Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «**Дослідження засобів та технологій обробки векторної графіки**»
за освітньою програмою **12 Інженерія програмного забезпечення**
зі спеціальності: **121 Інженерія програмного забезпечення**

Виконав: студент групи «ПЗ2221»: /Сергій БАГНО/

Керівник: /доц. Вадим ГОРЯЧКІН/

Нормоконтролер: /Світлана ВОЛКОВА/

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань
Студент _____
(підпис)

Дніпро – 2024 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems

Department Computer information technology

Explanatory Note

to Master's Thesis
master

on the topic: «Study of means and technologies of processing vector graphics»

according to educational curriculum **12 software engineering**
in the Speciality: **121 software engineering**

Done by the student of the group PZ2221: / Serhii BAHNO /

Scientific Supervisor: /Vadim HORYACHKIN/

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2024

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем
Кафедра: Комп’ютерні інформаційні технології
Рівень вищої освіти: магістр
Освітня програма: Інженерія програмного забезпечення
Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
_____ січень 2024 р.

ЗАВДАННЯ

На кваліфікаційну роботу Магістр

студенту Багно Сергій Сергійович

1. Тема дипломної роботи: Дослідження засобів та технологій обробки векторної графіки.
   Керівник роботи: Горячкін Вадим Миколайович
   затверджені наказом 1196 ст від 05.12.2022 року
2. Строк подання студентом роботи 23.12.2023 року
3. Вихідні дані до дипломної роботи: пояснювальна записка, демонстраційна програма-графічний редактор.
4. Зміст пояснювальної записки (перелік питань до розробки):
   4.1. Аналітична частина: Вступ, Розділ 1;
   4.2. Основна частина: Розділ 2, Розділ 3, Розділ 4, Висновки, Список використаних джерел, Додатки;
5. Перелік демонстраційного матеріалу:
   5.1. презентація;
   5.2. демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Вступ	03.03.23 – 14.04.23	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	15.04.23 – 22.08.23	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	23.08.23 – 18.10.23	
4	Постановка задачі, технічне завдання	19.10.23 – 20.10.23	30%
5	Техніко-економічні показники	21.10.23 – 07.11.23	
6	Розробка інструментальних засобів дослідження	08.11.23 – 13.11.23	
7	Виконання досліджень	14.11.23 – 18.11.23	60%
8	Оформлення тез доповідей	19.11.23 – 23.11.23	
9	Оформлення статті у фаховий журнал	24.11.23 – 28.11.23	
10	Оформлення пояснювальної записки	29.11.23 – 05.12.23	
11	Розробка демонстраційних матеріалів	02.09.23 – 15.09.23	100%
12	Подання кваліфікаційної роботи до кафедри	23.12.23	
13	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	23.01.24	

Студент _____ /Сергій БАГНО/

Керівник роботи _____ /Вадим ГОРЯЧКІН/

ЗМІСТ

ВСТУП

Актуальність. В даний час в наше життя все більше проникає інформація, що вимагає спеціальних методів обробки. Багато галузей техніки, що стосуються отримання, обробки, зберігання та передачі інформації, що значною мірою орієнтуються в даний час на розвиток систем, в яких інформація має характер зображень. Зображення, яке можна розглядати як двовимірний сигнал є значно більш ємним носієм інформації, ніж звичайний одновимірний (тимчасовий) сигнал.

В останні роки суттєво зріс інтерес до цифрової обробки зображень, тому зовсім не випадково, що цифрова обробка та розпізнавання зображень є одним з напрямків, що інтенсивно розвиваються.

Цифрове зображення – графічна форма представлення даних, призначена для зорового сприйняття. Будучи закодованим за допомогою особливого алгоритму та записаним на носій, цей масив даних стає файлом.

У сучасному процесі поліграфічного виробництва всі ілюстрації та елементи оформлення представлені цифровими зображення різних типів.

У комп’ютерних системах, коли одержувачем інформації є людина, велике значення мають методи покращення зображень, що дозволяють підвищити помітність деталей, що цікавлять на зображенні. Крім того, при попередній обробці зображень, що виконується в автоматичних комп’ютерних системах, також важливу роль відіграє попередня обробка зображень, що дозволяє сформувати простір ознак об’єктів.

Огляд літератури показав слабку вивченість комп’ютерної графіки як виду особливого мистецтва. Серйозною проблемою в контексті сучасної реальності є недостатня увага до навчання комп’ютерної графіки, а така ж відсутність необхідних умов, коштів та кваліфікованих фахівців у даній області. Тому необхідно створити всі умови для реалізації навчання комп’ютерної графіки. Потреба у застосуванні інформаційних комп’ютерних технологій у всіх галузях сучасного суспільства підтверджує доцільність, своєчасність та актуальність даного дослідження.

Об’єктом дослідження є цифрові векторні зображення.

Предметом дослідження є методи та алгоритми обробки векторних зображень.

З огляду на це **метою роботи** є дослідження методів і алгоритмів цифрової обробки зображень, що забезпечують якісне та чітке відображення а також розробка рекомендації щодо застосування найоптимальніших алгоритмів.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

– провести огляд предметної області цифрової обробки зображень;

– проаналізувати існуючі способи опису цифрових зображень;

– проаналізувати існуючі методи та алгоритми цифрової обробки зображень;

– дослідити ефективність алгоритмів оцінки параметрів зображень;

– обґрунтувати вибір інструментальних засобів розробки програмного засобу;

– розробити програму для обробки векторної графіки;

– показати основні режими функціонування програмного засобу;

– організувати тестування та налагодження програмного засобу.

Використані методи: системно-структурний та порівняльний аналіз – використаний при аналізі проблем програмного забезпечення, призначеного для обробки векторної графіки; методи формально-логічного аналізу – при обґрунтуванні вибору методів розробки програмного засобу; економіко-статистичні методи – при дослідження тенденцій програмних засобів; системний підхід – заснований на представленні досліджуваного об’єкта в ясно організованій цілісності.

Основною гіпотезою, покладеною в основу даної роботи, є можливість обробки цифрових svg форматів для цілей масштабування до будь-якого розміру без втрати якості, в першу чергу, зменшення розміру файлу, використовуючи алгоритм стиснення та обробки.

Здебільшого цифрові векторні зображення мають використовуватися при розробці веб-сайтів, де найчастіше розробники в силу нестачі часу, забувають

про найголовніше – якісне відображення зображенні та використовують растрові зображення.

Ця робота присвячена аналізу існуючих моделей та алгоритмів, призначених для цифрової обробки зображень, зокрема векторні зображення svg. Використання нижчеописаних моделей та алгоритмів дозволить скоротити обсяг пам’яті, необхідної для зберігання зображень, а також забезпечити якісне відображення зображень на веб-сторінках.

РОЗДІЛ 1. ГРАФІКА, ЯК СУЧАСНИЙ МЕТОД ПЕРЕДАЧІ ІНФОРМАЦІЇ

1.1. Комп’ютерна графіка

Комп’ютерна графіка, це такий науково-технологічний напрямок, що займається завданнями щодо створення, обробки та зберігання зображень за допомогою комп’ютера та його апаратних та програмних можливостей. Зображення на ПК зберігаються у вигляді двійкового коду – координат, кольору, позначеного в будь-якій колірній моделі. Сьогодні розглянемо, які види комп’ютерної графіки існують.

2D є найбільш поширеним типом комп’ютерної графіки та анімації. Це технологія, що сягає корінням у минуле, але є нестаріючою класикою. Така графіка проста, візуально приваблива і легко сприймається споживачами.

Якщо ж говорити по-науковому, то 2D, або двомірна графіка (two dimentional), це плоске зображення, побудоване на основі двох осей – горизонтальної (x) і вертикальної (y). Двовимірна графіка – це проста картинка, яка виглядає плоскою, внаслідок того, що в ньому застосовуються тільки два виміри – ширина і висота. Незважаючи на подібний вигляд у ілюстрації можна досягти обсягу за допомогою світла та тіней, але не реалістичності, за винятком фотографій. 2D малюнки зазвичай використовують для створення логотипів, макетів веб-сайтів, рекламних банерів, інтерфейсів, мультиплікації та кінематографу.

Приклад 2D-графіки зображений на рисунку 1.1:

Рисунок 1.1 – Приклад 2D-графіки

2D-графіку можна розділити на растрову, векторну та фрактальну.

Векторний малюнок можна уявити у вигляді елементарних геометричних об’єктів: точки, прямі, криві, кола, багатокутники, тощо. Фігурам надаються будь-які якості, наприклад, товщина ліній, колір заливки. Для створення ілюстрацій використовуються формули та координати. Наприклад, щоб намалювати трикутник потрібно вказати його вершини, колір заповнення та обведення. Для складних малюнків використовують набір геометричних фігур, які збираються разом як аплікація з паперу на уроці праці в початковій школі, але при цьому зберігається можливість надалі редагувати картинку, що вийшла [1].

Даний вид графіки використовується для створення ілюстрацій, іконок, логотипів та технічних креслень, проте процес відтворення фотореалістичних зображень буде украй складним у такому форматі.

Перевагами векторної графіки вважаються:

- мінімальний обсяг займаної пам’яті на ПК;
- трансформація та масштабування без втрати якості;
- вигляд завжди однаково, незалежно від характеристик пристрою відображення.

Негативними сторонами векторів є:

- неможливість подання всіх зображень за допомогою примітивів;
- трудомісткий процес переведення растрових зображень у векторні;
- відсутність автоматичного введення;
- проблеми із сумісністю програм перегляду та створення [2-5].

Приклад векторної графіки зображений на рисунку 1.2:

Рисунок 1.2 – Приклад векторної графіки

Векторні картинки потрібні на підприємствах, котрі займаються проектуванням, конструкторських бюро, в рекламних агентствах, друкарнях, тощо. Графічні редактори, що працюють з цими ілюстраціями, є: Adobe Illustrator, Corel Draw, AutoCad, ArhiCad.

Растрові зображення являють собою щось схоже на картатий аркуш паперу, де одна клітина, це одна точка-піксель, а утворювані ними рядки і стовпці збираються в матрицю (растр). У кожного пікселя свій колір та місце, де він розташований. У комплексі всі пікселі утворюють зображення.

Приклад кривих та векторних картинок зображений на рисунку 1.2:

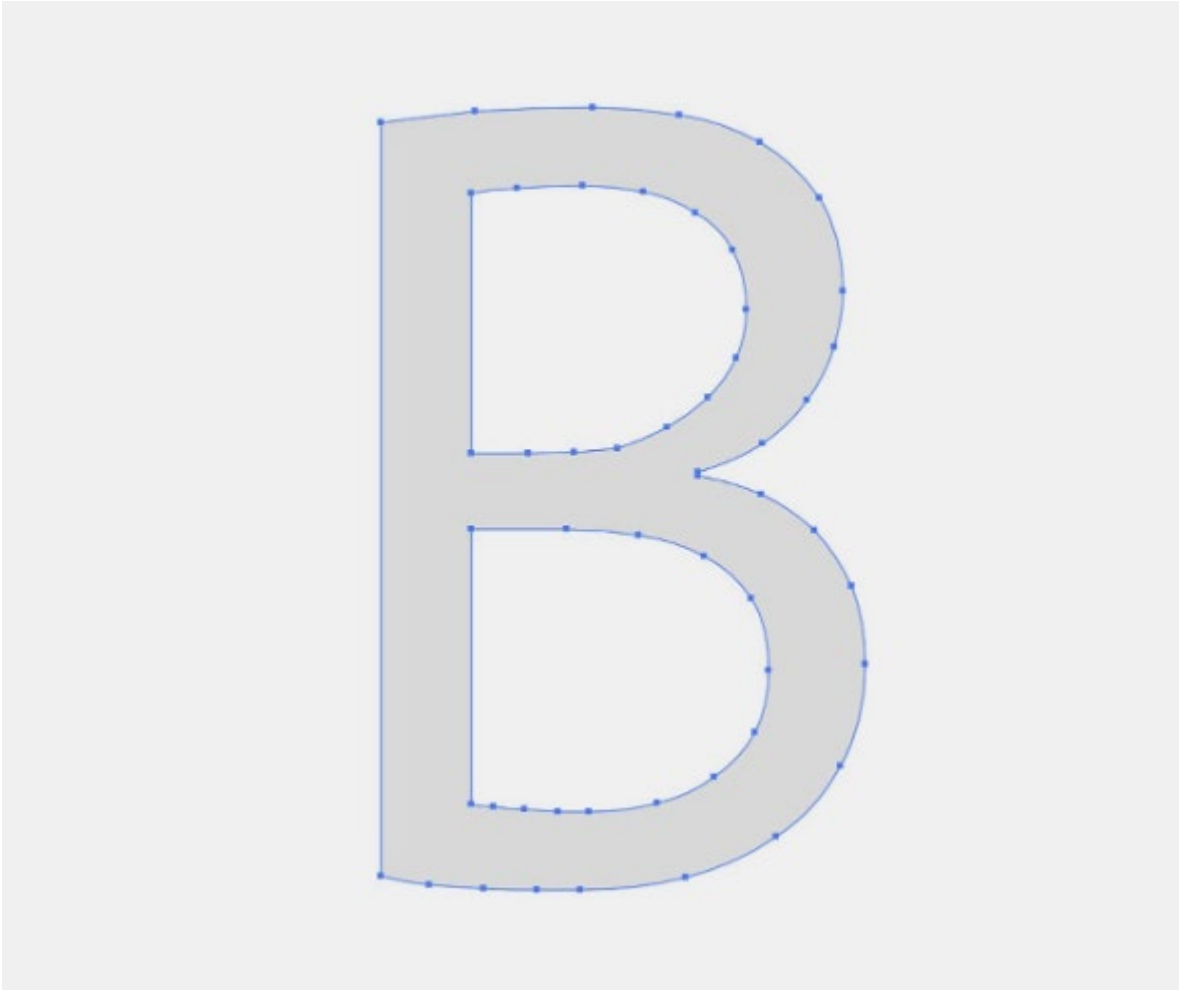

Рисунок 1.3 – Приклад кривих для векторних картинок

З цим видом графіки кожен знайомий, тому що всім доводилося колись робити фотографії. Саме фотографії є найпоширенішим видом 2D графіки, тому можемо побачити їх абсолютно скрізь.

Растрові зображення мають такі характеристики:

- роздільна здатність – кількість пікселів, що припадають на одиницю площі;
- розмір – ширина та висота в пікселях;
- колірний простір – метод відображення кольорів у координатах будь-якої системи кольорів;
- глибина кольору – найбільше відтінків кольорів, які можуть містити зображення.

До плюсів растру належить:

– реалістичність;
– можливість автоматизованого введення інформації;
– швидка обробка важких ілюстрацій;
– адаптивність під всілякі пристрої та програми перегляду.

До мінусів растрових зображень можна віднести наступне:

– великий розмір пам’яті;
– неможливість деформації та масштабування без втрати якості [7].

Приклад растрової графіки зображений на рисунку 1.4:

Рисунок 1.4 – Приклад растрової графіки

З растрової графікою працюють дизайнери інтер’єрів, аніматори, художники, веб-розробники, графічні дизайнери. До найпоширеніших редакторів можна віднести: Adobe Photoshop.

Приклад конвертації растрової графіки у векторну зображену на рисунку 1.5:

Рисунок 1.5 – Приклад конвертації растрової графіки у векторну

Перевага растрової графіки – у колірних можливостях. Вона дозволяє добре передавати градієнти та колірні переходи на складних ілюстраціях. Але при збільшенні зображення його якість погіршуватиметься.

Перевага векторної графіки – у нескінченному розмірі. Логотип у форматі svg можна збільшувати скільки завгодно і його контур не постраждає. Але зробити складну ілюстрацію з безліччю кольорів та дрібних деталей буде складніше.

Для фотографій та складних ілюстрацій найкраще підійде растрова графіка. Для простих ілюстрацій та логотипів – векторна.

Візуальне порівняння растрової та векторної графіки зображено на рисунку 1.6:

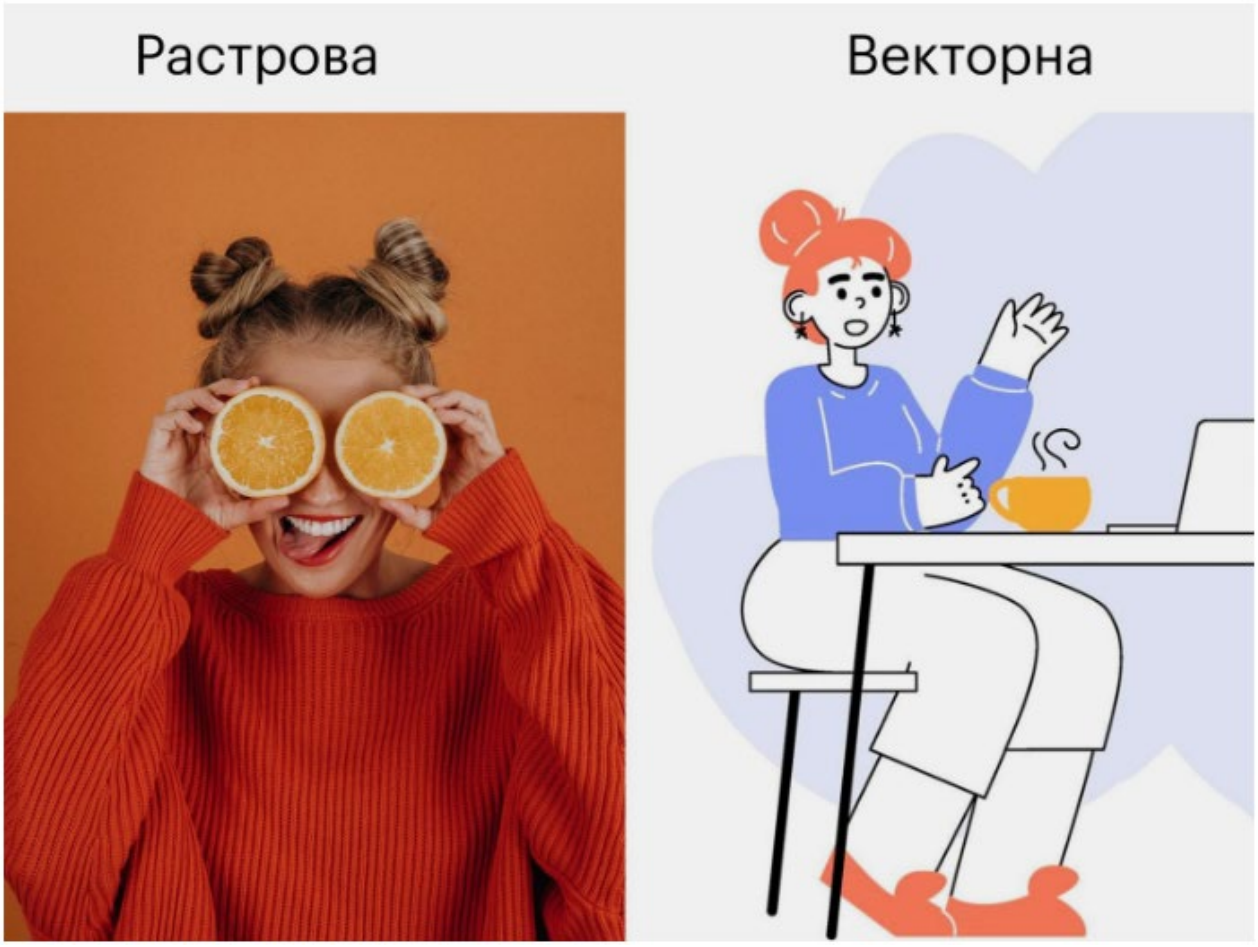

Рисунок 1.6 – Візуальне порівняння растрової та векторної графіки

Фрактальна графіка почала поширюватися зовсім недавно, при цьому вона є одним з перспективних видів комп’ютерної графіки, що найшвидше розвиваються і перспективних.

У фрактальній графіці реалізовано принцип успадкування геометричних якостей, що передаються від одного елемента до іншого. Заснована дана модель на математичних обчисленнях (формулах) і оскільки детального опису дрібних складових не потрібно, то описати такий об’єкт можна декількома рівняннями, результати яких надалі машина відображає автоматично, і не вимагає зберігання в пам’яті комп’ютера будь-яких об’єктів. Фрактальний принцип відображення графіки знайшов широке застосування у багатьох галузях комп’ютерної графіки,

науки та мистецтва. Фрактали широко застосовуються в растровій, векторній та 3D графіці [8, 9].

Цей вид графіки широко використовується для оформлення рекламних листівок, дискотек та веб-сайтів, методами фрактальної графіки часто моделюють турбулентні потоки та створюють різні візерунки.

Переваги фрактальної графіки:

– невеликий розмір;
– можливість масштабування;
– реалістичність.

Недоліки фрактальної графіки:

– необхідність якісного комп’ютерного обладнання;
– обмеження у вихідних математичних фігурах.

Приклад фрактальної графіки зображено на рисунку 1.7:

Рисунок 1.7 – Приклад фрактальної графіки

Найбільш поширені формати фрактальної графіки – BMP, PCX. Можна відзначити кілька програм для генерування фракталів: Fractal Explorer, Apophysis, Mandelbulb3D.

Кожен із представлених видів 2D графіки має свої унікальні риси, саме тому вибір між ними має ґрунтуватися на поставленому завданні.

На відміну від 2D, 3D-графіка формується у трьох вимірах – до параметрів додається вісь глибини (z). Це означає, що стає можливим переміщення об’єктів та персонажів не лише по горизонталі та вертикалі, а ще вперед та назад. Третя вісь також дозволяє обертати та представляти об’єкти з різних сторін. Таким чином, 3D-графіка додає обсягу зображень, дозволяючи глядачеві оцінити розміри об’єкта та відстань до нього [10].

Тривимірні моделі можуть бути двох типів:

- полігональна – сукупність вершин, ребер та граней, які визначають форму багатогранного об’єкта, обволікаючи порожній 3D простір;
- воксельна – сукупність елементів об’ємного зображення, що містить значення растру, які викладаються в об’ємні моделі об’єктів, що мають нутрощі.

Приклад 3D графіки зображено на рисунку 1.8:

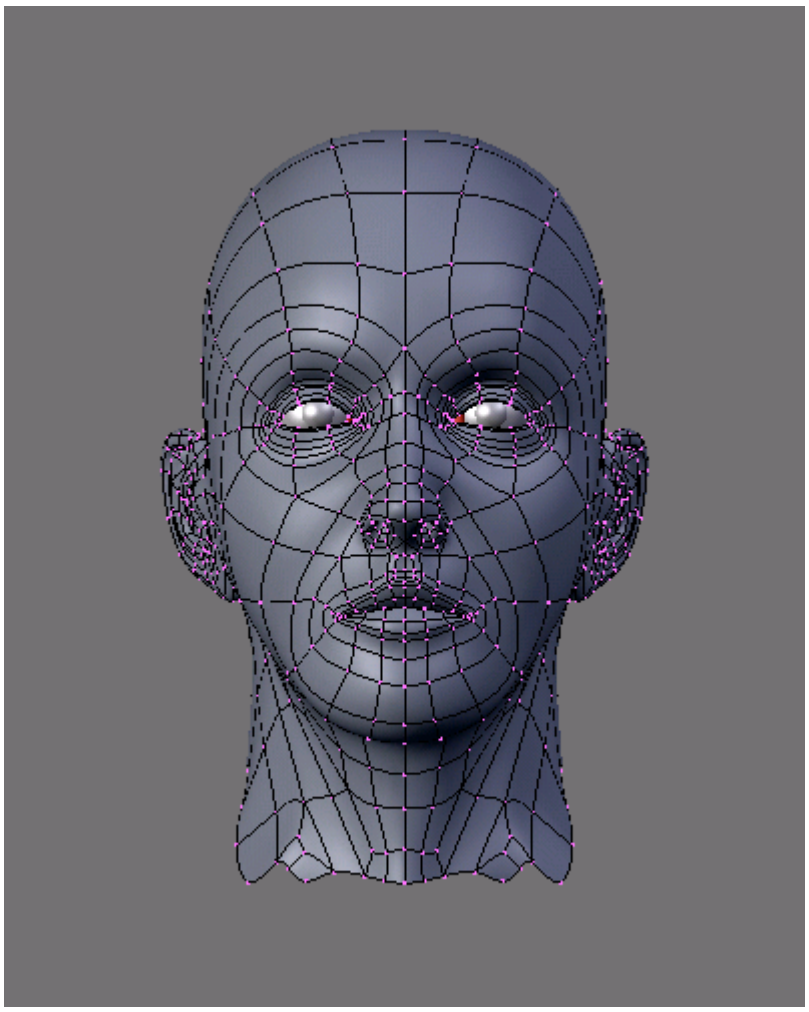

Рисунок 1.8 – Приклад 3D графіки

Тривимірна графіка зустрічається повсюдно і використовується у створенні зображень у всіляких сферах діяльності людини: машинобудування, архітектура, дизайн інтер'єру, реклама, ігрова та кіно індустрія, інтерактивні навчальні проекти. Можна виділити такі редактори: 3ds Max, Autodesk Maya, Cinema 4D, Blender.

Різниця між 2D та 3D графікою. Єдина реальна відмінність – вісь z. Однак додавання глибини змушує всю графіку виглядати інакше. Персонажі та об'єкти зазвичай більш деталізовані та мають тіні, яких у них не було б у 2D-графіці. Фігури більш обтічні, кольори та текстури наближені до реальних [11-19].

Інша важлива відмінність між цими двома типами графіки – це програмне забезпечення, яке використовується для їх створення та анімації. 2D-анімація створюється в основному за допомогою After Effects. 3D-графіка вимагає навичок у Cinema 4D, Houdini, Autodesk Maya, 3Ds Max або Blender.

1.2. Області застосування векторної графіки

Векторна графіка дуже корисна при підготовці зображень для сайтів, і в цьому її головне застосування веб-майстрами. Завдяки тому, що зображення, отримані переведенням з вектора в растр, майже завжди унікальні, вони оцінюються пошуковими системами при ранжируванні картинок. Якщо врахувати ту обставину, що з векторною графікою немає проблем із фонами, текстурами та формами, стає зрозуміло: кліпарту векторних малюнків саме місце на полиці веб-дизайнера.

Системи CAD/CAM використовуються сьогодні у різних галузях інженерної конструкторської діяльності від проектування мікросхем до створення літаків. Провідні інженерні та виробничі компанії, такі як Boeing, зрештою, рухаються до повністю цифрового представлення конструкції літаків.

Приклад векторної графіки в інженерії зображено на рисунку 1.9:

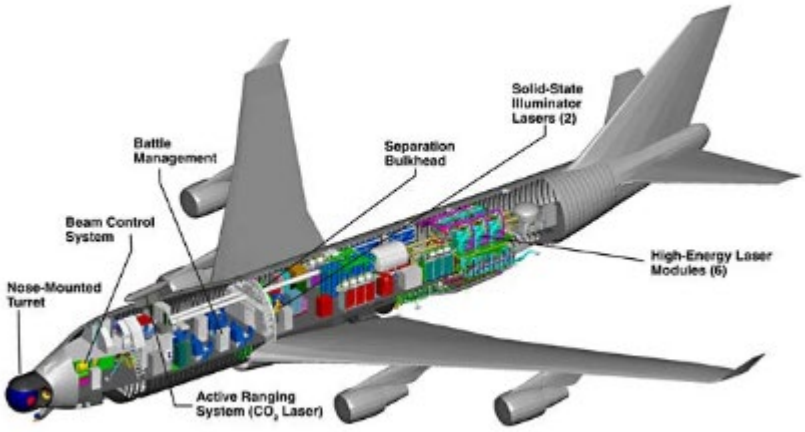

Рисунок 1.9 – Приклад векторної графіки в інженерії

Архітектура є іншою важливою сферою застосування для CAD/CAM і зовсім недавно створених систем класу walkthrough (прогулянки навколо об’єкта, що проектується з метою його вивчення та оцінки). Такі фірми, як McDonald’s, вже з 1987 року використовують машинну графіку для архітектурного дизайну, розміщення місць, планування приміщень та проектування кухонного обладнання. Є ряд ефектних застосувань векторної графіки у галузі

проектування стадіонів та дизайну спортивного інвентарю, новий парк у Балтіморі.

Приклад векторної графіки в архітектурі зображено на рисунку 1.10:

Рисунок 1.10 – Приклад векторної графіки в архітектурі

Медицина стала дуже привабливою сферою застосування комп'ютерної векторної графіки, наприклад: автоматизоване проектування імплантантів, особливо для кісток та суглобів, дозволяє мінімізувати необхідність внесення змін протягом операції, що скорочує час перебування на операційному столі.

Приклад векторної графіки у медицині зображено на рисунку 1.11:

Рисунок 1.11 – Приклад векторної графіки у медицині

Згідно з проведеними дослідженнями, аж до початку дев’яностих років доходи від використання векторної графіки в науково-інженерних додатках були значно вищими, ніж доходи в галузі бізнесу та інших галузях, які безпосередньо не пов’язані з наукою. Однак у 1991 році доходи були поділені рівною мірою, а баланс тепер стійко зсувається у бік нетехнічних додатків. А з 1998 року близько двох третин усіх доходів від комп’ютерної графіки надійде саме з нетехнічних сфер застосування. "Класична" векторна графіка досі використовується в різних додатках бізнесу, включаючи розробку концепції, тестування та створення нових продуктів, але бізнес також став провідним споживачем систем мультимедіа [20-23].

Приклад векторної графіки у дизайні зображено на рисунку 1.12:

Рисунок 1.12 – Приклад векторної графіки у дизайні

Висновки по розділу 1

Таким чином, можна зробити висновок, що векторна графіка на сьогоднішній день не тільки не втратила своєї актуальності для сучасних інституцій, але продовжує бути затребуваною, розвиватися та вдосконалюватися. Векторні зображення зручні для роботи спеціалістів з усього світу та дозволяють вирішувати нові, більш складні завдання, що постають перед ними. Незважаючи на роботу з простими елементами, векторні зображення можуть бути досить різноманітними, втілювати нові технологічні підходи та нові ідеї. При цьому повністю зберігається якість та чіткість ліній, що є важливим аспектом.

Жодна область застосування комп’ютера не може похвалитися таким різноманіттям типів форматів файлів, як комп’ютерна графіка. Кожна більш менш солідна софтверна компанія вважає своїм обов’язком зробити будь-який, але графічний редактор, а на додаток до нього, само собою зрозуміло, створюється свій власний формат файлів, в якому, як запевняють розробники, цей редактор зберігає шедеври, створені за його допомогою. Як результат такого підходу склалася ситуація, коли вже ніхто не в змозі охопити всю різноманітність типів існуючих графічних форматів.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ОБРОБКИ ВЕКТОРНОЇ ГРАФІКИ

2.1. Огляд існуючих форматів векторної графіки

Растрові зображення зберігають у файлі у вигляді прямокутної таблиці, у кожній клітинці якої записаний двійковий код кольору відповідного пікселя. Такий файл зберігає дані і про інші властивості графічного зображення, а також алгоритм його стискування.

Векторні зображення зберігають у файлі як перелік об’єктів і значень їх властивостей: координат, розмірів, кольорів заповнення і штриху (контуру) тощо.

Формат файлу впливає на об’єм пам’яті, який займає цей файл.

