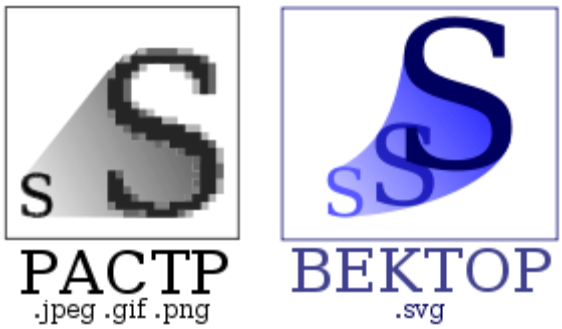

Рисунок 1.4 - Різна поведінка растрового і векторного об'єктів при збільшенні.

Векторні зображення позбавлені цього недоліку через принципово інший спосіб їх формування. Векторні примітиви в зображенні задаються наборами числових параметрів. При збільшенні, ці параметри масштабуються, а самі примітиви просто перемальовуються з нуля.

Бажання веб-дизайнерів мати на веб-сторінках картинки, що залишаються чіткими при будь-якому масштабі, привели до розробки веб-консорціумом W3C графічного формату SVG - Scalable Vector Graphics, «масштабована векторна графіки».

Зображення в форматі SVG створюється декларативно, тобто описом розмірів, форми і розміщення примітивів.

Опис зображення створюється в структурі вихідного тексту html-сторінки спеціальним XML-тегом <svg> ... </ svg>. Браузер читає html-файл і відображає в своєму вікні його контент. Але тут є нюанс - в браузері повинен бути в наявності інтерпретатор svg-описів, інакше тег <svg> ... </ svg> буде проігноровано. Нажаль, не всі браузери однаково добре працюють з svg-тегом. Найгірше становище - у Microsoft Internet Explorer, якому для показу svg-зображень потрібен додатковий зовнішній плагін Adobe SVG Viewer. Mozilla Firefox і Chrome мають вбудовані svg-переглядач, тому проблем з візуалізацією не виникає.

Наведемо опис простого зображення у форматі svg.

```
<svg width="200px" height="100px" >
<rect x="50" y="40" width="100px" height="50px" fill="magenta"/>
<circle cx="40px" cy="45px" r="30px" fill="rgba(0,230,75,0.4)"
stroke="rgba(200,0,0, 0.75)" stroke-width="5"/>
</svg>
```

Якщо цей текст зберегти в файлі svg-1.html і відкрити в Firefox, то без жодних додаткових хитрувань ви побачите результат (рис. 1.5)

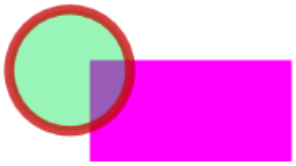

Рисунок 1.5 - SVG-зображення.

Про правила опису svg-зображень написано багато, і ми не ставимо перед собою мету переповідати тут все це. Наша ближня мета - лише ознайомитися з принципами декларативної векторної графіки і тільки її ми і будемо намагатися досягти. Дамо лише пару посилань на веб-ресурси, де ви зможете прочитати більше і докладніше:

https://developer.mozilla.org/ru/docs/Web/SVG

http://svg-art.ru/

Приклади побудови простих примітивів у форматі SVG

Проста лінія за допомогою тега line з усього двома параметрами - точками початку (x1 і x2) і кінця (y1 і y2) (рис. 1.6):

```
<Svg>
<Line x1 = "0" y1 = "0" x2 = "200" y2 = "200" stroke-width = "1" stroke = "rgb
(0,0,0)" />
</ Svg>
```

Також можна додати атрибути або стилі stroke і stroke-width, щоб задати колір і ширину:

```
style = "stroke-width: 1; stroke: rgb (0,0,0);"
```

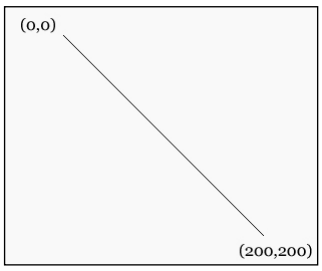

Рисунок 1.6 - Відрізок прямої.

Ламана лінія (рис.1.7). Синтаксис аналогічний попередньому, використовується тег `polyline`, атрибут `points` задає точки:

```
<Svg>
<Polyline points = "0,0 50,0 150,100 250,100 300,150" fill = "rgb (249,249,249)" stroke-width = "1" stroke = "rgb (0,0,0)" />
</ Svg>
```

Звернемо увагу на те, що координати точки перераховуються через кому, а точка від точки відділяється пропусками.

