НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри _____
А.І.Петренко
(підпис) (ініціали, прізвище)
"___"_____2016 р.

Дипломна робота

першого (бакалаврського)**рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: «Canvas як засіб включення графічних та мультимедійних компонентів на Web-сайти та його аналіз»
Виконав: студент IV курсу, групи ДА-21
(шифр групи)

Будник Сергій Анатолійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник к.т.н., доц. Цурін О.П. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічний проф., док. ек. н., Семенченко Н.В. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)
Нормоконтроль ст.. викладач Британ О.А. _____

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

Київ – 2016 року

Національний технічний університет України

«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський) (перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

«___»_____2016 р.

ЗАВДАННЯ на дипломний проект (роботу) студенту

Буднику Сергію Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)) Canvas як засіб включення графічних та мультимедійних компонентів на Web-сайти та його аналіз

керівник проекту (роботи))Цурін О.П., к.т.н., доц.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 2016 р. №_____

2. Строк подання студентом проекту (роботи) 10.06.2016

3. Вихідні дані до проекту (роботи)
Форма реалізації – у вигляді Веб-додатку з використанням тегу Canvas. Для розробки анімації та ігри використовувати різні сторінки. Дослідити алгоритми для побудови 3D графіків та графічного редактора. Для реалізації використовувати технологію HTML5.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)
   1. Розглянути тег Canvas, як нову сторінку в історії мультимедіа.
   2. Розглянути можливості для використання Canvas.
   3. Програмна реалізація гри, побудови 3D графіків та графічного редактора.
   4. Описати функціонально вартісний аналіз розробленого додатку.

5. Проаналізувати результати роботи програми, зробити висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Приклади використання HTML5 Canvas – проектор.
2. Порівняння HTML5 Canvas та Flash – плакат.
3. Презентація роботи додатків – плакат, проектор.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Семенченко Н.В. проф. док.ек.н		
Основна частина			

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2016	
2	Збір інформації	15.02.2016	
3	Вивчення алгоритмів для побудови ігри, 3D графіків та графічного редактора	28.02.2016	
4	Розробка алгоритму та структури системи	10.03.2016	
5	Розробка плану тестування	15.03.2016	
6	Розробка програмної моделі	25.03.2016	
7	Розробка опису	25.04.2016	
8	Тестування додатку та отримання результатів	30.04.2016	
9	Оформлення дипломної роботи	31.05.2016	
10	Отримання допуску до захисту та подача роботи в ДЕК	10.06.2016	

Студент _____ Будник С.А.
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ Цурін О.П.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

бакалаврської дипломної роботи Будника Сергія Анатолійовича
на тему «Canvas як засіб включення графічних та мультимедійних компонентів на Web-сайти та його аналіз»

Мета роботи полягає у аналізі нововведення HTML5, а саме тегу Canvas, який дозволяє включати графічні та мультимедійні компоненти на Web-сайти. Метою дослідження є створення лабораторної роботи з курсу «Комп'ютерна графіка» та розробка програмного продукту для демонстрування можливостей HTML5 Canvas, таких як робота з зображеннями, побудова 3D поверхонь та демонстрація гри. Проект був реалізований на мові HTML з використанням CSS та JS.

Практичні результати роботи – демонстрація можливостей мультимедіа та графіки, виконаних з допомогою HTML5 Canvas. Демонстрація можливостей уставлена в Moodle за адресою http://info.kpi.ua/mod/lesson/view.php?id=186 у розділ «Робота за Canvas» (Урок 3) курсу «Веб програмування».

Загальний обсяг роботи: 95 сторінок, 21 рисунок, 10 таблиць, 1 додаток на 9 стр., 2 додаток на 9 стр., 15 посилань.

Ключові слова: мультимедіа, графіка, алгоритми малювання, кросбраузерність, HTML5, Canvas, Web-сайт, Web-додаток.

АННОТАЦИЯ

бакалаврской дипломной работы Будника Сергея Анатольевича
на тему «Canvas как средство включения графических и мультимедийных компонентов на Web-сайты и его анализ»

Цель работы заключается в анализе нововведения HTML5, а именно тега Canvas, который позволяет включать графические и мультимедийные компоненты на Web-сайты. Целью исследования является создание лабораторной работы по курсу «Компьютерная графика» и разработка программного продукта для демонстрации возможностей HTML5 Canvas, таких как работа с изображениями, построение 3D поверхностей и демонстрация игры. Проект был реализован на языке HTML с использованием CSS и JS.

Практические результаты работы - демонстрация возможностей мультимедиа и графики, выполненных с помощью HTML5 Canvas. Демонстрация возможностей вставлена в Moodle по адресу http://info.kpi.ua/mod/lesson/view.php?id=186 в раздел «Работа с Canvas» (Урок 3) курса «Веб программирование».

Общий объем работы 95 страниц, 21 рисунок, 10 таблиц, 1 приложение на 9 стр. 2 приложение на 9 стр., 15 ссылок.

Ключевые слова: мультимедиа, графика, алгоритмы рисования, кроссбраузерность, HTML5, Canvas, Web-сайт, Web-приложение.

ANNOTATION

on Sergey Budnik bachelor's degree

on «Canvas as a way to include graphics and multimedia components on the Web-site and its analysis"

Purpose of project is to analyze innovation of HTML5, namely tag Canvas, which allows us to include graphics and multimedia components on the Web-site. The study is the establishment of the laboratory work on the course "Computer Graphics" and development of software to demonstrate capabilities of HTML5 Canvas, such as imaging, 3D surface construction and demonstration of the game. The project was implemented in HTML using CSS and JS.

Practical results - demonstration of multimedia and graphics made using HTML5 Canvas. Demonstration inserted into Moodle at http://info.kpi.ua/mod/lesson/view.php?id=186 in "Working on Canvas» (Lesson 3) the course "Web programming".

The total amount of work: 95 pages, 21 figures, 10 tables, 1 appendix for 9 p. 2 appendix for 9 p., 15 references.

Keywords: multimedia, graphics, drawing algorithms, Cross-browser compatibility, HTML5, Canvas, Web-site, Web-application.

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПЗ – програмне забезпечення.

ПП – програмний продукт.

API – прикладний програмний інтерфейс (англ. application programming interface).

HTML – Мова розмітки гіпертексту (англ. HyperText Markup Language).

XML – Розширювана мова розмітки (англ. Extensible Markup Language).

CSS – Каскадні таблиці стилів (англ. Cascading Style Sheets).

JS — прототипна-орієнтована сценарна мова програмування. (англ. JavaScript).

DOM — програмний інтерфейс, що дозволяє програмам і скриптам отримати доступ до вмісту HTML, XHTML і XML-документів, а також його анулювання, структуру і оформлення таких документів. (англ. Document Object Model).

SVG – мова розмітки масштабованої векторної графіки, створена Консорціумом Всесвітньої павутини (W3C) і входить в підмножину розширюваної мови розмітки XML, призначена для опису двовимірної векторної і змішаної векторно-растрової графіки в форматі XML (англ. Scalable Vector Graphics).

Flash — мультимедійна платформа компанії Adobe Systems для створення веб-додатків або мультимедійних презентацій.

ВСТУП

З розвитком інформаційних технологій, в умовах різноманітності передаваємої інформації та контенту, гостро постає задача коректного відображення мультимедіа на всіх типах електронних пристроїв. У багатьох передових IT компаніях вже ставиться задача використання мультимедіа як частини HTML5. Тому виникає необхідність у новому мультимедійному програвачі, а також організовувати керування ним. Описані вище задачі неминуче призводять до необхідності розв'язувати проблеми, що виникають при роботі з новим типом програвача мультимедіа, оскільки, зазвичай, різноманітність пристроїв є достатньо високою, і потрібно забезпечити коректне відображення на кожному з них.

Є декілька способів відображення різних типів мультимедіа. В даній роботі будуть приведені найбільш відомі з них.

Метою дипломної роботи є розробка веб додатку за допомогою HTML5 Canvas. Типовою архітектурою веб-додатку, реалізованому на HTML5 Canvas є така:

Тобто користувач вносить свої дані та всі ходи, що він зробив до браузера, а код, реалізований на мові javascript запам'ятовує їх, обробляє та відсилає користувачеві отримані результати. У дипломній роботі розглядатиметься весь

додаток, від написання самого додатку до частини, яка обробляє отримані дані від користувача, і відправляє їх назад.

Показником успіху роботи є готовий веб додаток, який:

• вирішує основні задачі, які ставляться до відображення графіки (збереження, аналіз даних і т.д.);

• розроблений з урахуванням кращих практик і специфічних архітектурних рішень, які дають можливість забезпечити апаратне прискорення, а також використання бібліотек;

• вирішує специфічні проблеми, які не вирішують інші системи (наприклад, набагато зручніше мати справу з великим числом елементів);

• містить фільтри обробки зображень.

В першому розділі проведено загальний опис способів вирішення проблеми відображення різних типів мультимедіа та проведено порівняння з аналогами HTML5.

В другому розділі приведено приклади використання Canvas, і показано, де його використовувати вигідніше.

В третьому розділі описуються програми та алгоритми для представлення різних можливостей HTML5 Canvas.

В четвертому розділі проведений функціонально-вартісний аналіз програмного продукту.

1. CANVAS – НОВА СТОРІНКА В ІСТОРІЇ МУЛЬТИМЕДІА

Canvas (холст) — елемент HTML5 для створення растрового двомірного зображення. Зазвичай використовується спільно з javascript. Використовується в основному для відображення графіків і игровый елементів в браузерних іграх, вставки відео, створення повноцінного плеєра. Canvas також використовується в WebGL для апаратного прискорення 3D-графіки. В результаті можна створювати навіть 3D ігри, які працюють у вікні браузера. Розглянемо canvas та його особливості, недоліки та приклади застосування.

1.1 Canvas та його історія

Початок відліку блоку знаходиться зліва зверху. Від нього і будується кожен елемент блоку. Розмір простору координат не обов'язково відображає розмір фактичної відображуваної площі. За замовчуванням його ширина дорівнює трьомстам пікселям, а висота ста п'ятдесяти.

Вперше елемент canvas був представлений компанією Apple в движку WebKit для Mac OS X з метою подальшого його використання в додатках Dashboard і Safari. Ситуацію з відсутністю canvas в IE виправила компанія Google випустила власне розширення, написане на JavaScript, під назвою ExplorerCanvas. На сьогоднішній день canvas частіше використовується для побудови графіків, простих анімацій та ігор в браузерах. Група WHATWG пропонує використовувати canvas як стандарт для створення графіки в нових поколіннях веб-додатків. Організація Mozilla Foundation веде проект під назвою 3D Canvas, метою якого є додати низькорівневу підтримку графічних прискорювачів для відображення тривимірних зображень через HTML-елемент

canvas. Поряд з цим існують бібліотеки, реалізують роботу з тривимірними моделями, серед них three.

1.2 Можливості та особливості

Canvas використовується, як правило, для малювання графіків для статей і ігрового поля в деяких браузерних іграх. Але також може використовуватися для вбудовування відео в сторінку і створення повноцінного плеєра. Використовується в WebGL для апаратного прискорення 3D-графіки. Компанія Google випустила JavaScript бібліотеку explorercanvas, яка дозволяла працювати з Canvas в браузерах IE7 та IE8. Canvas може ускладнити завдання роботам по розпізнаванню Капчі. При використанні canvas з сервера завантажується не картинка, а набір точок (або алгоритм промальовування), за якими браузер промальовує картинку (капчу).

Canvas дозволяє розмістити на полотні: зображення, відео, текст. Залити все це суцільним кольором, або обвести контури чи навіть додати градієнт. Додавання тіней схожих на властивості css3 box-shadow і text-shadow. І, нарешті, побудова фігур за допомогою вказівки контрольних точок. Причому можна змінювати ширину ліній, так і кисть малювання ліній, стиль з'єднань ліній.

Зміна висоти або ширини полотна видалить всі його вміст і налаштування, простіше кажучи він створиться заново; Початок відліку (точка 0,0) знаходиться в лівому верхньому кутку. Але її можна зрушувати; 3D контексту немає, є окремі розробки, але вони не стандартизовані; Колір тексту можна вказувати аналогічно CSS, втім, як і розмір шрифту.