Формат EPS (Encapsulated PostScript) має особливе значення для поліграфії. Не слід вважати його форматом виключно для точкових зображень. Він заснований мовою PostScript, розробленою фірмою Adobe Systems Inc та призначений для опису будь-якої графічної та текстової інформації. PostScript широко використовується у видавничих системах будь-якого рівня: від скромних домашніх друкарень до спеціалізованих робочих станцій. Мова PostScript використовується всіма професійними та напівпрофесійними пристроями виводу: лазерними принтерами, плотерами, фотонабірними автоматами. Ці пристрої мають програмний або апаратний інтерпретатор PostScript, що здійснює розтеризацію. В рамках цього формату можливе зберігання векторної та точкової графіки, шрифтів, растрованих зображень та інформації про растрування, контурів відсіювання та колірних профілів. Як і сама мова PostScript, формат EPS є універсальним форматом опису не лише точкових, а й векторних зображень, а також текстової інформації та шрифтів. Підтримує більшість колірних моделей. Можливе використання стиснення за алгоритмами LZW, JPEG та CCITT. Основні моделі кольорів для EPS: RGB, CMYK [24].

Використовуйте EPS, коли вам необхідно надіслати векторний логотип або інше векторне зображення клієнту, дизайнеру чи співробітнику друкарні. З

файлом EPS не доводиться турбуватися про якість зображення: незалежно від розміру, воно завжди відображатиметься чітко та якісно.

Не використовуйте EPS, коли:

– ви маєте справу з фотографіями чи художньою графікою. EPS може містити растрові зображення, але цей тип файлу призначений для векторних зображень.
– вам потрібно розмістити файл онлайн. Легше зберегти зроблений вами макет Photoshop одразу в JPEG, PNG або GIF.

AI (Adobe Illustrator Artwork) – це векторний формат файлів, створених програмою Adobe Illustrator. У Adobe Illustrator велика кількість версій – Adobe Illustrator 3, Adobe Illustrator 4, Adobe Illustrator 5 тощо, файл може бути відкритим у новішій версії програми (Adobe Illustrator 10, CS, CS2 і т. д.), але не може бути відкритий у більш старій версії програми (Adobe Illustrator 8, 7, 6 тощо), хоча з версії Adobe Illustrator 10 підтримує можливість імпорту файлів новіших версій. Формат забезпечує дуже високу якість малюнків, але по ряду параметрів погано сумісний з іншими програмами (наприклад, різні ефекти Adobe Illustrator та градієнтна заливка можуть не передаватись до інших форматів) [25].

Формат AI по суті є форматом PGF, сумісність із PDF здійснюється за рахунок вбудовування повної копії PGF-даних у файл PDF-формату. Такий же подвійний підхід використовується при збереженні сумісних файлів EPS в останніх версіях Illustrator.

Використовуйте AI, коли:

– вам потрібно відредагувати дизайн вектор. AI-файли дозволяють переміщати та змінювати кожен елемент у вашому дизайні парою кліків;
– ви повинні створити логотип, значок або фірмовий знак. Будь-яка векторна графіка, створена в Illustrator, може бути збільшена до будь-якого розміру без втрати якості;

– ви створюєте односторінковий друкований документ: буклет, постер, візитку, листівку або нотатку.

Не використовуйте AI, коли вам потрібно редагувати растрове зображення. Якщо у композиції використовується растрова графіка (фотографія або малюнок), Illustrator менш зручний, оскільки має обмежену кількість інструментів для такого редагування. У Photoshop доступні більш складні налаштування, наприклад: зміна кольору, контрастність та яскравість.

SVG (Scalable Vector Graphics). Це мова розмітки, що розробляється W3C консорціумом з 1999 р. Мова служить для опису графічних даних на web-сторінці як тексту (XML).

Термін SVG може означати не тільки саму мову розмітки, але й формат одержуваного зображення.

На SVG можна створювати нескладні векторні та змішані векторно-растрові зображення. Мову розробляли, ґрунтуючись на ідеї технології VML від Microsoft та PGML. На відміну від VML, SVG не є чиєюсь власністю, повністю відкритий і безкоштовний.

SVG-рисунок складається з набору геометричних фігур, описаних у форматі XML: лінія, еліпс, багатокутник тощо. Ви можете візуально оцінити частину можливостей формату SVG та познайомитись із синтаксисом розмітки на сторінці онлайн-редактора.

Основною перевагою SVG перед іншими форматами, що застосовуються в Інтернеті, є те, що SVG-картинка є не що інше, як простий текстовий файл. Такий файл можна відкрити та відредагувати будь-яким текстовим редактором, а також серверним скриптом. Також SVG може похвалитися підтримкою анімації, причому не покадрової, як у GIF, а з дуже широкою системою керування, що базується на мові SMIL. Мабуть, єдиним суттєвим недоліком SVG є велика вага складних малюнків, але цю неприємність легко усунути за допомогою технології gzip-стиснення [26].

Використовуйте SVG, коли:

– вам необхідно створити іконки, значки, дрібні векторні графіки для сайту. SVG займають менше місця, ніж JPEG чи GIF;
– вам необхідно створити векторну графіку для Інтернету та бути впевненим, що вона буде коректно та якісно відображатись у всіх браузерах та пристроях.

Не використовуйте SVG, коли вам потрібно створити графіку великого розміру чи з великою кількістю дрібних деталей.

Формат файлу CDR – векторне зображення або малюнок, створений за допомогою програми CorelDraw. Цей формат файлу розроблений компанією Corel для використання у власних програмних продуктах. CDR-файли не підтримуються багатьма програмами, призначеними для редагування зображень. Однак файл можна експортувати за допомогою CorelDraw до інших, більш поширених та популярних форматів зображень.

Також можна відкрити файл CDR програмою Corel Paint Shop Pro. Для кращої сумісності Corel рекомендує зберігати файли в CorelDraw форматі CDR версії 9.0 або раніше.

Для відкриття файлу CDR версії 10 і раніше можна використовувати програму Adobe Illustrator [27].

Використовуйте CDR, коли:

– ви повинні створити логотип, значок або фірмовий знак. Будь-яка векторна графіка, створена CorelDRAW, може бути збільшена до будь-якого розміру без втрати якості;
– ви створюєте односторінковий документ: буклет, постер, візитку або листівку;
– вам потрібно створити векторну графіку і ви вважаєте за краще працювати в CorelDRAW, а не в Adobe Illustrator.

Не використовуйте CDR, коли вам потрібно відредагувати растрове зображення (фотографію або рисунок). CorelDRAW має обмежену кількість інструментів для редагування растрових елементів. Крім того рекомендовано конвертувати CDR у більш популярні векторні формати – наприклад, в AI або EPS, тому що зараз формат CDR використовується все рідше.

CMX (Corel Presentation Exchenge). Це формат графічних програм корпорації Corel, призначений для передачі малюнків між різними програмами. Формат підтримується CorelDraw Graphics Suite X4 – Free Trial, Corel Paint Shop Pro Photo X2 – Free Trial, Adobe Illustrator 8.

PDF розшифровується як Portable Document Format і є універсальним форматом, який допомагає правильно зображати як векторну, так і растрову графіку. Він добре підтримується різними графічними програмами. PDF-файли коректно відображаються незалежно від програми, операційної системи або браузера.

PDF підтримує багатосторінкові документи, тому цей формат часто використовується для друку. Вам ніколи не відмовить у друкарні, якщо ви принесете правильно підготовлений PDF-файл. Будь-яке зображення, яке ви створювали у програмах Adobe Photoshop або Illustrator, може бути експортовано безпосередньо до PDF.

Використовуйте PDF, коли:

- ви готуєте файли до друку. Багато друкарень із задоволенням приймають PDF і працюють з ним як основний формат;
- ви хочете використовувати багатосторінкові документи в Інтернеті. Нелогічно використовувати PDF для однієї іконки або логотипу, але цей формат відмінно підходить для електронних книг, журналів та буклетів. Цей файл зручно переглядати, завантажувати та друкувати.

Не використовуйте PDF, коли вам потрібно міняти свій дизайн. Ви можете редагувати растрові зображення у Photoshop, а векторну графіку – у Illustrator. Коли закінчите, вже можете поєднати їх у PDF [28-35].

2.2. Огляд редакторів векторної графіки

Графічний редактор – це спеціальна програма або комплекс програм, які використовуються для створення або редагування зображення на комп'ютері.

Існуючі графічні редактори можуть створити нове зображення, так і відредагувати вже наявне. За допомогою цих програм можна переміщувати зображення, розгортати їх, видаляти чи копіювати елементи. Зображення, що вийшло, при цьому можна відразу роздрукувати і зберегти.

У векторних графічних редакторах створюють та редагують векторні зображення форматів CDR, EPS, AI та інші.

Далі розглянемо найбільш популярні редактори для векторної графіки. CorelXara – програма нового покоління. Вона використовується для створення графічного зображення на сторінці за один раз і формування блоку тексту за один раз. Програма дозволяє виконувати з малюнками, градієнтним заповненням, зображеннями та діапозитивами такі дії, про які ви могли лише мріяти. Хоча Corel рекламує CorelXara 1.5 як додаток до CorelDraw для створення графіки Web, але завдяки високій продуктивності, засобам для роботи з Web та спеціалізованому інструментарію CorelXara перевершує CorelDraw у багатьох відношеннях.

Завдяки можливостям масштабування векторної графіки та текстурам растрових зображень двовимірні об'єкти починають все більше нагадувати тривимірні. Намалюйте об'єкт. Накладіть текстуру (растрове зображення) або зафарбуйте її (матеріал). Визначте рівень прозорості. Потім перемістіть зображення та відредагуйте на свій смак. Інтерфейс CorelXara елегантний і простий. Піктограми у верхньому ряду забезпечують доступ до повнокольорових візуальних наборів кольорів, заповнень, штрихування, растрових зображень, шрифтів та графічних вставок (кліпартів).

Найпотужніший на сьогодні інструментальний засіб для графіки Web – зовнішній модуль CorelXara для Netscape Navigator та Microsoft Internet Explorer – дозволяє безпосередньо з браузера збільшувати масштаб зображення до 25000%. Завдяки компактності файлу та високій продуктивності перед

векторною графікою відкриваються блискучі перспективи в області розробки сторінок Web. CorelXara може далеко ще не все, але у деяких відносинах ця програма немає собі рівних. Якщо ви готуєте складні оригінал-макети, якщо починаєте користуватися пакетами для малювання або любите працювати з прозорими шарами, CorelXara стане гарним доповненням до вашого комплекту інструментів [36].

Adobe Illustrator – це програма, призначена для роботи з векторною графікою. За допомогою Adobe Illustrator дизайнери створюють барвисті ілюстрації, іконки, патерни, логотипи, різноманітні макети для друку та багато іншого.

Основні функції Adobe Illustrator:

- робота із геометричними фігурами. В ілюстраторі можна спритно працювати з фігурами – об’єднувати їх, віднімати, доповнювати іншими контурами і так далі. Геометричні форми часто є основами для іконок та логотипів, тому робота з ними – річ фундаментальна;
- створення з ілюстраціями. Можна робити яскраві ілюстрації з фігур, ліній та контурів, застосовувати різні ефекти та текстури. В Adobe Illustrator можна створювати будь-які форми, накладати їх один на одного за допомогою шарів та застосовувати різні ефекти: градієнти, тіні, текстури та інше;
- трасування растрової графіки. Ілюстратор дає можливість переводити растрове зображення у векторне за кілька кліків;
- редагування контурів. Можна малювати самостійно інструментом "Перо" або редагувати чужі ілюстрації. Можна завантажувати проекти в форматі Ai зі стоків і редагувати кожен елемент;
- застосування заливок. В Adobe Illustrator можна легко застосовувати різні заливки до об’єктів і контурів;
- підготовка до друку. Візитки, плакати чи величезні рекламні щити. Так як векторна графіка масштабується до будь-яких розмірів, в Ілюстратор можна створювати рекламні щити розміром з будинок. Параметри

експорту файлу дозволяють налаштувати вильоти, колірний профіль для друку та вибрати формат;

- робота з монтажними галузями. За допомогою монтажних областей зручно, наприклад, робити банери для сайту у різних розмірах;
- перетворення тексту на криві. Будь-який текст в Adobe Illustrator можна перевести в криві, тобто зробити векторними. Це дозволить працювати з ним так, ніби це векторна форма – можна змінювати форму літер, зробити логотип або гарний напис.

Можна зробити в Adobe Illustrator:

- логотипи;
- шрифти та написи;
- іконки;
- патерни;
- ілюстрації;
- макети для друку;
- графіку для соціальних мереж та сайтів;
- інфографіку;
- графіку для анімації;
- стікери.

Adobe Illustrator – це одна з найпопулярніших програм у дизайнерській сфері. Проект, створений у Adobe Illustrator, можна зберегти у 5 власних форматах: AI, PDF, EPS, FXG та SVG. Також є можливість конвертувати у інші формати [37].

CorelDRAW – це програма, яка є графічним редактором. Вона працює із векторними зображеннями, дає можливість створювати унікальні шаблони. Користувачі роблять контури, складають схеми, малюють логотипи.

Програма використовується дизайнерами для різних цілей:

- обробка зображень;
- проектування меблів;

- створення візуального контенту;
- начерки для зображення;
- моделювання об’єктів.

Під час роботи з програмою фахівці відзначають її особливості. Завдяки оновленню останньої версії виправлено багато помилок, користувачам відкриті додаткові функції.

Основні плюси:

- різні шрифти;
- підтримка форматів;
- підходить для плакатів;
- інструменти створення реклами;
- каталог форм;
- підтримка операційних систем Windows, MAC, OS.

Якщо порівнювати програму з такими інструментами, як Adobe Illustator, є додаткові особливості. Дизайнерам подобається імпортувати файли і не звертати увагу на формат. Готові макети можна використовувати для офсетного друку.

Якщо перейти до властивостей програми, можна дізнатися про список підтримуючих форматів.

Основні типи:

- AI;
- PFB;
- BMP;
- BMP;
- CGM;
- CDR.

Програма легко відкриває бібліотеки символів CSL. Якщо людина працює з курсором, йому буде цікавий ресурс CUR. У вікно редактора можна завантажувати файли Microsoft Word.

Доступні варіанти:

– DOC;
– DOCX;
– RTF.

Багато дизайнерів займаються виключно кресленнями. Маючи справу з бібліотекою AutoCAD, важливо щоб програма читала бінарний формат DWG. Ще редактор відкрито для графічного розширення DXF.

Продукти Adobe різноманітні та радують, що система розуміє міжплатформний формат PDF. Це дає можливість завантажувати електронні документи та обробляти інформацію [38].

Висновки по розділу 2

Графічні файли досить складно влаштовані, на відміну скажемо від простого текстового файлу. Згодом виникла необхідність отримання графічних файлів з певними вимогами. Наприклад, комп'ютерному художнику необхідна дуже висока якість картинки, пересічному користувачеві – хороша якість, але не дуже великий обсяг, веб-дизайнеру потрібно при мінімальному обсязі отримати більш менш пристойне зображення. Але вимоги переносимості файлів між різними програмами змусили виділити кілька певних форматів, що стали, кожен у своїй області, стандартами де-факто. Так, для зображень в Інтернеті, як правило, використовуються формати JPEG і GIF, для зберігання зображень – JPEG, у видавничій справі панує TIFF і т.д.

Тому комп'ютерна графіка є однією з галузей інформатики, що найбільш розвиваються, і в багатьох випадках виступає «локомотивом», що тягне за собою всю комп'ютерну індустрію. Навчання комп'ютерної графіки має проходити на основі єдності освіти та виховання, творчої діяльності, поєднання практичної роботи з розвитком у людей здібностей сприймати і розуміти витвори мистецтва.

РОЗДІЛ 3. ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Постановка задачі

Головним завданням роботи є розробка веб програмного забезпечення, який буде працювати у будь-якому сучасному браузері та надасть можливість створювати, редагувати та конвертувати у інший формат векторні картинки. Даний програмний засіб надасть більші можливості для створення векторних картинок. Для користування даним програмним забезпеченням не потрібно робити будь-які реєстрації у системі, а потрібно мати лише доступ до мережі інтернет та посилання за яким можна перейти у програмне забезпечення і зразу можна починати користуватись.

Програмний засіб потрібно реалізувати у такий спосіб, щоб він був зручним та простим у використанні для різних категорії користувачів. Програмний продукт повинен бути доступним, але містити весь необхідний функціонал, який зустрічається в системах, які уже реалізовані або в процесі реалізації.

Увесь користувацький інтерфейс буде поділений на 1 вкладку, на якій є усе необхідне для створення, редагування та збереження картинки.

Сервіс повинен швидко працювати і швидко реагувати на будь-які дії користувача. Це важливо для ефективної праці користувача.

Сервіс повинен мати достатню надійність, зручність у використанні та передбачати можливість внесення змін для розширення його функціоналу.

Сервіс повинен мати простий, ненав’язливий, але оригінальний дизайн – для проекту це одна з найголовніших критерій.

Сервіс потрібно розробити таким чином, що він був сумісний з усіма браузерами.

3.2. Вибір типу програмного забезпечення

Обираючи майбутню архітектуру та середовища розробки векторного редактора слід зважати на те, що проект буде розроблений у вигляді веб-системи, раціональним буде використання архітектури клієнт-сервер. Оскільки система не містить складної логічної структури буде оптимально використати просту та зрозумілу архітектуру. Однією з основних причин використання саме цієї моделі взаємодії є те, що функціонал системи буде чітко поділений (на клієнтську та серверну частину), таким чином обчислювальні навантаження також зменшиться.

Клієнтська частина системи буде складатися з безлічі компонентів, основними з яких будуть: сторінки, чи окремі її частини, які в свою чергу стануть самостійними компонентами для того, щоб забезпечити лаконічність коду, завдяки чому не буде дублювання коду.

Клієнтська частина є найважливішою складовою системи, адже саме вона буде містити основну логіку користувача, саме від неї буде залежати великий відсоток успіху системи.

Серверна частина системи повинна надавати доступ до бази даних. Також вона відповідатиме за конвертування уже отриманих даних у комфортний для користувача вигляд та формат. Також саме в ній буде відбуватися реалізація бізнес-логіки з сервером.

Переваги та недоліки системи клієнт/сервер. До переваг системи клієнт/сервер слід віднести:

- сильний централізований захист;
- центральне сховище файлів;
- можливість спільного використання серверами доступного технічного та програмного забезпечення;
- просту керованість при великій кількості користувачів;
- централізовану організацію, що запобігає втраті даних на комп’ютерах.

Недоліками цієї системи є:

- дороге технічне забезпечення;
- дорогі серверні операційні системи та клієнтські ліцензії;
- крім того, часто потрібний адміністратор мережі.

3.3. Обґрунтування вибору інструментальних засобів для розробки клієнтської частини

JavaScript – це мова програмування, яку розробники використовують для створення інтерактивних веб-сторінок. Функції JavaScript можуть покращити зручність взаємодії користувача з веб-сайтом: від оновлення стрічки новин у соціальних мережах до відображення анімації та інтерактивних карт. JavaScript є мовою програмування розробки скриптів для виконання на стороні клієнта, що робить її однією з базових технологій у всесвітній мережі Інтернет. Наприклад, карусель зображення, що випадає по кліку меню і кольори елементів, що динамічно змінюються, на веб-сторінці, які ви бачите під час перегляду сторінок в Інтернеті, виконані за допомогою JavaScript.

Історично веб-сторінки були статичними, схожими на сторінки книги. Статична сторінка здебільшого відображала інформацію у фіксованому вигляді та не виконувала всього того, що ми зараз очікуємо від сучасного сайту. Мова JavaScript виникла як технологія на стороні браузера, що дозволяє зробити веб-програми динамічнішими. Використовуючи її, браузери могли реагувати на взаємодію з користувачем та змінювати розташування контенту на веб-сторінці.

У міру розвитку мови розробники JavaScript створили бібліотеки, фреймворки та практики програмування та почали використовувати її за межами веб-браузерів. Сьогодні JavaScript можна використовувати для розробки як на стороні клієнта, так і сервері.

Всі мови програмування працюють шляхом переведення англійського синтаксису в машинний код, який виконує операційна система. JavaScript у широкому розумінні можна віднести до категорії скриптових або інтерпретованих мов. Код JavaScript інтерпретується, тобто безпосередньо

перекладається код машинної мови движком JavaScript. В інших мовах програмування компілятор обробляє весь код машинного на окремому етапі. Таким чином, усі скриптові мови є мовами програмування, але не всі мови програмування є скриптовими.

JavaScript – це комп'ютерна програма, яка виконує код JavaScript. Перші двигуни JavaScript були лише інтерпретаторами, але всі сучасні двигуни використовують для підвищення продуктивності JIT-компіляцію або компіляцію під час виконання.

JavaScript на стороні клієнта відноситься до того, як JavaScript працює у вашому браузері. У цьому випадку движок JavaScript знаходиться всередині коду браузера. Всі основні веб-браузери мають власні вбудовані движки JavaScript.

Нижче наведено огляд того, як працює JavaScript на стороні клієнта.

- Браузер завантажує веб-сторінку, коли ви її відвідуєте.
- У процесі завантаження браузер перетворює сторінку та всі її елементи, такі як кнопки, ярлики та поля, що випадають, в структуру даних, звану об'єктною моделлю документа (DOM).
- Двигун JavaScript браузера перетворює код JavaScript на байткод. Цей код є посередником між синтаксисом JavaScript та машиною.
- Різні події, такі як клацання миші по кнопці, викликають виконання зв'язаного блоку JavaScript. Потім двигун інтерпретує байткод і вносить зміни до DOM.
- Браузер відображає новий DOM.

JavaScript на стороні сервера відноситься до мови кодування в логіці внутрішнього сервера. У цьому випадку движок JavaScript знаходиться безпосередньо на сервері. Функція JavaScript на стороні сервера може звертатися до бази даних, виконувати різні логічні операції та реагувати на різні події, які запускає операційна система сервера. Основна перевага скриптингу на стороні сервера полягає в тому, що ви можете значно налаштувати відповідь сайту відповідно до ваших вимог, прав доступу та інформаційних запитів сайту.

Динамічна поведінка – це здатність оновлювати відображення веб-сторінки для створення нового контенту в міру необхідності. Різниця між JavaScript на стороні клієнта та на стороні сервера полягає у генеруванні нового контенту. Код на стороні сервера динамічно генерує новий контент, використовуючи логіку програми та змінюючи дані з бази даних. JavaScript на стороні клієнта, з іншого боку, динамічно генерує новий контент усередині браузера, використовуючи логіку інтерфейсу користувача і змінюючи контент веб-сторінки, яка вже знаходиться на стороні клієнта. Сенс трохи відрізняється в цих двох контекстах, але вони пов’язані, і обидва підходи працюють разом для покращення користувальницького досвіду.

Інша різниця між двома видами використання JavaScript, крім реалізації динамічних функцій, полягає в ресурсах, до яких код може отримати доступ. На стороні клієнта браузер контролює середовище JavaScript. Код може отримати доступ тільки до тих ресурсів, які дозволяє браузер. Наприклад, він не може записати вміст на жорсткий диск, якщо ви не натиснете на кнопку завантаження. З іншого боку, функції на стороні сервера можуть отримати доступ до всіх ресурсів серверної машини за необхідності.

Можливості JavaScript у браузері обмежені задля безпеки користувача. Мета полягає в запобіганні доступу недобросовісної веб-сторінки до особистої інформації або нанесення шкоди користувачам.

Приклади таких обмежень включають:

JavaScript на веб-сторінці не може читати/записувати довільні файли на жорсткому диску, копіювати або запускати програми. Він не має прямого доступу до системних функцій ОС.

Сучасні браузери дозволяють йому працювати з файлами, але з обмеженим доступом, і надають його, тільки якщо користувач виконує певні дії, такі як перетягування файлу у вікно браузера або його вибір за допомогою тега <input>.

Існують способи взаємодії з камерою/мікрофоном та іншими пристроями, але вони потребують явного дозволу користувача. Таким чином, сторінка з

підтримкою JavaScript не може непомітно включити веб-камеру, спостерігати за тим, що відбувається.

Різні вікна/вкладки не знають одне про одного. Іноді одне вікно, використовуючи JavaScript, відкриває інше вікно. Але навіть у цьому випадку JavaScript з однієї сторінки не має доступу до іншої, якщо вони прийшли з різних сайтів (з іншого домену, протоколу чи порту).

JavaScript може легко взаємодіяти з сервером, з якого надійшла поточна сторінка. Але його здатність отримувати дані з інших сайтів/доменів обмежена. Хоча це можливо в принципі, для чого потрібна явна згода (виражена в заголовках HTTP) із віддаленою стороною. Знову ж таки, це обмеження безпеки.

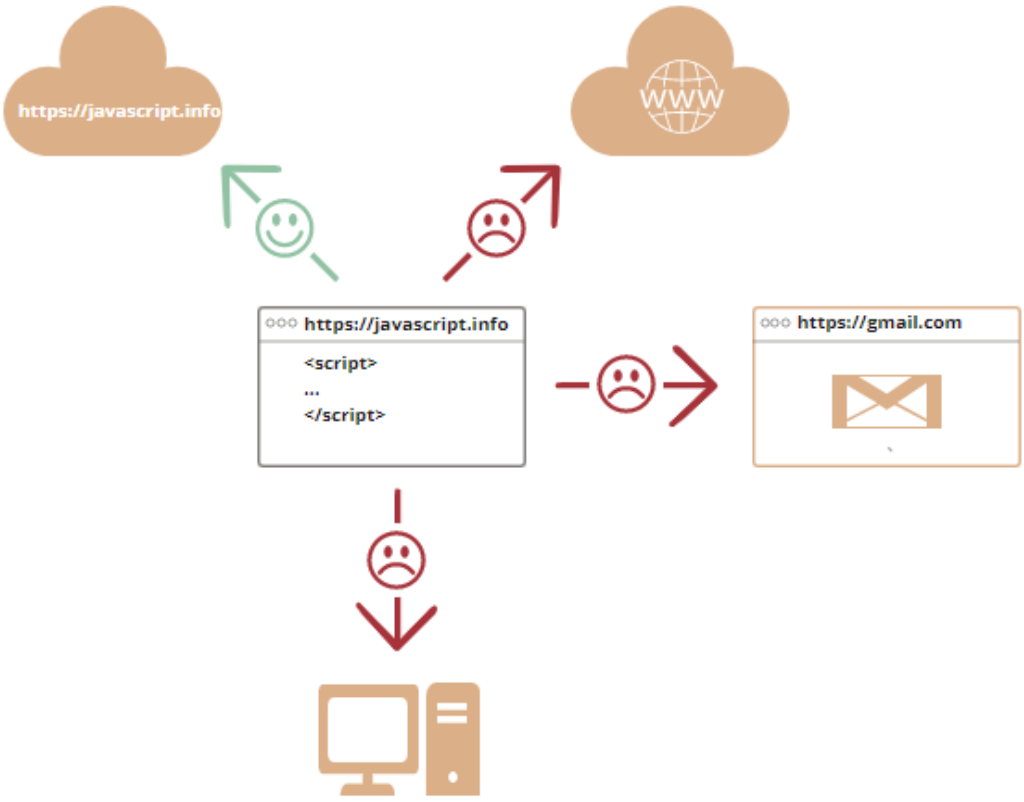

Рисунок 3.1 – Обмеження Java Script

Бібліотеки JavaScript – це колекції попередньо написаних фрагментів коду, які веб-розробники можуть повторно використовувати для виконання стандартних функцій JavaScript. Код бібліотеки JavaScript підключається до іншого коду проекту за необхідності.

Нижче наведено приклади використання бібліотек JavaScript.

- візуалізація даних. Візуалізація даних вкрай важлива для користувачів, щоб переглядати статистику, наприклад, у панелі адміністратора, панелі приладів та показниках продуктивності. У таких бібліотеках, як Chart.js, ApexCharts та Algolia Places, є вбудовані функції, які можна використовувати для створення веб-застосунків, що відображають дані у вигляді графіків і карт;
- маніпулювання DOM. Ви можете використовувати такі бібліотеки, як jQuery та Umbrella JS, щоб спростити розробку веб-сайтів, оскільки вони надають код для стандартних функцій сайту, таких як анімація меню, галереї зображень, кнопки, лайтбокси та багато іншого;
- форми. У всіх веб-розробках використовуються форми, за допомогою яких відвідувачі сайту можуть зв’язатися з будь-ким, замовити товар або зареєструватися на заходи. Деякі бібліотеки JavaScript, такі як wForms, LiveValidation, Validanguage та qForms, спрощують функції форми, включаючи її перевірку, компонування, умови та перетворення;
- математичні та текстові функції. Багато веб-програм повинні вирішувати математичні рівняння, обробляти дати, час і текст. Замість того, щоб надсилати всі такі запити на сервер, ефективніше обробляти деякі з них на стороні клієнта. Веб-розробники роблять це за допомогою бібліотек JavaScript, таких як Date.js, Sylvester та JavaScript URL Library.

Як і бібліотеки JavaScript, фреймворки JavaScript є набором попередньо написаних фрагментів коду, які виконують різні функції і можуть бути використані повторно. Однак якщо бібліотеки JavaScript – це спеціалізований інструмент для використання на вимогу, фреймворки JavaScript – це повний набір інструментів, який допомагає сформувати та організувати будь-який веб-додаток.

Переваги мови JavaScript:

- простота освоєння та використання. Синтаксис JavaScript сформований з урахуванням мови програмування Java, його легко вивчити, щоб написати

код. Розробники використовують JavaScript майже в кожному сайті та мобільному додатку для створення сценаріїв на стороні клієнта. В останнє десятиліття Node.js також набув значної популярності для кодування бекенда. Багато великих платформ потокового мовлення та відео було розроблено на Node.js;

- незалежність від платформи. На відміну від інших мов програмування, JavaScript можна вставити в будь-яку веб-сторінку та використовувати з іншими механізмами та мовами веб-розробки. Написавши на ньому будь-що, ви можете виконувати код JavaScript на будь-якій машині. Таким чином, JavaScript робить розробку програм незалежної від платформи;
- зниження навантаження на сервер. Ви можете використовувати JavaScript для зниження навантаження на сервер та перевантаження мережі, тому що він може виконувати логічні операції та робити більшу частину роботи сервера на стороні клієнта. Наприклад, розглянемо процес заповнення реєстраційної форми. JavaScript швидко перевіряє, чи ввели ви 10-значний номер для поля мобільного телефону. Якби ці запити надсилалися на сервер, ваша сторінка перезавантажувалася б при кожній помилці, що зробило б процес реєстрації дуже повільним та стомлюючим;
- поліпшення інтерфейсу користувача. JavaScript створює веб-сайти, які роблять зручним пошук та обробку складної інформації. Розробники застосовують JavaScript для розширення функціональності та читабельності, а також для підвищення ефективності взаємодії користувача з сайтом;
- підтримка паралелізму. JavaScript може виконувати кілька різних наборів вказівок паралельно. На сервері Node.js може обробляти безліч відповідей на запити сервера, потребуючи при цьому меншої пропускної спроможності [39-49].

3.3.1. Огляд фреймворку Vue

Vue.js – це середовище JavaScript, яке розробники цінують за його універсальність. Результати опитування State of JavaScript 2022 показали, що майже половина респондентів використовують Vue.js, а рівень задоволеності фреймворком перевищує 80%.

Позначений як "Прогресивне середовище JavaScript", Vue.js, без сумніву, може запропонувати щось привабливе звичайному розробнику.

Vue.js – це JavaScript-фреймворк модель-подання-подання (MVVM) для створення інтерфейсів користувача (UI) і односторінкових додатків. Творцем Vue.js є колишній інженер Google Еван Ю, який швидко розчарувався у використанні AngularJS у проектах Google. Він вирішив отримати кращі риси Angular і створити щось неймовірно легке.

Vue.js – це легкий інтерфейсний JavaScript-фреймворк.