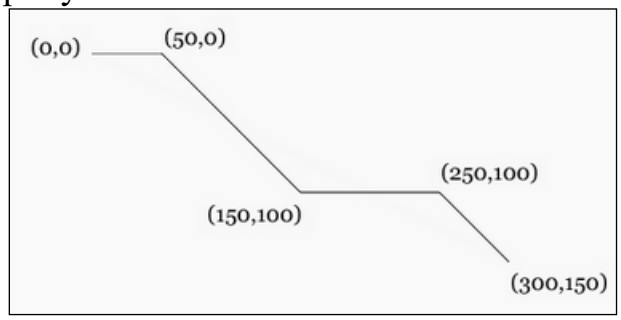

Рисунок 1.7 – Ламана лінія

Прямокутник (рис.1.8). Будується тегом rect, можна додати деякі атрибути:

```
<svg>
<rect width="200" height="200" fill="rgb(234,234,234)" stroke-width="1" stroke="rgb(0,0,0)"/>
</svg>
```

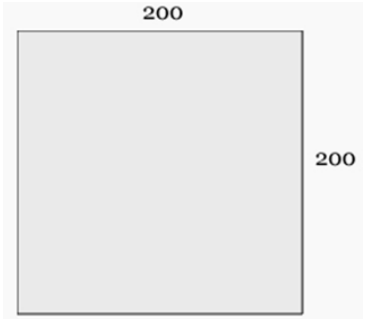

Рисунок 1.8 – Прямокутник

Ще тег rect вміє малювати прямокутник із закругленими кутами. Для цього треба просто серед параметрів цього тега вказати радіуси заокруглення rx і ry, докладніше читайте

http://xiper.net/learn/svg/svg-essentials/rectangles.

Окружність (рис.1.9). Викликається тегом circle, в прикладі з допомогою атрибута r задаємо радіус, cx і cy задають координати центру:

```
<svg>
<circle cx="102" cy="102" r="100" fill="rgb(234,234,234)" stroke-width="1"
stroke="rgb(0,0,0)"/>
</svg>
```

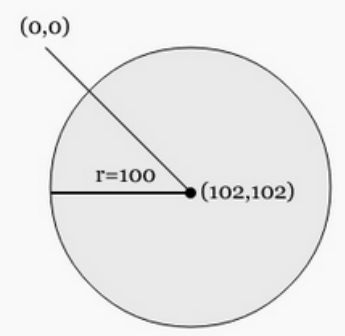

Рисунок 1.9 – Окружність

Еліпс (рис.1.10). Викликається тегом ellipse, працює аналогічно circle, але можна поставити два радіусу - rx і ry:

```
<svg>
<ellipse cx="100" cy="50" rx="100" ry="50" fill="rgb(234,234,234)" stroke-width="1"
stroke="rgb(0,0,0)"/>
</svg>
```

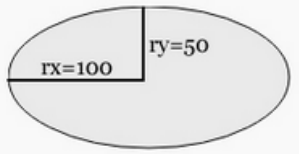

Рисунок 1.10 – Еліпс.

Багатокутник (рис.1.11). Викликається тегом polygon, багатокутник може мати різну кількість сторін:

```
<svg>
<polygon points="70.444,218.89 15.444,118.89 70.444,18.89 180.444,18.89 235.444,118.89
180.444,218.89" fill="rgb(234,234,234)" stroke-width="1" stroke="rgb(0,0,0)"/>
</svg>
```

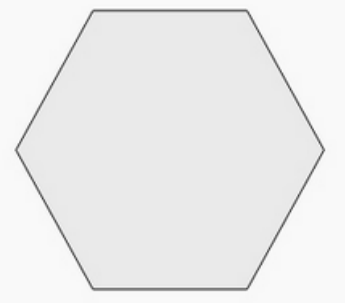

Рисунок 1.11 – Многоугольник.

Як видно, від полілінії полігон відрізняється тільки тим, що він замикається відрізком від останньої точки масиву `points` до його початкової точки.

Використання графічних редакторів

ДЛЯ СТВОРЕННЯ SVG-ЗОБРАЖЕНЬ

Для створення складних зображень в форматі SVG «ручна» технологія може бути застосована погано. Набагато простіше все виходить за допомогою векторного графічного процесора, здатного зберігати створений документ у форматі SVG. Серед таких можна по-рекомендувати або Adobe Illustrator, Inkscape, CorelDraw. Існують і інші, інформацію про них ви можете знайти самостійно.