У випадку, якщо вам немає необхідності перемальовувати полотно, але потрібно проводити маніпуляції з ним, то ви можете «сфотографувати» все полотно і зберегти в змінну. І працювати вже з цим малюнком, не змушуючи канву відрисовувати після кожної маніпуляції. Якщо має оновлюватися не

зображення, а лише його частина, то ви можете брати певну зону на полотні і малювати її заново. Браузери можуть оптимізувати анімації, які йдуть одночасно, зменшивши число reflow і repaint до одного, що у свою чергу приведе до підвищення точності анімації. Наприклад анімації на JavaScript, синхронізовані з CSS transitions або SVG SMIL. Плюс до всього, якщо виконується анімація в табі, який невидимий, браузери не будуть продовжувати перемальовування, що приведе до меншого використання CPU, GPU, пам'яті і, як наслідок, знизить витрату батареї в мобільних пристроях. Для цього використовуйте requestAnimationFrame. Всі поточні браузери мають фільтр розмиття зображення при його збільшенні. Його варто використовувати, якщо ви часто попіксельно обробляєте картинку. Шляхом зменшення картинки, наприклад, в два рази і подальшого апаратного збільшення її з допомогою фільтра. Якщо гра дозволяє окремо обробляти фон і елементи гри, то має сенс зробити два полотна друг над іншому. Для очищення канви кращим засобом буде використання clearRect, однак, якщо очищати тільки необхідні ділянки, то швидкість зросте ще більше.

1.3 Переваги та недоліки HTML5 Canvas

Переваги:

• На відміну від SVG набагато зручніше мати справу з великою кількістю елементів;

• Має апаратне прискорення;

• Можна маніпулювати кожним пікселем;

• Можна застосовувати фільтри обробки зображень;

• Є багато бібліотек.

Недоліки:

• Навантажує процесор і оперативну пам'ять;

• Через обмеження збирача сміття, немає можливості очистити пам'ять;

• Необхідно самому обробляти події з об'єктами;

• Погана продуктивність при великому розширенні;

• Доводиться вимальовувати окремо кожен елемент.

1.4 Порівняння Canvas з його аналогами

Для показання доцільності використовування canvas порівняємо його з найвідомішими аналогами SVG та Flash.

1.4.1 Порівняння Canvas та Flash

Порівнювати ми будемо за декількома критеріями:

• Підтримка сучасними браузерами

• Файловий розподіл та структура

• Мобільні пристрої

• Ігри

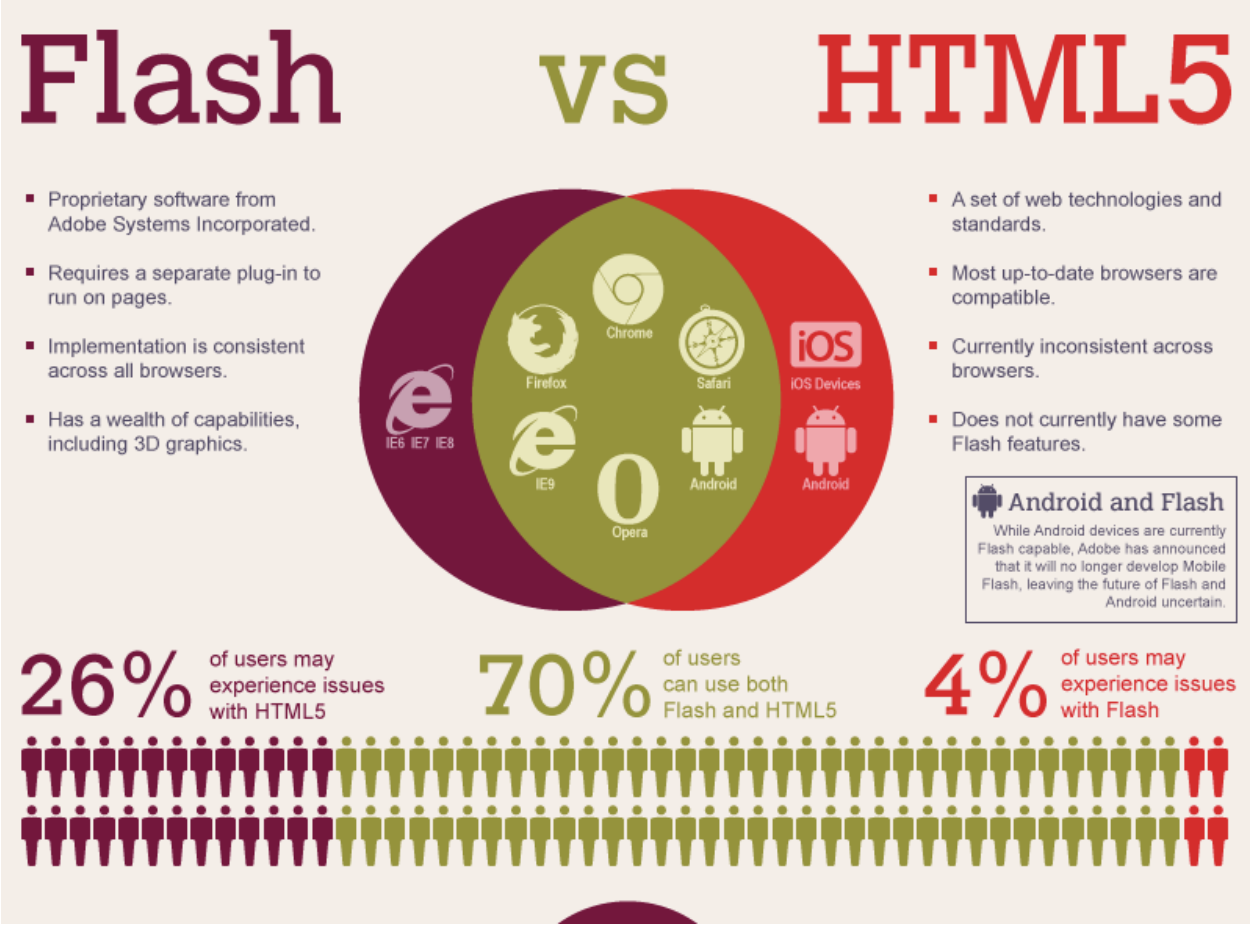

Рисунок 1 - Flash vs HTML5

Підтримка сучасними браузерами.

На даний момент, 99% десктопних браузерів підтримують Flash Player, та 82% підтримують HTML 5 Canvas. Але за даними, наведеними на Рис.2 бачимо, що більшість використовує новітні версії браузерів для настільного комп'ютера, тож можемо вважати нічию у цьому пункті [2].

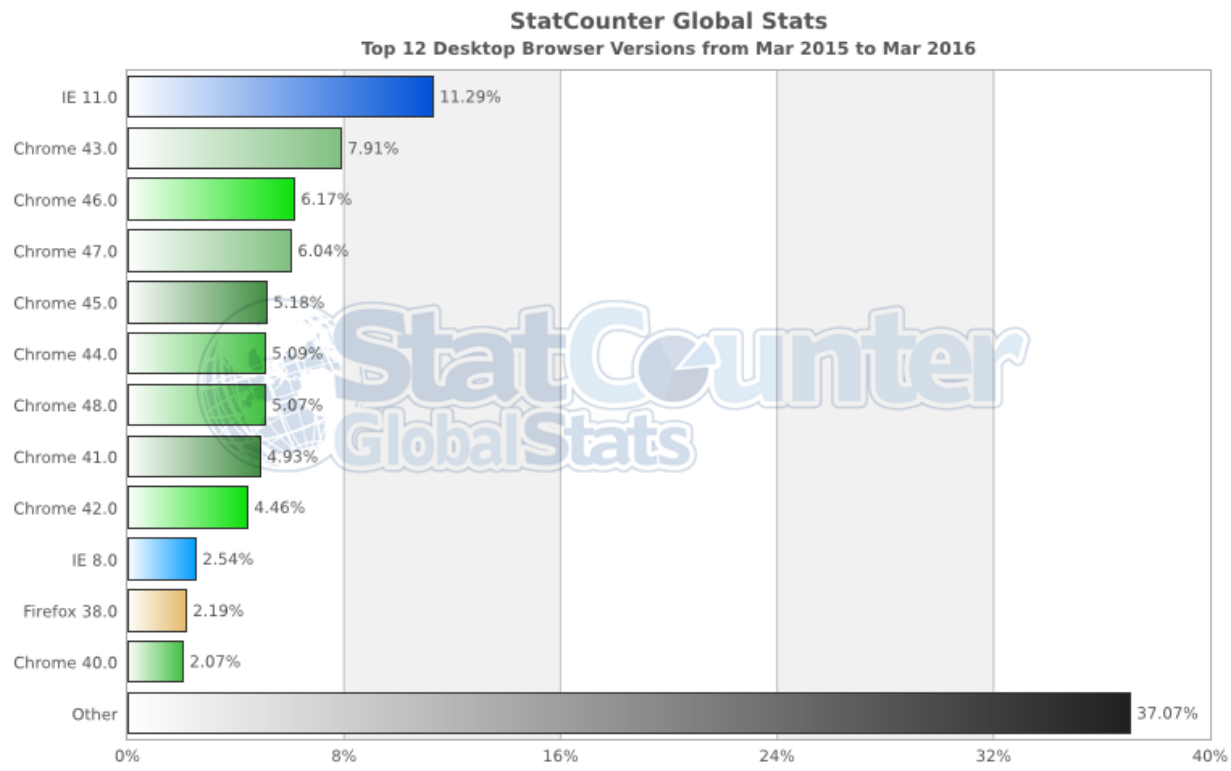

Рисунок 2 - Розподіл користування браузерами для ПК за останній рік [3].

Файловий розподіл та структура.

Для Flash технології буде достатнім один файл з розширенням swf. Це дуже важливо для ігр і зручно для зберігання чи транспортування. Що стосується HTML5, то тут за самою природою HTML відбувається розділення коду у різні файли. Тому треба бути обачним з їхнім розташуванням та вказанням вірного шляху до них. Хоча, з іншого боку, кожен окремий файл виконує свою функцію, і при редагуванні частини коду є можливість ввести поправки, не хвилюючись за весь проект в цілому. Також сама структура нового стандарту дозволяє пошуковим системам заглядати всередину створених з його допомогою сайтів і додатків, чого не можна сказати про Flash. За допомогою HTML5 розробники можуть створювати проекти, що складаються з безлічі взаємозалежних частин, і тепер пошукові роботи вміють розуміти ці зв’язки. А проект на Flash для роботів

Google представляє з себе незрозумілий для них вміст. Тому у даному пункті невелику перемогу здобув HTML5.

Мобільні пристрої.

Змоменту релізу HTML5 однією з головних переваг нової мови вважається її здатність працювати на мобільних пристроях. Той факт, що багато Android і iOS-пристроїв не підтримують Flash (97% мобільних браузерів підтримують HTML5 Canvas, а у Flash цей показник 0%), дозволяє використовувати його тільки на PC - цей ринок в 2013 році впав на 10 відсотків у порівнянні з 2012 [4]. За інформацією Бюро інтерактивної реклами, практично кожен другий житель США є власником мобільного телефону з доступом в інтернет, і кожен п’ятий перегляд веб-сторінок відбувається з мобільного пристрою. Цифри зростають щомісяця, і компанії, що створюють інтерактивні відео з використанням технології Flash, автоматично втрачають величезну аудиторію, яка могла б переглядати цей контент на своїх смартфонах і планшетах [5]. У даному пункті перемогу здобув HTML5.

Ігри.

За інформацією блога Digital Buzz, 32% часу, витраченого користувачами мобільних пристроїв, йде на ігри. За допомогою HTML5 розробники зможуть створювати іграшки, які будуть працювати на всіх девайсах. Згідно зі звітом розробника open-source фреймворків для створення веб-додатків Sencha, понад 60% розробників повністю або частково перевели свої проекти на HTML5, і більше 70% з них заявили про те, що в цьому році користуються даною технологією більше і частіше ніж в минулому [6]. Тож в данному пункті також перемагає HTML5.

Таблиця 1 - Порівняння векторної та растрової графіки у різних браузерах [14]

Браузер	**Вектор, HTML5 (fps)**	**Вектор, Flash (fps)**	**Растр, HTML5 (fps)**	**Растр, Flash (fps)**
Chrome 14.0.835.186	9	18	37	14
Firefox 6.0.2	11	17	48	14
IE 9.02	10	20	47	16
Opera 11.51	16	19	9	15
Safari 5.1	2	19	12	15

Висновок.