Архітектура MVVM Vue.js забезпечує відділення логіки уявлення від бізнес-логіки, яка представляє уявлення та модель відповідно. Потім модель уявлення діє як посередник, готуючи об’єкти даних до подання. Але, крім архітектури, найбільш примітною особливістю Vue.js є директиви. Директиви розширюють HTML за допомогою HTML-атрибутів, розширюючи функціональні можливості HTML-додатків. Ці директиви є вбудованими, але вони також можуть бути визначені користувачем. По суті вони дозволяють маніпулювати об’єктною моделлю документа (DOM). DOM – це свого роду інтерфейс, зокрема API для документів HTML та XML, який інструктує машини про те, як структурувати текст. Концепція директив існує як в Angular, так і Vue.js, але ви можете підключити Vue.js до будь-якої частини серверного додатка і вручну налаштувати більш інтерактивний досвід для споживачів. Коротше кажучи, ця функція дозволяє елементам HTML інкапсулювати повторно використовуваний код. Фреймворки JavaScript, такі як Vue.js, сприяють більшій організації, коли доходить до розробки інтерфейсу. Vue.js може взяти веб-сторінку і розбити її на компоненти, що повторно використовуються, кожен з яких має свої власні елементи HTML, CSS і JavaScript. Інші розумні інструменти Vue.js включають

Vuex та Vue-Router. Вони працюють з основним програмуванням Vue.js, надаючи наступне:

Керування станом, коли елементи керування інтерфейсу користувача, такі як текстові поля, кнопки OK і т. д., можуть вимагати управління за межами поточної сторінки, яку відвідує користувач (Vuex).

Маршрутизація — процес, який відбувається, коли вам потрібно синхронізувати URL-адресу з поданнями у вашій програмі (Vue-Router).

Vue.js — універсальна мова, тому ви можете використовувати її для різних цілей.

По-перше, Vue.js добре підходить для будь-яких проектів, в яких задіяні HTML, CSS та JavaScript. І саме тому Vue.js в основному використовується для створення інтерфейсів і будь-яких веб-розробок. Але оскільки Vue.js легкий, це також добрий вибір для швидкого прототипування.

Односторінкові програми (SPA) також знаходяться у сфері компетенції Vue.js. У SPA є лише одна сторінка для перегляду інформації. Користувачі отримують доступ до нової інформації, оскільки сайт динамічно переписує сторінку при взаємодії з користувачем. Як і React Native, Vue.js дозволяє розробляти нативні мобільні програми. Розробники можуть використовувати NativeScript для створення мобільних додатків на Android та iOS із загальним кодом JavaScript. Може бути корисно знати, що глобальні веб-сайти, що використовують Vue.js, включають такі, як Facebook, Netflix і Google, що демонструє, наскільки потужна ця структура.

Розробники використовують Vue.js з низки причин. Зрозуміло, що Vue.js є популярним, але його справжня привабливість полягає в його технічній проникливості. Ось деякі особливості, що відображають можливості Vue.js:

– легкий дизайн. Vue.js називають прогресивним, тому що він призначений для поступового впровадження. Це означає, що програмування на Vue.js означає починати з малого та масштабувати за власним бажанням. В результаті Vue.js досить простий і не відрізняється складністю більших фреймворків. Плюс варто відзначити, що розмір фреймворку Vue.js менше

21 кілобайта. Так що будьте впевнені, вам не доведеться мати справу із затримкою важкого програмного забезпечення на ваших робочих машинах;

- двостороння прив’язка даних. Маніпуляції з DOM – це двонаправлена подія. Те, що відбувається у виставі, впливає на модель, і навпаки. Хоча потік даних в одному напрямку може бути менш складним на мозковому рівні, одностороння прив’язка даних перешкоджає процесу розробки. Зміни моделі відбиваються на уявленні, але з навпаки. Отже, внесення змін до інтерфейсу користувача вимагає додаткової роботи;
- DOM-рендерінг. Vue.js – постачальник віртуального DOM. (Vue.js безкоштовний і має відкритий вихідний код, тому плата не стягується) Віртуальний DOM – це засіб зберігання уявлення інтерфейсу користувача в пам’яті. Через узгодження віртуальний DOM синхронізує зміни із «справжнім» DOM пізніше. Віртуальні DOM вигідні, тому що негайне оновлення «реальної» DOM може бути стомлюючим та повільним. Використовуючи віртуальний DOM, розробники можуть бачити зміни, які вони вносять у виставу, фактично не надсилаючи ці зміни для постійного рендерингу;
- Компонентний. Кожна частина програми або веб-сторінки Vue.js діє як окремий компонент. Перевага тут у тому, що ви можете повторно використовувати компоненти на власний розсуд. В цілому, такий підхід до розробки забезпечує кращу читаність коду, оскільки буде менше безладу. Простіше модульне тестування є додатковою перевагою, тому що ви можете перевірити, як найдрібніші частини програми працюють самі по собі;
- Vue.js має велику та активну спільноту. І останнє, але не менш важливе: ця спільнота заповнена талановитими розробниками Vue.js, а це означає, що вам легко знайти таланти для вашої команди розробників програмного забезпечення;

- реактивність. Vue.js – це реактивний MVC-фреймворк. Подання (view) змінюється у міру зміни моделі;
- прогресивність. Ядро Vue.js ідеально підходить для впровадження у існуючий проект. Так, сайт готового продукту може продовжувати працювати, наприклад, на jQuery (раніше використовуваній бібліотеці), але частина модулів поступово переписуватиметься на Vue до повноцінного переходу;
- простота. Почати працювати з фреймворком можна без базових знань у веб-розробці. Низький поріг входження — причина популярності у розробників-початківців;
- невелика вага. Фреймворк займає близько 20 кБ, тому реалізовані у ньому проекти швидше завантажуються і краще ранжуються пошуковими роботами;
- швидка розробка. Шаблони, безліч документації та інструкцій, широке співтовариство ентузіастів дозволяють вирішити будь-яку проблему, що виникає під час створення проектів на Vue.js [50-52].

3.3.2. Огляд фреймворка React

React JS і React Native – два досить популярні інструменти, які використовуються для програмування безлічі програм і веб-сайтів.

Що ще цікавіше, кожна з їхніх унікальних особливостей – це їхня невідома історія. У 2011 році Facebook усвідомив, що їм потрібна більш нова модель з розширеним інтерфейсом користувача, щоб задовольнити зростаючі потреби програми.

Постійні оновлення та розробки вимагали перебудови всієї програми, тому Джордон Уолк, інженер Facebook, вигадав прототип React JS. Це нововведення змінило правила гри компанії. Пізніше, в 2015 році, вони випустили React Native, який створив вражаючий фундамент для розробки програм як для смартфонів, так і для веб-сайтів.

React використовують для створення інтерфейсів односторінкових та багатосторінкових програм, розробки великих сайтів. Наприклад, з його допомогою написано стрімінговий сервіс Netflix та реалізовано стрічки новин найбільших соціальних мереж.

Фреймворк призначений:

- для створення функціональних інтерактивних веб-інтерфейсів, працюючи з якими, не потрібно постійно оновлювати сторінку;
- швидкої та зручної реалізації окремих компонентів та сторінок цілком – елементи в React легко використовувати повторно;
- легка розробка складних програмних структур — їх просто описувати, якщо використовувати реалізований у React підхід;
- доопрацювання нової функціональності інтерфейсу з будь-яким початковим стеком технологій: фреймворк не залежить від решти інструментарію і добре працюватиме, на чому б не був написаний код;
- розробки односторінкових та багатосторінкових додатків (SPA та PWA). Це сайти, які функціонують як програми та веб-сервіси та мають відповідний інтерфейс;
- роботи із серверною частиною сайту або розробки інтерфейсів мобільних додатків. У таких випадках React використовують спільно з інструментами, що адаптують веб-технології для інших цілей.

React JS – це бібліотека JavaScript з відкритим вихідним кодом, яка використовується для створення інтерфейсів для додатків, особливо для Інтернету. Ви можете використовувати функцію під назвою «компоненти», яка є невеликою частиною коду, для створення складних інтерфейсів користувача. React JS в основному включає два сегменти: один з них – це компоненти, які включають фрагменти HTML-коду, видимі у ваших інтерфейсах, а інший сегмент – це HTML-документ, в якому відбуватиметься процес рендерингу компонентів, які ви використовуєте.

React JS простий та зручний в експлуатації. Як тільки ви оволодієте React JS і познайомитеся з його основами, вам не доведеться витрачати багато часу на його вивчення. Розуміння основних вимог React JS практично не потребує часу та може бути виконане протягом кількох годин.

Переваги React JS:

- створення складних веб-додатків – це просто. Використання HTML для створення динамічних програм – складне завдання, оскільки вимагає складного кодування. Однак з React JS кодування може працювати без будь-яких незручностей;
- компоненти можна використовувати повторно. React JS має безліч компонентів з різними елементами керування та логікою. Ви можете легко повторно використовувати ці компоненти, де вони потрібні. Завдяки цьому розробка та супровід ваших програм стає набагато простіше;
- підвищена продуктивність. Віртуальна модель DOM має багато переваг, що підвищують продуктивність. Які б компоненти React ви не написали, вони перетворяться на цю віртуальну DOM, а не на фізичну DOM. Таким чином, продуктивність буде відносно вищою;
- доступність SEO-дружніх інструментів. Розробники мають доступ до багатьох інструментів React JS. Ви можете працювати з певними компонентами React, вивчати, а також редагувати їх з точністю та легкістю. У той час, як консервативним фреймворкам JavaScript складно впоратися з SEO, React JS пропонує підтримку для ефективної роботи серед пошукових систем. Це доступно завдяки віртуальній моделі DOM, яка відображає та повертається до браузера як звичайну веб-сторінку.

Недоліки React JS:

- React JS фреймворки мають тенденцію регулярно змінюватись та розвиватися. Темпи розробки React JS дуже великі, що ускладнює відстеження постійних оновлень розробникам. Це також виявилося для багатьох незручністю;

- задовільна документація. Через швидкий темп оновлень React JS немає можливості для точної документації. Розробники повинні стежити за оновленнями та самі писати інструкції;
- відсутність повних наборів інструментів. React JS фокусується тільки на рівнях інтерфейсу розроблюваних додатків. Розробники повинні використовувати інші фреймворки, щоб отримати доступ до інших наборів інструментів;
- складна навігація за даними У React JS важко переміщатися між даними. React JS не підтримує паралельну обробку даних, і вам потрібно знайти батьківський вузол, щоб перейти до наступної ієрархії вузла дерева.

React Native, що з’явився після React JS, є JavaScript-фреймворком з відкритим вихідним кодом . Розробники використовують React Native для розробки мобільних та веб-додатків, що підходять для iOS, Android, а також Windows. Він орієнтований на розробку кросплатформових мобільних програм лише з використанням JavaScript. У цьому відношенні React Native та React JS дуже схожі, але React Native використовує власні компоненти як будівельні блоки. Більше того, React Native найкраще підходить для мобільних пристроїв, а не для інтернету.

Переваги React Native:

- програми, розроблені на React Native, добре працюють як із системами Android, так і з iOS. React Native пропонує кроссплатформенну доступність, і це найкраща частина;
- живе перезавантаження. Цікавою особливістю React Native є живе перезавантаження. Будь-які зміни, які ви вносите до коду програм, будуть змінені в процесі розробки. Будь-які зміни в логіці фіксуються на екрані і ви можете побачити ці зміни практично відразу;
- висока продуктивність. React Native повністю задовольняє потреби розробників, які працюють над мобільними програмами. Він забезпечує

основу підвищення продуктивності з допомогою використання графічних процесорів (GPU);

- економічний. Подібно до React JS, React Native також має функцію повторного використання компонентів коду. Повторне використання означатиме, що це заощадить багато часу та грошей, необхідних для розробки програми з нуля;
- спільнота. React Native був розроблений на хакатоні спільноти розробників. Це означає, що фреймворк постійно розвивається та підтримується спільнотою. Присутність команди допомагає поєднати знання різних умів із фреймворком.

Недоліки React Native:

- React Native все ще перебуває на стадії розробки, тому багато хто називає його незрілим. Хоча React Native постійно вдосконалюється, існує ймовірність зниження продуктивності вашої програми.
- важко вчитися. Якщо ви новачок у цій галузі як розробник додатків, React Native може бути складнішою структурою для вивчення.
- відсутність надійного захисту. React Native розглядається як не дуже безпечний фреймворк, якщо ви намагаєтеся розробляти програми, пов’язані з фінансами, банківською справою або скрізь, де є особисті дані.
- ініціалізація займає більше часу. Ініціалізація середовища виконання для просунутих пристроїв займає більше часу з React Native, що багатьом незручно.
- має проблеми з ліцензуванням та патентами. Оскільки React Native контролюється Facebook, постійно виникають проблеми з ліцензіями та патентами. Facebook має право зупинити розробників, якщо він не схвалює програму або у разі будь-яких порушень.

Порівняння React JS та React Native, яка в них різниця:

- базова похідна. React JS можна спочатку розглядати як базову похідну від React DOM. React Native, з іншого боку, сам собою може розглядатися як базова похідна. Це означає, що при зміні компонентів синтаксис та робочий процес не змінюються;
- мета кожного. React JS спочатку являє собою бібліотеку JavaScript, яка допомагає розробникам програмувати рівні інтерфейсу користувача з підвищеною продуктивністю і динамікою. React Native, з іншого боку, сам по собі є закінченим фреймворком, який корисний для створення кросплатформових додатків як для мобільних пристроїв, так і для Інтернету;
- відображення коду браузера та розробок.. Для рендерингу браузера React JS використовує функцію віртуальної DOM. У React JS розроблені програми відображають HTML-код в інтерфейсі користувача. React Native використовує власні API-інтерфейси для рендерингу компонентів у мобільному додатку. У React Native рендеринг інтерфейсу користувача відбувається за допомогою JSX, який є абревіатурою JavaScript;
- стиль та анімація. React JS використовує CSS для створення стилів для компонентів кодування, а анімація в React JS створюється за допомогою CSS аналогічно веб-розробці. React Native використовує таблицю стилів для стилізації. У React Native вам необхідно використовувати анімований API для включення анімації до різних компонентів програми;
- придатність. React JS – найкращий вибір для розробки веб-додатка з підвищеною продуктивністю, швидкодією та динамічною функціональністю. Беручи до уваги, що якщо ви хочете створити мобільний додаток з високою продуктивністю, то React Native має бути вашою головною перевагою.

React JS та React Native необхідні розробникам додатків. Їхні функціонуючі системи постійно розвиваються, та їх гнучкість подобається декільком розробникам. Так чи інакше, і React JS, і React Native оптимальні для певних типів програм. У той час як React JS – це скоріше бібліотека JavaScript, React

Native – це сам по собі цілий фреймворк. Це робить React JS вихідною точкою та серцем Native. Таким чином, обидва здаються такими, що доповнюють один одного.

Обидві структури мають свої сильні сторони та обмеження. У середовищі, де технології розвиваються стабільними темпами, ви будете засипані поліпшеннями та оновленнями, які іноді можуть завдавати клопоту. Тим не менш, не можна очікувати, що все працюватиме ідеально. Все, що вам потрібно, це достатньо навичок і знань, і все готове [53-56].

3.3.3. Огляд фреймворка Angular

AngularJS – це структура, що використовує мову програмування JavaScript. AngularJS – це інструмент розробки інтерфейсу з відкритим вихідним кодом, створений і підтримуваний Google. AngularJS корисний для тестування та розробки клієнтських додатків.

Складові фреймворку Angular:

- компоненти. Компоненти – це великі складові програми, які не залежать один від одного. Наприклад, один компонент – це стрічка новин, інший – шапка сайту. Додаток будується з них, як із блоків. Зазвичай кожен компонент зберігається у окремому файлі. Для нього можна створити свої HTML-шаблон та CSS-стилі. Вони можуть бути в тому ж файлі, що і компонент, а можуть підключатися окремо. Створюється готовий блок інтерфейсу зі структурою, стилями та певною логікою поведінки;
- модулі. Це також складові частини програми, але інші. Вони керують компонентами. Якщо компонент — це область програми, модуль відповідає за керування нею. Точка входу в програму, код для анімації або навігації – це всі модулі.

Головний модуль є у кожному проекті. Додаткові додаються в міру потреби та виконують конкретні завдання. Вони потрібні, щоб не перевантажувати основний модуль зайвою функціональністю та не робити його надто громіздким.

- форми. Більшість додатків на Angular – form-based, тобто засновані на формах. Форма – це структура, в яку користувач вводить якісь дані, а потім відправляє їх на сервер. Блок для написання коментаря чи зворотний зв’язок — це форма. Angular робить роботу з формами простішим: їх не доводиться писати з нуля. Для них вже створені шаблони, які потрібно адаптувати до нового завдання;
- сервіси. Вони схожі на компоненти, але більш вузькоспеціалізовані. Вони можуть визначатися як на рівні модуля, так і на рівні компонента або програми. У сервісах реалізується спеціальна логіка. Вони підключаються до програми як звичайний клас і використовуються для зберігання глобального стану програми. Також використовуються як постачальник даних;
- директиви. Це складові програми, які змінюють структуру або поведінку сторінки. Компоненти також належать до директив. Але крім них існують ще два види: структурні директиви та директиви, що змінюють зовнішній вигляд чи поведінку елементів. Вони потрібні, щоб застосувати одну дію до всіх екземплярів одного компонента, наприклад, зміна валюти у всіх картках товару.

Переваги Angular:

- велика кількість можливостей. Angular допомагає прив’язувати компоненти програми один до одного, передавати дані, анімувати інтерфейси та ін. Для простих проектів його функціональність може бути надмірною, але для складних SPA-додатків вона незамінна;
- універсальне застосування. Фреймворк дозволяє створювати не лише веб-застосунки. З його допомогою можна писати код, який може бути адаптований до іншого середовища. Наприклад, програма зможе працювати в мобільній або десктопній операційній системі. За допомогою Angular можна створити навіть програму для доповненої реальності;
- детальний style guide. Особливість Angular – докладна документація. Вона містить рекомендації до побудови та розробки додатків, style guide – гайд

за стилем програмування на Angular. Це зручно для розробників, які вперше зіштовхнулися із фреймворком. Єдність стилю допомагає програмістам краще розуміти код один одного;

– підтримка від Google. Розробники Angular – співробітники Google, а підтримка великої корпорації допомагає фреймворку розвиватися. При цьому завдяки вільній ліцензії та відкритому вихідному коду розвивати його можуть і сторонні розробники;

Недоліки Angular:

– складність у вивченні. Angular вважається одним із найскладніших фронтенд-фреймворків. Його можливо нелегко вивчити з нуля самостійно. Крім того, для початку роботи потрібно знати не тільки "чистий" JavaScript, але і TypeScript, який на ньому заснований;
– відсутність сумісності між старою та новою версіями. Незважаючи на схожі назви, AngularJS та Angular несумісні та принципово різні. Тому розробникам, які стикаються з legacy-кодом на AngularJS, потрібно вивчити основи роботи із застарілим фреймворком. Концепції та правила нового Angular не підійдуть.

10 кращих фреймворків AngularJS:

– Mobile Angular UI. Мобільний інтерфейс користувача Angular допомагає розробникам створювати мобільні програми HTML5 з використанням Bootstrap і AngularJS. Він пропонує основні мобільні компоненти та надійні бібліотеки в якості мобільної платформи інтерфейсу користувача. Mobile Angular UI містить перемикачі, бічні панелі, накладання, області, що прокручуються, і розташовані зверху/знизу панелі навігації. За допомогою простих директив AngularJS мобільний інтерфейс Angular спрощує створення мобільного додатка. Інтерфейс користувача слідує синтаксису Bootstrap 3, який спрощує перетворення настільного веб-додатка в мобільний. У комплект входить невеликий файл CSS для

повністю адаптивного інтерфейсу користувача з сенсорним екраном для мобільних додатків;

- Ionic — це адаптивний набір для розробки програмного забезпечення з відкритим кодом, який використовує веб-технології для створення нативних мобільних додатків. Ionic – це кроссплатформенний фреймворк для розробки гібридних мобільних додатків. Ionic потребує AngularJS, щоб використати весь свій потенціал для спрощення процесу розробки інтерфейсу. Ionic створений, щоб допомогти розробникам перетворити потужні програми HTML5 для мобільних платформ. Ionic підтримує такі технології, як Cordova, Phonegap, Trigger.io та ElectronJS. Ionic використовує компоненти, директиви та розширення AngularJS для створення прогресивних нативних та гібридних додатків;
- Angular Material – це бібліотека компонентів інтерфейсу користувача, розроблена Google для проектів Angular. Angular Material допомагає розробникам Angular прискорити розробку зрозумілих, послідовних та інтерактивних інтерфейсів користувача. Як фреймворк Angular CSS, він складається з шаблонів, панелей адміністратора і більше 30 компонентів інтерфейсу користувача. AngularJS Material містить інструменти для налаштування компонентів CSS з типографікою, кнопками, прапорцями, темами та багатьма іншими. Бібліотека матеріалів AngularJS дозволяє розробникам створювати веб-програми, що відповідають традиційним методам проектування, незалежності від пристроїв та переносимості через браузер;
- Bootstrap — це безкоштовне інтерфейсне середовище з відкритим вихідним кодом, яке включає компоненти, макети та інструменти JS. Angular UI Bootstrap – це інфраструктура AngularJS, розроблена за допомогою Bootstrap. Для роботи Bootstrap потрібний jQuery. Використання Bootstrap безпосередньо з AngularJS конфліктує з деякими функціями Angular. Використання Angular UI Bootstrap усуває залежність від jQuery, що дозволяє використовувати функції Bootstrap у коді

AngularJS. Фреймворк Angular UI Bootstrap спирається на деякі вбудовані директиви AngularJS, засновані на CSS, і включає деякі елементи розмітки Bootstrap;

- UI Grid раніше був відомий як "ng-grid". Angular UI Grid не залежить від jQuery, тому що він повністю написаний на AngularJS. UI Grid має центральний модуль сітки, і всі його риси визначають модулі та директиви Angular. У Angular UI Grid є компонент, заснований на Angular UI Suite та Angular UI Bootstrap. Компонент відображає елементи даних всередині мереж, розроблених за допомогою AngularJS. Використання UI Grid спрощує виконання таких дій, як угруповання, сортування, віртуалізація, фільтрація, експорт, редагування у сітці та багато іншого. Розробники можуть керувати величезними наборами даних за допомогою Angular UI Grid. Це також допомагає для віртуалізації розширення рядків, закріплення стовпців та інтернаціоналізації;
- LumX – це інтерфейсна структура, розроблена за допомогою матеріального дизайну Google. Він дотримується рекомендацій Google і дозволяє розробникам створювати інтерфейси користувача, що відповідають керівним принципам Google. LumX спрощує керування показниками Google при розробці інтерфейсу користувача. LumX використовує AngularJS для створення веб-сайтів із шаблоном MVC (модель-представлення-контролер). LumX використовує jQuery для підвищення продуктивності веб-програми. LumX працює як препроцесор Sass і використовує такі технології, як Bourbon та Gulp, для покращення налаштування та продуктивності;
- Supersonic – це гібридне середовище інтерфейсу користувача, створене на основі Ionic. Supersonic поєднує веб-компоненти з JavaScript і CSS для створення власних інтерфейсів на основі AngularJS і HTML5. У Supersonic є компоненти CSS для мобільних пристроїв і компоненти власного інтерфейсу користувача зі стилями на основі CSS. Декларативний інтерфейс користувача спрощує інтеграцію API в компоненти веб-

додатків. Використання Supersonic підвищує швидкість розробки та проектування, оскільки зміни інтерфейсу користувача не вимагають перекомпіляції;

- Radian – це масштабована, динамічна, потужна та надійна платформа AngularJS. Radian дозволяє дизайнерам та розробникам вбудовувати графіки безпосередньо на веб-сторінки на основі HTML. Radian налаштовує HTML елементи для побудови графіків і графіків на основі параметрів. Radian використовує фреймворк AngularJS для реалізації HTML-елементів. Radian пов'язує атрибути HTML та JavaScript за допомогою бібліотеки побудови графіків D3.js. Браузери відображають елементи SVG для відображення графіків на веб-сторінці;
- Suave UI – це простий і зручний у використанні фреймворк, що використовує AngularJS для розробки інтерфейсів для веб-додатків. Визначення CSS, такі як кольорові кнопки, багаторівневі елементи, сітки, форми та багато іншого включені в Suave UI. Suave UI надає бібліотеку компонентів для розробки інтерфейсу користувача у вигляді пов'язаних файлів. Це знижує зусилля розробника щодо включення окремих компонентів. Одна команда з Bower може використовуватися для одночасного включення всіх компонентів;
- Angular Foundation – один із найпросунутіших інтерфейсних фреймворків. Angular Foundation – це комбінація кількох фреймворків, які використовуються для розробки веб-сайтів, програм та електронних листів для всіх платформ. Angular Foundation є зручним для розробки веб-сайтів, що включають семантичні компоненти HTML. Фреймворк Angular Foundation є гнучким, настроюваним і легко читається [57-62].

3.4. Огляд фреймворків для розробки серверної частини

Для правильного вибору фреймворку необхідно проаналізувати найбільш популярні рішення, що часто використовуються в реальних проектах. Основний упор при виборі фреймворку робиться на функціональність, поширеність,

наявність зрозумілої документації та навчальних матеріалів, можливість підтримки.

PHP є однією з найпопулярніших мов програмування для веб-розробок по всьому світу. PHP часто використовують у великих комерційних проектах. Інтернет-магазин техніки Ельдорадо, наприклад, використовує PHP для збереження та створення своїх внутрішніх систем. В даний час PHP використовує понад 82% сайтів [63].

Фреймворк (від англ. framework – каркас, оболонка, структура) – це платформа, що є програмою, що визначає структуру програмної системи; крім того, це програмне забезпечення, яке полегшує розробку та об’єднання різних компонентів одного великого програмного проекту.

У цій роботі будуть розглянуті три найбільш популярні фреймворки: Symfony, Laravel та Yii2.

Для правильного вибору фреймворку потрібно розглянути кожен окремо, розібрати подібності, відмінності, документацію, продуктивність, функціонал, зручність використання.

Спільні риси у даних фреймворків:

- наявність відкритого коду;
- наявність повноцінного функціоналу для реалізації серверної та клієнтських частин;
- підтримка ORM (Object Relationship Mapping);
- зрозуміла документація та наявність навчальних матеріалів.

Продуктивність та швидкість обробки запитів не завжди є найважливішим елементом. Вона важливіша, коли йдеться про обробку величезної кількості важливих даних. З цієї точки зору, Yii більше підходить. Laravel з представлених рішень є найповільнішим, однак сайти, що працюють на даному фреймворку, задовольняють усі запити користувача за швидкістю.

Кожен фреймворк має свої переваги та недоліки розробки.

У Symfony масштабна та сильна спільнота, Yii буде корисний для швидкого написання коду. Тут же більше підійде Laravel, тому що він найпростіший в освоєнні, має багато навчальних матеріалів і відмінно підходить для старту діяльності у веб-розробці.

Якщо говорити про можливість роботи з базами даних, то тут найкращим рішенням буде Symfony, він підтримує величезну кількість баз даних, у тому числі непопулярні GemFire, CouchDB. Laravel та Yii підтримують меншу кількість баз даних, проте обидва фреймворки взаємодіють з MySQL, здатною покрити більшість потреб.

Таблиця 3.1 – Бази даних які підтримують фреймворки, що розглядаються

Framework	Laravel	Yii	Symfony
Database	SQLite MySQL PostgreSQL Redis Microsoft BI MongoDB	MySQL SQLite Microsoft BI Oracle PostgreSQL MongoDB	Microsoft BI MongoDB MySQL N oSQL PostgreSQL CouchDB DynamoDB GemFire GraphDB MemBase MemCacheDB Oracle Apache Jackrabbit

При виборі фреймворку важливим пунктом є також наявність сильної спільноти та матеріалів для навчання. Symfony має сильну спільноту, яка постійно зростає та розвивається. Laravel виділяється величезною кількістю курсів, відео, статей для різних рівнів знань, що дозволяє вивчати його швидко та ефективно.

3.4.1. Laravel

Laravel – безкоштовний веб-фреймворк з відкритим кодом, призначений для розробки з використанням архітектурної моделі MVC. Laravel випущено під ліцензією MIT. Вихідний код проекту розміщується на GitHub. Система шаблонів Laravel Blade на відміну інших систем шаблонів дозволяє

використовувати PHP код в уявленнях. Крім того, він не погіршує продуктивність додатків, тому що файли шаблони зберігаються у .blade.php extension. Весь код перетворюється на чистий код PHP під час обробки програми [64].

Крім того, Laravel має чималу кількість ресурсів для навчання. Інформації в інтернеті достатньо, щоб самостійно вивчити фреймворк. Це одна з головних його переваг.

Особливості:

- велика кількість ресурсів на навчання;
- підтримує Composer для керування пакетами;
- багато пакетів та патчів для збільшення функціоналу;
- найпопулярніший фреймворк у 2015-2016.

3.4.2. Yii2

Yii – це популярний фреймворк для php-розробки, заснований на парадигмі MVC. Основна перевага – дуже висока швидкість роботи та, як наслідок, продуктивність. Yii не має власної системи представлення за замовчуванням, але при цьому дозволяє використовувати сторонні шаблонування, такі як Twig та Smarty. Таким чином при початковому використання Symfony, можна продовжити роботу з наступним проектом використовуючи Yii [65].

Однак незважаючи на високу продуктивність, фреймворк включає в себе велику кількість коду, що дає неабияке навантаження на сервер.

Особливості:

- підтримка Ajax;
- має функціонал роботи з формами, забезпечує їх побудову та валідацію;
- висока розширюваність;
- велика спільнота.

3.4.3. Symfony

Symfony – вільний фреймворк, написаний на PHP, який використовує патерн MVC. У фреймворку Symfony дуже потужна функціональність, продумана архітектура, а також розвинена спільнота. Фреймворк безкоштовний та поширюється під ліцензією MIT. Symfony пропонує швидку розробку та управління веб-додатками, що дозволяє легко вирішувати рутинні завдання веб-програміста.

Працює лише з PHP 5 і вище. Має підтримку безлічі баз даних (MySQL, PostgreSQL, SQLite або будь-яка інша PDO-сумісна СУБД).

Система шаблонів Symfony Twig забезпечує чисту та лаконічну розробку на відміну чистого PHP.

Symfony працює на повторно використовуваних компонентах та забезпечує найкращу модульність. Він також використовує модель та контролер для розробки веб-застосунків, які можуть виглядати складно для багатьох новачків. Крім того, Symfony є гарним прикладом модульної структури. Можна використовувати 30 компонентів, наданих Symfony у проекті за модульним принципом [66].

Особливості:

- LTS;
- широкий функціонал;
- стабільність;
- модульність;
- велике комьюніті.

Для більш наочного відображення всіх переваг та недоліків фреймворків

та обґрунтування вибору було складено таблицю 3.2:

Таблиця 3.2 – Порівняння фрейморків

	Symfony	Yii2	Laravel
Функціональність	Має відкритий вихідний код. Дозволяє працювати з великою кількістю БД. Наявність власної системи шаблонів Symfony Twig. Використовує модульний підхід. Найбільш велика підтримка баз даних.	Має відкритий вихідний код. Відсутня власна система шаблонів. Використання MVC та Немодульних компонентів. Підтримується набагато менше баз даних по порівняно з Symfony. Є найбільш продуктивним фреймворком.	Має відкритий вихідний код. Наявність власної системи шаблонів Laravel Blade. Не використовує модульний підхід. Підтримується набагато менше баз даних у порівнянні з Symfony.
Простота освоєння	Для розробки веб-додатків використовуються модель, та контролер що викликає труднощі для новачків.	Є більша кількість посібників з опису кожної функції та посилання на клас.	Простий у освоєнні, так як. є більша кількість підручників у Інтернет. Є найбільш підходящим для тих хто має мінімальні знання в PHP
Активність використання у розробці веб-додатків	Активно використовується в розробці	Активно використовується в розробці	Активно використовується в розробці

Закінчення таблиці 3.2

	Symfony	Yii2	Laravel
Забезпечення довгострокової підтримки	У травні 2021 року вийшла версія з довгостроковою підтримкою на 3 року	Немає інформації про те, коли вийде нова версія з довгостроковою підтримкою	У квітні 2022 року вийшла версія laravel 9.10 з підтримкою на 3 роки
Підтримка з боку спільноти	Велика спільнота	Велика спільнота	Велика спільнота

3.5. Вибір системи управління базами даних

Система управління базами даних (скорочено СУБД) – це програмне забезпечення для створення та роботи з базами даних.