Послідовність дій така:

а) В інтерактивному режимі, використовуючи графічний користувацький інтерфейс редактора, створюється потрібне зображення.

б) Командою «Файл - Зберегти як» зображення зберігається в файл типу svg.

Далі використання цього файлу може проводитися різними способами. Ми можемо, наприклад, просто перейменувати розширення «svg» на «html» і тоді, відкривши цей файл в браузері, ми можемо побачити створене зображення. Збільшуючи і зменшуючи веб-сторінку (в Firefox це Ctrl + і Ctrl-), можна переконатися в чіткості картинки при будь-якому масштабі.

У професійних випадках svg файл розміщується в області даних веб-сервера, а в html-коді сторінки робиться посилання на цей файл. Коли сервер буде передавати веб-сторінку браузеру, svg-код буде коректно підставлений в неї і користувач побачить це зображення. Цей варіант дій ми тут не розглядаємо.

Набір завдань для самостійного виконання

ЗАВДАННЯ 1. Командне(скриптове) малювання по канві

Командами малювання по канві веб-сторінки намалювати цифру номера варіанту і першу букву назви шрифту згідно номеру завдання, причому обриси цих букв повинні відповідати саме цьому шрифту.

Розмір канви 200 на 300 пікселів. Висота великої літери - приблизно 100 пікселів, інші розміри букв підібрати на око пропорційно їх обрису, заданому шрифтом. Заливка букв - на ваш розсуд, але не чорна. Те, що потрібно малювати, відзначено червоною рамкою на рис. 1.12.

Рисунок 1.12 - Варіанти завдання 1 - командне малювання по канві.

ЗАВДАННЯ 2. Декларативне малювання у форматі SVG

Варіанти svg-малюнків наведені на рис.1.13.

При виконанні малюнків суворого дотримання розмірів і пропорцій не потрібно, головне - передати ідею.

Створити в Notepad ++ вихідний текст малюнка в форматі svg згідно варіанта завдання. Зберегти його в файлі з розширенням html. Перевірити вірність побудов, відкривши файл в браузері (припустимо, Firefox).

Розмір канви 200 на 300 пікселів, малюнок займає майже всю канву (найменші поля близько 10% розміру канви).