Виходячи з результатів аналізу HTML5 є кращим вибором, якщо дивитись наперед. Бо творці контенту, які не використовують у своїй роботі HTML5 вже відстають від тих, хто це роблять, і скоро остаточно програють. Ті, хто все ще вважають Flash відмінним вибором для онлайн-відео та ігор, в найближчому майбутньому повинні будуть змінити свою думку, і до цього їх підштовхне все триваюче проникнення мобільних пристроїв і подальше поширення HTML5.

1.4.2 Порівняння Canvas та SVG

Вибираючи між Canvas і SVG, слід відштовхуватися від двох речей : характеру вирішуваних завдань і продуктивності .

Canvas безальтернативно слід використовувати для обробки зображень і відео, добре вирішує складні сцени з великою кількістю об'єктів і можна хитро використовувати в веб - рекламі.

SVG краще застосовувати для статичних зображень і документів з високою точністю і детальністю , наприклад , схем , креслень і т.д.

Обидві технології , можна спокійно застосовувати для відтворення графіків і діграмм і для двовимірних іграшок.

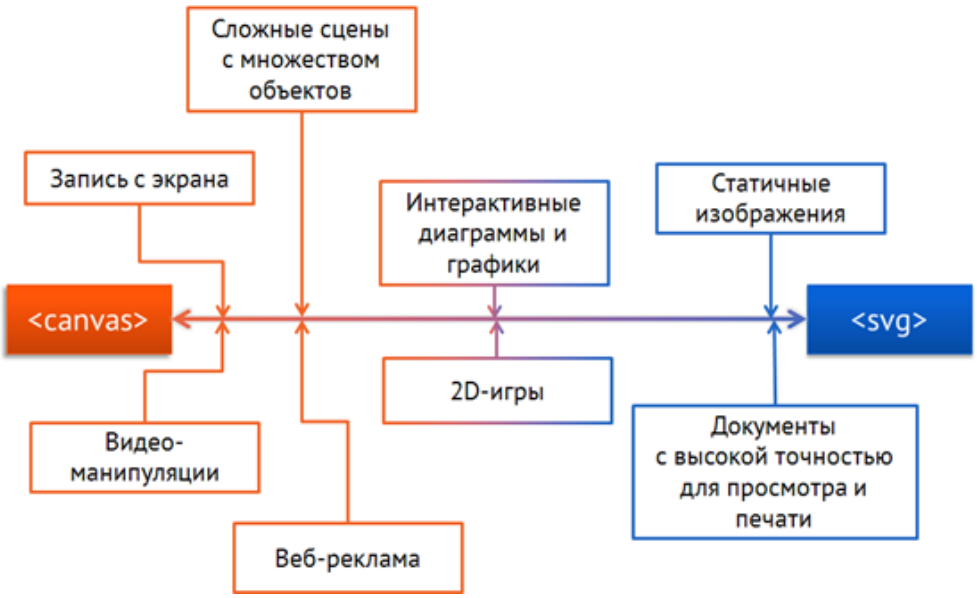

Рисунок 3 - Вибір залежить від поставленої задачі

Є такі переваги Canvas:

1. Зрозумілість. Canvas простий. Якщо ви знаєте JavaScript і мали досвід роботи з графікою через примітивні 2D-API, розібратися з Canvas вам не складе труднощів. API Canvas має близько 40 і 20 методів атрибутів, які ви легко можете розбити для себе на кілька груп.

2. Швидкість. Canvas швидкий. З підтримкою апаратного прискорення і тому швидкий. Canvas спроектований так, щоб бути швидким, хоча якісь речі і можуть на перший погляд здатися непрактичними (наприклад, потрібно спочатку намалювати весь шлях, перш ніж його прокреслити).

3. Інтеграція. Canvas - це звичайний HTML5 елемент і його можна вставити в будь-яке місце сторінки. Canvas можна комбінувати з відео, зображеннями та іншим контентом. Йому можна навіть зробити круглі куточки.

Переваги SVG:

1. Маркетинг навколо HTML5. SVG, формально включений до складу HTML5, став гарною червоною ганчіркою для демонстрації переваг нової версії

стандарту. HTML5 дозволяє працювати з векторною графікою - це звучить занадто добре, щоб замовчувати цей чудовий факт. Маркетинг, в свою чергу призводить до іншого фактору.

2. Підтримка в браузерах. Виробники браузерів не можуть ігнорувати той факт, що їм потрібно реалізовувати HTML5, якщо вони хочуть залишатися на плаву в браузерному ринку. Тому сьогодні ми бачимо базову підтримку SVG у всіх основних браузерах, причому місцями і з підтримкою апаратного прискорення при відображенні. Але маркетингу, звичайно, було б мало для того, щоб братися за SVG, якби не третій фактор.

3. SVG розвивається. Насправді, якщо ще кілька років тому SVG розглядався перевага як стандарт для опису графіки, то сьогодні на нього дивляться вже значно ширше. Потенційно, SVG - це не просто квадратики і кружечки з градієнтними заливаннями, зірочки і фрагменти тексту, а й можливість опису інтерфейсу для інтерактивної взаємодії, платформа для анімацій і візуалізації даних. У контексті зростання інтересу до розробки веб-додатків роль SVG буде теж рости.

Що стосується продуктивності , то тут все досить просто: до певного рівня залежності швидкості відтворення Canvas і SVG від розміру екрану і кількості об'єктів рухаються поруч , але далі виявляється , що SVG краще підходить для зображень великого розміру, а Canvas показує себе спритнішим при великій кількості об'єктів (рисунок 4).

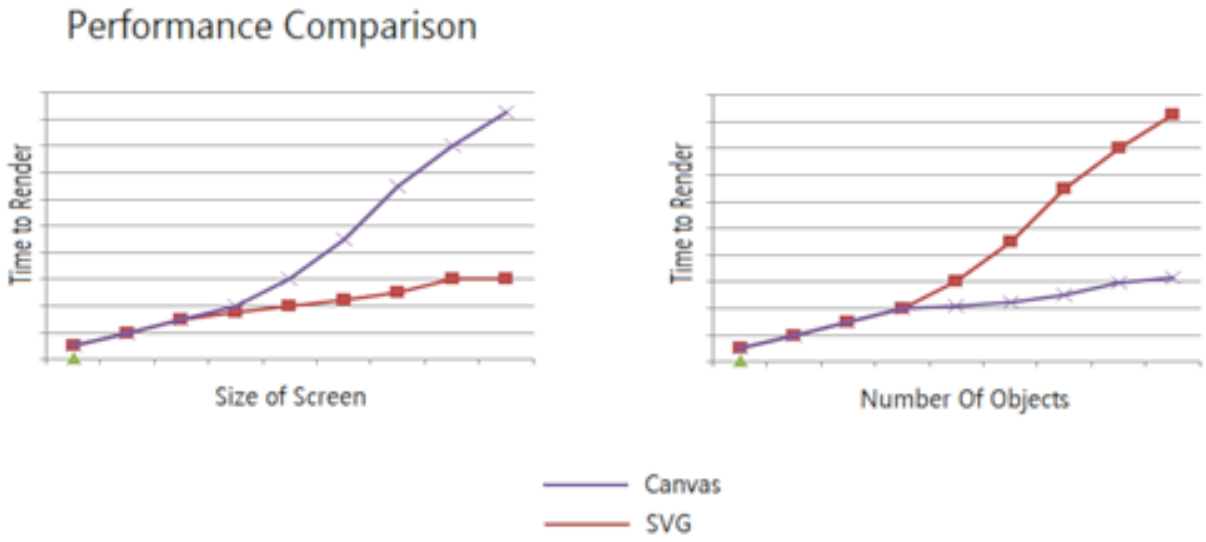

Рисунок 4 - Продуктивність Canvas vs SVG

Також были проведені тести, в результаті яких виявилося, що Canvas перемагає SVG у прирості швидкості в 200-500 % для відображення карт від Google.

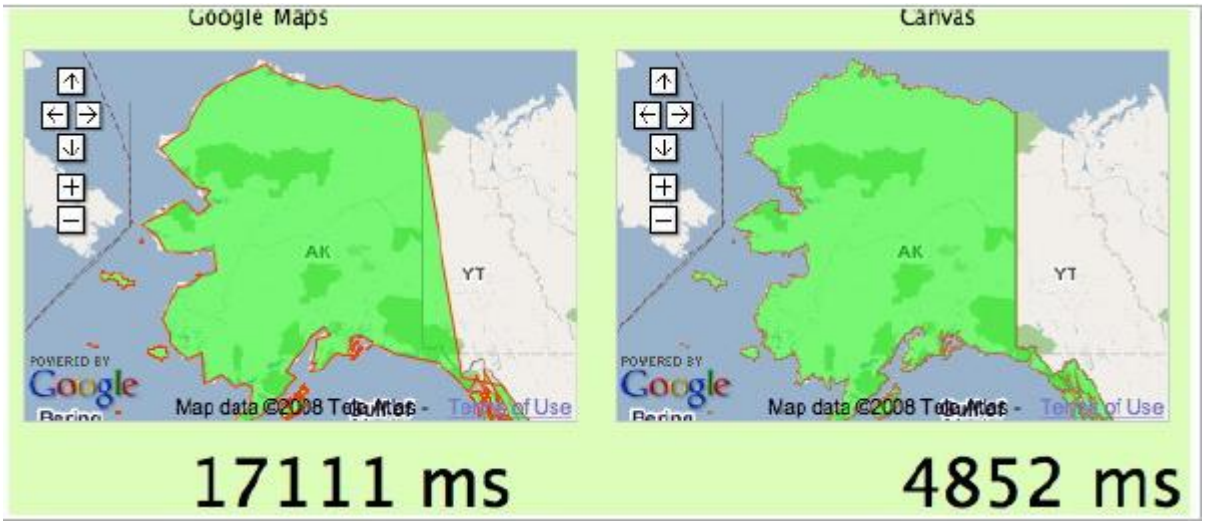

Рисунок 5 – Швидкодія на зображенні в Google Maps

Для уточнення питань продуктивності можна звернутися до дослідження [13], яким установлено закономірність між продуктивністю SVG, Canvas і параметрами зображення. В результаті виявляється цілком очевидним, що при збільшенні числа об'єктів (для SVG - векторних) продуктивність SVG падає

сильно (майже експоненціально), а Canvas - залишається на стабільному рівні. Тут варто зазначити, що розмір активної області при цьому не змінюється.

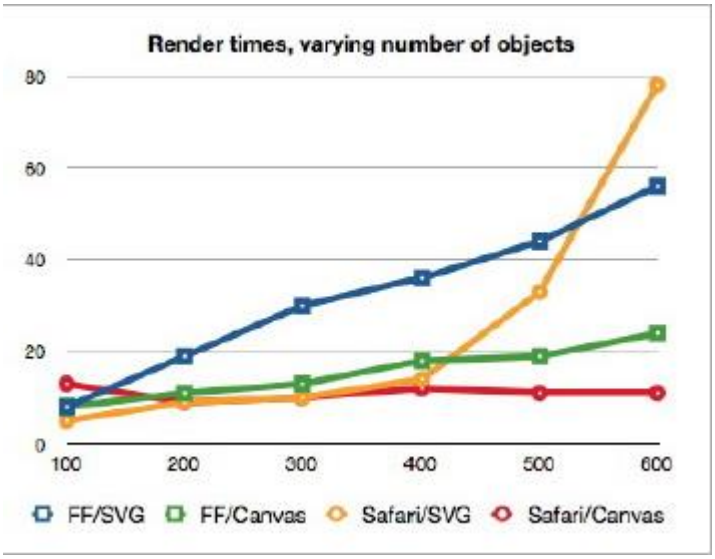

Рисунок 7 - Продуктивність Canvas і SVG при збільшенні числа об'єктів [13]

Таблиця 2 - Переваги Canvas та SVG

Canvas	SVG
Заснована на пікселях (динамічний png-файл)	Заснована на фігурах
Один елемент HTML	Кілька графічних елементів, які стають частиною моделі DOM
Змінюється тільки за допомогою сценаріїв	Змінюється за допомогою сценаріїв і CSS
Модель подій і взаємодія з користувачем засновані на пікселях (x, y)	Модель подій і взаємодія з користувачем є абстрактними (прямокутник, контур)
Продуктивність вище на невеликих поверхнях, при великому числі об'єктів (> 10 000) або для обох умов	Продуктивність вище на великих поверхнях, при невеликому числі об'єктів (<10 000) або для обох умов

Отож, підсумуємо:

Canvas - растрове полотно з доступом до окремих пікселей, який зберігає тільки останнім свій стан, яке можна витягти у вигляді масиву значень кольору і прозорості (RGBA) для різних пікселів. SVG - векторне масштабоване зображення з доступом до окремих об'єктів, легко піддається індексації.