Головна функція СУБД – це управління даними (які може бути як у зовнішній, і у оперативної пам’яті). СУБД обов’язково підтримує мови баз даних, а також відповідає за копіювання та відновлення інформації після будь-яких збоїв.

Реляційні та об’єктно-реляційні СУБД є одними із найпоширеніших систем. Вони є таблицями, у яких кожен стовпець упорядкований і має певну унікальну назву. Послідовність рядків визначається послідовністю введення інформації в таблицю. При цьому обробка стовпців та рядків може відбуватися у будь-якому порядку. Таблиці з даними пов’язані між собою спеціальними відносинами, завдяки чому з даними з різних таблиць можна працювати – наприклад, поєднувати їх за допомогою одного запиту.

Для управління реляційними базами даних застосовується спеціальна мова програмування – SQL.

Найбільш правильний підхід при виборі СУБД заснований на оцінці того, якою мірою існуючі системи задовольняють основним вимогам створюваного проекту інформаційної системи.

Існує кілька критеріїв вибору системи управління базами даних:

- особливості розробки додатків;
- моделювання даних;
- вимоги до робочого середовища;
- контроль роботи системи;
- особливості архітектури і функціональні можливості;
- продуктивність;
- надійність;
- змішані критерії.

Далі коротко розглянемо найкращі СУБД, які найчастіше використовуються при створенні проектів:

1. MySQL – вільна система управління базами даних. Розробка та підтримка сайта MySQL здійснює корпорація Oracle, що має на даний момент права на торговельну марку. Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією. Крім цього розробники створюють функціональність за замовленням ліцензійних користувачів, саме завдяки такому замовленню майже в найраніших версіях з’явився механізм реплікації.

Основні переваги MySQL:

- підтримка декількох одночасних запитів;
- оптимізація зв’язків з приєднанням багатьох даних за один прохід;
- записи фіксованої і змінної довжини;
- ODBC драйвер;
- гнучка система привілеїв і паролів;
- гнучка підтримка форматів чисел, рядків змінної довжини і міток часу;
- інтерфейс з мовами C і Perl, PHP;
- швидка робота, масштабованість;
- сумісність з ANSI SQL;

- безкоштовна в більшості випадків;
- хороша підтримка з боку провайдерів послуг хостингу;
- швидка підтримка транзакцій через механізм InnoDB [67].

2. PostgreSQL – ця система управління базами даних, що вільно розповсюджується, відноситься до об'єктно-реляційного типу СУБД. Як і у випадку з MySQL, робота з PostgreSQL ґрунтується на мові SQL, проте, на відміну від MySQL, PostgreSQL підтримує стандарт SQL-2011. Ця СУБД немає обмежень ні з максимальному розміру бази даних, ні з максимуму записів чи індексів у таблиці.

Якщо говорити про переваги PostgreSQL, то в першу чергу це надійність транзакцій та реплікацій, можливість успадкування та легка розширюваність. PostgreSQL підтримує різні розширення та варіанти мов програмування, такі як PL/Perl, PL/Python та PL/Java. Також є можливість завантажувати C-сумісні модулі.

Багато хто відзначає, що на відміну від MySQL дана СУБД має хорошу та докладну документацію, яка дає відповіді практично на всі питання.

Про те, що це масштабніша, ніж MySQL, СУБД, говорить і той факт, що PostgreSQL періодично порівнюють з такою потужною системою управління даних, як Oracle. Все це дозволяє говорити про PostgreSQL як про одну з найпоширеніших СУБД на даний момент [68].

3. SQLite на даний момент це одна з найкомпактніших СУБД. Також вона є вбудовуваною та реляційною.

SQLite дозволяє зберігати всі дані в одному файлі і завдяки своєму невеликому об'єму відрізняється завидною швидкодією. SQLite значно відрізняється від MySQL та PostgreSQL своєю структурою: двигун та інтерфейс цієї СУБД знаходяться в одній бібліотеці – і саме це дозволяє виконувати всі запити дуже швидко. Інші СУБД (MySQL, PostgreSQL, Oracle тощо) використовують парадигму «клієнт-сервер», коли взаємодія відбувається через мережевий протокол.

З недоліків можна відзначити відсутність системи користувачів та можливості збільшення продуктивності [69].

4. Oracle – це система, що відрізняється стабільністю вже не один десяток років, тому її вибирають корпорації, для яких важлива надійність відновлення після збоїв, налагоджена процедура бекапу, можливість масштабування та інші цінні можливості. До того ж ця СУБД забезпечує відмінну безпеку та ефектний захист даних.

Ця СУБД належить до об’єктно-реляційного типу. Назва походить від назви фірми Oracle, що розробила цю систему. Поряд із SQL СУБД використовує процедурне розширення під назвою PL/SQL, а також мову Java.

На відміну від інших СУБД, вартість купівлі та використання Oracle досить висока, і саме це найчастіше є перешкодою до її використання в невеликих фірмах [70].

5. MongoDB – СУБД відрізняється тим, що вона призначена для зберігання ієрархічних структур даних, і тому її називають документоорієнтованою (вона є документним сховищем без використання таблиць або схем). MongoDB має відкритий вихідний код.

Використовуючи ідентифікатор, ви можете виконувати швидкі операції над об’єктом. Також ця СУБД добре показує себе і за складних взаємодій. У першу чергу йдеться про швидкодію – у деяких випадках додаток, написаний на MongoDB, працюватиме швидше, ніж така ж програма, що використовує SQL, т.к. MongoDB відноситься до класу СУБД NoSQL і користується об’єктною мовою запитів, яка значно легша за SQL [71].

Однак ця мова має свої обмеження, і тому MongoDB слід використовувати у випадках, коли немає необхідності в складних і нетривіальних вибірках.

На рисунку 3.2 зображено рейтинг СУБД 2022 року:

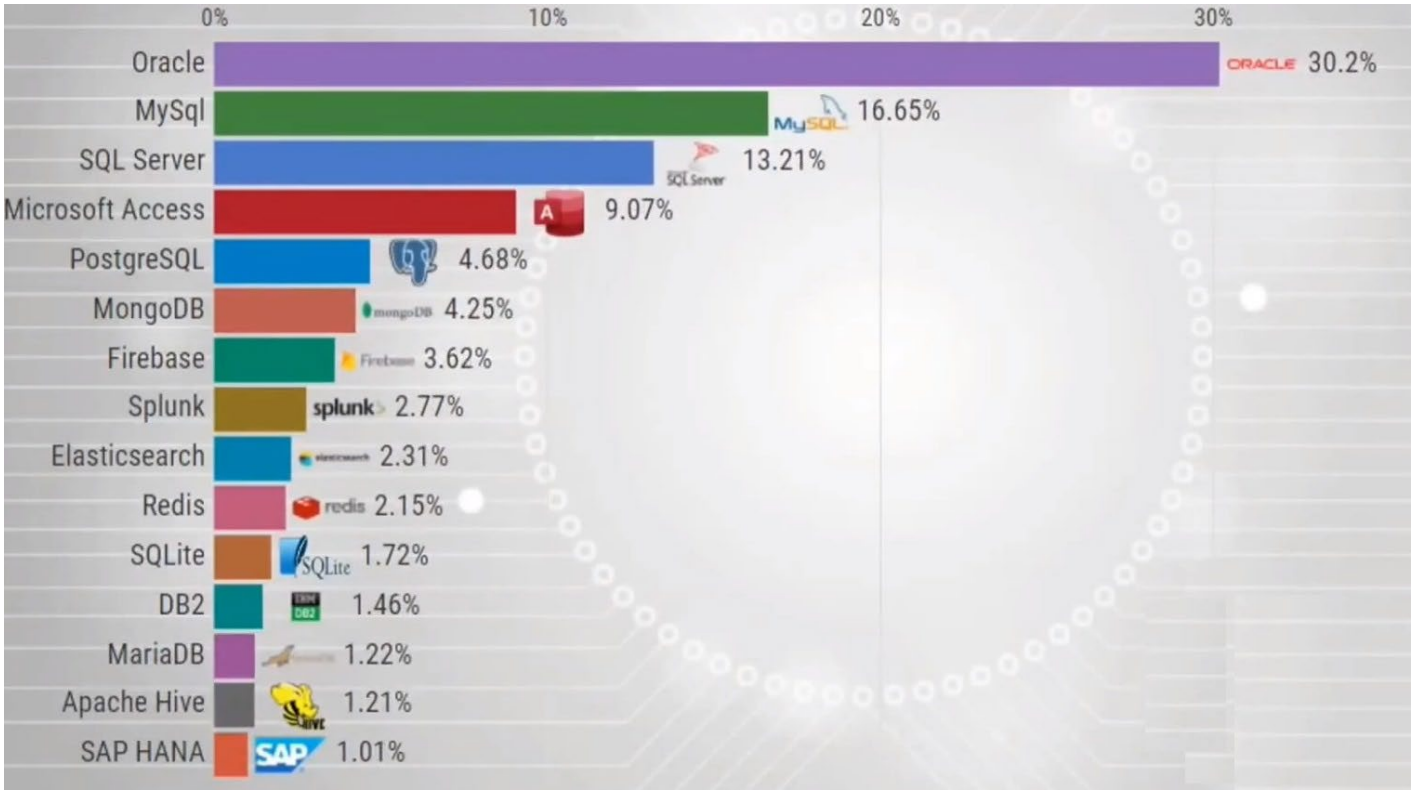

Рисунок 3.2 – Рейтинг СУБД 2022 року

У таблиці 3.3 наведено популярні системи управління базами даних:

Таблиця 3.3 – Популярні системи управління базами даних

БД	Розробник	Ліцензія	Написана на
Oracle	Oracle Corporation	Пропрієтарна	Assembly, C, C++
MySQL	Oracle Corporation	GPL v2 або пропрієтарна	C, C++
Microsoft SQL Server	Microsoft Corporation	Пропрієтарна	C, C++
PostgreSQL	PostgreSQL Global Development Group	Ліцензія PostgreSQL (безкоштовне програмне забезпечення з відкритим вихідним кодом, ліберальна ліцензія)	C
MongoDB	MongoDB Inc.	Різні варіанти ліцензування	C++, C, JavaScript
DB2	IBM	Пропрієтарна EULA	Assembly, C, C++
Redis	Salvatore Sanfilippo	Ліцензія BSD ANSI	C

Для проекту було обрано СУБД MySQL. Цей компонент зручний у роботі і широко застосовується у реальних комерційних проектах. СУБД дозволяє

працювати з різними таблицями. Можна взаємодіяти з MyISAM та InniDB таблицями. У MySQL постійно з’являються нові види таблиць, а також тут зручно працювати із транзакціями.

Проект має дві ліцензії. Відповідно до ліцензії GPL, розробники повинні надавати відкритий код своїх програм, проте не всі готові так чинити. Їх є комерційна ліцензія з урахуванням Oracle. Тут MySQL може тиражуватися разом із програмним забезпеченням, яке працює під ліцензією зі спеціального переліку Oracle.

MySQL лояльна до різних операційних систем, можна використовувати різні версії Windows, MacOS, Linux та інші. Також у СУБД є API для більшості мов програмування, включаючи PHP, Python, Java.

Таким чином, СУБД MySQL є оптимальним вибором для розробки проекту. Вона стабільно працює і може зберігати безліч даних. Це виявиться особливо корисним, якщо буде велика кількість користувачів.

РОЗДІЛ 4. ОПИС ПРОЕКТУВАННЯ ТА РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ ДЛЯ СТВОРЕННЯ ТА РЕДАГУВАННЯ ВЕКТОРНОЇ ГРАФІКИ

4.1. Вибір моделі розробки програмного засобу

Для розробки програмного забезпечення ми вибрали каскадну модель (Waterfall) – це модель процесу розробки програмного забезпечення, в якій процес розробки виглядає як потік, що послідовно проходить фази аналізу вимог, проектування, реалізації, тестування, інтеграції та підтримки.

Основна суть моделі Waterfall у тому, що етапи залежать один від одного і наступний починається, коли закінчено попередній, утворюючи таким чином каскадний рух уперед.

Паралелізм етапів у каскадній моделі, хоч і обмежений, але можливий для абсолютно незалежних робіт. При цьому інтеграція паралельних шматків все одно відбувається на наступному етапі, а не в рамках одного.

Команди різних етапів між собою не спілкуються, кожна команда відповідає чітко за свій етап.

Для замовників дана модель виглядає лінійно і з боку досить просто: із завершеного етапу проектування випливає програмування, а потім тестування – і так крок за кроком поки не буде досягнуто фінальної точки і мети, заради якої ведеться розробка.

Однак уявлення про простоту каскадної моделі є ілюзорним. Воно з’являється через обмежене не бачення клієнтом всього процесу, адже дана модель не передбачає залучення замовника до деталей процесів розробки, і демонструє зрозумілий і кінцевий результат роботи тільки на контрольних точках і наприкінці проекту.

Насправді каскадну модель не можна назвати простою, практично нею складно управляти. Внесення замовником значних змін у процес розробки з waterfall або опрацювання серйозних не передбачених проектом ризиків несуть руйнівний характер для всього процесу – модель доводиться перебудовувати, графіки перепланувати. Візуальний вигляд каскадної моделі зображено на рисунку 4.1:

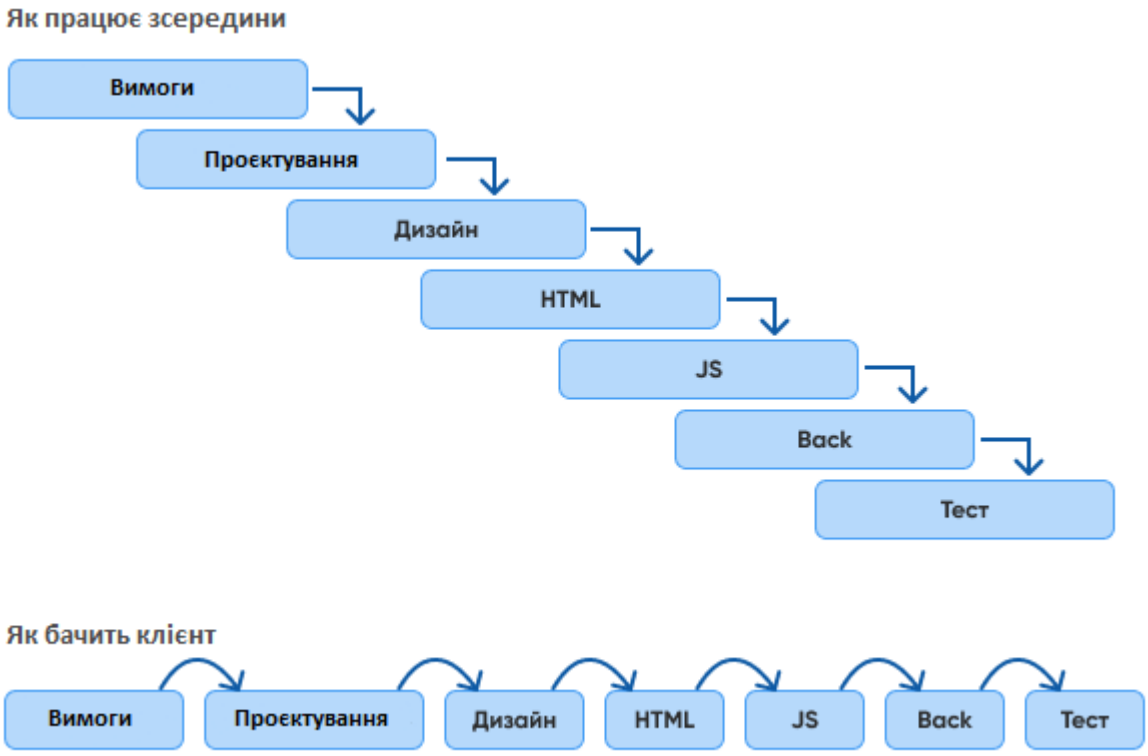

Рис. 4.1. Візуальний вигляд каскадної моделі

Наведемо переваги каскадної моделі:

– розробку легко контролювати. Замовник завжди знає, чим зараз зайняті програмісти, може керувати термінами та вартістю;
– вартість проекту визначається на початковому етапі. Усі кроки заплановані вже на етапі узгодження договору, ПЗ пишеться безперервно «від і до»;
– не потрібно наймати тестувальників із серйозною технічною підготовкою. Тестувальники зможуть спиратися на детальну технічну документацію.

Недоліки каскадної моделі:

– тестування розпочинається на останніх етапах розробки. Якщо в вимогах до продукту припустилися помилки, то виправити її коштуватиме дорого. тестувальники виявлять її, коли розробник уже написав код, а техніки – документацію;
– замовник бачить готовий продукт наприкінці розробки і лише тоді може дати зворотний зв’язок. Велика ймовірність того, що результат його не влаштує.

– розробники пишуть багато технічної документації, що затримує роботу. Чим більша документація у проекті, тим більше змін потрібно вносити та довше їх узгоджувати.

«Водоспад» підходить для розробки проектів у медичній та космічній галузі, де вже сформовано велику базу документів, на основі яких можна написати вимоги до нового ПЗ.

При роботі з каскадною моделлю основне завдання – написати докладні вимоги до розробки. На етапі тестування не має з’ясуватися, що у них є помилка, що впливає весь продукт [72].

4.2. Опис алгоритму розв’язування задачі

Візуальне моделювання в UML можна представити як деякий процес ступеневого спуску від найбільш загальній і абстрактній концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної інформаційної системи. Для досягнення цих цілей з автоматизації, спочатку будується модель у формі діаграми прецедентів тобто варіантів використання, яка описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування [73].

Таким чином, діаграма варіантів використання є вихідним концептуальним уявленням або концептуальною моделлю системи в процесі її проектування і розробки.

Далі слід врахувати, що кожен прецедент системи має певну поведінку, може знаходитись в певних станах, а також переходити з одного стану в інший. Тобто, він моделює всі зміни станів певного варіанту використання як його реакцію на зовнішні впливи.

В разі чого, для специфікації функціональності прецеденту, що має складну модель поведінки, програмну систему з автоматизації ремонтно-будівельної компанії представляється з точки зору теорії кінцевих автоматів, тобто у формі діаграми станів [74].

Модель ін’єкції залежностей зображено на рисунку 4.2:

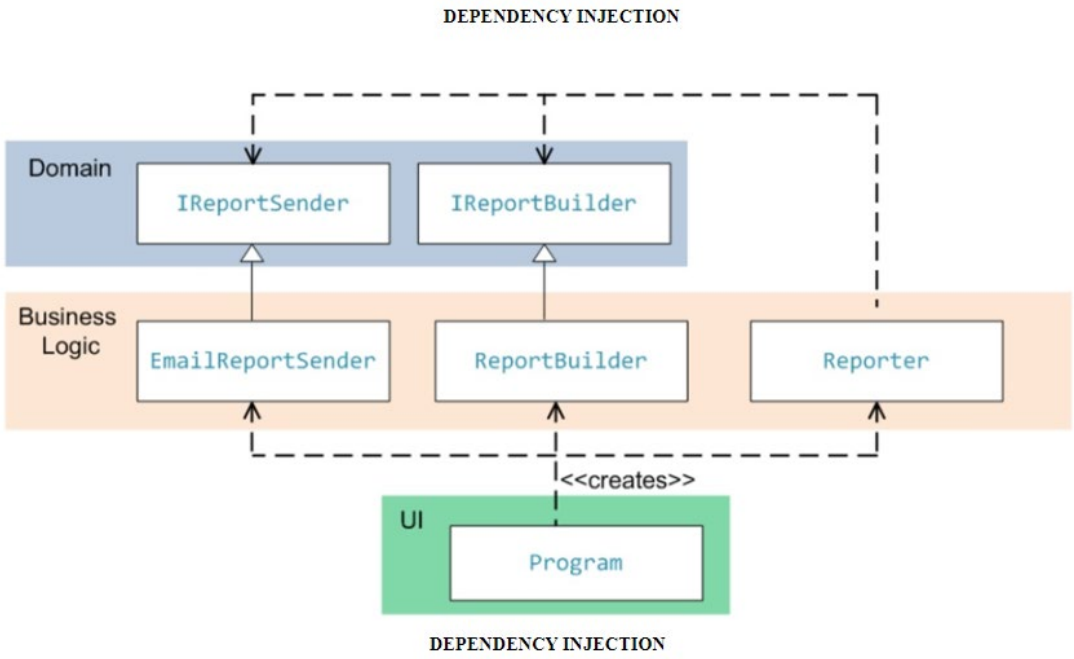

Рисунок 4.2 – Модель ін’єкції залежностей

Діаграми варіантів використання описують взаємини і залежності між групами варіантів використання і дійових осіб, які беруть участь в процесі див. рис. 3.2. Важливо розуміти, що діаграми варіантів використання не призначені для відображення проекту і не можуть описувати внутрішній устрій системи. Діаграми варіантів використання призначені для спрощення взаємодії з майбутніми користувачами системи, з клієнтами, і особливо знадобляться для визначення необхідних характеристик системи. Іншими словами, діаграми варіантів використання говорять про те, що система повинна робити, не вказуючи самі застосовувані методи.

Загальна діаграма варіантів використання зображена на рисунку 4.3:

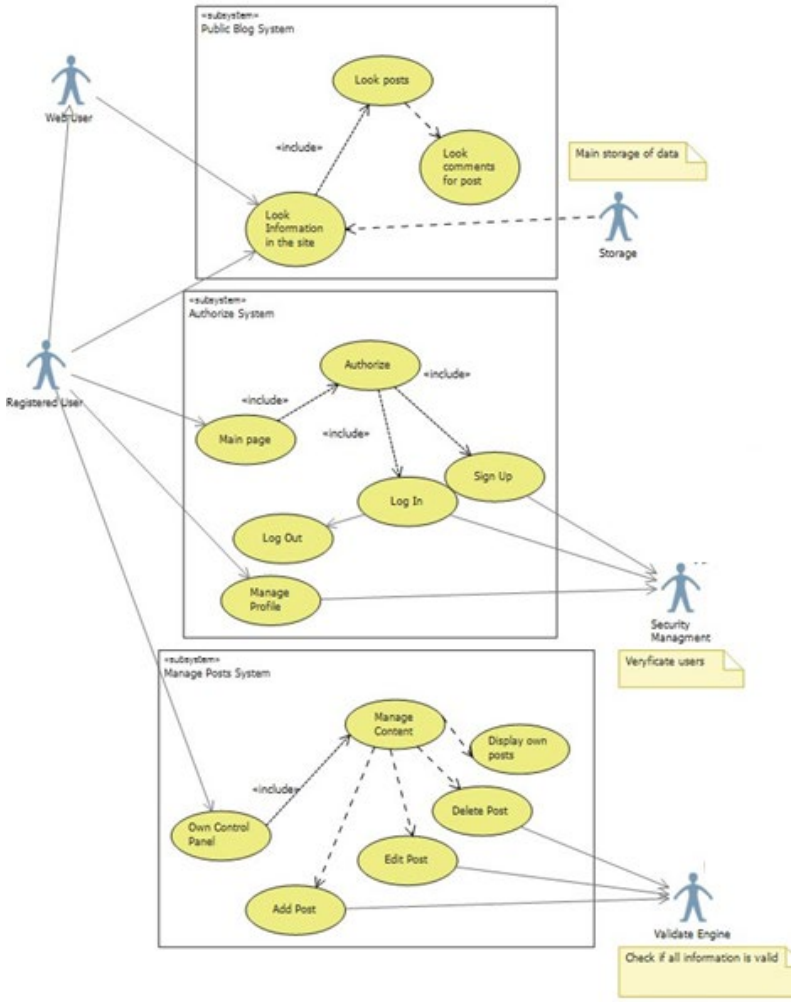

Рисунок 4.3 – Загальна діаграма варіантів використання

При роботі з варіантами використання важливо пам’ятати декілька простих правил:

- кожен варіант використання відноситься як мінімум до одної дійової особи;
- кожен варіант використання має ініціатора;
- кожен варіант використання призводить до відповідного результату.

Варіанти використання також можуть взаємодіяти з іншими варіантами використання. Єдиний актор проектованої системи, буде взаємодіяти із всіма варіантами використання. У модель включено асоціації, що відповідають за напрямки передачі інформації між актором і варіантами використання. Діаграма послідовностей зображена на рисунку 4.5:

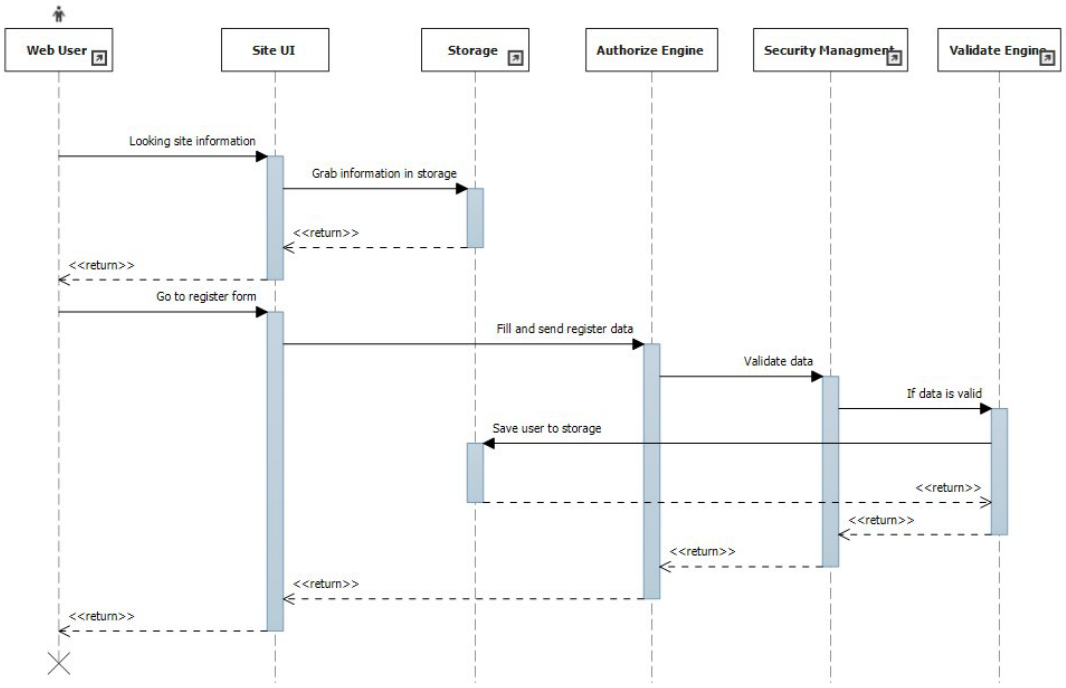

Рисунок 4.5 – Діаграма послідовностей

4.3. Основні режими функціонування програмного засобу

Головна сторінка програмного засобу зображено на рисунку 4.6:

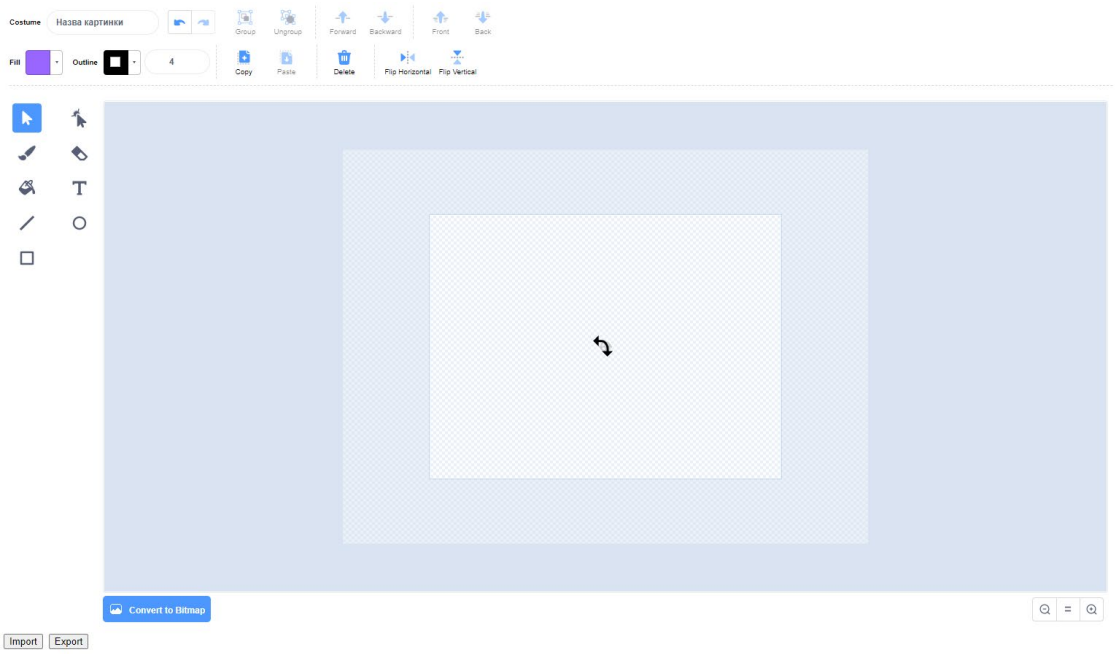

Рисунок 4.6 – Головна сторінка програмного засобу

Приклад створеної векторної картинки зображено на рисунку 4.7:

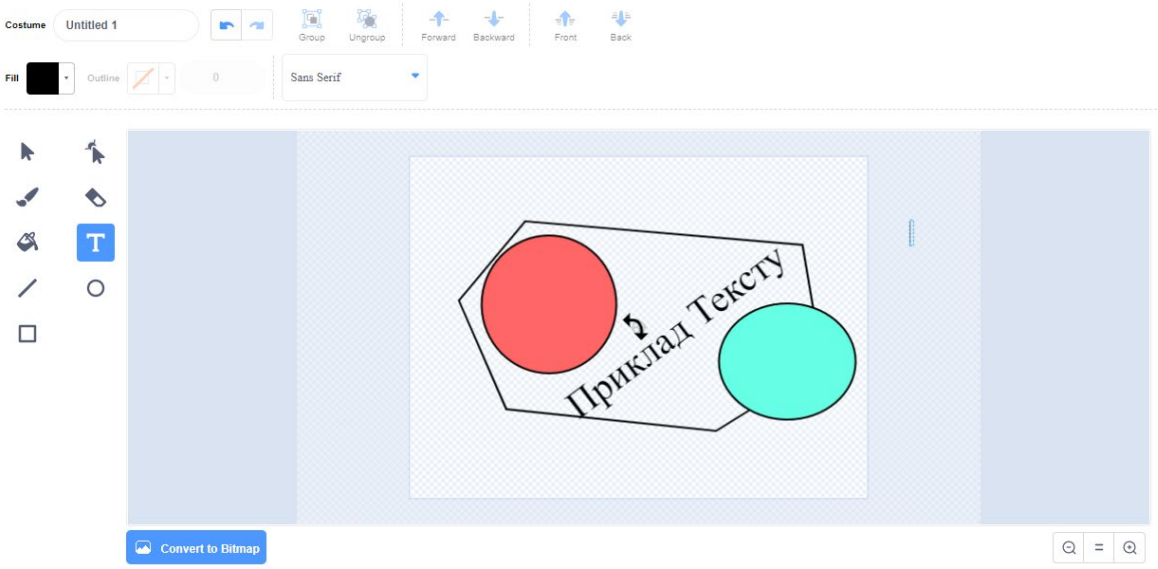

Рисунок 4.7 – Приклад створеної просто векторної картинки

Картинка у форматі svg і відкрита у Google Chrome зображена на рисунку 4.8:

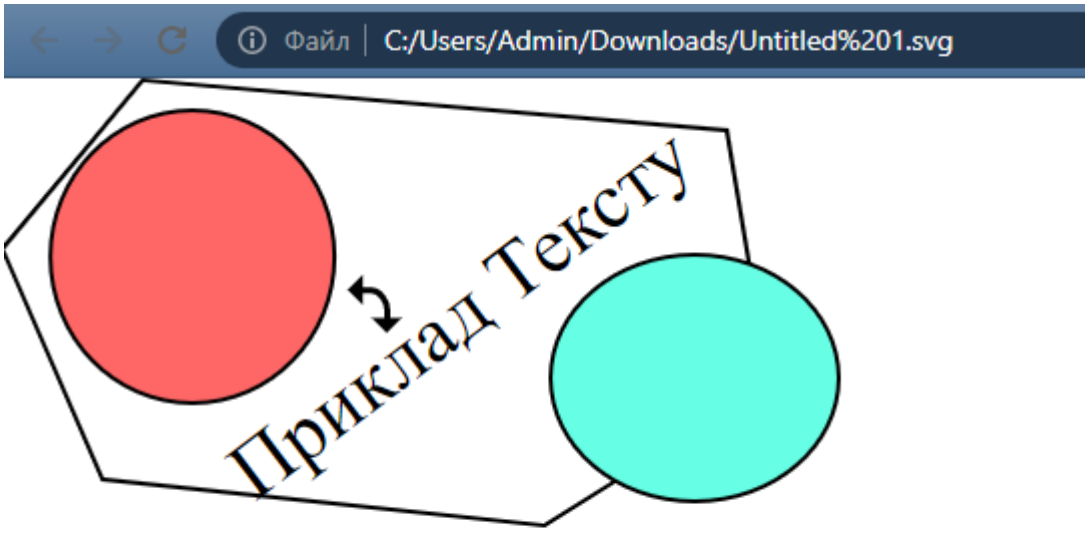

Рисунок 4.8 – Збережена картинка у форматі svg і відкрита у Google Chrome

Збережена картинка у форматі png і відкрита у Paint зображена на рисунку 4.9:

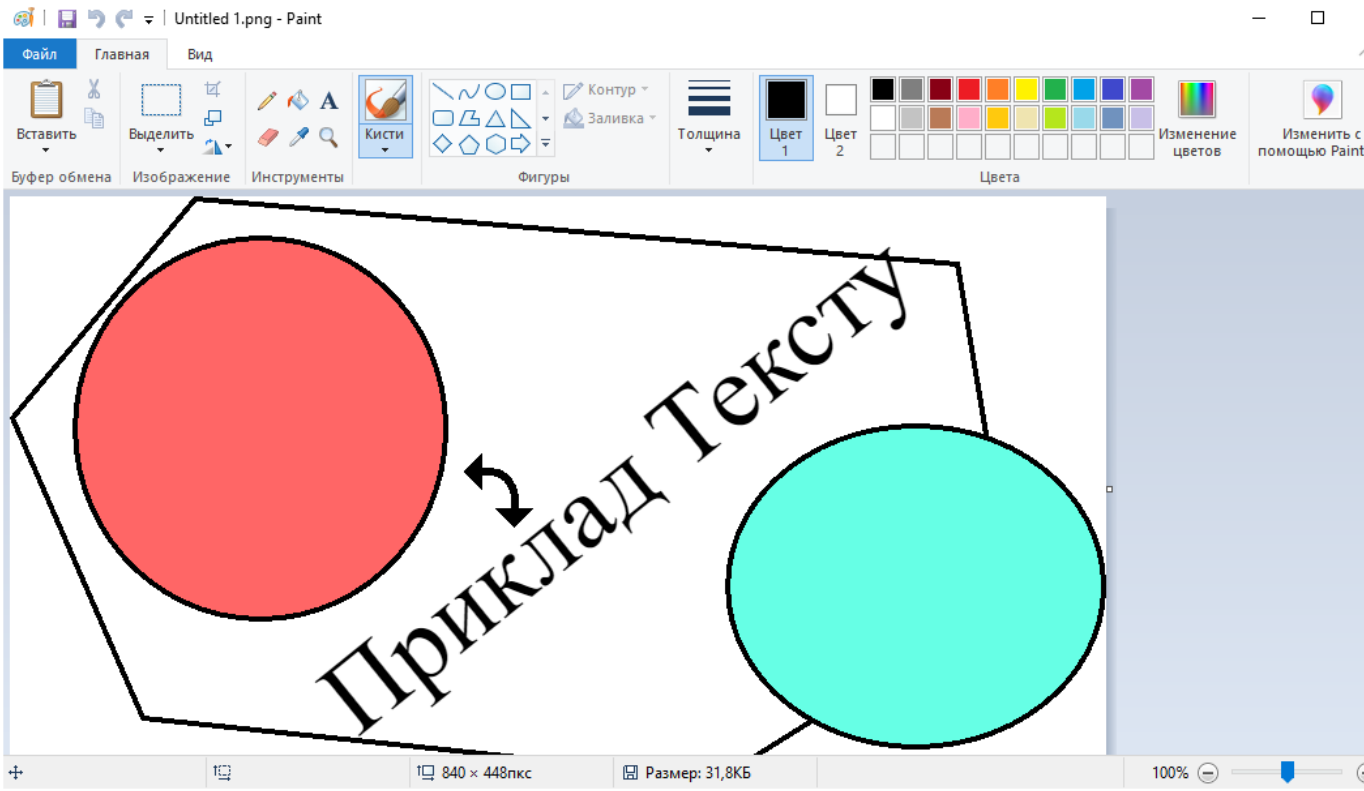

Рисунок 4.9 – Збережена картинка у форматі png і відкрита у Paint

4.4. Організація тестування веб-ресурсу

Тестування, як заключний етап розробки системи, виконує важливу роль в процесі створення високоякісного проекту. Після тестування замовнику надається готовий проект без помилок, з хорошою читабельністю, легкою логікою, зручністю і надійністю.

Для організації тестування системи розробили спеціальну методологію, згідно з якою була здійснена перевірка. Тестування можна проводити в різний спосіб, нижче наведено перелік правил, за допомогою яких провели тестування для програмного засобу.

Основні етапи:

Найперше було проведено функціональне тестування, тобто було перевірено, щоб кожна функція системи працювала відповідно до вимог технічного завдання.

Наступний етап тестування – це юзабіліті тестування, який призначений для оцінки системи з точки зору кінцевого користувача. Даний етап допомагає визначити відповідність продукту до очікувань користувачів, виявляє проблемні місця в інтерфейсі.

Навігаційне тестування:

- перевірили усі кнопки, форми і поля чи є зручними вони для використання;
- Тестування контенту:
- перевірили чи контент є інформативним, зрозумілим, структурованим і логічним.
- перевірили чи відсутні граматичні, орфографічні помилки.
- перевірили колірну палітру системи і розмір шрифтів.

Також було проведено тестування інтерфейсу користувача (UserInterface), яке виконується для перевірки відповідності графічного користувацького інтерфейсу системи до технічного завдання.

Було проведено тестування сумісності, яке виконується для перевірки роботи системи при різних програмних і апаратних конфігураціях. Кросплатформове тестування системи дозволяє оцінювати роботу сайту при різних ОС: Windows, iOS/Mac OS, Linux, Android, BlackBerry тощо.

Кросбраузерні тестування системи допомагає перевірити правильність роботи сайту в різних браузера: MozillaFirefox, GoogleChrome, Internet Explorer, Opera.

- перевірили чи навігація по системі є максимально простою;
- перевірили оптимізацію часу завантаження системи;
- перевірили чи кнопки мають достатній розмір для натиснення.

Бета-тестування – заключна стадія тестування. Як правило, це роблять кінцеві користувачі, які не є розробниками.

При бета-тестуванні система потрапляє в руки реальних користувачів, щоб виявити будь-які недоліки з їх точки зору, які виправляються для остаточної, релізної версії.

Висновки по розділу 4

Використовуючи зібрану інформацію та обрані проектні рішення було розроблено систему, яка повністю відповідає вимогам, описаним у специфікації вимог. Весь функціонал розробленої системи, описаний у специфікації вимог, успішно реалізований. Варто зауважити, що ідея створення даної сайту може активно розвиватись та доповнюватись новим функціоналом, що забезпечить системі подальший розвиток. Розроблена система задовольняє всім вимогам, поставленим на етапі постановки завдання.

ВИСНОВКИ

Усі галузі застосування – будь то інженерія та наука, бізнес та мистецтво/розваги – є сферою застосування векторної графіки. Зростаючий потенціал ПК забезпечує спокусливу основу капіталовкладень і зростання. І очікується стійке зростання промисловості у цій сфері до кінця цього десятиліття, особливо з огляду на те, що на початку цього десятиліття щорічне зростання становило близько 12%. Невідомо, як довго триватиме тенденція подвоєння капіталовкладень, особливо під впливом цін, проте очікується на стійке 10% щорічне підвищення в наступні 5 років. Звичайно, компанії продовжують формуватися, хоча інвестори зараз, здається, більше воліють вкладати гроші у програмне забезпечення, у т.ч. редактори векторної графіки. Сьогодні особливо привабливі для інвесторів компанії, що спеціалізуються на графічних інтерфейсах користувача, об’єктно-орієнтованих програмах, віртуальній реальності та програмному забезпеченні паралельних процесів.

Специфікація SVG включає об’єктну модель документа – DOM (Document Object Model), що полегшує обробку графічних об’єктів. Векторна графіка відрізняється зручністю використання для зображень, що складаються з елементів, які можуть бути розкладені на найпростіші геометричні об’єкти (лінії, кола, багатокутники, текст). Векторні дані масштабуються і піддаються різноманітних маніпуляцій (у тому числі обертання, витягування, стиснення). Векторні зображення адаптуються до різних пристроїв виведення та можуть бути перетворені на інший векторний формат. До недоліків векторної графіки відносять проблематичність її використання передачі складних зображень (наприклад фотографій). Візуалізація векторних зображень може вимагати значно більше часу, ніж растрового файлу такої ж складності, оскільки кожен елемент зображення має бути відтворений окремо та у певній послідовності. Усі комп’ютерні зображення, всі формати для їх зберігання і всі програми для їх обробки поділяються на два великі класи – векторні та растрові, – що відрізняються, перш за все, рівнем абстракції, застосованої до зображення. Можна сказати, що якщо векторна графіка намагається імітувати сприйняття

зображень людиною, то растровий формат зберігає графіку в тому вигляді, в якому вона найлегше перетравлюється комп'ютером. Відповідно, векторна графіка здебільшого створюється людиною з нуля прямо у векторному редакторі, а спроби генерувати її автоматично рідко коли призводять до задовільного результату. І навпаки, головний постачальник растрових зображень – фотографії, тобто у суттєвій своїй частині автоматичний процес з результатами, що легко оцифровуються. Векторне зображення складається з об'єктів – геометричних форм, складених з прямих, дуг кола та кривих Безьє. У всіх векторних форматах об'єкти можуть варіювати товщину та колір контуру, а замкнуті об'єкти – ще й колір заливки. Об'єкти можуть накладатися, частково або повністю затуляючи один одного. Як окремі об'єкти можуть включатися растрові зображення та рядки або абзаци тексту (букви яких можуть також зберігатися у вигляді геометричних форм, але допускають і більш високий рівень абстракції – поділ на власне текст, який можна редагувати, та параметри його оформлення).

Основні результати та висновки:

- проаналізовано різні види цифрових зображень, області застосування, а також переваги та недоліки;
- проаналізовано основні об'єкти векторної графіки, а також структура svg-зображення;
- через те, що більшість цифрових зображень є растровими, запропоновано метод перетворення растрових зображень на векторні трасування.
- розроблено діаграму варіантів використання, що дозволяє за допомогою певних інструментів без великих витрат часу створювати векторне svg-зображення;
- експериментально перевірено, що використання векторних зображень формату svg помітно покращує якісне відображення цифрового зображення.

– експериментально доведено, що з проектування Web-систем необхідно використовувати зображення формату svg, як мінімум для логотипів компаній, оскільки при масштабуванні сторінки не страждає якість відображення зображення загалом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lenburn, J. The great cartoon directors / Lenburn J. – NY. : DaCapo, 2013
2. Турлюн Л. Н. Комп'ютерна графіка як особливий вид сучасного мистецтва/Л. Н. Турлюн. – Барнаул /: Алтайський державний університет, 2014. – 100 с.
3. Яцюк О. Г. Основи графічного дизайну на базі комп'ютерних технологій: Довідковий та практичний посібник про сучасний графічний дизайн / О. Г. Яцюк. – Львів.: БХВ – Львів, 2014. – 240 с.
4. Маценко В. Г. Комп'ютерна графіка / В. Г. Маценко. – Чернівці : «Рута», 2019. – 341 с.
5. Tosiasu, L. Computer Animation: theory and practice / Tosiasu L. – T. :Departament of Information Science, 2019 – 13 p.
6. Richard, W. The Animator's Survival Kit / Richard W. – B. : Touchstone Pictures, 2015 – 223 p.
7. Ажгіхіна С. Г. Комп'ютерні технології в образотворчій та дизайнерській діяльності студентів вузів / С. Г. Ажгіхіна // Мистецтво та освіта. – 2014.
8. Ollie, J. The illusion of life: Disney animation / Ollie J. – E. : Disney editions, 2015 – 119 p.
9. Hayward, S. Computer for animation / Hayward S. – C. : Wobura, 2019 – 186 p.
10. Сухорукова Л. А. Дизайн: історія, теорія і практика / Л. А. Сухорукова, М. В. Мурашко. // ВІСНИК. – 2017. – №41. – С. 37–42.
11. Уваров А. С. Графіка для конструкторів / А. С. Уваров. – К.: «ДМК Пресс», 2018. – 306 с.
12. Браілов А. Ю. Інженерна комп'ютерна графіка / А. Ю. Браілов. – К.: «Каравела», 2017. – 176 с.
13. Ванін В. В. Комп'ютерна інженерна графіка у середовищі / В. В. Ванін, В. В. Перевертун, Т. О. Надкернична. – К.: «Каравела», 2015. – 336 с.

14. Lenburn, J. The encyclopedia of animated cartoons, 2nd edition / Lenburn J. – NY. : Facts on file, 1999 – 366 p
15. Овчиннікова Р.Ю. Дизайн у рекламі: основи графічного проектування: навчальний посібник, 2019. – 240с.
16. Вінічук, О.М. SVG-зображення як основний векторний формат графіки / О.Н.Вінічук // Техніка та технології: роль у розвитку сучасного товариства : Матеріали X Міжнародної науково-практичної конференції – Житомир.: Апріорі, 2017. – с. 24-28.
17. Вінічук, О.М. SVG – основний формат векторного зображення при проектуванні інформаційних web-систем / О.Н.Вінічук, Н.І.Лістопад // Радіоелектроніка та молодь у XXI столітті. Матеріали XXI Міжнародного молодіжного форуму (Україна, м. Харків, 25-27 березня 2017 року). – Харків: ХНУРЕ, 2017. – с. 303–304.
18. Вінічук, О.М. Використання векторної графіки при Проектуванні інформаційних web-систем / О.Н.Вінічук, Н.І.Лістопад / 53-та наукова конференція аспірантів, магістрантів та студентів (Республіка Білорусь, м. Мінськ, 3-4 травня 2017). – Мінськ: БДУІР, 2017.
19. Вінічук, О.М. Вплив атомної моделі Бору на розвиток атомної енергетики/О.М. Вінічук, О.О. Новікова // Великі перетворювачі природознавства: Нільс Бор: матеріали ювілейних XXV міжнародних читань (Республіка Білорусь, м. Мінськ, 16-17 березня 2017 року). – Мінськ; БДУІР, 2017. – с.227-228
20. Agree, W. Introduction to Mass communication / Agree w. – NY. : Longman, 2007 – 522 p.
21. Гейн А.Г., Житомирський В.Г., Лінецький О.В., Сапір М.В., Шолоховіч В.Ф. "Основи інформатики та обчислювальної техніки". – М.: Просвещение, 2020 р. – 254 с.
22. Stern, G. A program foe 3-Dimensional Animation / Stern G. – NY. : Nycograph, 1983 – 404 p.

23. Саймон М. Як створити власний мультфільм. Анімація двомірних персонажів [Текст] / Саймон М. – М.: НТ Прес, 2016. – 576 с.
24. Семакін І.Г., Хеннер Є.К. Інформатики. Задачник практикум у 2т. Том. 2. М.: Лабораторія Базових Знань, 2019. – 280 с.
25. Пічугін М. Ф. Комп'ютерна графіка / М. Ф. Пічугін, І. О. Канкін, В. В. Воротніков. – Київ : «Центр учбової літератури», 2021. – 344 с.
26. Павлова І.М. Практичні завдання для роботи у графічному редакторі Інформатика та освіта. – 2014. – № 1. – С. 35 – 44.
27. Моргунов Є.Б. Людські фактори в комп'ютерних системах. М.: Тривола, 2020-. – 272 с.
28. Рожкова Н. Г. Графічний дизайн і реклама на комп'ютері / Н. Г. Рожкова, П. П. Данилов, В. Н. Шитов. – К.: «Вільямс», 2019. – 320 с.
29. Анципа В. А. Растрові та векторні графічні зображення / / Інформатика та освіта. – 2020. – № 7. – С. 56-62.
30. Турлюн Л.М. Зародження комп'ютерної графіки в 50-60-х роках XX століття // Молодий учений. – 2012. – №5- с.569-570
31. Турлюн Л. Н. Роль комп'ютерних технологій та комп'ютерної графіки у формуванні професійних компетенцій бакалавра та магістра мистецтвознавства // ALMA MATER (ВІСНИК ВИЩО ШКОЛИ) 2022. №8.
32. Маценко В. Г. Комп'ютерна графіка / В. Г. Маценко. – Чернівці : «Рута», 2019. – 341 с.
33. Поляков А. Ю. Методи та алгоритми комп'ютерної графіки / А. Ю. Поляков, А. Ю. Бруснєцов. – Херсон, 2021. – 574 с.
34. Добробабенко Н. С. Фірмовий стиль: принципи розробки. – М., 2021. – 234 с.
35. Горобець С. М. Основи комп'ютерної графіки / С. М. Горобець. – К.: «Центр навчальної літератури», 2016. – 232 с.
36. Офіційний сайт CorelXara [Електронний ресурс] – режим доступу до ресурсу: https://www.xara.com/us/designer-pro/

37. Офіційний сайт Adobe Illustrator [Електронний ресурс] – режим доступу до ресурсу: https://www.adobe.com/ua/products/illustrator.html
38. Офіційний сайт CorelDRAW [Електронний ресурс] – режим доступу до ресурсу: https://www.coreldraw.com/en/
39. Freeman, Robson – Learning JavaScript Programming . – NY. : Longman, 2007 – 522 p.
40. Minnick, Holland – JavaScript for Dummies – NY. : Nycograph, 2015 – 404 p.
41. D. Crockford – How JavaScript Works – NY. : G. Braziller, 2005 – 195 p.
42. E. Brown – Learning JavaScript. A guide to creating modern websites – NY. : MIT Press, 2001 – 695 p.
43. D. Duckett – Javascript and jQuery. Interactive web development – NY. : DaCapo, 2003 – 273 p.
44. S. Stefanov – JavaScript. Templates . – NY. : Longman, 2007 – 522 p.
45. Resig, Bibo, Maras – JavaScript Ninja Secrets – NY. : MIT Press, 2014 – 422 p.
46. M. Haverbeke – Expressive JavaScript. Modern web programming – NY. : MIT Press, 2001 – 195 p.
47. N. Zakas – JavaScript. Performance optimization – NY. : MIT Press, 2021 – 555 p.
48. L. Atencio – Functional programming in JavaScript – NY. : MIT Press, 2009 – 342 p.
49. M. Fowler – JavaScript code refactoring – NY. : MIT Press, 2016 – 765 p.
50. Офіційний сайт Vue – [Електронний ресурс] — Режим доступу: https://v2.vuejs.org/v2/cookbook/index.html?redirect=true
51. Офіційний сайт Amazon – [Електронний ресурс] — Режим доступу: https://aws.amazon.com/ru/what-is/javascript/
52. Офіційний сайт JS – [Електронний ресурс] — Режим доступу: https://learn.javascript.ru/intro
53. М. Паціанський – Основи Redux. Без води! (2016) –2016. – 341 с.
54. Стоян Стефанов React.js Швидкий старт (2016) –2016. – 241 с.
55. М. Паціанський – React.js для початківців (2016) –2016. – 355 с.

56. Офіційний сайт React – [Електронний ресурс] — Режим доступу: https://reactjs.org/
57. Офіційний сайт Angular – [Електронний ресурс] — Режим доступу: https://angular.io/
58. Tony, W. The animator workbook / Tony W. – NY. : Read Business Information, 2019 – 142 p.
59. Jack, B. Structure of art [Текст] / Jack B. – NY. : G. Braziller, 2018 – 195 p.
60. Метод getCurrentPosition(location, error, configuration) — синтаксис № 3. 2016. – [Електронний ресурс] — Режим доступу: http://html5ru.com/css-i-html.html
61. Метод getCurrentPosition(location, error, configuration) — синтаксис № 4. 2016. [Електронний ресурс] — Режим доступу: http://html5ru.com/-preimushhestva-html5.html
62. Jack, B. Structure of art [Текст] / Jack B. – NY. : G. Braziller, 2018 – 195 p.
63. Istvan Barakonyi, Tamer Fahmy, Dieter Schmalstieg, "Remote collaboration using Augmented Reality Videoconferencing", Proceedings of Graphics Interface 2014, p.89–96, May 17–19, 2014, London, Ontario, Canada
64. Офіційний сайт – Laravel – [Електронний ресурс] – Режим доступу: https://laravel.ru/docs/v5
65. Офіційний сайт – Yii – [Електронний ресурс] – Режим доступу: https://www.yiiframework.com/doc/guide/2.0/en
66. Офіційний сайт – Symfony – [Електронний ресурс] – Режим доступу: https://symfony.com/doc/current/index.html
67. Офіційний сайт – MySql – [Електронний ресурс] – Режим доступу: https://www.mysql.com/
68. Офіційний сайт – PostgreSQL – [Електронний ресурс] – Режим доступу: https://www.postgresql.org/
69. Офіційний сайт – SQLite – [Електронний ресурс] – Режим доступу: https://sqlite.org/index.html
70. Офіційний сайт – Oracle – [Електронний ресурс] – Режим доступу: https://www.oracle.com/cis/
71. Офіційний сайт – MongoDB – [Електронний ресурс] – Режим доступу: https://www.mongodb.com/

72. Чверкун Денис. Основні властивості каскадної моделі. 2019 рік. – [Електронний ресурс] – Режим доступу: https://geekbrains.ru/posts /waterfall
73. Магданур Г.І. ASP.NET MVC Framework: навчальний посібник / Магдануров Г.І., Юнєв В.А. – СПб.: БХВ-Петербург, 2019. – 321 с.
74. Шматалюк А., Ферапонтов М., Громов А., Каменова М. Моделювання бізнесу. Методологія ARIS – 327 с. 2021.

ДОДАТОК А

Технічне завдання

ЗАТВЕРДЖУЮ
Проректор Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ОБРОБКИ ВЕКТОРНОЇ ГРАФІКИ

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ
1116130. 01271 – 01 12 01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____ Вадим ГОРЯЧКІН
Виконавець
_____Сергій БАГНО
Нормоконтролер
_____Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО

1116130. 01271 – 01 12 01

ДОСЛІДЖЕННЯ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ОБРОБКИ ВЕКТОРНОЇ ГРАФІКИ