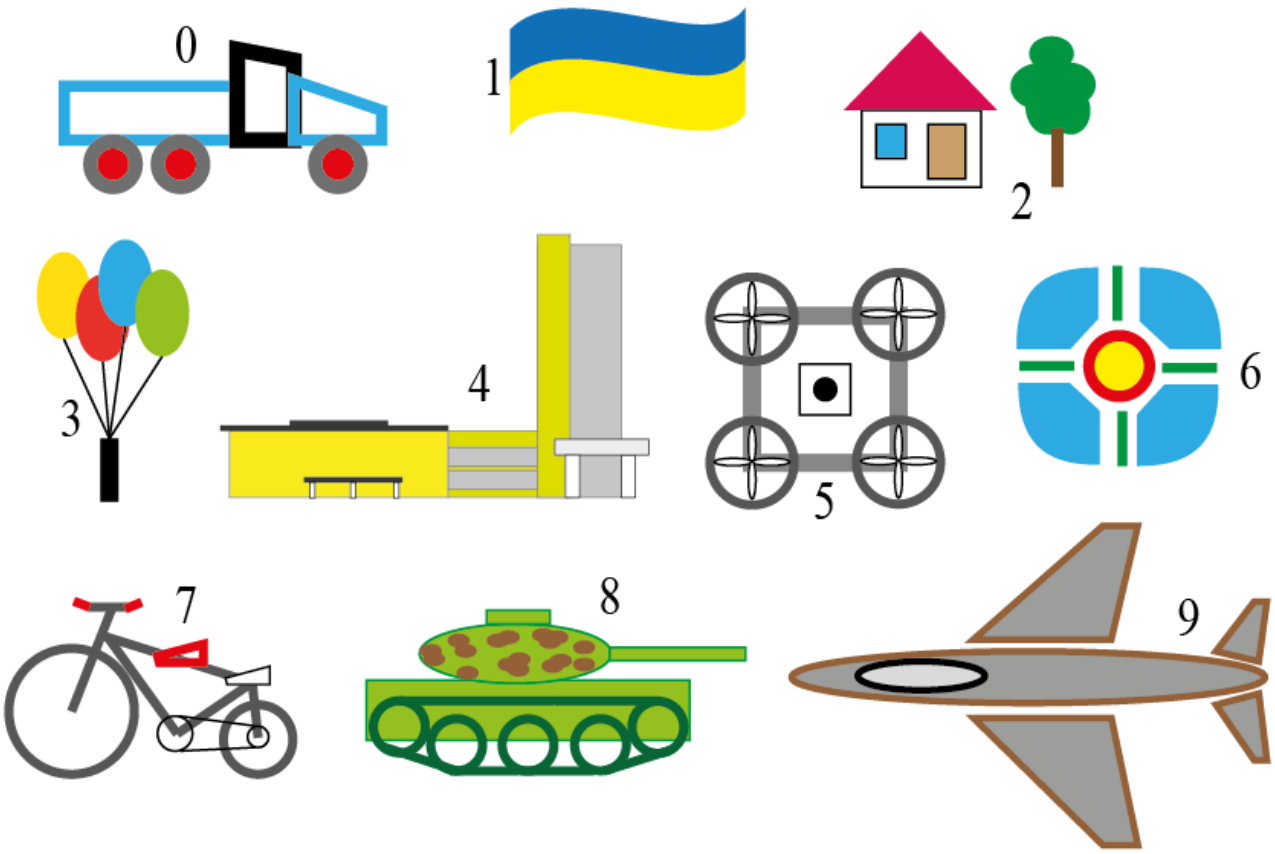

Рисунок 1.13 - Варіанти завдання 2 – декларативне малювання

ЗАВДАННЯ 3. СТВОРЕННЯ SVG-РИСУНКА з редактором

У векторному графічному редакторі якісно намалювати першу букву шрифту з завдання 1. Зберегти зображення в форматі svg. Відкрити svg-файл в Notepad++, вивчити, що в ньому міститься, бути готовим дати пояснення з усього, що там написано.

Варіант завдання обирається студентом по останній цифрі номера залікової книжки, або призначається викладачем.

В кінці виконання всіх завдань оформлюється звіт.

У звіт будуть включатися :

по завданню № 1:

а) Картинка з контурами букв і характерними точками на контурах;

б) Таблиця координат точок;

в) Початковий код веб-сторінки з командами малювання по канві;

г) Скріншот результату показу веб-сторінки в браузері.

по завданню №2:

а) Початкова картинка-завдання;

б) Ескіз з прикидками координат і розмірів;

в) Остаточний варіант svg-коду;

г) Скріншот остаточного зображення в браузері.

по завданню №3:

а) Початковий обрис букви кеглем 72 пункту;

б) Скріншот отриманого зображення (шматочок вікна браузера);

в) Код svg-файлу;

г) Письмове пояснення сенсу всіх літерних команд в атрибуті d тега path, які зустрічаються в вашому svg-файлі.

В кінці звіту слід написати висновки по всій роботі.

ДОДАТОК А

Титульна сторінка розрахунково-графічної роботи

Національний університет «Чернігівська політехніка»
Кафедра кібербезпеки та математичного моделювання

Розрахунково-графічна робота

з дисципліни „Комп'ютернаграфіка"

варіант № _____

виконав(ла)
студент(ка)

_____
(прізвище, ім'я, по-батькові)
перевірив

_____
оцінка_____балів

Підпис викладача _____

Чернігів 201_

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Базова

1. Алгоритми та методи комп’ютерної графіки. Тема 2. Векторне малювання. Методичні вказівки до самостійної роботи з комп’ютерної графіки для студентів напряму підготовки 6.050102 «Комп’ютерна інженерія»./ Укл. Нестеренко С.О. – Чернігів: ЧНТУ, 2015. - 22 с., рос. мовою (фактично дана розрахунково-графічна робота – це частковий переклад вищезазначених методичних рекомендацій із незначними змінами)

2. Блінова, Порєв. Комп'ютерна графіка - К .: "Юніор", 2004. - 456 с.

3. Фолі, ван Дем.Основи інтерактивної машинної графіки. В 2х книгах.- М.: Мир, 1985.

Ресурси ІНТЕРНЕТ

1. www.codeguru.com
2. www.graphicon.msu.su
3. http://fraktali.849pm.com/fracpc.html
4. http://www.codemanual.com/index_rus.shtml
5. http://cgw.pennnet.com/
6. http://www.sciencedirect.com
7. http://www.geom.umn.edu/software/cglist/