Для роботи з Canvas є спеціальний API з набором примітивних функцій. Для роботи з нутрощами SVG можна використовувати DOM і CSS.

JS-код для Canvas пишеться окремо від елемента <Canvas>. JS-код для SVG можна як писати окремо, так і вставляти прямо всередину SVG-документа.

І ще одна річ: на полотні можна спроектувати зображення або відео, але не елемент SVG.

Таблиця 3 – Порівняння найпопулярніших платформ

Критерії	Мультимедійні платформи		
	HTML5 Canvas	SVG	Flash
Де працює	Елементи Canvas працюють на будь-якому веб-браузері HTML5. Незважаючи на те, що вони не працюють в IE до версії 9, вони працюють на ОС IOS і Android смартфонів і планшетів.	SVG працюють в більшості сучасних браузерів, IE не підтримує його до версії 9. Він також працює на ОС IOS і Android смартфонів і планшетів.	Flash працює в будь-якому браузері, який має плагін для нього. Проте, Safari для Mac 10.7 (Lion) НЕ не поставляється з плагіном Flash-і Adobe більше не підтримує флеш на мобільних пристроях.
як створити	Більшість створення CANVAS здійснюється вручну, але деякі програми ілюстрації отримують підтримку HTML5, як правило, за допомогою плагінів.	Більшість векторних графічних програм може виводити файли як SVG.	Є багато комерційних додатків, доступних автору флеш-анімації.

Продовження таблиці 3

Критерії	Мультимедійні платформи		
	HTML5 Canvas	SVG	Flash
Тип графіки	Елемент CANVAS створює растрові дані на екрані, але він може малювати вектори. Більшість з них створені вручну на JavaScript.	SVG - це інструмент векторної графіки, але він може використовувати растрові файли, такі як JPG або PNG.	Flash повністю векторний, але можна добавити растрові зображення в ваших флеш-файли.
вбудований HTML	Елемент CANVAS є елементом HTML5 і так автоматично використовується в HTML.	SVG має деяку підтримку браузера для вбудованих елементів в HTML5 (IE 9, FF 12, Chrome 19 і Safari 5.1), але і в цілому XML вбудований.	Flash-файли SWF вкладається як файли і мають використовувати плагін для браузера, щоб працювати.
необхідні технології	Елемент CANVAS використовує JavaScript для перегляду зображень. Якщо JavaScript відключений, CANVAS буде порожнім.	SVG вимагає сучасні браузери, які підтримують XML або SVG рендеринг.	Flash вимагає плагін для відображення зображень.
Швидкодія	CANVAS дуже швидкий для рендеринга зображень, але це вимагає аналізу JavaScript.	SVG, коли використовується в HTML5, дуже швидкий, тому що він не вимагає додаткових завантажень.	Flash є найповільнішим, що вимагає час завантаження як SWF-файлу і вкладення.

Продовження таблиці 3

Критерії	Мультимедійні платформи		
	HTML5 Canvas	HTML5 Canvas	HTML5 Canvas
Пошукова оптимізація	CANVAS є основаним на тексті, і може бути прочитаний пошуковими системами.	SVG є основаним на тексті, і може бути прочитаний пошуковими системами.	Flash впроваджений SWF-файл, який може бути важким для пошукових систем, щоб розібрати.
Середній розмір готового проекту	1.7 Мб	1.5 Мб	> 2.5 Мб
Що може бути інтерактивним	Немає окремих елементів, що є клікабельними в CANVAS, тільки весь елемент.	SVG дозволяє клікати по елементам в зображенні.	Flash може мати інтерактивні точки, створені в будь-якому місці файлу.

Таблиця 4 - Кількість fps у різних браузерах та технологіях

Браузер	**HTML**	**Canvas**	**SVG**	**Flash**
Chrome 14.0.835.186	21	30	18	44
Firefox 6.0.2	9	38	2	44
IE 9.02	16	34	25	50
Opera 11.51	29	28	12	50
Safari 5.1	23	31	51	50

1.5 Висновки

В даному розділі були розглянуті основні переваги обраних технологій для реалізації прототипу системи та їх порівняльна характеристика з іншими альтернативами, в результаті чого виявлено переваги та недоліки HTML5

поріняно з іншими рішеннями. Було встановлено доцільність використання технології HTML5 Canvas у рамках даної дипломної роботи.

Також наведена класифікація основних пунктів, за якими доцільно порівнювати мультимедійні платформи.

2. ВИКОРИСТАННЯ CANVAS

Коли в браузері Mosaic з'явилася можливість вставляти на сторінки картинки, це дало мережі потужний поштовх вперед. Але аж до теперішнього моменту картинки залишаються статичними. Так, можна робити анімовані gif-зображення, оновлювати стилі картинок на льоту за допомогою JS, генерувати їх на стороні сервера. Але в будь - якому випадку — як тільки картинка була відкрита в браузері, ми не можемо змінити її вміст.

Для цього може бути застосований елемент canvas, призначений для створення динамічно змінюваних зображень.

Сам по собі він дуже простий. Все, що ви вказуєте в параметрах тега — це розміри полотна (зазвичай це 360 на 240 пікселів). Це парний тег. Але те, що ви помістите в його середині, призначене тільки для браузерів, що цей елемент не підтримують. Наприклад:

```
<canvas id="my-first-canvas" width="360" height="240">
<p>No canvas support? Have an old-fashioned image instead:</p>
<img src="puppy.jpg" alt="a cute puppy">
</canvas>
```

Користувачі браузерів без підтримки canvas побачать фото милого щеняти.

Вся робота по візуалізації покладається на JavaScript. Насамперед, треба вказати, з яким елементом ми працюємо і в якому контексті. «Контекст» в даному випадку — це API, і він на сьогоднішній день лише один — двомірний (як видно, є куди рости).

Тепер ми можемо почати малювати на цьому двовимірному полотні елемент canvas, використовуючи API, задокументований у специфікації HTML5. Набір інструментів тут схожий з тим, що ви знайдете в будь-якому графічному редакторі, наприклад, в Illustrator-і: лінії, заливка, градієнти, тіні, фігури, криві Безьє. Різниця лише в тому, що замість малювання мишкою, потрібно користуватися командами в JavaScript.

Двомірний API включає безліч різних методів: fillStyle, fillRect, lineWidth, shadowColor і ще багато інших.

Теоретично, будь-яке зображення, яке можна створити в тому ж Ілюстратора, може бути точно так само створено за допомогою елемента canvas. На практиці, втім, це може виявитися досить трудомістким і дасть на виході величезну кількість коду. З іншого боку, це і не зовсім те, для чого був canvas придуманий.

Звичайно, це гарно, що використовуючи JavaScript і canvas, можна створювати векторні зображення на льоту, але якщо вони робляться хоть трохи складними, ця робота не буде виправдовувати себе.

Зміст і головна функція canvas полягають в тому, що його вміст можна динамічно оновлювати, відрисовуючи нові елементи у відповідь на дії користувача. Ця його здатність реагувати на події, ініційовані відвідувачем сторінки, робить можливим створення таких інструментів і ігор, які раніше вимагали б застосування сторонніх технологій, таких як Флеш.

Одним з перших прикладів можливостей canvas став проект Bespin від Mozilla Labs — додаток, є простим текстовим редактором для кодинга, яке працює у вікні браузера.

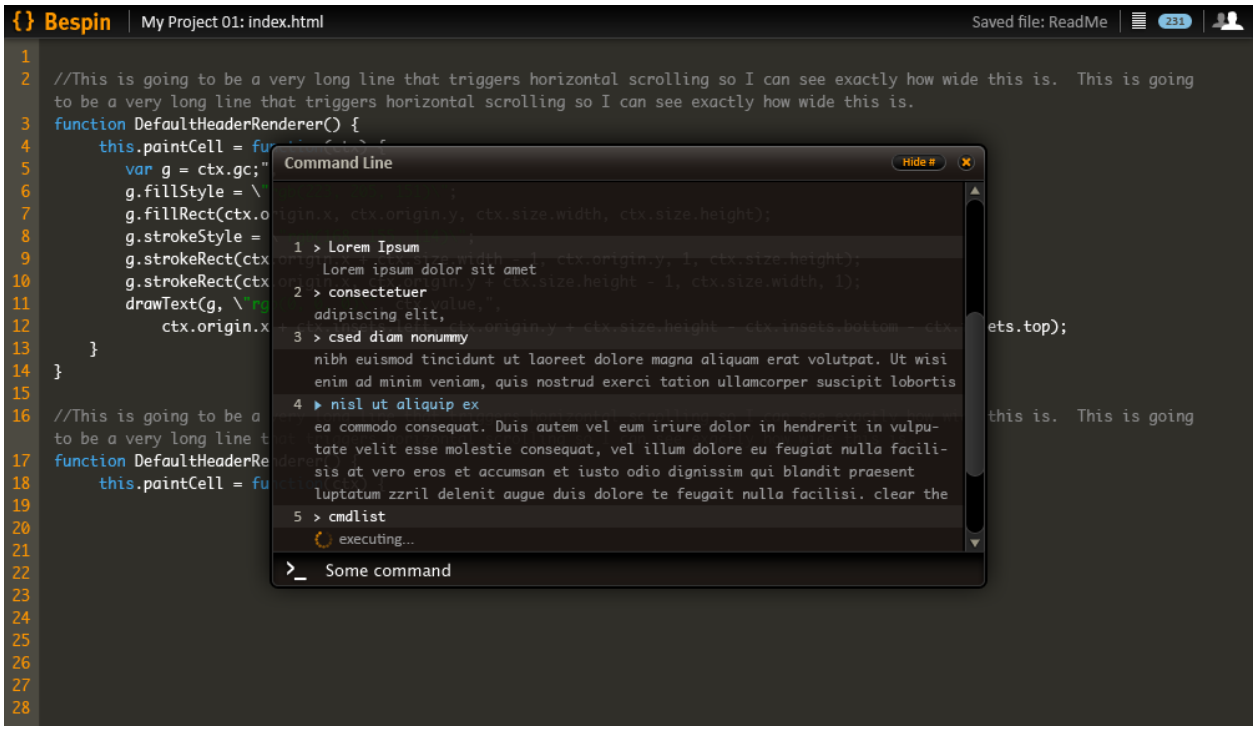

Рисунок 8 - Веб-додаток Bespin, створений за допомогою елемента canvas

Це дуже потужна річ. Але, також, вона — гарний приклад того, що не треба робити з canvas, бо текстовий редактор, по своїй суті, призначений для роботи з текстом. Bespin успішно працює з текстом всередині елемента canvas, з одним лише застереженням — це вже не текст. Це просто набір векторних форм, схожих на нього.

Кожна сторінка в мережі може бути описана за допомогою DOM — Document Object Model, «об'єктна модель документа». DOM може включати в себе багато різних «вузлів», найважливіші з них — елементи, атрибути і текстові об'єкти. Цих трьох видів «цеглинок» достатньо, щоб побудувати практично будь-яку сторінку, яку ви тільки можете собі уявити. У свою чергу, елемент canvas не має DOM — його вміст не можна розділити на окремі частини і представити як дерево вузлів.