Текст програми

Листів 64

2024

АНОТАЦІЯ

ЗМІСТ

2. ТЕКСТ ПРОГРАМИ

```
var LiveInput = (0, _liveInputHoc2.default)(_input2.default);

var ModeToolsComponent = function ModeToolsComponent(props) {

    var messages = (0, _reactIntl.defineMessages)({

        brushSize: {

            'id': 'paint.modeTools.brushSize',

            'defaultMessage': 'Size'

        },

        eraserSize: {

            'id': 'paint.modeTools.eraserSize',

            'defaultMessage': 'Eraser size'

        },

        copy: {

            'id': 'paint.modeTools.copy',

            'defaultMessage': 'Копіювати'

        },

        paste: {

            'id': 'paint.modeTools.paste',

            'defaultMessage': 'Вставити'

        },
```

```
delete: {
    'id': 'paint.modeTools.delete',
    'defaultMessage': 'Видалити'
},
curved: {
    'id': 'paint.modeTools.curved',
    'defaultMessage': 'Curved'
},
pointed: {
    'id': 'paint.modeTools.pointed',
    'defaultMessage': 'Pointed'
},
thickness: {
    'id': 'paint.modeTools.thickness',
    'defaultMessage': 'Thickness'
},
flipHorizontal: {
    'id': 'paint.modeTools.flipHorizontal',
    'defaultMessage': 'Горизонтально'
},
```

```
    flipVertical: {
        'id': 'paint.modeTools.flipVertical',
        'defaultMessage': 'Вертикально'
    },
    filled: {
        'id': 'paint.modeTools.filled',
        'defaultMessage': 'Filled'
    },
    outlined: {
        'id': 'paint.modeTools.outlined',
        'defaultMessage': 'Outlined'
    }
});

switch (props.mode) {
    case _modes2.default.BRUSH:
    /* falls through */
    case _modes2.default.BIT_BRUSH:
    /* falls through */
    case _modes2.default.BIT_LINE:
```

```
        {
            var currentIcon = (0, _format.isVector)(props.format) ? _brush4.default : props.mode === _modes2.default.BIT_LINE ? _line2.default : _brush2.default;
            var currentBrushValue = (0, _format.isBitmap)(props.format) ? props.bitBrushSize : props.brushValue;
            var changeFunction = (0, _format.isBitmap)(props.format) ? props.onBitBrushSliderChange : props.onBrushSliderChange;
            var currentMessage = props.mode === _modes2.default.BIT_LINE ? messages.thickness : messages.brushSize;
            return _react2.default.createElement(
                'div',
                { className: (0, _classnames2.default)(props.className, _modeTools2.default.modeTools) },
                _react2.default.createElement(
                    'div',
                    null,
                    _react2.default.createElement('img', {
                        alt: props.intl.formatMessage(currentMessage),
                        classNameName: _modeTools2.default.modeToolsIcon,
                        draggable: false,
                        src: currentIcon
                    })
```

```
                ),
                _react2.default.createElement(LiveInput, {
                    range: true,
                    small: true,
                    max: _strokeWidth.MAX_STROKE_WIDTH,
                    min: '1',
                    type: 'number',
                    value: currentBrushValue,
                    onSubmit: changeFunction
                })
            );
        }
    case _modes2.default.BIT_ERASER:
    /* falls through */
    case _modes2.default.ERASER:
        {
            var _currentIcon = (0, _format.isVector)(props.format) ? _eraser4.default
: _eraser2.default;
            var currentEraserValue = (0, _format.isBitmap)(props.format) ?
props.bitEraserSize : props.eraserValue;
            var _changeFunction = (0, _format.isBitmap)(props.format) ?
```

```
props.onBitEraserSliderChange : props.onEraserSliderChange;

            return _react2.default.createElement(

                'div',

                { className: (0, _classnames2.default)(props.className,
_modeTools2.default.modeTools) },

                _react2.default.createElement(

                    'div',

                    null,

                    _react2.default.createElement('img', {

                        alt: props.intl.formatMessage(messages.eraserSize),

                        className: _modeTools2.default.modeToolsIcon,

                        draggable: false,

                        src: _currentIcon

                    })

                ),

                _react2.default.createElement(LiveInput, {

                    range: true,

                    small: true,

                    max: _strokeWidth.MAX_STROKE_WIDTH,

                    min: '1',
```

```
                type: 'number',
                value: currentEraserValue,
                onSubmit: _changeFunction
            })
        );
    }
case _modes2.default.RESHAPE:
    return _react2.default.createElement(
        'div',
        { className: (0, _classnames2.default)(props.className,
_modeTools2.default.modeTools) },
        _react2.default.createElement(
            _inputGroup2.default,
            { className: (0,
_classnames2.default)(_modeTools2.default.modDashedBorder,
_modeTools2.default.modLabeledIconHeight) },
            _react2.default.createElement(_labeledIconButton2.default, {
                disabled: !props.hasSelectedUncurvedPoints,
                hideLabel: (0, _hideLabel.hideLabel)(props.intl.locale),
                imgSrc: _curvedPoint2.default,
                title: props.intl.formatMessage(messages.curved),
```

```
                onClick: props.onCurvePoints
            }),
            _react2.default.createElement(_labeledIconButton2.default, {
                disabled: !props.hasSelectedUnpointedPoints,
                hideLabel: (0, _hideLabel.hideLabel)(props.intl.locale),
                imgSrc: _straightPoint2.default,
                title: props.intl.formatMessage(messages.pointed),
                onClick: props.onPointPoints
            })
        ),
        _react2.default.createElement(
            _inputGroup2.default,
            { className: (0,
_classnames2.default)(_modeTools2.default.modLabeledIconHeight) },
            _react2.default.createElement(_labeledIconButton2.default, {
                hideLabel: (0, _hideLabel.hideLabel)(props.intl.locale),
                imgSrc: _delete2.default,
                title: props.intl.formatMessage(messages.delete),
                onClick: props.onDelete
            })
```

```
            )
        );
    case _modes2.default.BIT_SELECT:
    /* falls through */
    case _modes2.default.SELECT:
        return _react2.default.createElement(
            'div',
            { className: (0, _classnames2.default)(props.className,
_modeTools2.default.modeTools) },
            _react2.default.createElement(
                _inputGroup2.default,
                { className: (0,
_classnames2.default)(_modeTools2.default.modDashedBorder,
_modeTools2.default.modLabeledIconHeight) },
                _react2.default.createElement(_labeledIconButton2.default, {
                    hideLabel: (0, _hideLabel.hideLabel)(props.intl.locale),
                    imgSrc: _copy2.default,
                    title: props.intl.formatMessage(messages.copy),
                    onClick: props.onCopyToClipboard
                }),
                _react2.default.createElement(_labeledIconButton2.default, {
```

```
                disabled: !(props.clipboardItems.length > 0),
                hideLabel: (0, _hideLabel.hideLabel)(props.intl.locale),
                imgSrc: _paste2.default,
                title: props.intl.formatMessage(messages.paste),
                onClick: props.onPasteFromClipboard
            })
        ),
        _react2.default.createElement(
            _inputGroup2.default,
            { className: (0,
_classnames2.default)(_modeTools2.default.modDashedBorder,
_modeTools2.default.modLabeledIconHeight) },
            _react2.default.createElement(_labeledIconButton2.default, {
                hideLabel: (0, _hideLabel.hideLabel)(props.intl.locale),
                imgSrc: _delete2.default,
                title: props.intl.formatMessage(messages.delete),
                onClick: props.onDelete
            })
        ),
        _react2.default.createElement(
            _inputGroup2.default,
```

```
                { className: (0,
_classnames2.default)(_modeTools2.default.modLabeledIconHeight) },
                _react2.default.createElement(_labeledIconButton2.default, {
                    hideLabel: props.intl.locale !== 'en',
                    imgSrc: _flipHorizontal2.default,
                    title: props.intl.formatMessage(messages.flipHorizontal),
                    onClick: props.onFlipHorizontal
                }),
                _react2.default.createElement(_labeledIconButton2.default, {
                    hideLabel: props.intl.locale !== 'en',
                    imgSrc: _flipVertical2.default,
                    title: props.intl.formatMessage(messages.flipVertical),
                    onClick: props.onFlipVertical
                })
            )
        );
    case _modes2.default.BIT_TEXT:
    /* falls through */
    case _modes2.default.TEXT:
        return _react2.default.createElement(
```

```
            'div',
            { className: (0, _classnames2.default)(props.className,
_modeTools2.default.modeTools) },
            _react2.default.createElement(
              _inputGroup2.default,
              null,
              _react2.default.createElement(_fontDropdown2.default, {
                onUpdateImage: props.onUpdateImage
              })
            )
          );
        case _modes2.default.BIT_RECT:
        /* falls through */
        case _modes2.default.BIT_OVAL:
          {
            var fillIcon = props.mode === _modes2.default.BIT_RECT ?
_rectangle2.default : _oval2.default;
            var outlineIcon = props.mode === _modes2.default.BIT_RECT ?
_rectangleOutlined2.default : _ovalOutlined2.default;
            return _react2.default.createElement(
              'div',
```

```
            { className: (0, _classnames2.default)(props.className,
_modeTools2.default.modeTools) },
            _react2.default.createElement(
              _inputGroup2.default,
              null,
              _react2.default.createElement(_labeledIconButton2.default, {
                highlighted: props.fillBitmapShapes,
                imgSrc: fillIcon,
                title: props.intl.formatMessage(messages.filled),
                onClick: props.onFillShapes
              })
            ),
            _react2.default.createElement(
              _inputGroup2.default,
              null,
              _react2.default.createElement(_labeledIconButton2.default, {
                highlighted: !props.fillBitmapShapes,
                imgSrc: outlineIcon,
                title: props.intl.formatMessage(messages.outlined),
                onClick: props.onOutlineShapes
```

```
        })
    ),
    props.fillBitmapShapes ? null : _react2.default.createElement(
        _inputGroup2.default,
        null,
        _react2.default.createElement(
            _label2.default,
            { text: props.intl.formatMessage(messages.thickness) },
            _react2.default.createElement(LiveInput, {
                range: true,
                small: true,
                max: _strokeWidth.MAX_STROKE_WIDTH,
                min: '1',
                type: 'number',
                value: props.bitBrushSize,
                onSubmit: props.onBitBrushSliderChange
            })
        )
    )
);
```

```
            }
        default:
            // Leave empty for now, if mode not supported
            return _react2.default.createElement('div', { className: (0,
_classnames2.default)(props.className, _modeTools2.default.modeTools) });
    }
};

ModeToolsComponent.propTypes = {
    bitBrushSize: _propTypes2.default.number,
    bitEraserSize: _propTypes2.default.number,
    brushValue: _propTypes2.default.number,
    className: _propTypes2.default.string,
    clipboardItems: _propTypes2.default.arrayOf(_propTypes2.default.array),
    eraserValue: _propTypes2.default.number,
    fillBitmapShapes: _propTypes2.default.bool,
    format: _propTypes2.default.oneOf(Object.keys(_format2.default)),
    hasSelectedUncurvedPoints: _propTypes2.default.bool,
    hasSelectedUnpointedPoints: _propTypes2.default.bool,
    intl: _reactIntl.intlShape.isRequired,
```

```
    mode: _propTypes2.default.string.isRequired,
    onBitBrushSliderChange: _propTypes2.default.func.isRequired,
    onBitEraserSliderChange: _propTypes2.default.func.isRequired,
    onBrushSliderChange: _propTypes2.default.func.isRequired,
    onCopyToClipboard: _propTypes2.default.func.isRequired,
    onCurvePoints: _propTypes2.default.func.isRequired,
    onDelete: _propTypes2.default.func.isRequired,
    onEraserSliderChange: _propTypes2.default.func,
    onFillShapes: _propTypes2.default.func.isRequired,
    onFlipHorizontal: _propTypes2.default.func.isRequired,
    onFlipVertical: _propTypes2.default.func.isRequired,
    onOutlineShapes: _propTypes2.default.func.isRequired,
    onPasteFromClipboard: _propTypes2.default.func.isRequired,
    onPointPoints: _propTypes2.default.func.isRequired,
    onUpdateImage: _propTypes2.default.func.isRequired
};

var mapStateToProps = function mapStateToProps(state) {
    return {
        mode: state.scratchPaint.mode,
```

```
        format: state.scratchPaint.format,
        fillBitmapShapes: state.scratchPaint.fillBitmapShapes,
        bitBrushSize: state.scratchPaint.bitBrushSize,
        bitEraserSize: state.scratchPaint.bitEraserSize,
        brushValue: state.scratchPaint.brushMode.brushSize,
        clipboardItems: state.scratchPaint.clipboard.items,
        eraserValue: state.scratchPaint.eraserMode.brushSize
    };
};
var mapDispatchToProps = function mapDispatchToProps(dispatch) {
    return {
        onBrushSliderChange: function onBrushSliderChange(brushSize) {
            dispatch((0, _brushMode.changeBrushSize)(brushSize));
        },
        onBitBrushSliderChange: function onBitBrushSliderChange(bitBrushSize) {
            dispatch((0, _bitBrushSize.changeBitBrushSize)(bitBrushSize));
        },
        onBitEraserSliderChange: function onBitEraserSliderChange(eraserSize) {
            dispatch((0, _bitEraserSize.changeBitEraserSize)(eraserSize));
        },
```

```
        onEraserSliderChange: function onEraserSliderChange(eraserSize) {
            dispatch((0, _eraserMode.changeBrushSize)(eraserSize));
        },
        onFillShapes: function onFillShapes() {
            dispatch((0, _fillBitmapShapes.setShapesFilled)(true));
        },
        onOutlineShapes: function onOutlineShapes() {
            dispatch((0, _fillBitmapShapes.setShapesFilled)(false));
        }
    };
};

exports.default = (0, _reactRedux.connect)(mapStateToProps,
mapDispatchToProps)((0, _reactIntl.injectIntl)(ModeToolsComponent));

/***/ }),
/* 342 */
/***/ (function(module, exports, __webpack_require__) {

"use strict";
```

```
Object.defineProperty(exports, "__esModule", {
    value: true
});

var _createClass = function () { function defineProperties(target, props) { for (var i = 0; i < props.length; i++) { var descriptor = props[i]; descriptor.enumerable = descriptor.enumerable || false; descriptor.configurable = true; if ("value" in descriptor) descriptor.writable = true; Object.defineProperty(target, descriptor.key, descriptor); } } return function (Constructor, protoProps, staticProps) { if (protoProps) defineProperties(Constructor.prototype, protoProps); if (staticProps) defineProperties(Constructor, staticProps); return Constructor; }; }();

var _paper = __webpack_require__(2);

var _paper2 = _interopRequireDefault(_paper);

var _reactRedux = __webpack_require__(6);

var _lodash = __webpack_require__(5);
```

```
var _lodash2 = _interopRequireDefault(_lodash);

var _propTypes = __webpack_require__(1);

var _propTypes2 = _interopRequireDefault(_propTypes);

var _react = __webpack_require__(0);

var _react2 = _interopRequireDefault(_react);

var _fontDropdown = __webpack_require__(343);

var _fontDropdown2 = _interopRequireDefault(_fontDropdown);

var _fonts = __webpack_require__(53);

var _fonts2 = _interopRequireDefault(_fonts);

var _font = __webpack_require__(68);
```

var _selection = __webpack_require__(3);

var _fontDropdown3 = __webpack_require__(109);

var _fontDropdown4 = _interopRequireDefault(_fontDropdown3);

function _interopRequireDefault(obj) { return obj && obj.__esModule ? obj : { default: obj }; }

function _classCallCheck(instance, Constructor) { if (!(instance instanceof Constructor)) { throw new TypeError("Cannot call a class as a function"); } }

function _possibleConstructorReturn(self, call) { if (!self) { throw new ReferenceError("this hasn't been initialised - super() hasn't been called"); } return call && (typeof call === "object" || typeof call === "function") ? call : self; }

function _inherits(subClass, superClass) { if (typeof superClass !== "function" && superClass !== null) { throw new TypeError("Super expression must either be null or a function, not " + typeof superClass); } subClass.prototype = Object.create(superClass && superClass.prototype, { constructor: { value: subClass, enumerable: false, writable: true, configurable: true } }); if (superClass) Object.setPrototypeOf ? Object.setPrototypeOf(subClass, superClass) : subClass.__proto__ = superClass; }

```
var FontDropdown = function (_React$Component) {
    _inherits(FontDropdown, _React$Component);

    function FontDropdown(props) {
        _classCallCheck(this, FontDropdown);

        var _this = _possibleConstructorReturn(this, (FontDropdown.__proto__ ||
Object.getPrototypeOf(FontDropdown)).call(this, props));

        (0, _lodash2.default)(_this, ['getFontStyle', 'getFontName',
'handleChangeFontSerif', 'handleChangeFontSansSerif',
'handleChangeFontHandwriting', 'handleChangeFontMarker',
'handleChangeFontCurly', 'handleChangeFontPixel', 'handleChangeFontChinese',
'handleChangeFontJapanese', 'handleChangeFontKorean', 'handleOpenDropdown',
'handleClickOutsideDropdown', 'setDropdown', 'handleChoose']);
        return _this;
    }

    _createClass(FontDropdown, [{
        key: 'getFontStyle',
        value: function getFontStyle(font) {
```

```
switch (font) {
    case _fonts2.default.SERIF:
        return _fontDropdown4.default.serif;
    case _fonts2.default.SANS_SERIF:
        return _fontDropdown4.default.sansSerif;
    case _fonts2.default.HANDWRITING:
        return _fontDropdown4.default.handwriting;
    case _fonts2.default.MARKER:
        return _fontDropdown4.default.marker;
    case _fonts2.default.CURLY:
        return _fontDropdown4.default.curly;
    case _fonts2.default.PIXEL:
        return _fontDropdown4.default.pixel;
    case _fonts2.default.CHINESE:
        return _fontDropdown4.default.chinese;
    case _fonts2.default.JAPANESE:
        return _fontDropdown4.default.japanese;
    case _fonts2.default.KOREAN:
        return _fontDropdown4.default.korean;
    default:
```

```
                return '';
            }
        }
    }, {
        key: 'getFontName',
        value: function getFontName(font) {
            switch (font) {
                case _fonts2.default.CHINESE:
                    return '中文';
                case _fonts2.default.KOREAN:
                    return '한국어';
                case _fonts2.default.JAPANESE:
                    return '日本語';
                default:
                    return font;
            }
        }
    }, {
        key: 'handleChangeFontSansSerif',
        value: function handleChangeFontSansSerif() {
```

```
        if (this.dropDown.isOpen()) {
            this.props.changeFont(_fonts2.default.SANS_SERIF);
        }
    }
}, {
    key: 'handleChangeFontSerif',
    value: function handleChangeFontSerif() {
        if (this.dropDown.isOpen()) {
            this.props.changeFont(_fonts2.default.SERIF);
        }
    }
}, {
    key: 'handleChangeFontHandwriting',
    value: function handleChangeFontHandwriting() {
        if (this.dropDown.isOpen()) {
            this.props.changeFont(_fonts2.default.HANDWRITING);
        }
    }
}, {
    key: 'handleChangeFontMarker',
```

```
    value: function handleChangeFontMarker() {
        if (this.dropDown.isOpen()) {
            this.props.changeFont(_fonts2.default.MARKER);
        }
    }
}, {
    key: 'handleChangeFontCurly',
    value: function handleChangeFontCurly() {
        if (this.dropDown.isOpen()) {
            this.props.changeFont(_fonts2.default.CURLY);
        }
    }
}, {
    key: 'handleChangeFontPixel',
    value: function handleChangeFontPixel() {
        if (this.dropDown.isOpen()) {
            this.props.changeFont(_fonts2.default.PIXEL);
        }
    }
}, {
```

```
    key: 'handleChangeFontChinese',
    value: function handleChangeFontChinese() {
        if (this.dropDown.isOpen()) {
            this.props.changeFont(_fonts2.default.CHINESE);
        }
    }
}, {
    key: 'handleChangeFontJapanese',
    value: function handleChangeFontJapanese() {
        if (this.dropDown.isOpen()) {
            this.props.changeFont(_fonts2.default.JAPANESE);
        }
    }
}, {
    key: 'handleChangeFontKorean',
    value: function handleChangeFontKorean() {
        if (this.dropDown.isOpen()) {
            this.props.changeFont(_fonts2.default.KOREAN);
        }
    }
```

```
}, {
    key: 'handleChoose',
    value: function handleChoose() {
        if (this.dropDown.isOpen()) {
            this.dropDown.handleClosePopover();
            this.props.onUpdateImage();
        }
    }
}, {
    key: 'handleOpenDropdown',
    value: function handleOpenDropdown() {
        this.savedFont = this.props.font;
        this.savedSelection = (0, _selection.getSelectedLeafItems)();
    }
}, {
    key: 'handleClickOutsideDropdown',
    value: function handleClickOutsideDropdown(e) {
        e.stopPropagation();
        this.dropDown.handleClosePopover();
```

```
        // Cancel font change
        var _iteratorNormalCompletion = true;
        var _didIteratorError = false;
        var _iteratorError = undefined;

        try {
            for (var _iterator = this.savedSelection[Symbol.iterator](), _step;
!(_iteratorNormalCompletion = (_step = _iterator.next()).done);
_iteratorNormalCompletion = true) {
                var item = _step.value;

                if (item instanceof _paper2.default.PointText) {
                    item.font = this.savedFont;
                }
            }
        } catch (err) {
            _didIteratorError = true;
            _iteratorError = err;
        } finally {
            try {
                if (!_iteratorNormalCompletion && _iterator.return) {
```

```
                _iterator.return();
            }
        } finally {
            if (_didIteratorError) {
                throw _iteratorError;
            }
        }
    }

    this.props.changeFont(this.savedFont);
    this.savedFont = null;
    this.savedSelection = null;
  }
}, {
  key: 'setDropdown',
  value: function setDropdown(element) {
    this.dropDown = element;
  }
}, {
  key: 'render',
```

```
value: function render() {
    return _react2.default.createElement(_fontDropdown2.default, {
        componentRef: this.setDropdown,
        font: this.props.font,
        getFontName: this.getFontName,
        getFontStyle: this.getFontStyle,
        onChoose: this.handleChoose,
        onClickOutsideDropdown: this.handleClickOutsideDropdown,
        onHoverChinese: this.handleChangeFontChinese,
        onHoverCurly: this.handleChangeFontCurly,
        onHoverHandwriting: this.handleChangeFontHandwriting,
        onHoverJapanese: this.handleChangeFontJapanese,
        onHoverKorean: this.handleChangeFontKorean,
        onHoverMarker: this.handleChangeFontMarker,
        onHoverPixel: this.handleChangeFontPixel,
        onHoverSansSerif: this.handleChangeFontSansSerif,
        onHoverSerif: this.handleChangeFontSerif,
        onOpenDropdown: this.handleOpenDropdown
    });
}
```

```
    }]);


    return FontDropdown;
}(_react2.default.Component);


FontDropdown.propTypes = {
    changeFont: _propTypes2.default.func.isRequired,
    font: _propTypes2.default.string,
    onUpdateImage: _propTypes2.default.func.isRequired
};


var mapStateToProps = function mapStateToProps(state) {
    return {
        font: state.scratchPaint.font
    };
};
var mapDispatchToProps = function mapDispatchToProps(dispatch) {
    return {
        changeFont: function changeFont(font) {
            dispatch((0, _font.changeFont)(font));
```

```
        }
    };
};

exports.default = (0, _reactRedux.connect)(mapStateToProps,
mapDispatchToProps)(FontDropdown);


/***/ }),
/* 343 */
/***/ (function(module, exports, __webpack_require__) {

"use strict";


Object.defineProperty(exports, "__esModule", {
    value: true
});

var _classnames = __webpack_require__(19);
```

```
var _classnames2 = _interopRequireDefault(_classnames);

var _propTypes = __webpack_require__(1);

var _propTypes2 = _interopRequireDefault(_propTypes);

var _react = __webpack_require__(0);

var _react2 = _interopRequireDefault(_react);

var _button = __webpack_require__(43);

var _button2 = _interopRequireDefault(_button);

var _dropdown = __webpack_require__(102);

var _dropdown2 = _interopRequireDefault(_dropdown);

var _inputGroup = __webpack_require__(37);
```

```
var _inputGroup2 = _interopRequireDefault(_inputGroup);

var _fonts = __webpack_require__(53);

var _fonts2 = _interopRequireDefault(_fonts);

var _fontDropdown = __webpack_require__(109);

var _fontDropdown2 = _interopRequireDefault(_fontDropdown);

function _interopRequireDefault(obj) { return obj && obj.__esModule ? obj : { default: obj }; }

var ModeToolsComponent = function ModeToolsComponent(props) {
    return _react2.default.createElement(
        _dropdown2.default,
        {
            className: (0, _classnames2.default)(_fontDropdown2.default.modUnselect, _fontDropdown2.default.fontDropdown),
            enterExitTransitionDurationMs: 60,
```

```
        popoverContent: _react2.default.createElement(
          _inputGroup2.default,
          { className: _fontDropdown2.default.modContextMenu },
          _react2.default.createElement(
            _button2.default,
            {
              className: (0,
_classnames2.default)(_fontDropdown2.default.modMenuItem),
              onClick: props.onChoose,
              onMouseOver: props.onHoverSansSerif
            },
            _react2.default.createElement(
              'span',
              { className: _fontDropdown2.default.sansSerif },
              props.getFontName(_fonts2.default.SANS_SERIF)
            )
          ),
          _react2.default.createElement(
            _button2.default,
            {
```

```
                    className: (0,
_classnames2.default)(_fontDropdown2.default.modMenuItem),
                    onClick: props.onChoose,
                    onMouseOver: props.onHoverSerif
                },
                _react2.default.createElement(
                    'span',
                    { className: _fontDropdown2.default.serif },
                    props.getFontName(_fonts2.default.SERIF)
                )
            ),
            _react2.default.createElement(
                _button2.default,
                {
                    className: (0,
_classnames2.default)(_fontDropdown2.default.modMenuItem),
                    onClick: props.onChoose,
                    onMouseOver: props.onHoverHandwriting
                },
                _react2.default.createElement(
                    'span',
```

```
                { className: _fontDropdown2.default.handwriting },
                props.getFontName(_fonts2.default.HANDWRITING)
            )
        ),
        _react2.default.createElement(
            _button2.default,
            {
                className: (0,
_classnames2.default)(_fontDropdown2.default.modMenuItem),
                onClick: props.onChoose,
                onMouseOver: props.onHoverMarker
            },
            _react2.default.createElement(
                'span',
                { className: _fontDropdown2.default.marker },
                props.getFontName(_fonts2.default.MARKER)
            )
        ),
        _react2.default.createElement(
            _button2.default,
```

```
                {
                    className: (0,
_classnames2.default)(_fontDropdown2.default.modMenuItem),
                    onClick: props.onChoose,
                    onMouseOver: props.onHoverCurly
                },
                _react2.default.createElement(
                    'span',
                    { className: _fontDropdown2.default.curly },
                    props.getFontName(_fonts2.default.CURLY)
                )
            ),
            _react2.default.createElement(
                _button2.default,
                {
                    className: (0,
_classnames2.default)(_fontDropdown2.default.modMenuItem),
                    onClick: props.onChoose,
                    onMouseOver: props.onHoverPixel
                },
                _react2.default.createElement(
```

```
            'span',
            { className: _fontDropdown2.default.pixel },
            props.getFontName(_fonts2.default.PIXEL)
          )
        ),
        _react2.default.createElement(
          _button2.default,
          {
            className: (0,
_classnames2.default)(_fontDropdown2.default.modMenuItem),
            onClick: props.onChoose,
            onMouseOver: props.onHoverChinese
          },
          _react2.default.createElement(
            'span',
            { className: _fontDropdown2.default.chinese },
            props.getFontName(_fonts2.default.CHINESE)
          )
        ),
        _react2.default.createElement(
```

```
                _button2.default,
                {
                    className: (0,
_classnames2.default)(_fontDropdown2.default.modMenuItem),
                    onClick: props.onChoose,
                    onMouseOver: props.onHoverJapanese
                },
                _react2.default.createElement(
                    'span',
                    { className: _fontDropdown2.default.japanese },
                    props.getFontName(_fonts2.default.JAPANESE)
                )
            ),
            _react2.default.createElement(
                _button2.default,
                {
                    className: (0,
_classnames2.default)(_fontDropdown2.default.modMenuItem),
                    onClick: props.onChoose,
                    onMouseOver: props.onHoverKorean
                },
```

```
                _react2.default.createElement(
                    'span',
                    { className: _fontDropdown2.default.korean },
                    props.getFontName(_fonts2.default.KOREAN)
                )
            )
        ),
        ref: props.componentRef,
        tipSize: .01,
        onOpen: props.onOpenDropdown,
        onOuterAction: props.onClickOutsideDropdown
    },
    _react2.default.createElement(
        'span',
        { className: (0, _classnames2.default)(props.getFontStyle(props.font), _fontDropdown2.default.displayedFontName) },
        props.getFontName(props.font)
    )
  );
};
```

```
ModeToolsComponent.propTypes = {
    componentRef: _propTypes2.default.func.isRequired,
    font: _propTypes2.default.string,
    getFontName: _propTypes2.default.func.isRequired,
    getFontStyle: _propTypes2.default.func.isRequired,
    onChoose: _propTypes2.default.func.isRequired,
    onClickOutsideDropdown: _propTypes2.default.func,
    onHoverChinese: _propTypes2.default.func,
    onHoverCurly: _propTypes2.default.func,
    onHoverHandwriting: _propTypes2.default.func,
    onHoverJapanese: _propTypes2.default.func,
    onHoverKorean: _propTypes2.default.func,
    onHoverMarker: _propTypes2.default.func,
    onHoverPixel: _propTypes2.default.func,
    onHoverSansSerif: _propTypes2.default.func,
    onHoverSerif: _propTypes2.default.func,
    onOpenDropdown: _propTypes2.default.func
};
exports.default = ModeToolsComponent;
```

/***/ }),

/* 344 */

/***/ (function(module, exports, __webpack_require__) {

// Imports

var ___CSS_LOADER_API_IMPORT___ = __webpack_require__(14);

exports = ___CSS_LOADER_API_IMPORT___(false);

// Module

exports.push([module.i, "/* DO NOT EDIT\n@todo This file is copied from GUI and should be pulled out into a shared library.\nSee https://github.com/LLK/scratch-paint/issues/13 */\n\n/* DO NOT EDIT\n@todo This file is copied from GUI and should be pulled out into a shared library.\nSee https://github.com/LLK/scratch-paint/issues/13 */\n\n/* ACTUALLY, THIS IS EDITED ;)\nTHIS WAS CHANGED ON 10/25/2017 BY @mewtaylor TO ADD A VARIABLE FOR THE SMALLEST\nGRID UNITS.\n\nALSO EDITED ON 11/13/2017 TO ADD IN CONTANTS FOR LAYOUT FROM `layout-contents.js`*/\n\n/* layout contants from `layout-constants.js`, minus 1px */\n\n.font-dropdown_mod-menu-item_hwOca {\n display: -webkit-box;\n display: -webkit-flex;\n display: -ms-flexbox;\n display: flex;\n margin: 0 -.25rem;\n min-width: 6.25rem;\n padding: calc(2 * .25rem);\n padding-left: calc(3 * .25rem);\n padding-right: calc(3 * .25rem);\n white-space: nowrap;\n width: 8.5rem;\n cursor: pointer;\n -webkit-transition: 0.1s ease;\n -o-transition: 0.1s ease;\n transition: 0.1s ease;\n -webkit-box-align: center;\n -webkit-align-items: center;\n -ms-flex-align: center;\n align-items:

center;\n}\n\n.font-dropdown_mod-menu-item_hwOca:hover {\n background: #4C97FF;\n color: white;\n}\n\n.font-dropdown_mod-context-menu_3llFm {\n display: -webkit-box;\n display: -webkit-flex;\n display: -ms-flexbox;\n display: flex;\n -webkit-box-orient: vertical;\n -webkit-box-direction: normal;\n -webkit-flex-direction: column;\n -ms-flex-direction: column;\n flex-direction: column;\n}\n\n.font-dropdown_mod-unselect_130OF {\n -webkit-user-select: none;\n -moz-user-select: none;\n -ms-user-select: none;\n user-select: none;\n}\n\n.font-dropdown_displayed-font-name_3cU-U {\n font-size: .8rem;\n}\n\n.font-dropdown_font-dropdown_3vjc6 {\n -webkit-box-align: center;\n -webkit-align-items: center;\n -ms-flex-align: center;\n align-items: center;\n color: #575e75;\n display: -webkit-box;\n display: -webkit-flex;\n display: -ms-flexbox;\n display: flex;\n font-size: 1rem;\n -webkit-box-pack: justify;\n -webkit-justify-content: space-between;\n -ms-flex-pack: justify;\n justify-content: space-between;\n width: 8.5rem;\n height: 2rem;\n}\n\n.font-dropdown_serif_tMSQM {\n font-family: 'Serif';\n}\n\n.font-dropdown_sans-serif_24kX9 {\n font-family: 'Sans Serif';\n}\n\n.font-dropdown_serif_tMSQM {\n font-family: 'Serif';\n}\n\n.font-dropdown_handwriting_Y7s5d {\n font-family: 'Handwriting';\n}\n\n.font-dropdown_marker_3AmLD {\n font-family: 'Marker';\n}\n\n.font-dropdown_curly_1UQYh {\n font-family: 'Curly';\n}\n\n.font-dropdown_pixel_3aRC6 {\n font-family: 'Pixel';\n}\n\n.font-dropdown_chinese_zV1Hj {\n font-family: \"Microsoft YaHei\", \"微软雅黑\", STXihei, \"华文细黑\";\n}\n\n.font-dropdown_japanese_2SlYs {\n font-family: \"ヒラギノ角ゴ Pro W3\", \"Hiragino Kaku Gothic Pro\", Osaka, \"メイリオ\", Meiryo, \"ＭＳ Ｐゴシック\", \"MS PGothic\";\n}\n\n.font-dropdown_korean_1Fx37 {\n font-family: \"Malgun Gothic\";\n}\n", ""]);

// Exports

exports.locals = {

"mod-menu-item": "font-dropdown_mod-menu-item_hwOca",

"modMenuItem": "font-dropdown_mod-menu-item_hwOca",

"mod-context-menu": "font-dropdown_mod-context-menu_3llFm",

"modContextMenu": "font-dropdown_mod-context-menu_3llFm",

"mod-unselect": "font-dropdown_mod-unselect_130OF",

"modUnselect": "font-dropdown_mod-unselect_130OF",

"displayed-font-name": "font-dropdown_displayed-font-name_3cU-U",

"displayedFontName": "font-dropdown_displayed-font-name_3cU-U",

"font-dropdown": "font-dropdown_font-dropdown_3vjc6",

"fontDropdown": "font-dropdown_font-dropdown_3vjc6",

"serif": "font-dropdown_serif_tMSQM",

"sans-serif": "font-dropdown_sans-serif_24kX9",

"sansSerif": "font-dropdown_sans-serif_24kX9",

"handwriting": "font-dropdown_handwriting_Y7s5d",

"marker": "font-dropdown_marker_3AmLD",

"curly": "font-dropdown_curly_1UQYh",

"pixel": "font-dropdown_pixel_3aRC6",

"chinese": "font-dropdown_chinese_zV1Hj",

"japanese": "font-dropdown_japanese_2SlYs",

"korean": "font-dropdown_korean_1Fx37"

```
};

module.exports = exports;



/***/ }),

/* 345 */

/***/ (function(module, exports, __webpack_require__) {


var api = __webpack_require__(13);

        var content = __webpack_require__(346);


        content = content.__esModule ? content.default : content;


        if (typeof content === 'string') {

          content = [[module.i, content, '']];

        }


var options = {};


options.insert = "head";
```

options.singleton = false;

var update = api(content, options);

module.exports = content.locals || {};

/***/ }),

/* 346 */

/***/ (function(module, exports, __webpack_require__) {

// Imports

var ___CSS_LOADER_API_IMPORT___ = __webpack_require__(14);

exports = ___CSS_LOADER_API_IMPORT___(false);

// Module

exports.push([module.i, "/* DO NOT EDIT\n@todo This file is copied from GUI and should be pulled out into a shared library.\nSee https://github.com/LLK/scratch-paint/issues/13 */\n\n/* DO NOT EDIT\n@todo This file is copied from GUI and should be pulled out into a shared library.\nSee https://github.com/LLK/scratch-paint/issues/13 */\n\n/* ACTUALLY, THIS IS EDITED ;)\nTHIS WAS CHANGED ON 10/25/2017 BY @mewtaylor TO ADD

A VARIABLE FOR THE SMALLEST\nGRID UNITS.\n\nALSO EDITED ON 11/13/2017 TO ADD IN CONTANTS FOR LAYOUT FROM `layout-contents.js`*/\n\n/* layout contants from `layout-constants.js`, minus 1px */\n\n.mode-tools_mode-tools_UREem {\n display: -webkit-box;\n display: -webkit-flex;\n display: -ms-flexbox;\n display: flex;\n min-height: 3rem;\n -webkit-box-align: center;\n -webkit-align-items: center;\n -ms-flex-align: center;\n align-items: center;\n}\n\n.mode-tools_mode-tools-icon_3yoZ2 {\n margin-right: calc(2 * .25rem);\n width: 2rem;\n height: 2rem;\n}\n\n[dir=\"ltr\"] .mode-tools_mod-dashed-border_3Bmy_ {\n border-right: 1px dashed #D9D9D9;\n padding-right: calc(3 * .25rem);\n}\n\n[dir=\"rtl\"] .mode-tools_mod-dashed-border_3Bmy_ {\n border-left: 1px dashed #D9D9D9;\n padding-left: calc(3 * .25rem);\n}\n\n.mode-tools_mod-labeled-icon-height_kRA3W {\n display: -webkit-box;\n display: -webkit-flex;\n display: -ms-flexbox;\n display: flex;\n height: 2.85rem; /* for the second row so the dashed borders are equal in size */\n -webkit-box-align: center;\n -webkit-align-items: center;\n -ms-flex-align: center;\n align-items: center;\n}\n", ""]);