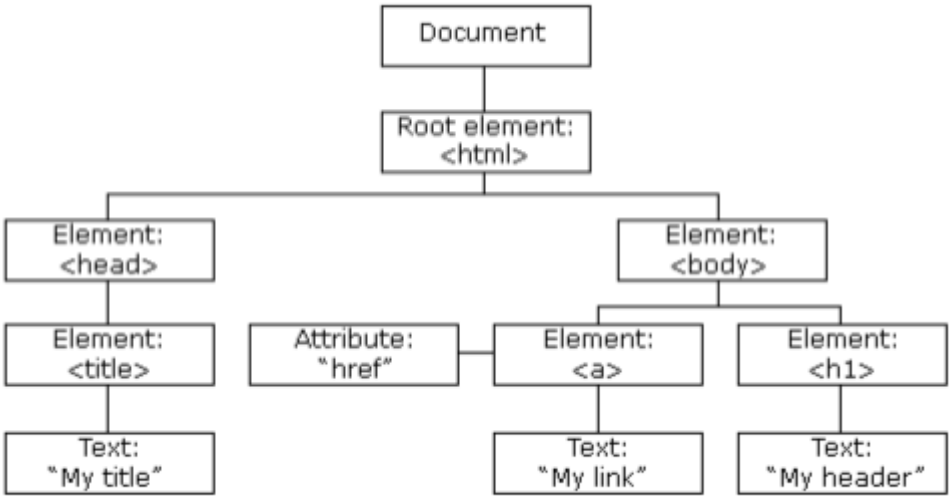

Рисунок 9 - DOM

Пристрої для читання з екрану і інші технології для схожих завдань ґрунтуються на можливості аналізу DOM для розуміння структури та змісту документа. Немає DOM — немає доступу.

Така «ізольованість» canvas — велика проблема для HTML5. На щастя, W3C працює над її рішеннями. Так як цей момент досить серйозний, я сподіваюсь, шо вони вирішать все швидко, але в той же час не хочеться, щоб це гальмувало розвиток решти специфікації.

Виходячи з вищевикладеного, можна припустити, що на даний момент веб-дизайнерам немає сенсу застосовувати canvas в своїх проектах. Це не так.

Коли більшість використовують JavaScript на своєму сайті, він виконує винятково роль додаткового поліпшення. Ті відвідувачі, у яких він відключений, не будуть позбавлені доступу до інформації, хіба що тільки певних другорядних зручностей. Такий багаторівневий підхід під назвою «Ненав'язливий JavaScript» може бути застосований і до canvas. Замість створення нового контенту, він буде використовуватися для кращого уявлення вже наявного.

Припустимо, у вас є таблиця з якимось даними. Ви хочете проілюструвати спостереження в них тенденції за допомогою графіка. У разі якщо дані статичні — можна просто створити графік-картинку, використовуючи Google Chart API, наприклад. Але якщо дані можна змінювати по ходу справи, то було б добре створити графік, який кожен раз відрисовується кожен раз знову з урахуванням змін. Canvas тут відмінно підійде — можна використовувати JavaScript для вилучення вмісту елемента table і створення на його основі математично розрахованої ілюстрації.

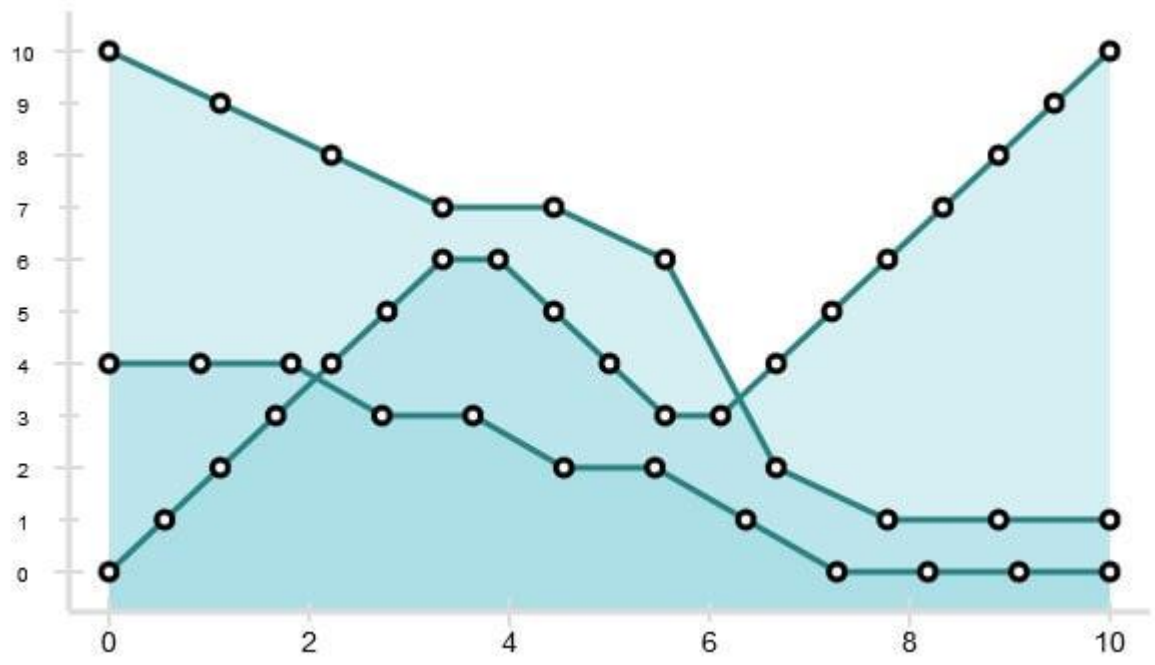

Рисунок 10 - Приклад графіків, згенерованих за допомогою canvas

Є й інший варіант. Насправді, елемент canvas — не єдиний API для генерації динамічних зображень. SVG — Scalable Vector Graphics, масштабована векторна графіка — XML формат, який може використовуватися для опису тих самих фігур, що і canvas. І виходячи з суті XML як такого, вміст SVG теоретично «зрозумілі» для пристроїв для читання з екрану. Але на практиці, SVG не справив

виключного враження на розробників, на відміну від canvas, який, хоч і з'явився зовсім недавно, вже широко підтримується доступними браузерами. Його можливості навіть реалізовані в IE, за допомогою додаткової JavaScript-бібліотеки.

Враховуючи гасла WHATWG про те, що треба «мостити натоптані стежки» і «не винаходити велосипед», може здатися дивним, що вони вважали за краще включити в HTML5 canvas, коли вже існує аналогічна технологія SVG. Як це часто відбувається, специфікація HTML5 насправді лише документує багато чого з того, що вже підтримується браузерами. Canvas не був придуманий для HTML5 — він вперше з'явився в Safari і був створений Apple. Коли розробники інших браузерів побачили цю ідею, вона їм сподобалася і була скопійована.

2.1 Висновки

В даному розділі наведено аналітичний огляд використання Canvas. Було показано, що HTML5 Canvas краще використовувати для тих цілей, де растрова графіка перемагає векторну. У світі браузерів, де виживає найбільш пристосований, canvas в даний момент набирає силу і популярність. Як тільки він стане більш «відкритим» для доступу ззовні, його позиції будуть надійно закріплені.

3. РЕАЛІЗАЦІЯ

Для більшого сприйняття можливостей використання Canvas у різних сферах мультимедіа в даній роботі представлені декілька варіацій Canvas.

3.1. Робота з зображеннями

Елемент Canvas в HTML5 використовує пікселі для створення графічних зображень на екрані. У цьому розділі описано 3 способи роботи з пікселями з використанням елемента Canvas, що дозволяють створювати спеціальні фотоефекти. За допомогою цих методів є можливість розробляти унікальні зображення для ефектного художнього оформлення веб-сайтів, блогів, відеоігор, рекламних оголошень, ілюстрацій і багатьох інших елементів.

У цих прикладах показано, як: створити круглу рамку навколо фотографії; істотно змінити колірну композицію; перетворити кольорове зображення в чорно-біле.

Елемент Canvas підтримує immediate mode для відображення пікселів безпосередньо на екрані, що робить його ефективним засобом розробки спеціальних фотоефектів. Підтримка миттєвого режиму дозволяє зчитувати і записувати кожен піксель елемента Canvas, що дає можливість створювати складні візуальні ефекти, змінюючи характеристики пікселів. При імпорті і відображенні фотографії в елементі Canvas зображення перетворюється в масив кольорових пікселів, кожен з яких має значення червоного, синього, зеленого кольорів, а також альфа-фактора (визначає прозорість). Оскільки елемент Canvas дозволяє змінювати кольори окремих пікселів, а також малювати в масиві

пікселів точки, лінії і фігури, з його допомогою можна з легкістю створювати привабливі фотоефекти. Приклади коду написані з використанням JavaScript і елемента Canvas в HTML5. Приклади мають найпростішу структуру і призначені для демонстрації основних принципів використання функцій елемента Canvas. При необхідності можна з легкістю розширити цей код, реалізувавши власні фотоефекти відповідно до вимог конкретних веб-додатків.

3.1.1. Створення рамок для фотографій за допомогою елемента Canvas

У цьому підрозділі пояснюється, як створювати різні рамки і спеціальні візуальні ефекти на веб-сторінках за допомогою елемента Canvas HTML5.

В додатках включений окремий приклад коду з коментарями, в якому фотографія парусного корабля поміщається в круглу рамку. У цьому прикладі коду показано, як використовувати технологію елемента Canvas для завантаження фотографії корабля і її відображення на екрані. Далі описано створення кола за допомогою методу полотна arc і додавання круглої білої рамки навколо зображення.

Рисунок 11 – Приклад роботи рамки для фотографії

В рамках обговорення розглядається структура прикладу коду елемента Canvas і порядок взаємодії всіх компонентів один з одним. У цьому прикладі використовується стандартний заголовок HTML5, <!Doctype html>, за яким браузери розпізнають специфікацію HTML5.

У тезі body програми використовується функція onload для виклику функції displayImage при завантаженні сторінки. Початкове зображення корабля завантажується в тіло, що дозволяє порівнювати з ним зображення, яке буде змінено за допомогою елемента Canvas. Тег canvas входить в тіло програми. Вказується вихідна ширина і висота полотна, а також атрибут id. Атрибут ID необхідний для додавання елемента полотна в об'єктну модель сторінки.

Код сценарію містить дві функції: displayImage і getColorData. Функція displayImage викликається при завантаженні сторінки. Функція getColorData викликається з функції displayImage. На початку розділу сценарію створюється глобальна змінна для створення порожнього файлу зображення, який буде використовуватися пізніше.

Функція displayImage викликається при завантаженні сторінки. Вона отримує полотно за допомогою ID елемента canvas в основному коді. Потім функція отримує об'єкт полотна CanvasRenderingContext2D, готуючи його до обробки операціями малювання, і використовує функцію DrawImage для завантаження зображення в контекст. Після ініціалізації контексту як двомірного полотна можна приступати до малювання на полотні.

На завершення в функції вказується шлях до джерела зображення. Оскільки завантаження зображення може займати деякий час, визначається подія, яка викликає функцію по завершенні завантаження. Після того як зображення було завантажене воно виводиться на екран, а потім викликається функція changeImage, яка завершує операції малювання.

Функція changeImage передбачає зміну зображення шляхом створення кола навколо центру зображення виконується за допомогою методу arc. Розмір зображення 200 × 200 пікселів. Радіус кола - 150 пікселів. Оскільки центр окружності збігається з центром зображення і межа окружності має товщину 100 пікселів, пікселі на зовнішніх кордонах заміщаються білим кольором. Окружність визначається за допомогою властивостей strokeStyle і lineWidth, а також методів beginPath, arc, closePath і stroke.

3.1.2. Перетворення кольорових фотографій в чорно-білі з допомогою елемента canvas

У цьому підрозділі описується порядок перетворення кольорових фотографій в чорно-білі з допомогою елемента canvas в HTML5, що дозволяє, наприклад, домогтися художнього ефекту старовинної фотографії.

В додатках включений окремий приклад коду з коментарями, в якому показано, як перетворити кольорову фотографію в чорно-білу. У цьому прикладі коду показано, як використовувати елемент canvas, щоб завантажити фотографію корабля на веб-сторінку і відобразити її на екрані. Після цього в програмі зчитуються значення червоного, зеленого і синього кольорів для кожного пікселя і обчислюються відповідні значення з відтінків сірого. Після обчислення цього значення програма змінює значення червоного, зеленого і синього кольору для кожного пікселя. В кінці прикладу коду наведені матеріали для обговорення, в яких більш детально розглядається реалізація необхідного поводження в коді.

Рисунок 12 - Перетворення кольорової фотографії в чорно-білу

У тезі body програми використовується функція onload для виклику функції displayImage при завантаженні сторінки. Початкове зображення завантажується в основну частину програми, щоб можна було порівнювати з ним зображення, яке буде змінено за допомогою елемента canvas. Подія onerror додається в елемент img і визначає, завантажено чи зображення. Елемент canvas входить в основну частину програми. Вказується вихідна ширина і висота елемента canvas, а також атрибут id. Він необхідний для додавання елемента полотна в об'єктну модель сторінки.

Код сценарію містить чотири функції: displayImage, getColorData, putColorData і noPhoto. Функція displayImage викликається при завантаженні сторінки. Функції getColorData і putColorData викликаються з функції displayImage. Функція noPhoto викликається в разі виникнення помилки при завантаженні фотографії. На початку розділу сценаріїв створюються глобальні змінні для створення порожнього файлу зображення, який буде використовуватися пізніше. Також створюються змінні, що визначають висоту і

ширину елемента canvas. Крім того, обчислюється довжина масиву зображення, визначаючи число 4-байтових значень в масиві.

Функція displayImage викликається при завантаженні сторінки. Вона отримує canvas, використовуючи ідентифікатор (ID) елемента canvas в коді основної частини програми. Потім виходить об'єкт CanvasRenderingContext2D Object елемента canvas, що дозволяє підготуватися до отрисовке. Для завантаження зображення в контекст використовується функція DrawImage. Після ініціалізації контексту як двовимірного елемента canvas можна виконати отрисовку.

На завершення в функції вказується шлях до джерела зображення. Оскільки завантаження зображення може займати деякий час, визначається подія, яка викликає функцію по завершенні завантаження. Після завантаження зображення воно відображається, і викликаються функції getColorData і putColorData, які завершують відрисовку.

За допомогою функції getColorData зображення копіюється за допомогою методу getImageData в об'єкт myImage і обробляється перебором значень в масиві даних зображення і отриманням значень перших трьох байтів, відповідних червоному, зеленому і синьому кольору (альфа-фактор не враховується). Далі витягнуті значення складаються, а їх сума ділиться на три. Результат приводиться до цілого числа і записується в значення відповідних трьох байтів. Це значення відповідає колірному відтінку, але представляє відтінок сірого від 0 до 255. У результаті виходить чорно-біле зображення, яке виглядає, як вихідне кольорове.

Функція putColorData записує змінені дані зображення в елемент canvas за допомогою методу putImageData. Найчастіше рекомендується виконувати фактичне опрацювання другого зображення і відображати результат тільки після завершення обробки.

3.1.3. Зміна кольорів фотографій за допомогою елемента Canvas

У цьому підрозділі описується застосування елемента Canvas в HTML5 для зміни одного або декількох кольорів на фотографії, що дозволяє створювати яскраві зображення і ефектні веб-сторінки.

В додатках включений окремий приклад коду з коментарями, в якому кольорова фотографія змінюється так, щоб в її палітрі збереглися тільки синій і зелений кольори. Далі для кожного пікселя видаляються значення, відповідні червоному кольору, в результаті чого залишаються тільки значення синього і зеленого кольору. У цьому прикладі з фотографії видаляється один колір (червоний). При необхідності можна видаляти, наприклад, два кольори або змінювати частку видалення кольору. Наприклад, можна сформувати унікальну кольорову палітру, знизивши вміст червоного кольору на 50%.

Рисунок 13 - Зміна кольору на фотографії

У тезі body програми використовується функція onload для виклику функції displayImage при завантаженні сторінки. Початкове зображення

завантажується в тег <body>, що дозволяє порівнювати з ним зображення, яке буде змінено за допомогою Canvas. Подія onerror додається в елемент img і визначає, завантажено чи зображення. Елемент canvas входить до складу тега <body>. У цьому тезі вказується вихідна ширина і висота елемента Canvas, а також атрибут id. Він є необхідним для додавання елемента Canvas в об'єктну модель сторінки.

Код сценарію містить чотири функції: displayImage, getColorData, putColorData і noPhoto. Функція displayImage викликається при завантаженні сторінки. Функції getColorData і putColorData викликаються з функції displayImage. Функція noPhoto викликається, якщо при завантаженні фотографії виникає помилка. На початку розділу сценаріїв в коді створюються глобальні змінні для створення порожнього файлу зображення, який буде використовуватися пізніше. Також створюються змінні, що визначають висоту і ширину елемента canvas. Крім того, обчислюється довжина масиву зображення на основі числа 4-байтових значень в масиві.

Функція displayImage викликається при завантаженні сторінки. Вона отримує елемент canvas за допомогою ID елемента canvas в коді тега <body>. Потім виходить об'єкт CanvasRenderingContext2D Object елемента canvas, що дозволяє підготуватися до отрисовке; для завантаження зображення в контекст використовується функція DrawImage. Після ініціалізації контексту як двовимірного елемента canvas можна виконати отрисовку.

Далі в функції вказується шлях до джерела зображення. Оскільки завантаження зображення може займати деякий час, визначається подія, яка викликає функцію по завершенні завантаження. Після того, як зображення було завантажене воно відображається, і викликаються функції getColorData і putColorData, які виконують отрисовку.

Зображення корабля має зберігатися на вашому комп'ютері. Виклик зображення з іншого домену не підтримується. Якщо джерело пікселів для

елемента canvas знаходиться в іншому домені, зображення розглядається як загроза безпеці і не підлягає зміні. Додаткові відомості про загрози безпеки між доменами см. В документації по canvas. Для виконання сценарію можна використовувати будь-який файл з ім'ям kestral.png.

За допомогою функції getColorData зображення копіюється за допомогою функції getImageData в об'єкт myImage і обробляється перебором значень в масиві даних зображення і присвоєння байтам, відповідним червоному кольору, нульових значень. Масив для зображення розміром 200 на 200 пікселів містить 40 000 осередків. Кожна осередок містить 4 байта даних, в результаті чого загальний розмір масиву становить 160 000 байт. У програмі цикл виконується 160 000 разів. Кожен раз зчитується 4 байта. У кожному проході циклу першому байту присвоюється нульове значення. В результаті з палітри зображення виключається червоний колір. Байти зеленого і синього кольорів не змінюються.

Для читання і запису пікселів на екрані кожного пікселя необхідно привласнити колір. Значення кольору складається з чотирьох компонентів: червоний, зелений, синій і альфа-канал. Наприклад, білий піксель складається з 100% червоного, 100% зеленого, 100% синього і 100% альфа-каналу. (Альфа-канал відображає ступінь прозорості кольору.) Значення кольору для кожного пікселя в елементі canvas зберігається у вигляді масиву розміром 4 байта. Всі пікселі елемента canvas зберігаються в одному масиві. Відлік починається з верхнього лівого і закінчується в правому нижньому кутку. Розмір масиву визначається перемножением висоти елемента canvas на його ширину і на число 4 (розмір масиву для одного пікселя).

Чотири байта відповідають відсотковим значенням червоного, зеленого, синього кольору і альфа-каналу для кожного пікселя. (Значення байтів знаходяться в діапазоні від 0 до 255 для десяткових чисел і від # 0 до #FF для шістнадцятирічних.) У коді, наведеному в даному розділі, показано, як в миттєвому режимі Canvas виконується запис кольору пікселів на екрані.

Функція putColorData записує змінені дані зображення в елемент canvas за допомогою методу putImageData. Найчастіше рекомендується виконувати фактичне опрацювання другого зображення і відображати результат тільки після завершення обробки.

3.2 Відображення трьохмірної графіки

Припустимо, нам потрібно візуалізувати тривимірний об'єкт, показуючи його різні уявлення на відображаємому пристрої, такому як монітор комп'ютера. Також припустимо, що відображається об'єкт складається виключно з точок (x, y, z):

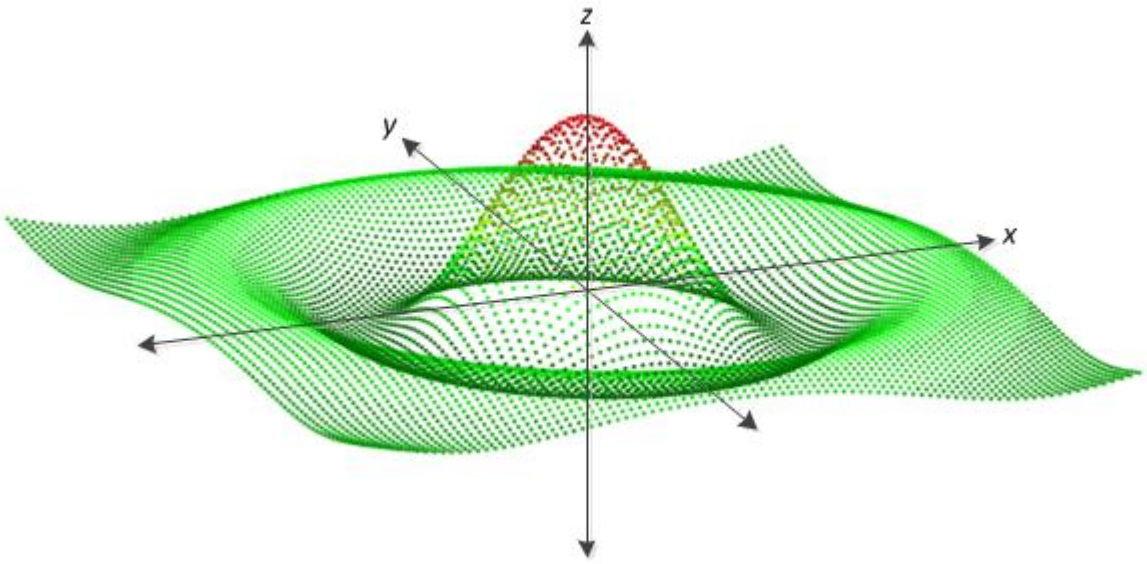

Рисунок 14 - Математична поверхня на основі точок

Якщо впровадити об'єкт в систему координат xyz так, щоб його вихідна точка розташовувалася в центрі монітора, а площину xy збігалася з площиною монітора, спостерігач побачить проекцію уявлення тривимірного об'єкту на двомірну площину xy або монітор.

Об'єкт, показаний на рис. 8, є представником математичної поверхні на основі точок:

$$z = f(x,y) = \frac{4\sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}} \quad (1)$$

Для створення зображення 8100 рівномірно розподілених точок (x, y) з площини xy були використані для розрахунку висоти z поверхні над площиною xy з використанням заданої формули f (x, y). Говорячи іншими словами, кожна точка поверхні (x, y, z) в системі координат xyz задається (x, y, f (x, y)).

Як наслідок, попередня поверхня буде представлятися "плоскою" при перегляді зверху. Тобто дивлячись прямо вниз по осі z, можна буде побачити контурне зображення поверхні:

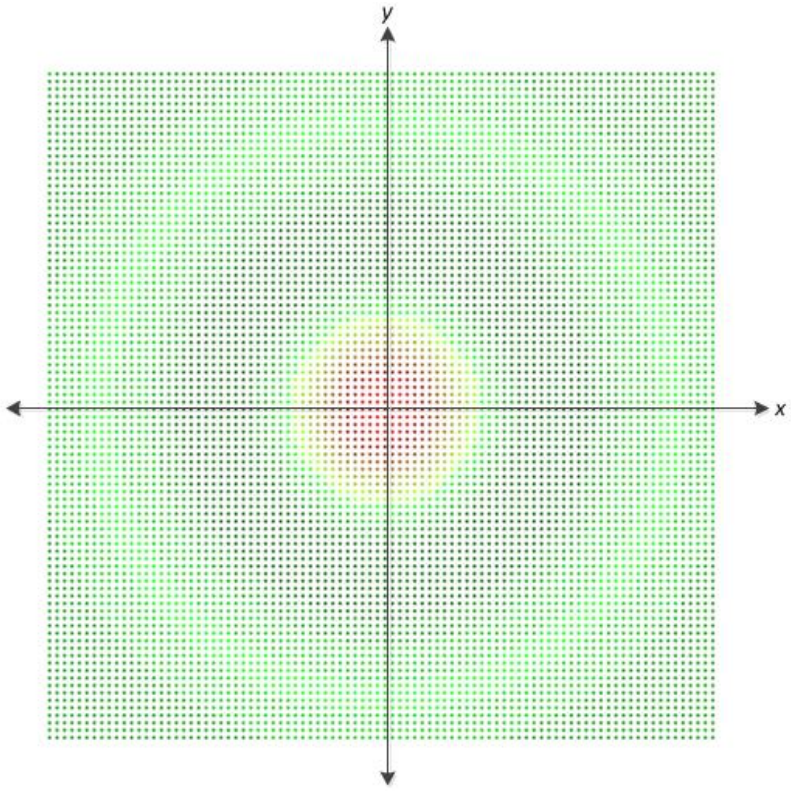

Рисунок 15 - Контурне зображення поверхні

Це контурне зображення поверхні показує 8100 рівномірно розподілених точок xy, використовуваних для розрахунку координат z поверхні. Контурне зображення безумовно може бути корисним саме по собі (наприклад, в якості топографічної карти). Але лише обертання цих тривимірних точок навколо осей x, y і z дозволяє побачити справжню природу тривимірного об'єкту.