// Exports

exports.locals = {

"mode-tools": "mode-tools_mode-tools_UREem",

"modeTools": "mode-tools_mode-tools_UREem",

"mode-tools-icon": "mode-tools_mode-tools-icon_3yoZ2",

"modeToolsIcon": "mode-tools_mode-tools-icon_3yoZ2",

"mod-dashed-border": "mode-tools_mod-dashed-border_3Bmy_",

"modDashedBorder": "mode-tools_mod-dashed-border_3Bmy_",

```
        "mod-labeled-icon-height": "mode-tools_mod-labeled-icon-
height_kRA3W",

        "modLabeledIconHeight": "mode-tools_mod-labeled-icon-height_kRA3W"

};

module.exports = exports;




/***/ }),

/* 347 */

/***/ (function(module, exports) {



module.exports = "data:image/svg+xml,%3C?xml version='1.0' encoding='UTF-8'
standalone='no'?%3E %3Csvg width='20px' height='20px' viewBox='0 0 20 20'
version='1.1' xmlns='http://www.w3.org/2000/svg'
xmlns:xlink='http://www.w3.org/1999/xlink'%3E %3C!-- Generator: Sketch 48.2
(47327) - http://www.bohemiancoding.com/sketch --%3E %3Ctitle%3Ecopy
v2%3C/title%3E %3Cdesc%3ECreated with Sketch.%3C/desc%3E
%3Cdefs%3E%3C/defs%3E %3Cg id='Page-1' stroke='none' stroke-width='1'
fill='none' fill-rule='evenodd'%3E %3Cg id='copy-v2'%3E %3Cg id='copy'
transform='translate(3.000000, 2.000000)'%3E %3Cpolyline id='Path-3' stroke-
opacity='0.5' stroke='%23575E75' stroke-linecap='round' stroke-linejoin='round'
stroke-dasharray='1,2' points='0.503173828 3 0.503173828 15.5 13
15.5'%3E%3C/polyline%3E %3Cpath d='M2,1.00684547 C2,0.450780073
2.45303631,0 2.99703014,0 L10,0 L10,2.34995317 C10,3.26124887 10.7336617,4
11.6500468,4 L14,4 L14,13.0046024 C14,13.5543453 13.544239,14
```

12.9975267,14 L3.00247329,14 C2.44882258,14 2,13.5500512 2,12.9931545 L2,1.00684547 Z' id='Rectangle-4' fill='%234C97FF'%3E%3C/path%3E %3Cpath d='M11,0 L14,3 L11.9989566,3 C11.4472481,3 11,2.55733967 11,2.00104344 L11,0 Z' id='Rectangle-5' fill='%234C97FF'%3E%3C/path%3E %3Cpath d='M9.8115942,9.1884058 L8.6884058,9.1884058 L8.6884058,10.3115942 C8.6884058,10.6859903 8.38647343,11 8,11 C7.61352657,11 7.3115942,10.6859903 7.3115942,10.3115942 L7.3115942,9.1884058 L6.1884058,9.1884058 C5.81400966,9.1884058 5.5,8.88647343 5.5,8.5 C5.5,8.11352657 5.81400966,7.8115942 6.1884058,7.8115942 L7.3115942,7.8115942 L7.3115942,6.6884058 C7.3115942,6.31280193 7.61352657,6 8,6 C8.38647343,6 8.6884058,6.31280193 8.6884058,6.6884058 L8.6884058,7.8115942 L9.8115942,7.8115942 C10.1859903,7.8115942 10.5,8.11352657 10.5,8.5 C10.5,8.88647343 10.1859903,9.1884058 9.8115942,9.1884058 Z' id='Fill-1' stroke='%23FFFFFF' stroke-width='0.25' fill='%23FFFFFF'%3E%3C/path%3E %3C/g%3E %3C/g%3E %3C/g%3E %3C/svg%3E"

/***/ }),

/* 348 */

/***/ (function(module, exports) {

module.exports = "data:image/svg+xml,%3C?xml version='1.0' encoding='UTF-8' standalone='no'?%3E %3Csvg width='20px' height='20px' viewBox='0 0 20 20' version='1.1' xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'%3E %3C!-- Generator: Sketch 48.2 (47327) - http://www.bohemiancoding.com/sketch --%3E %3Ctitle%3Epaste

v2%3C/title%3E %3Cdesc%3ECreated with Sketch.%3C/desc%3E %3Cdefs%3E%3C/defs%3E %3Cg id='Page-1' stroke='none' stroke-width='1' fill='none' fill-rule='evenodd'%3E %3Cg id='paste-v2'%3E %3Cg id='paste' transform='translate(3.000000, 2.000000)'%3E %3Cpolyline id='Path-3' stroke-opacity='0.5' stroke='%23575E75' stroke-linecap='round' stroke-linejoin='round' stroke-dasharray='1,2' transform='translate(6.748413, 6.750000) scale(1, -1) translate(-6.748413, -6.750000) ' points='0.5 0.5 0.5 13 12.9968262 13'%3E%3C/polyline%3E %3Cpath d='M2,3.00684547 C2,2.45078007 2.45303631,2 2.99703014,2 L10,2 L10,4.34995317 C10,5.26124887 10.7336617,6 11.6500468,6 L14,6 L14,15.0046024 C14,15.5543453 13.544239,16 12.9975267,16 L3.00247329,16 C2.44882258,16 2,15.5500512 2,14.9931545 L2,3.00684547 Z' id='Rectangle-4' fill='%234C97FF'%3E%3C/path%3E %3Cpath d='M11,2 L14,5 L11.9989566,5 C11.4472481,5 11,4.55733967 11,4.00104344 L11,2 Z' id='Rectangle-5' fill='%234C97FF'%3E%3C/path%3E %3Cpath d='M8.34791833,12.8771885 C8.26180668,12.9633001 8.14699113,13.0063559 8.03217559,13.0063559 C7.9030081,13.0063559 7.78819256,12.9633001 7.70208091,12.8771885 L5.86503222,11.0401398 C5.73586474,10.8966203 5.69280891,10.7100451 5.76456862,10.5378218 C5.83632834,10.3655985 5.99419971,10.2651349 6.18077497,10.2651349 L6.92707599,10.2651349 L7.28587456,7.66743321 C7.31457845,7.46650601 7.41504205,7.27993075 7.57291342,7.16511521 C7.73078479,7.03594773 7.94606393,6.97853995 8.13263919,7.00724384 C8.47708582,7.06321642 8.74977273,7.33733852 8.79282856,7.66743321 L9.16597907,10.2651349 L9.86922427,10.2651349 C10.0557995,10.2651349 10.2136709,10.3799504 10.2854306,10.5521737 C10.3571903,10.7100451 10.3141345,10.9109723 10.184967,11.0401398 L8.34791833,12.8771885 Z' id='Fill-1' fill='%23FFFFFF'%3E%3C/path%3E %3C/g%3E %3C/g%3E %3C/g%3E %3C/svg%3E"

/***/ }),

/* 349 */

/***/ (function(module, exports) {

module.exports = "data:image/svg+xml,%3C?xml version='1.0' encoding='UTF-8'?%3E %3Csvg width='20px' height='20px' viewBox='0 0 20 20' version='1.1' xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'%3E %3C!-- Generator: Sketch 50.2 (55047) - http://www.bohemiancoding.com/sketch --%3E %3Ctitle%3Edelete%3C/title%3E %3Cdesc%3ECreated with Sketch.%3C/desc%3E %3Cdefs%3E%3C/defs%3E %3Cg id='delete' stroke='none' stroke-width='1' fill='none' fill-rule='evenodd'%3E %3Cg id='Delete-Icon' transform='translate(2.000000, 1.500000)' fill='%234C97FF' fill-rule='nonzero'%3E %3Cpath d='M2,3.25 L14,3.25 C14.4437164,3.25 14.7904293,3.63311343 14.7462779,4.07462779 L13.6363275,15.1741315 C13.5468672,16.0687347 12.7940775,16.75 11.8950124,16.75 L4.10498756,16.75 C3.20592253,16.75 2.45313279,16.0687347 2.36367248,15.1741315 L1.25372211,4.07462779 C1.20957067,3.63311343 1.55628356,3.25 2,3.25 Z M8.75,12 L8.75,7 C8.75,6.58578644 8.41421356,6.25 8,6.25 C7.58578644,6.25 7.25,6.58578644 7.25,7 L7.25,12 C7.25,12.4142136 7.58578644,12.75 8,12.75 C8.41421356,12.75 8.75,12.4142136 8.75,12 Z M11.25,12 L11.25,7 C11.25,6.58578644 10.9142136,6.25 10.5,6.25 C10.0857864,6.25 9.75,6.58578644 9.75,7 L9.75,12 C9.75,12.4142136 10.0857864,12.75 10.5,12.75 C10.9142136,12.75 11.25,12.4142136 11.25,12 Z M6.25,12 L6.25,7 C6.25,6.58578644 5.91421356,6.25 5.5,6.25 C5.08578644,6.25 4.75,6.58578644 4.75,7 L4.75,12 C4.75,12.4142136 5.08578644,12.75 5.5,12.75 C5.91421356,12.75 6.25,12.4142136 6.25,12 Z M1.5,4 L14.5,4 L1.5,4 Z M1.5,3

L14.5,3 C15.0522847,3 15.5,3.44771525 15.5,4 C15.5,4.55228475 15.0522847,5 14.5,5 L1.5,5 C0.94771525,5 0.5,4.55228475 0.5,4 C0.5,3.44771525 0.94771525,3 1.5,3 Z M9.25,3.25 L9.25,2 C9.25,1.86192881 9.13807119,1.75 9,1.75 L7,1.75 C6.86192881,1.75 6.75,1.86192881 6.75,2 L6.75,3.25 L9.25,3.25 Z M7,0.25 L9,0.25 C9.96649831,0.25 10.75,1.03350169 10.75,2 L10.75,4.75 L5.25,4.75 L5.25,2 C5.25,1.03350169 6.03350169,0.25 7,0.25 Z' id='Combined-Shape'%3E%3C/path%3E %3C/g%3E %3C/g%3E %3C/svg%3E"

/***/ }),

/* 350 */

/***/ (function(module, exports) {

module.exports = "data:image/svg+xml,%3C?xml version='1.0' encoding='UTF-8' standalone='no'?%3E %3Csvg width='20px' height='20px' viewBox='0 0 20 20' version='1.1' xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'%3E %3C!-- Generator: Sketch 43.2 (39069) - http://www.bohemiancoding.com/sketch --%3E %3Ctitle%3Ecurved-point%3C/title%3E %3Cdesc%3ECreated with Sketch.%3C/desc%3E %3Cdefs%3E%3C/defs%3E %3Cg id='Page-1' stroke='none' stroke-width='1' fill='none' fill-rule='evenodd'%3E %3Cg id='curved-point'%3E %3Cpath d='M2,15 C2,10.5818452 5.58151214,7 10.000744,7 C14.4184879,7 18,10.5818452 18,15' id='Stroke-3' stroke='%234C97FF' stroke-width='0.75' fill-opacity='0.25' fill='%234C97FF' stroke-linecap='round' stroke-linejoin='round'%3E%3C/path%3E %3Cpath d='M3,7 L17,7' id='Stroke-7' stroke='%234C97FF' stroke-width='0.75' stroke-linecap='round' stroke-linejoin='round'%3E%3C/path%3E %3Ccircle id='Oval-4' fill-opacity='0.25'

fill='%234C97FF' cx='10' cy='7' r='3'%3E%3C/circle%3E %3Ccircle id='Oval-4' fill='%234C97FF' cx='10' cy='7' r='2'%3E%3C/circle%3E %3Ccircle id='Oval-5' fill='%234C97FF' cx='3' cy='7' r='1'%3E%3C/circle%3E %3Ccircle id='Oval-5-Copy' fill='%234C97FF' cx='17' cy='7' r='1'%3E%3C/circle%3E %3C/g%3E %3C/g%3E %3C/svg%3E"

/***/ }),

/* 351 */

/***/ (function(module, exports) {

module.exports = "data:image/svg+xml,%3C?xml version='1.0' encoding='UTF-8' standalone='no'?%3E %3Csvg width='20px' height='20px' viewBox='0 0 20 20' version='1.1' xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'%3E %3C!-- Generator: Sketch 43.2 (39069) - http://www.bohemiancoding.com/sketch --%3E %3Ctitle%3Eflip-horizontal%3C/title%3E %3Cdesc%3ECreated with Sketch.%3C/desc%3E %3Cdefs%3E%3C/defs%3E %3Cg id='Page-1' stroke='none' stroke-width='1' fill='none' fill-rule='evenodd'%3E %3Cg id='flip-horizontal'%3E %3Cg transform='translate(2.000000, 3.000000)'%3E %3Ccircle id='Oval' fill='%23575E75' opacity='0.5' cx='8' cy='0.75' r='1'%3E%3C/circle%3E %3Ccircle id='Oval' fill='%23575E75' opacity='0.5' cx='8' cy='13.25' r='1'%3E%3C/circle%3E %3Ccircle id='Oval-Copy' fill='%23575E75' opacity='0.5' cx='8' cy='3.875' r='1'%3E%3C/circle%3E %3Ccircle id='Oval-Copy-2' fill='%23575E75' opacity='0.5' cx='8' cy='7' r='1'%3E%3C/circle%3E %3Ccircle id='Oval-Copy-3' fill='%23575E75' opacity='0.5' cx='8' cy='10.125' r='1'%3E%3C/circle%3E %3Cpath d='M16,3.08425423 L16,10.9157458

C16,11.4342626 15.2574491,11.6956996 14.8235798,11.3282353 L10.2019293,7.41103711 C9.93269025,7.18445835 9.93269025,6.81408922 10.2019293,6.58751046 L14.8235798,2.67176469 C15.2574491,2.30430042 16,2.56573745 16,3.08425423' id='Fill-11' fill='%234C97FF' opacity='0.5'%3E%3C/path%3E %3Cpath d='M0,10.9157458 L0,3.08425423 C0,2.56573745 0.742550911,2.30430042 1.17470525,2.67176469 L5.79807074,6.58896289 C6.06730975,6.81554165 6.06730975,7.18591078 5.79807074,7.41248954 L1.17470525,11.3282353 C0.742550911,11.6956996 0,11.4342626 0,10.9157458' id='Fill-14' fill='%234C97FF'%3E%3C/path%3E %3C/g%3E %3C/g%3E %3C/g%3E %3C/svg%3E"

/***/ }),

/* 352 */

/***/ (function(module, exports) {

module.exports = "data:image/svg+xml,%3C?xml version='1.0' encoding='UTF-8' standalone='no'?%3E %3Csvg width='20px' height='20px' viewBox='0 0 20 20' version='1.1' xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'%3E %3C!-- Generator: Sketch 43.2 (39069) - http://www.bohemiancoding.com/sketch --%3E %3Ctitle%3Eflip-vertical%3C/title%3E %3Cdesc%3ECreated with Sketch.%3C/desc%3E %3Cdefs%3E%3C/defs%3E %3Cg id='Page-1' stroke='none' stroke-width='1' fill='none' fill-rule='evenodd'%3E %3Cg id='flip-vertical'%3E %3Cg id='flip-horizontal' transform='translate(10.000000, 10.000000) rotate(90.000000) translate(-10.000000, -10.000000) translate(2.000000, 3.000000)'%3E %3Ccircle id='Oval' fill='%23575E75' opacity='0.5' cx='8' cy='0.75' r='1'%3E%3C/circle%3E

%3Ccircle id='Oval' fill='%23575E75' opacity='0.5' cx='8' cy='13.25' r='1'%3E%3C/circle%3E %3Ccircle id='Oval-Copy' fill='%23575E75' opacity='0.5' cx='8' cy='3.875' r='1'%3E%3C/circle%3E %3Ccircle id='Oval-Copy-2' fill='%23575E75' opacity='0.5' cx='8' cy='7' r='1'%3E%3C/circle%3E %3Ccircle id='Oval-Copy-3' fill='%23575E75' opacity='0.5' cx='8' cy='10.125' r='1'%3E%3C/circle%3E %3Cpath d='M16,3.08425423 L16,10.9157458 C16,11.4342626 15.2574491,11.6956996 14.8235798,11.3282353 L10.2019293,7.41103711 C9.93269025,7.18445835 9.93269025,6.81408922 10.2019293,6.58751046 L14.8235798,2.67176469 C15.2574491,2.30430042 16,2.56573745 16,3.08425423' id='Fill-11' fill='%234C97FF' opacity='0.5'%3E%3C/path%3E %3Cpath d='M0,10.9157458 L0,3.08425423 C0,2.56573745 0.742550911,2.30430042 1.17470525,2.67176469 L5.79807074,6.58896289 C6.06730975,6.81554165 6.06730975,7.18591078 5.79807074,7.41248954 L1.17470525,11.3282353 C0.742550911,11.6956996 0,11.4342626 0,10.9157458' id='Fill-14' fill='%234C97FF'%3E%3C/path%3E %3C/g%3E %3C/g%3E %3C/g%3E %3C/svg%3E"

/***/ }),

/* 353 */

/***/ (function(module, exports) {

module.exports = "data:image/svg+xml,%3C?xml version='1.0' encoding='UTF-8' standalone='no'?%3E %3Csvg width='20px' height='20px' viewBox='0 0 20 20' version='1.1' xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'%3E %3C!-- Generator: Sketch 43.2 (39069) - http://www.bohemiancoding.com/sketch --%3E %3Ctitle%3Estraight-

point%3C/title%3E %3Cdesc%3ECreated with Sketch.%3C/desc%3E %3Cdefs%3E%3C/defs%3E %3Cg id='Page-1' stroke='none' stroke-width='1' fill='none' fill-rule='evenodd'%3E %3Cg id='straight-point' fill='%234C97FF'%3E %3Cpolyline id='Path-2' stroke='%234C97FF' stroke-width='0.75' fill-opacity='0.25' stroke-linecap='round' stroke-linejoin='round' points='2 15 10 7 18 15'%3E%3C/polyline%3E %3Ccircle id='Oval-4' fill-opacity='0.25' cx='10' cy='7' r='3'%3E%3C/circle%3E %3Ccircle id='Oval-4' cx='10' cy='7' r='2'%3E%3C/circle%3E %3C/g%3E %3C/g%3E %3C/svg%3E"

/***/ }),

/* 354 */

/***/ (function(module, exports) {

module.exports = "data:image/svg+xml,%3C?xml version='1.0' encoding='UTF-8'?%3E %3Csvg width='20px' height='20px' viewBox='0 0 20 20' version='1.1' xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'%3E %3C!-- Generator: Sketch 50.2 (55047) - http://www.bohemiancoding.com/sketch --%3E %3Ctitle%3Eoval-outlined%3C/title%3E %3Cdesc%3ECreated with Sketch.%3C/desc%3E %3Cdefs%3E%3C/defs%3E %3Cg id='oval-outlined' stroke='none' stroke-width='1' fill='none' fill-rule='evenodd'%3E %3Cg id='Group' transform='translate(4.000000, 4.000000)' fill='%23575E75'%3E %3Cpolygon id='Fill-1' points='0 9.33333333 1.33333333 9.33333333 1.33333333 2.66666667 0 2.66666667'%3E%3C/polygon%3E %3Cpolygon id='Fill-2' points='1.33333333 2.66666667 2.66666667 2.66666667 2.66666667 1.33333333 1.33333333 1.33333333'%3E%3C/polygon%3E %3Cpolygon id='Fill-3' points='1.33333333

10.6666667 2.66666667 10.6666667 2.66666667 9.33333333 1.33333333 9.33333333'%3E%3C/polygon%3E %3Cpolygon id='Fill-4' points='2.66666667 1.33333333 9.33333333 1.33333333 9.33333333 0 2.66666667 0'%3E%3C/polygon%3E %3Cpolygon id='Fill-5' points='9.33333333 2.66666667 10.6666667 2.66666667 10.6666667 1.33333333 9.33333333 1.33333333'%3E%3C/polygon%3E %3Cpolygon id='Fill-6' points='10.6666667 9.33333333 12 9.33333333 12 2.66666667 10.6666667 2.66666667'%3E%3C/polygon%3E %3Cpolygon id='Fill-7' points='9.33333333 10.6666667 10.6666667 10.6666667 10.6666667 10.6666667 9.33333333 9.33333333 9.33333333'%3E%3C/polygon%3E %3Cpolygon id='Fill-8' points='2.66666667 12 9.33333333 12 9.33333333 10.6666667 2.66666667 10.6666667'%3E%3C/polygon%3E %3C/g%3E %3C/g%3E %3C/svg%3E"

/***/ }),

/* 355 */

/***/ (function(module, exports) {

module.exports = "data:image/svg+xml,%3C?xml version='1.0' encoding='UTF-8'?%3E %3Csvg width='20px' height='20px' viewBox='0 0 20 20' version='1.1' xmlns='http://www.w3.org/2000/svg' xmlns:xlink='http://www.w3.org/1999/xlink'%3E %3C!-- Generator: Sketch 50.2 (55047) - http://www.bohemiancoding.com/sketch --%3E %3Ctitle%3Erectange-outlined%3C/title%3E %3Cdesc%3ECreated with Sketch.%3C/desc%3E %3Cdefs%3E%3C/defs%3E %3Cg id='rectange-outlined' stroke='none' stroke-width='1.33333333' fill='none' fill-rule='evenodd'%3E %3Crect id='rectangle-icon' stroke='%23575E75' x='4.5' y='4.5' width='11' height='11'%3E%3C/rect%3E

%3C/g%3E %3C/svg%3E"

/***/ }),

/* 356 */

/***/ (function(module, exports, __webpack_require__) {

"use strict";

ДОДАТОК Б


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МІНІСТЕРСТВО ІНФРАСТРУКТУРИ УКРАЇНИ
УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ ТА ТЕХНОЛОГІЙ
СХІДНИЙ НАУКОВИЙ ЦЕНТР ТРАНСПОРТНОЇ АКАДЕМІЇ НАУК
ABSTRACTS
OF THE XVII INTERNATIONAL CONFERENCE
«MODERN INFORMATION AND COMMUNICATION
TECHNOLOGIES ON A TRANSPORT, IN INDUSTRY AND
EDUCATION»
13-14, December, 2023
АВТОМАТИЗАЦІЯ
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ
КІБЕРБЕЗПЕКА
КОМУНІКАЦІЯ
НАВЧАННЯ
СУЧАСНІ ІНФОРМАЦІЙНІ ТА КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ НА ТРАНСПОРТІ, В ПРОМИСЛОВОСТІ ТА ОСВІТІ
ТЕЗИ
XVII МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ
13-14 ГРУДНЯ 2023
ДНІПРО
2023

Дослідження засобів та технологій обробки векторної графіки

Багно С. С., Горячкін В.М., Український державний університет науки і технологій

Останнім часом великою популярністю користується цифрова обробка зображень. Цифрове зображення є графічною формою інформації, яка призначена для візуального сприйняття. Після кодування за допомогою спеціальних алгоритмів та збереження на носії, цей набір даних стає файлом. У сучасному поліграфічному виробництві практично всі ілюстрації та елементи дизайну представлені у вигляді різних типів цифрових зображень.

Комп'ютерна графіка - це науково-технологічний напрямок, який займається створенням, обробкою та зберіганням зображень за допомогою комп'ютера та його апаратних та програмних можливостей. Векторна графіка є одним із типів комп'ютерної графіки, вона використовується для створення зображень на основі векторів та математичних об'єктів.

Векторна графіка застосовується в різних сферах, включаючи інженерію та науку, бізнес, мистецтво та сферу розваг. Векторна графіка особливо корисна у формуванні зображень для веб-сайтів і є основним інструментом веб-майстрів. Оскільки перетворені з векторного формату в растровий, зображення майже завжди є унікальними, пошукові системи використовують їх для ранжирування зображень. Растрові зображення зберігаються у файлах у вигляді таблиці, де кожна клітинка містить двійковий код кольору відповідного пікселя. Такий файл зберігає дані про інші характеристики графічного зображення, а також алгоритм його стискання.

Графічний редактор – це спеціальна програма або набір програм, призначених для створення або редагування зображень на комп'ютері. Такі редактори дозволяють як створювати нові зображення, так і редагувати наявні. Вони дають можливість переміщати, обертати, видаляти або копіювати елементи зображення. Результат можна одразу ж зберегти або надрукувати.

При виборі архітектури та середовища розробки векторного редактора для веб-системи важливо врахувати переваги архітектури клієнт-сервер. Використання такої моделі взаємодії має кілька переваг, зокрема чітке розділення функціоналу між клієнтською та серверною частинами, що дозволяє зменшити обчислювальне навантаження. Клієнтська частина системи при цьому складається з різноманітних компонентів, таких як окремі сторінки чи їх елементи, що функціонують як самостійні частини. Така структура сприяє лаконічності коду та уникненню його дублювання. Клієнтська частина відіграє ключову роль у системі, оскільки містить основну логіку користувача, від якої залежить значна частина успіху системи. Серверна частина має надавати доступ до бази даних, а також конвертувати дані у зручний формат для користувача, реалізовувати взаємодію із сервером.

Розвиток потужностей комп’ютерної техніки створює стійку основу для інвестицій у компанії, що спеціалізуються на графічних інтерфейсах користувача, об’єктно-орієнтованих програмах, віртуальній реальності та програмному забезпеченні паралельних процесів. Внаслідок цього очікується стійке зростання у цій сфері до кінця цього десятиліття, що підтверджується тенденціям на його початку, коли щорічне зростання світового ринку послуг у сфері комп’ютерної графіки становило близько 12%. Невідомо, як довго триватиме така тенденція, проте найближчими роками очікується на її продовження з стійке щорічним ростом щонайменше у 10%.

Виконані в роботі дослідження дозволяють покращити роботу з об’єктами векторної графіки та надають нові можливості при формуванні зображень для веб-сайтів і будуть корисними для веб-майстрів, які працюють з ними.

ДОДАТОК В

Технічне завдання

ЗАТВЕРДЖУЮ
Проректор Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ОБРОБКИ ВЕКТОРНОЇ ГРАФІКИ

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01271 – 01 – ЛЗ

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____ Вадим ГОРЯЧКІН
Виконавець
_____Сергій БАГНО
Нормоконтролер
_____Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО

1116130.01271 – 01

ДОСЛІДЖЕННЯ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ОБРОБКИ ВЕКТОРНОЇ ГРАФІКИ

Технічне завдання

Листів 24

2024

ЗМІСТ

ВСТУП

На даний час в наше життя все більше проникає інформація, що вимагає спеціальних методів обробки. Багато галузей техніки, що стосуються отримання, обробки, зберігання та передачі інформації, що значною мірою орієнтуються в даний час на розвиток систем, в яких інформація має характер зображень. Зображення, яке можна розглядати як двовимірний сигнал є значно більш ємним носієм інформації, ніж звичайний одновимірний (тимчасовий) сигнал.

В останні роки суттєво зріс інтерес до цифрової обробки зображень, тому зовсім не випадково, що цифрова обробка та розпізнавання зображень є одним з напрямків, що інтенсивно розвиваються.

Цифрове зображення – графічна форма представлення даних, призначена для зорового сприйняття. Будучи закодованим за допомогою особливого алгоритму та записаним на носій, цей масив даних стає файлом.

У сучасному процесі поліграфічного виробництва всі ілюстрації та елементи оформлення представлені цифровими зображення різних типів.

У комп’ютерних системах, коли одержувачем інформації є людина, велике значення мають методи покращення зображень, що дозволяють підвищити помітність деталей, що цікавлять на зображенні. Крім того, при попередній обробці зображень, що виконується в автоматичних комп’ютерних системах, також важливу роль відіграє попередня обробка зображень, що дозволяє сформувати простір ознак об’єктів.

Огляд літератури показав слабку вивченість комп’ютерної графіки як виду особливого мистецтва. Серйозною проблемою в контексті сучасної реальності є недостатня увага до навчання комп’ютерної графіки, а така ж відсутність необхідних умов, коштів та кваліфікованих фахівців у даній області. Тому необхідно створити всі умови для реалізації навчання комп’ютерної графіки. Потреба у застосуванні інформаційних комп’ютерних технологій у всіх галузях сучасного суспільства підтверджує доцільність, своєчасність та актуальність даного дослідження.

Об’єктом дослідження є цифрові векторні зображення.

Предметом дослідження є методи та алгоритми обробки векторних зображень.

З огляду на це метою роботи є дослідження методів і алгоритмів цифрової обробки зображень, що забезпечують якісне та чітке відображення а також розробка рекомендації щодо застосування найоптимальніших алгоритмів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести огляд предметної області цифрової обробки зображень;
- проаналізувати існуючі способи опису цифрових зображень;
- проаналізувати існуючі методи та алгоритми цифрової обробки зображень;
- дослідити ефективність алгоритмів оцінки параметрів зображень;
- обґрунтувати вибір інструментальних засобів розробки програмного засобу;
- розробити програму для обробки векторної графіки;
- показати основні режими функціонування програмного засобу;
- організувати тестування та налагодження програмного засобу.

Використані методи: системно-структурний та порівняльний аналіз – використаний при аналізі проблем програмного забезпечення, призначеного для обробки векторної графіки; методи формально-логічного аналізу – при обґрунтуванні вибору методів розробки програмного засобу; економіко-статистичні методи – при дослідження тенденцій програмних засобів; системний підхід – заснований на представленні досліджуваного об’єкта в ясно організованій цілісності.

Основною гіпотезою, покладеною в основу даної роботи, є можливість обробки цифрових svg форматів для цілей масштабування до будь-якого розміру без втрати якості, в першу чергу, зменшення розміру файлу, використовуючи алгоритм стиснення та обробки.

Здебільшого цифрові векторні зображення мають використовуватися при розробці веб-сайтів, де найчастіше розробники в силу нестачі часу, забувають про найголовніше – якісне відображення зображенні та використовують растрові зображення.

Ця робота присвячена аналізу існуючих моделей та алгоритмів, призначених для цифрової обробки зображень, зокрема векторні зображення svg. Використання нижчеописаних моделей та алгоритмів дозволить скоротити обсяг пам’яті, необхідної для зберігання зображень, а також забезпечити якісне відображення зображень на веб-сторінках.

1 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 1196 ст від 05.12.2022 року ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем магістерських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Векторна графіка є важливою складовою сучасного інформаційного простору та графічного дизайну. Засоби та технології обробки векторної графіки відіграють ключову роль у створенні високоякісних зображень, логотипів, ілюстрацій та інших графічних елементів. Дослідження цієї теми визначить сучасні тенденції та можливості, які вона пропонує.

Мета дослідження: Визначення сучасних засобів та технологій обробки векторної графіки, їх функціональність та застосування в галузі дизайну та інформаційних технологій.

2.1 Функціональне призначення

Дослідження засобів та технологій обробки векторної графіки має на меті вивчення сучасних інструментів та підходів, спрямованих на створення, редагування та оптимізацію векторних зображень. Основні функціональні завдання включають:

1. Аналіз Засобів Векторної Графіки:

Вивчення основних функціональних можливостей векторних графічних редакторів.

Порівняння інструментів, визначення їхнього призначення та області застосування.

2. Дослідження Технологій Програмування:

Вивчення бібліотек та інструментів для обробки векторної графіки у програмному забезпеченні.

Аналіз інтеграції векторної графіки у веб-розробці, мобільних додатках та інших програмних проектах.

3. Застосування в Професійному Дизайні:

Визначення ролі векторної графіки у графічному дизайні, створенні логотипів, ілюстраціях та інших графічних елементах.

Аналіз технологічних вимог та особливостей використання векторних зображень у дизайнерських процесах.

4. Практичне Застосування у Виробництві:

Вивчення використання векторної графіки у виробничих процесах, таких як CNC-обробка та створення макетів для виробництва.

Аналіз можливостей оптимізації виробничих процесів за допомогою векторної графіки.

5. Перспективи та Інновації:

Вивчення тенденцій розвитку векторної графіки та її вплив на сучасні технології.

Аналіз можливостей впровадження новаторських рішень, таких як штучний інтелект та машинне навчання, у обробку векторної графіки.

Дослідження забезпечить обґрунтоване розуміння сучасних можливостей та переваг використання засобів та технологій обробки векторної графіки у різних галузях, відкриваючи перспективи для подальшого розвитку та вдосконалення цієї сфери.