Побудова тривимірних точок

Щоб намалювати певне уявлення поверхні, наприклад таке, як на рис. 8, пристрою відображення потрібні тільки координати x і y точок, так як відображається тільки проекція поверхні на площину xy.

Поворот тривимірних точок

Для початку ми розміщуємо всі точки поверхні n (n = 8100) в матриці 3 x nP, відомої як матриця координат уявлення. Тобто у нас є стовпці n в P, і кожен стовпець представляє точку на поверхні:

$$P = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ z_1 & z_2 & \cdots & z_n \end{bmatrix}$$

Для обертання цієї поверхні навколо осей x, y або z ми виконуємо множення P за допомогою відповідної матриці обертання R 3 x 3:

P '= RP

Тут P - вихідне уявлення поверхні, а P '- уявлення після повороту навколо однієї з осей координат, визначених R. Так як осей координат три, то є три матриці обертання для R:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Наприклад, якщо вам потрібно повернути поверхню, показану на рис. 3, на π / 12 радіан (15 °) навколо осі x, то R_x (приблизно) приймає наступний вигляд:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.966 & -0.259 \\ 0 & 0.259 & 0.966 \end{bmatrix}$$

Для створення уявлення поверхні P 'після повороту ми просто виконуємо матричне множення R_xP наступним чином:

$$P' = R_xP = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.966 & -0.259 \\ 0 & 0.259 & 0.966 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ z_1 & z_2 & \cdots & z_n \end{bmatrix} =$$

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ 0.966y_1 - 0.259z_1 & 0.966y_2 - 0.259z_2 & \cdots & 0.966y_n - 0.259z_n \\ 0.259y_1 + 0.966z_1 & 0.259y_2 + 0.966z_2 & \cdots & 0.259y_n + 0.966z_n \end{bmatrix}$$

Потім, щоб відобразити поверхню, отримані числові (x, y) точки P' шикуються на відображаємому пристрої.

Масштабування і трансляція

Як описано вище, для повороту поверхні навколо осі ми виконуємо:

P '= RP (2)

Аналогічним чином для масштабування поверхні можна виконати таке ж матричне множення, але з іншої матрицею перетворення 3 x 3 S:

$$P' = SP, \quad \text{где} \quad S = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix}$$

В цьому випадку S масштабує уявлення поверхні по осях x, y і z з коефіцієнтами α, β і γ відповідно.

Трансляція (або зсув) - це переміщення поверхні в нове положення на пристрої відображення. Для трансляції представлення поверхні ми зміщуємо кожну точку поверхні (xi, yi, zi) в нову точку, додаючи постійні (які можуть приймати нульове значення) до кожної з координат точок:

(Xi + d_x, yi + dy, zi + dz)

В цьому випадку d_x, dy і dz представляють бажані величини зміщення по осях x, y і z відповідно.

З точки зору матриці трансляцію можна виконати за допомогою матричного додавання і матриці трансляції T величиною 3 x n, як зазначено нижче:

$$P' = P + T, \quad \text{где} \quad T = \begin{bmatrix} d_x & d_x & \cdots & d_x \\ d_y & d_y & \cdots & d_y \\ d_z & d_z & \cdots & d_z \end{bmatrix}$$

Лістинг програми відображається в додатках.

Першим етапом є вибір, що цікавить поверхні *f* (x, y) і відповідного діапазону для x і y. Як показано на рис. 8, ми припускаємо:

$$f(x, y) = \frac{4 \sin \sqrt{x^2 + y^2}}{\sqrt{x^2 + y^2}}$$

Допустимий діапазон для *f*:

$$-9 \leq x \leq 9$$

$$-9 \leq y \leq 9$$

Далі потрібно вибрати відповідне число тривимірних точок для побудови в цій області площині xy розміром 18 x 18. Якщо дозволяти за п'ять точок для кожного блоку, ми отримаємо 5 · 18 x 5 · 18 = 8100 точок поверхні форми (x, y, z) = (x, y, f (x, y))

Після створення всіх точок поверхні для них будуть обрані кольору на основі їх координат по осі z. Тобто колір точки залежить від "висоти" точки "над" площиною xy. Є 11 таких кольорів "висоти": `["#060", "#090", "#0C0", "#0F0", "#9F0", "#9C0", "#990", "#960", "#930", "#900", "#C00"]`

Як можна було побачити вище, колір, який представляє мінімальне значення координати z, - це # 060 (темно-зелений). # 060 потім "зростає" до # C00 (середній червоний), максимального значення координати z. Іншими словами, мінімальне і максимальне значення координати z використовуються для лінійного розподілу (по вертикалі) 11 кольорів "висоти" серед точок на основі координати z кожної з точок.

Для кожної точки 3 x 1 в масиві points ми створюємо маленький кольоровий прямокутник (що набагато швидше, ніж візуалізація маленького кола) і маємо його, використовуючи координати x і y точки, перемножені на емпірично виведену постійну.

Перед відрисовкой точок (прямокутників) на пристрої відображення ми сортуємо точки по їх відносного розташуванню на осі z. Це гарантує, що точки, розташовані найдалі від очей користувача, малюються першими, а найближчі - останніми.

3.3 Гра «Pacman»

Pac-Man (яп. パック マン Паку-ман) — відео гра, яка вперше вийшла в 1980 в Японії. Вона стала новим стилем гри, яка не передбачає насильства, тому гра позиціонувалася і для хлопчиків, і для дівчаток.

Ігровий процес

Pac-Man [Пе́кмен] — кругла жовта істота, що складається з рота, що відкривається і закривається. Завдання гравця — зібрати всі білі крапки на рівні, уникаючи зіткнень з привидами. Рівень закінчується, коли з'їдені всі крапки. Привиди на початку рівня знаходяться в центрі екрану в невеликій кімнаті, звідки вони виходять по одному, а один привид починає рівень поза кімнатою. Якщо Пекмен «з'їсть» велику білу крапку — «енерджайзер» (англ. Energizer), то на кілька секунд привиди стають вразливими і самі уникають зустрічі з Пекменом (час, на який це відбувається, скорочується в міру зростання рівня, і починаючи з певного моменту, єдиним ефектом енерджайзера є зміна напрямку руху привидом). З'їдені в такій ситуації привиди не зникають, але відбігають від гравця, а йому нараховуються додаткові очки. Якщо зайти в коридор, який веде праворуч або ліворуч за кордон екрана, то Пекмен вийде з протилежного боку (це можуть зробити і привиди, однак, перед заходом в коридор сповільнюються, Пекмен натомість зберігає швидкість).

Дана гра реалізована за допомогою HTML5 Canvas, CSS3 та JS. Також було підключено спеціальну бібліотеку Modernizr JavaScript library 1.5.

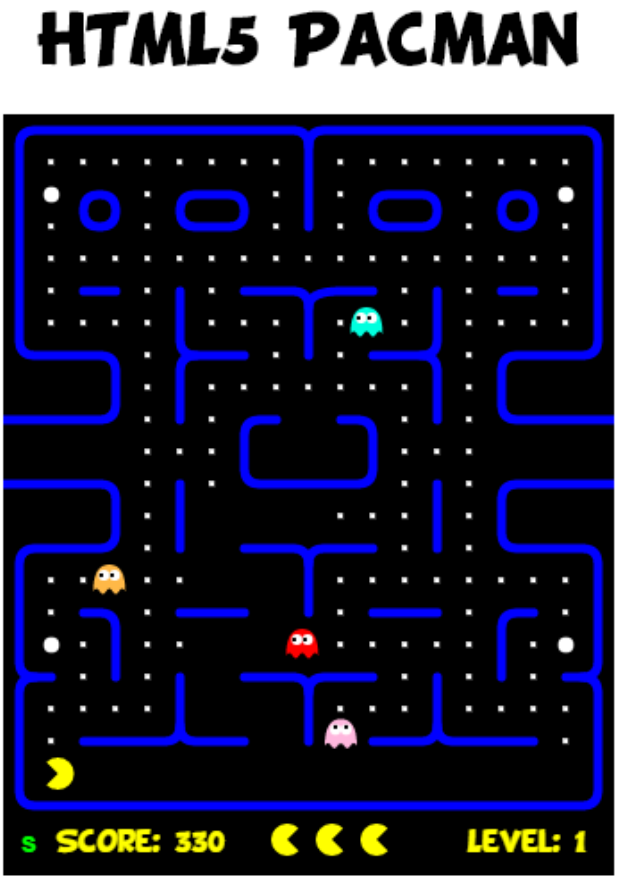

Рисунок 16 - Гра Pac-Man

У тезі head прогами написаний CSS. Він необхідний для коретного і гарного відображення гри в браузері. У тезі body програми використовується елемент canvas. У цьому тезі вказується вихідна ширина і висота елемента Canvas, а також атрибут id. Він є необхідним для додавання елемента Canvas в об'єктну модель сторінки.

Код сценарію містить банато функцій, але головними є: init, mainLoop, draw і move. Функція init викликається при завантаженні сторінки. Вона ініціалізує веб-сторінку для роботи з Canvas та підключенням аудіофайлів для різних частин гри, таких як: початок, з’їдання енерджайзера, смерть тощо. Функція mainLoop викликається з функцією init для промалювання полотна та

початку гри. Функція draw викликаєтьсядля малювання самої гри, та використовується майже в усих інших функціях. Функція move використовується для «ходів» гравця та руху привидів. Після усих функцій в коді створюється лабіринт, який буде використовуватися впродовж всієї гри. Також створюються змінні, що визначають висоту і ширину елемента canvas. Крім того, обчислюється кількість привидів, очок, рівень гри та кількість «життя» у гравця.

Також, при бажанні, є можливість вимкнути звук, чи поставити на паузу клавішами S та P відповідно.

3.4 Висновки

В даному розділі було реалізовано роботу з зображеннями, розглянуто та реалізовано алгоритми побудови трьохвимірних поверхонь та реалізовано аркадну гру «Pac-Man» за допомогою HTML5 Canvas, CSS3 та JS. Ці мультимедійні продукти презентують доцільність та актуальність використання нового стандарту HTML.

4. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для демонстрації мультимедійних можливостей HTML5 Canvas. Інтерфейс користувача був розроблений за допомогою HTML5, CSS3 та Javascript у середовищі розробки JetBrains WebStorm.

Програмний продукт призначено для використання в браузерах на персональних комп'ютерах під управлінням будь-якої операційної системи.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.1 Постановка задачі

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки системи аналізу нелінійних нестаціонарних процесів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки даних та відклик користувачеві у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

4.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мультимедійної платформи;

F_2 – вибір конкретного браузеру;

F_3 – жанр гри.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мультимедійна платформа Flash;

б) мультимедійна платформа HTML5 Canvas;

Функція F_2:

а) браузер Google Chrome;

б) браузер Amigo.

Функція F_3:

а) гра, ідея якої взята з прототипу, що був зроблений після 2000-х;

б) гра, ідея якої взята з прототипу, що був зроблений до 2000-х;.

4.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рисунок 17). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 5).