2.2 Експлуатаційне призначення

Дослідження засобів та технологій обробки векторної графіки спрямоване на оптимізацію та раціоналізацію процесів створення, редагування та використання векторних зображень. Експлуатаційне призначення включає наступні аспекти:

1. Вдосконалення Робочих Процесів:

Забезпечення оптимальних умов для дизайнерів та розробників шляхом вибору ефективних інструментів та технологій обробки векторної графіки.

Спрощення та прискорення створення та редагування графічних елементів.

2. Інтеграція з Іншими Програмними Засобами:

Забезпечення сумісності з іншими графічними та програмними рішеннями для зручного обміну та обробки векторних зображень.

Можливість інтеграції з веб-сервісами, мобільними додатками та іншими платформами.

3. Оптимізація Дизайнерських Процесів:

Сприяння розробці високоякісних графічних рішень для логотипів, ілюстрацій, рекламних матеріалів тощо.

Забезпечення інструментів для швидкого створення та адаптації дизайнерських концепцій.

4. Забезпечення Працездатності у Виробничому Середовищі:

Впровадження технологій, що підтримують виробничі процеси, такі як CNC-обробка та виготовлення макетів.

Максимізація якості виробничих графічних матеріалів.

5. Підтримка Інновацій та Розвитку:

Забезпечення платформи для експериментів з новими технологіями, такими як штучний інтелект, машинне навчання та розширена реальність у контексті векторної графіки.

Розгляд можливостей для впровадження новаторських ідей у процеси роботи з векторною графікою.

Експлуатаційне призначення дослідження засобів та технологій обробки векторної графіки полягає у покращенні продуктивності та якості роботи з векторними зображеннями, забезпечуючи оптимальні умови для творчих та виробничих процесів.

3 ОПИС ПРОЕКТУВАННЯ ТА РОЗРОБКИ ПРОГРАМНОГО ЗАСОБУ ДЛЯ СТВОРЕННЯ ТА РЕДАГУВАННЯ ВЕКТОРНОЇ ГРАФІКИ

3.1 Вибір моделі розробки програмного засобу

Для розробки програмного забезпечення ми вибрали каскадну модель (Waterfall) – це модель процесу розробки програмного забезпечення, в якій процес розробки виглядає як потік, що послідовно проходить фази аналізу вимог, проектування, реалізації, тестування, інтеграції та підтримки.

Основна суть моделі Waterfall у тому, що етапи залежать один від одного і наступний починається, коли закінчено попередній, утворюючи таким чином каскадний рух уперед.

Паралелізм етапів у каскадній моделі, хоч і обмежений, але можливий для абсолютно незалежних робіт. При цьому інтеграція паралельних шматків все одно відбувається на наступному етапі, а не в рамках одного.

Команди різних етапів між собою не спілкуються, кожна команда відповідає чітко за свій етап.

Для замовників дана модель виглядає лінійно і з боку досить просто: із завершеного етапу проектування випливає програмування, а потім тестування – і так крок за кроком поки не буде досягнуто фінальної точки і мети, заради якої ведеться розробка.

Однак уявлення про простоту каскадної моделі є ілюзорним. Воно з’являється через обмежене не бачення клієнтом всього процесу, адже дана модель не передбачає залучення замовника до деталей процесів розробки, і демонструє зрозумілий і кінцевий результат роботи тільки на контрольних точках і наприкінці проекту.

Насправді каскадну модель не можна назвати простою, практично нею складно управляти. Внесення замовником значних змін у процес розробки з waterfall або опрацювання серйозних не передбачених проектом ризиків несуть руйнівний характер для всього процесу – модель доводиться перебудовувати, графіки перепланувати.

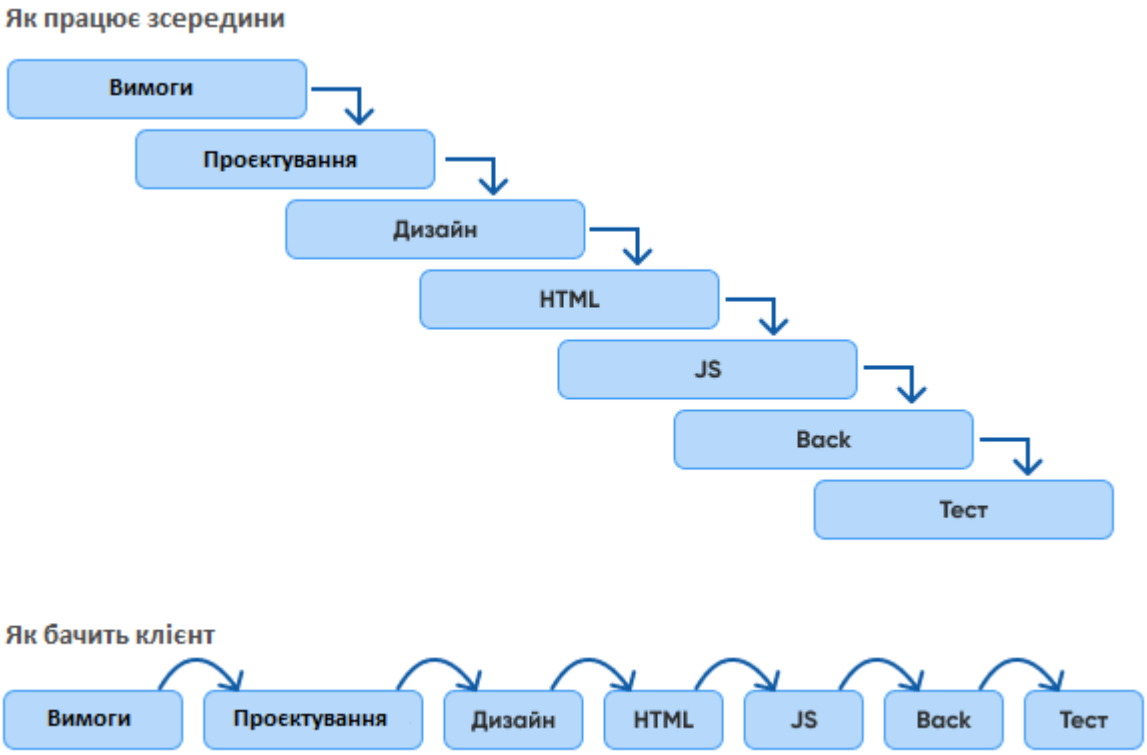

Рис. 3.1. Візуальний вигляд каскадної моделі

Наведемо переваги каскадної моделі:

Розробку легко контролювати. Замовник завжди знає, чим зараз зайняті програмісти, може керувати термінами та вартістю.

Вартість проекту визначається на початковому етапі. Усі кроки заплановані вже на етапі узгодження договору, ПЗ пишеться безперервно «від і до».

Не потрібно наймати тестувальників із серйозною технічною підготовкою. Тестувальники зможуть спиратися на детальну технічну документацію.

Недоліки каскадної моделі:

Тестування розпочинається на останніх етапах розробки. Якщо в вимогах до продукту припустилися помилки, то виправити її коштуватиме дорого. Тестувальники виявлять її, коли розробник уже написав код, а техніки – документацію.

Замовник бачить готовий продукт наприкінці розробки і лише тоді може дати зворотний зв’язок. Велика ймовірність того, що результат його не

влаштує.

Розробники пишуть багато технічної документації, що затримує роботу. Чим більша документація у проекті, тим більше змін потрібно вносити та довше їх узгоджувати.

«Водоспад» підходить для розробки проектів у медичній та космічній галузі, де вже сформовано велику базу документів, на основі яких можна написати вимоги до нового ПЗ.

При роботі з каскадною моделлю основне завдання – написати докладні вимоги до розробки. На етапі тестування не має з’ясуватися, що у них є помилка, що впливає весь продукт.

3.2 Опис алгоритму розв’язування задачі

Візуальне моделювання в UML можна представити як деякий процес ступеневого спуску від найбільш загальній і абстрактній концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної інформаційної системи. Для досягнення цих цілей з автоматизації, спочатку будується модель у формі діаграми прецедентів тобто варіантів використання, яка описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування

Таким чином, діаграма варіантів використання є вихідним концептуальним уявленням або концептуальною моделлю системи в процесі її проектування і розробки.

Далі слід врахувати, що кожен прецедент системи має певну поведінку, може знаходитись в певних станах, а також переходити з одного стану в інший. Тобто, він моделює всі зміни станів певного варіанту використання як його реакцію на зовнішні впливи.

В разі чого, для специфікації функціональності прецеденту, що має складну модель поведінки, програмну систему з автоматизації ремонтно-будівельної компанії представляється з точки зору теорії кінцевих автоматів, тобто у формі діаграми станів

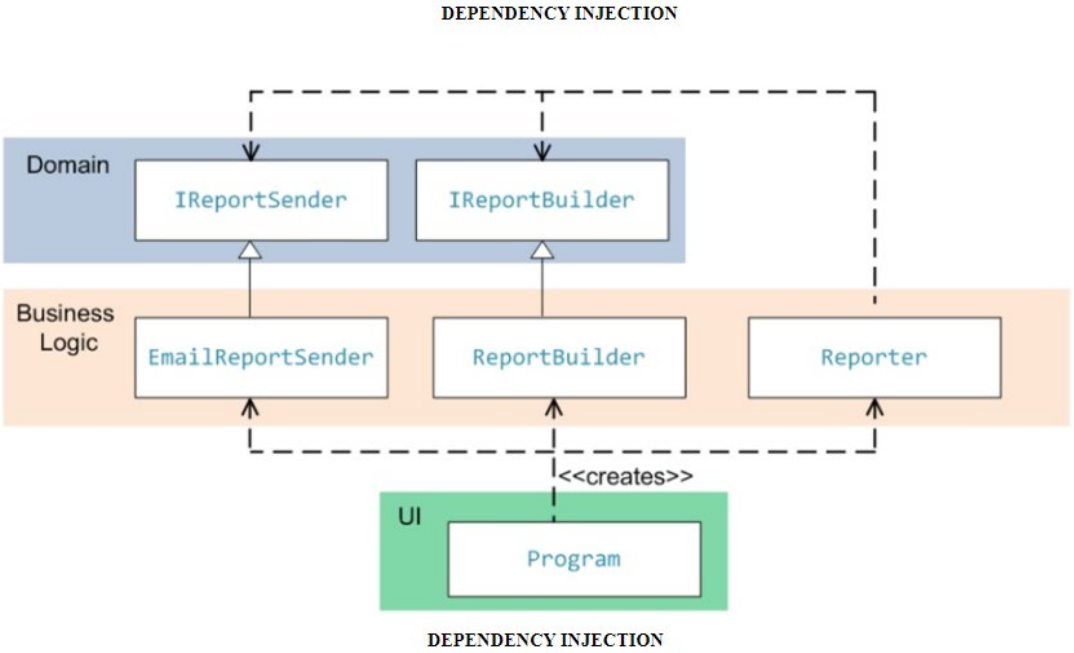

Рисунок 3.2 – Модель ін’єкції залежностей

Діаграми варіантів використання описують взаємини і залежності між групами варіантів використання і дійових осіб, які беруть участь в процесі див. рис. 3.2. Важливо розуміти, що діаграми варіантів використання не призначені для відображення проекту і не можуть описувати внутрішній устрій системи. Діаграми варіантів використання призначені для спрощення взаємодії з майбутніми користувачами системи, з клієнтами, і особливо знадобляться для визначення необхідних характеристик системи. Іншими словами, діаграми варіантів використання говорять про те, що система повинна робити, не вказуючи самі застосовувані методи.

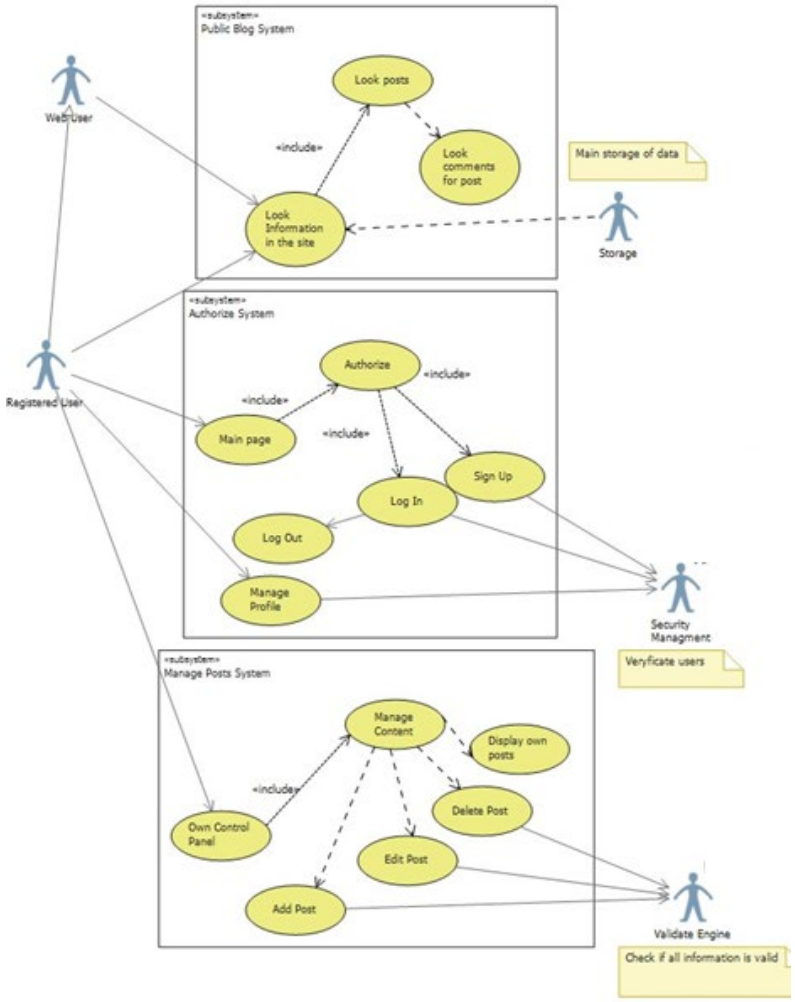

Рисунок 3.3 – Загальна діаграма варіантів використання

При роботі з варіантами використання важливо пам’ятати декілька простих правил:

- кожен варіант використання відноситься як мінімум до одної дійової особи;
- кожен варіант використання має ініціатора;
- кожен варіант використання призводить до відповідного результату.

Варіанти використання також можуть взаємодіяти з іншими варіантами використання. Єдиний актор проектованої системи, буде взаємодіяти із всіма варіантами використання. У модель включено асоціації, що відповідають за напрямки передачі інформації між актором і варіантами використання.

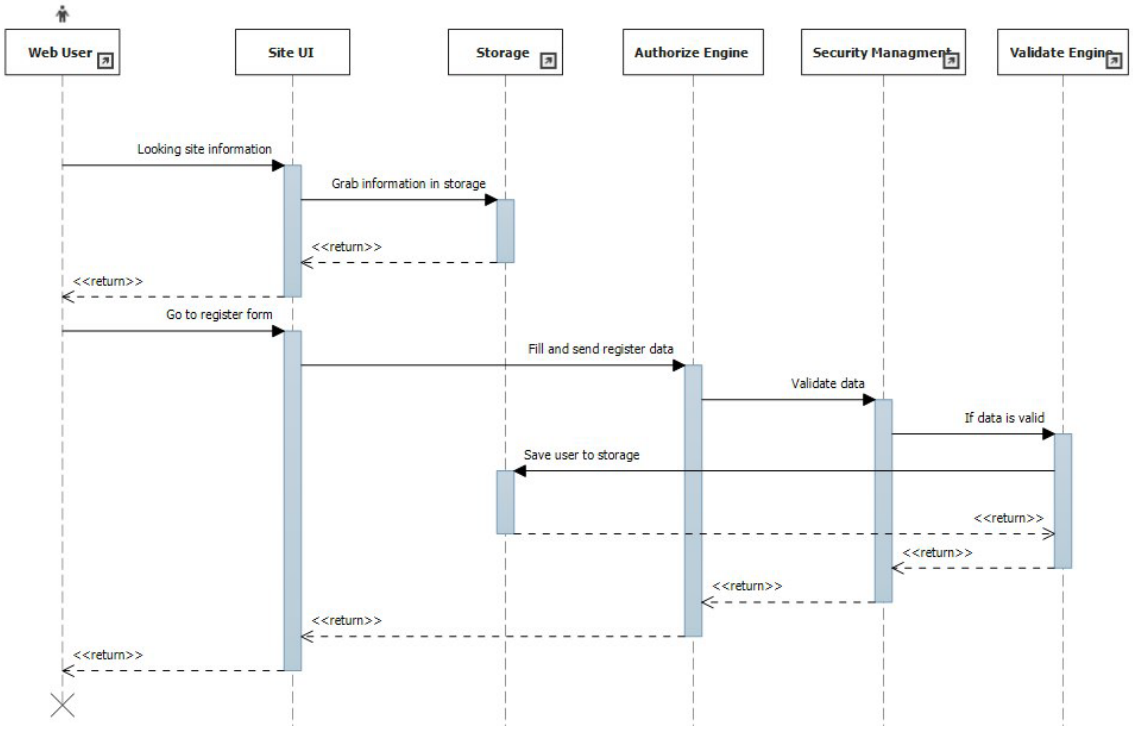

Рисунок .5 – Діаграма послідовностей

3.3 Основні режими функціонування програмного засобу

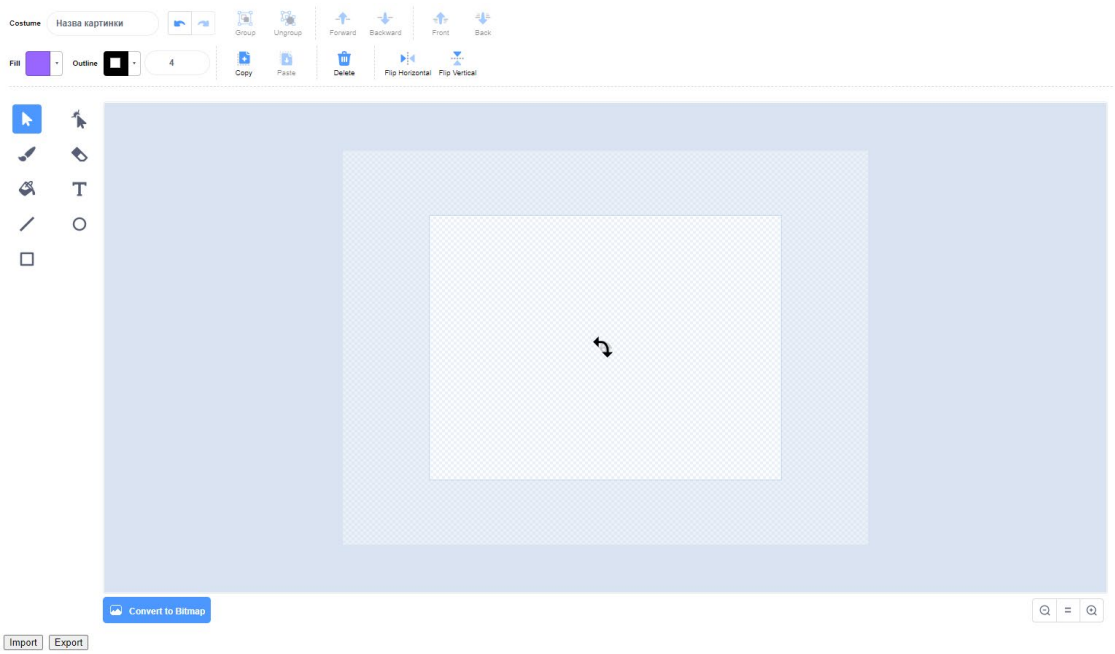

Рисунок 4.6 – Головна сторінка програмного засобу

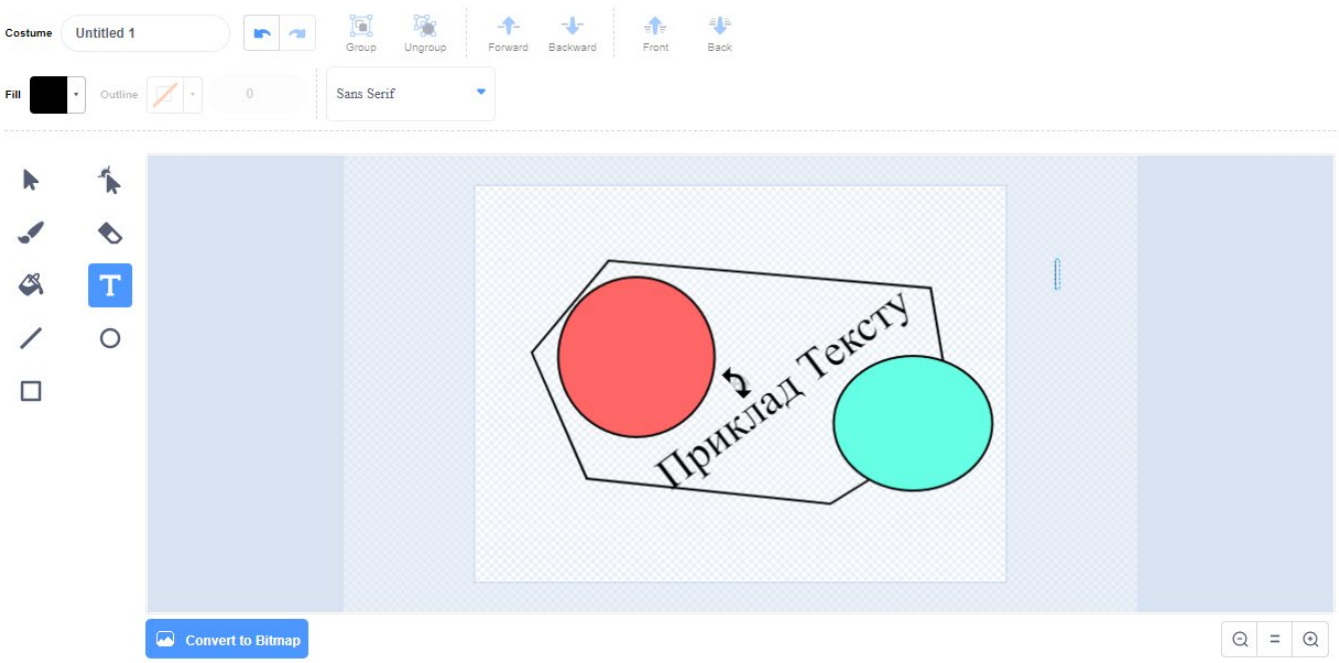

Рисунок 4.7 – Приклад створеної просто векторної картинки

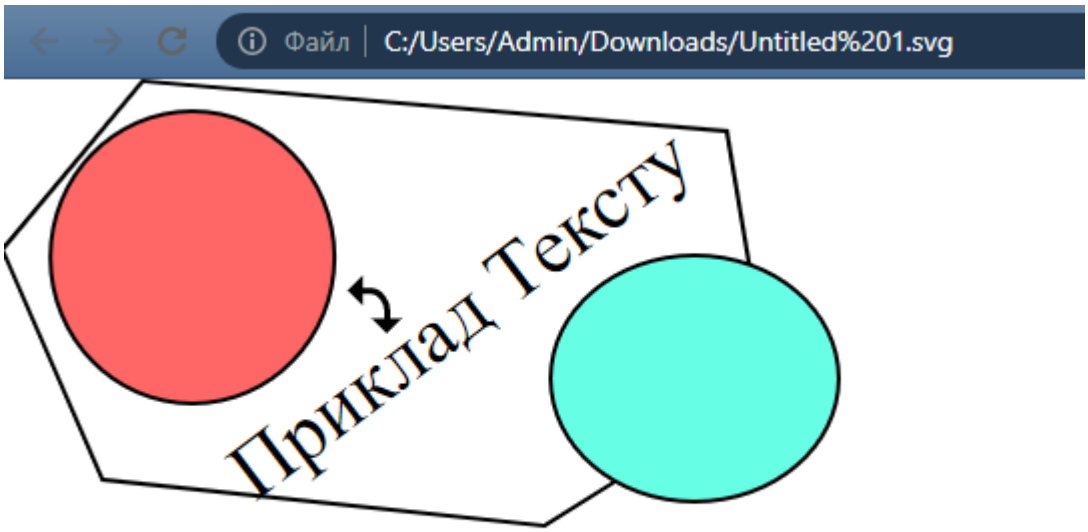

Рисунок 4.8 – Збережена картинка у форматі svg і відкрита у Google Chrome

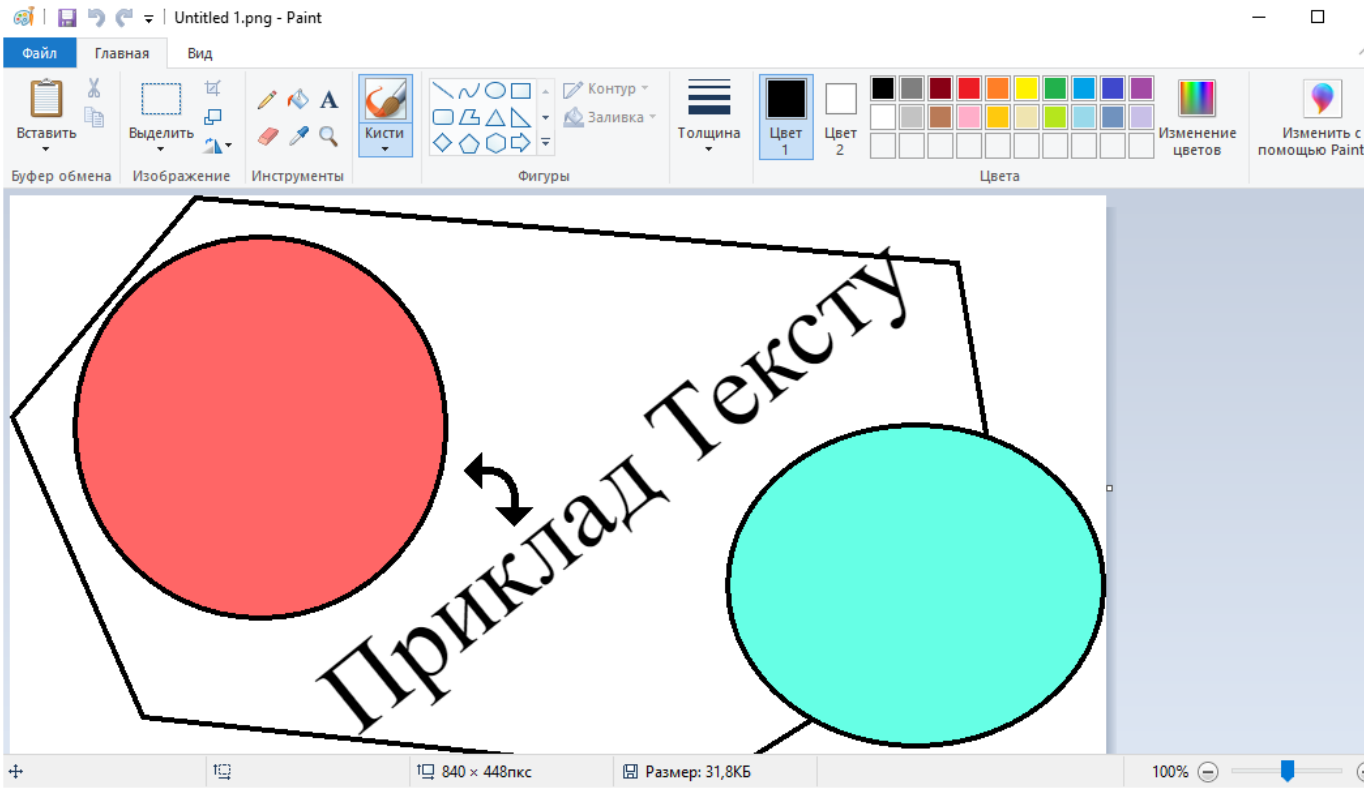

Рисунок 4.9 – Збережена картинка у форматі png і відкрита у Paint

3.4 Організація тестування веб-ресурсу

Тестування, як заключний етап розробки системи, виконує важливу роль в процесі створення високоякісного проекту. Після тестування замовнику надається готовий проект без помилок, з хорошою читабельністю, легкою логікою, зручністю і надійністю.

Для організації тестування системи розробили спеціальну методологію, згідно з якою була здійснена перевірка. Тестування можна проводити в різний спосіб, нижче наведено перелік правил, за допомогою яких провели тестування для програмного засобу.

Основні етапи:

Найперше було проведено функціональне тестування, тобто було перевірено, щоб кожна функція системи працювала відповідно до вимог технічного завдання.

Наступний етап тестування – це юзабіліті тестування, який призначений для оцінки системи з точки зору кінцевого користувача. Даний етап допомагає визначити відповідність продукту до очікувань користувачів, виявляє

проблемні місця в інтерфейсі.

Навігаційне тестування:

- перевірили усі кнопки, форми і поля чи є зручними вони для використання;

Тестування контенту:

- перевірили чи контент є інформативним, зрозумілим, структурованим і логічним.
- перевірили чи відсутні граматичні, орфографічні помилки.
- перевірили колірну палітру системи і розмір шрифтів.

Також було проведено тестування інтерфейсу користувача (UserInterface), яке виконується для перевірки відповідності графічного користувацького інтерфейсу системи до технічного завдання.

Було проведено тестування сумісності, яке виконується для перевірки роботи системи при різних програмних і апаратних конфігураціях. Кросплатформове тестування системи дозволяє оцінювати роботу сайту при різних ОС: Windows, iOS/Mac OS, Linux, Android, BlackBerry тощо.

Кросбраузерні тестування системи допомагає перевірити правильність роботи сайту в різних браузера: MozillaFirefox, GoogleChrome, Internet Explorer, Opera.

- перевірили чи навігація по системі є максимально простою;
- перевірили оптимізацію часу завантаження системи;
- перевірили чи кнопки мають достатній розмір для натиснення.

Бета-тестування – заключна стадія тестування. Як правило, це роблять кінцеві користувачі, які не є розробниками.

При бета-тестуванні система потрапляє в руки реальних користувачів, щоб виявити будь-які недоліки з їх точки зору, які виправляються для остаточної, релізної версії.

3.5 Вимоги до надійності

Вимоги до надійності наступні:

- при оновленні даних забезпечити показ відповідних повідомлень про стан роботи програми та результати виконання операцій;
- наявність архівної копії тексту програми на зовнішньому носії;
- наявність резервної копії бази даних на зовнішньому носії.

3.6 Вимоги експлуатації

Працювати з програмою може людина, що має навички роботи з комп’ютерами та ознайомлена з керівництвом користувача програмного продукту.

3.7 Вимоги до складу та параметрів технічних засобів

Продукти, що розробляється повинен використовуватись на пристроях, що мають наступні характеристики:

- роздільна здатність дисплею – HD (1980x1020);
- оперативна пам’ять – 4 ГБ;
- вбудована пам’ять – 16 ГБ;
- операційна система – WINDOWS;
- частота процесора – 1.6 ГГц;
- комунікації – microUSB.

3.8 Вимоги до інформаційної та програмної сумісності

Програмні продукти розробляється для всіх видів операційних систем сімейства “WINDOWS” починаючи від версії 10 та наступні версії.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документації мають входити:

- специфікація;
- текст програми;
- опис програми;
- керівництво користувача для користування мобільним додатком.

Вся документація програмних додатків повинна задовольняти вимоги до програмної документації.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиця 5.1 – Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	04.09.23 –15.09.23
Робочий проект	Програмування та відлагодження програми.	18.09.23 – 22.09.23
	Тестування програми	25.09.23 – 27.09.23
	Розробка, узгодження і затвердження програмної документації.	28.09.23 – 29.09.23

6 ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки доц. Горячкін В. М.

7 ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Показники та їх розрахунок не були описані у документах бо розробка мобільного додатку несе в собі навчальний характер, а не комерційний.