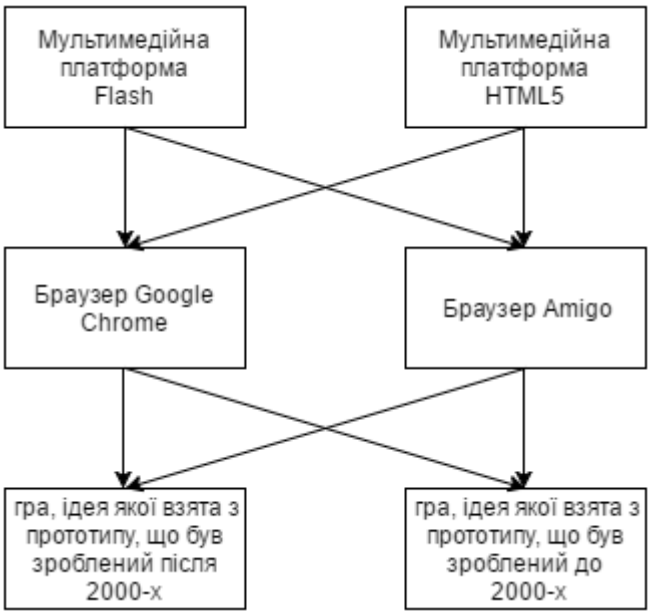

Рисунок 17 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Таблиця 5 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Займає менше часу при написанні коду	Не підтримує роботу на мобільних пристроях
	Б	Код швидко виконується, підтримує роботу на мобільних пристроях	Займає більше часу при написанні коду
F2	А	Простий у використанні, підтримує більшість типів мультимдіа	Використовує багато оперативної пам’яті
	Б	Найоптимальніший для використання тільки у межах РФ	Багато непотрібного софту, що заважає роботі ПК
F3	*А*	Новітні алгоритми для використання в іграх	Важкі в написанні коду
	Б	Легкі в написанні коду, величезна потенційна аудиторія	Молодша аудиторія їх не сприймає

Функція *F1:*

Оскільки в межах дипломної роботи ставиться написання програми, що працює на всіх пристроях, та демонструє зручне мультимедіа засобами HTML5 Canvas, тому варіант а) має бути відкинутий.

Функція *F2:*

Оскільки планується розглядати широкий спектр мультимедіа, шо підтримується більшістю браузерів, та перший є найбільш популярним браузером у світі, то варіант б) має бути відкинутий

Функція *F3:*

Обидва варіанти є гідними реалізації, тому вважаємо варіанти а) та б) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1б – F2а – F3а
2. F1б – F2а – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.2 Обґрунтування системи параметрів ПП

4.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – завантаженість підключення до Інтернету;

– *X2* – об’єм оперативної пам’яті, що використовується;

– *X3* – час відклику на натискання клавіш;

– *X4* – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від завантаженості лінії зв’язку.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії під час натискання клавіш.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

4.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 2.

За даними таблиці 6 будуються графічні характеристики параметрів – рисунки 18 – 21.

Таблиця 6 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Завантаженість підключення до Інтернету	X1	Мб/с	700	100	60
Об’єм оперативної пам’яті, що використовується	X2	Мб	2000	800	300
Час відклику на натискання клавіш	X3	мс	200	100	50
Потенційний об’єм програмного коду	X4	кількість строк коду	1800	1000	500

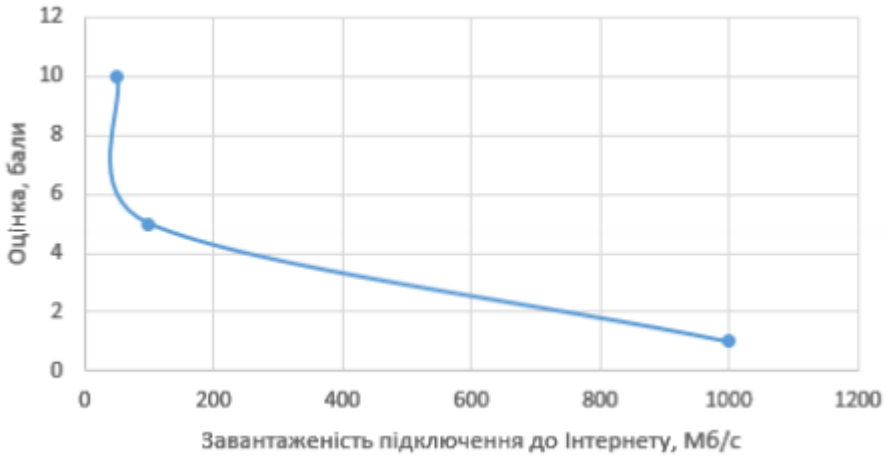

Рисунок 18 – X1, завантаженість підключення до Інтернету

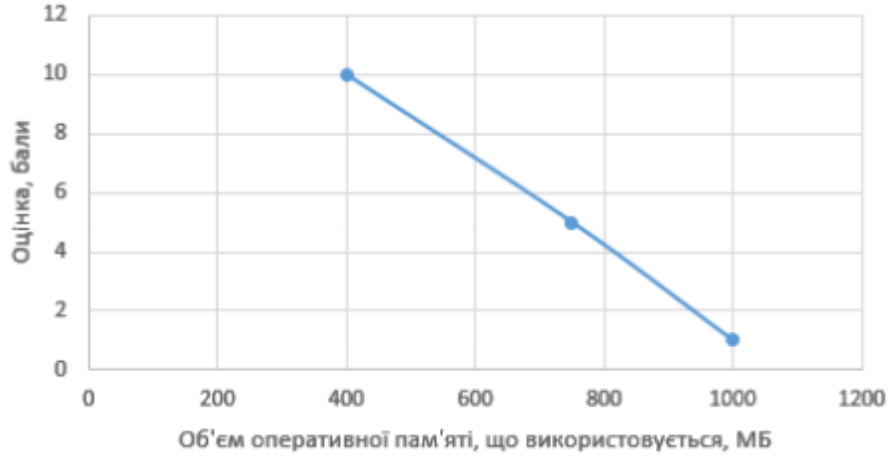

Рисунок 19 – X2, об’єм оперативної пам’яті, що використовується

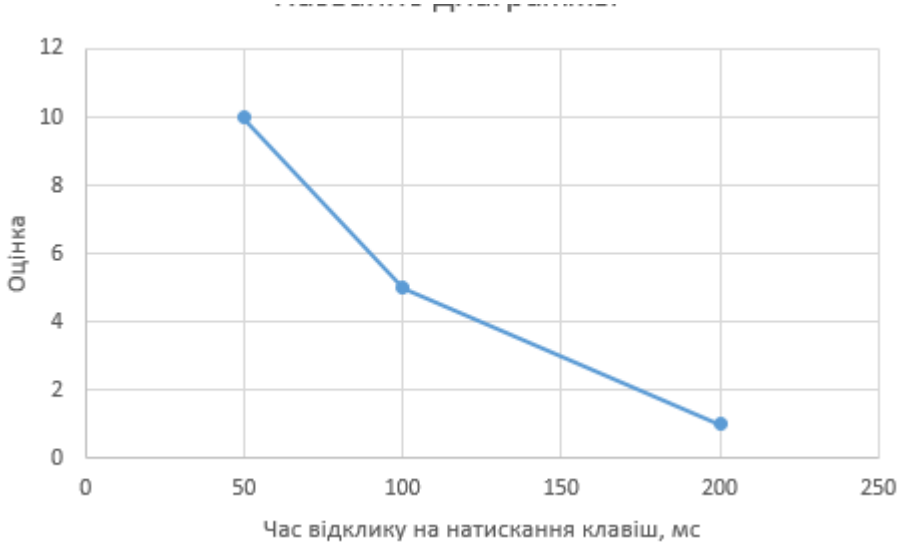

Рисунок 20 – X3, час відклику на натискання клавіш

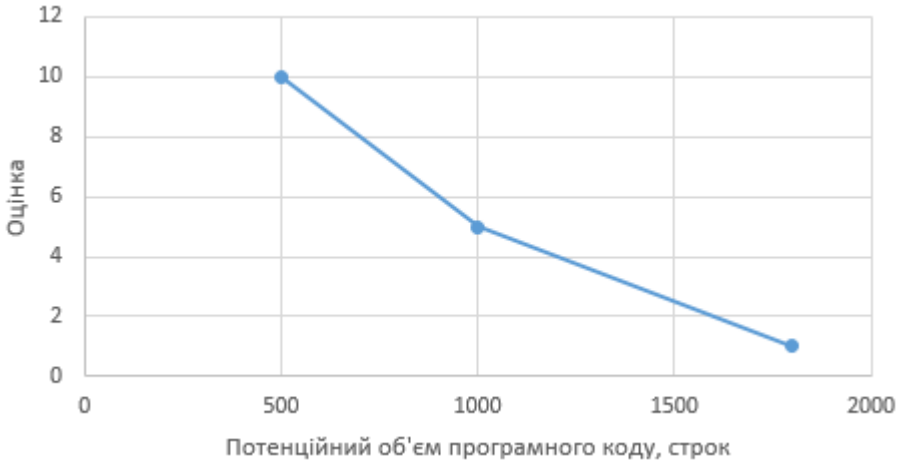

Рисунок 21 – X4, потенційний об’єм програмного коду

4.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який має найбільш зручний інтерфейс та зрозумілу взаємодію з користувачем

Таблиця 7 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Завантаженість підключення до Інтернету	Мб/с	3	2	2	1	1	2	2	13	-4,5	20,25
X2	Об’єм оперативної пам’яті	Мб	1	1	1	2	2	1	1	9	-8,5	72,25
X3	Час відклику на натискання клавіш	Мс	2	3	3	3	3	3	4	21	3,5	12,25
X4	Об’єм програмного коду	кількість строк коду	4	4	4	4	4	4	3	27	9,5	20,25
	Разом		10	10	10	10	10	10	10	70	0	195

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 7.

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \quad (3)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5. \quad (4)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \quad (5)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 195. \quad (6)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 195}{7^2(4^3 - 4)} = 0{,}8 > W_k = 0{,}67 \quad (7)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.

Таблиця 8 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	>	>	<	<	<	0,5
X1 і X3	<	>	>	>	>	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	>	>	>	>	>	>	>	>	1,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	<	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}. \quad (8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j. \quad (9)$$

Як видно з таблиці 5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 9 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	0,5	1,5	1,5	4,5	0,281	16,25	0,275	59,125	0,274
X2	1,5	1,0	1,5	1,5	5,5	0,344	21,25	0,36	77,875	0,361
X3	0,5	0,5	1,0	1,5	3,5	0,219	12,25	0,208	44,875	0,207
X4	0,5	0,5	0,5	1,0	2,5	0,156	9,25	0,157	34,125	0,158
Всього:					16	1	59	1	216	1

4.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм оперативної пам’яті, що використовується) та *X1* (завантаженість підключення до Інтернету) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*час відклику на натискання клавіш) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 80 мс або варіанту б) 55мс.

Абсолютне значення параметра *X4 (*потенційний об’єм програмного коду) обрано не найкращим (не мінімальним), тобто це значення відповідає або варіанту а) 1500 або варіанту б) 800

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\ , \qquad (10)$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

Таблиця 10 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри, що беруть участь у реалізації функцій	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	Б	X1	100	5	0,274	1,37
F2	А	X2	500	6,7	0,361	2,42
F3	А	X3	80	5,5	0,207	1,14
	Б		55	7,5	0,207	1,55
	А	X4	1500	2,5	0,158	0,395
	Б		800	6,4	0,158	1,01

За даними з таблиці 6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \qquad (11)$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 1,37 + 2,42 + 1,14 + 0,395 = 5,325

K_{K2} = 1,37 + 2,42 + 1,55 + 1,01 = 6,35

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (12)$$

де T_P – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт;$K_{СК}$ – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови

програмування;$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$Т_1$ = 90·1.7·0.8 = 122.4 людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =27 людино-днів, $К_П$ =0.9,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$Т_2$ = 27 · 0.9 · 0.8 = 19.44 людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I$ = (122.4 + 19.44 + 4.8 + 19.44) · 8 = 1328,64 людино-годин;

$Т_{II}$ = (122.4 + 19.44 + 6.91 + 19.44) · 8 = 1345.52 людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці бере участь один програміст з окладом 6000 грн., один фінансовий аналітик з окладом 9000грн. Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} грн., \quad (13)$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$С_ч = \frac{6000 + 9000}{2 \cdot 21 \cdot 8} = 44{,}64 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д, \quad (14)$$

де $С_ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 44{,}64 \cdot 1328.64 \cdot 1.2 = 71172{,}59$ грн.

II. $С_{ЗП} = 44{,}64 \cdot 1345.52 \cdot 1.2 = 72076{,}82$ грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 71172{,}59 \cdot 0.22 = 24429$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.3677 = 67281{,}38 \cdot 0.22 = 24739$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 6000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 6000 \cdot 0,2 = 16800$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 16800 \cdot (1 + 0.2) = 20280$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 20280 \cdot 0,22 = 4461,6$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,156 \cdot 0,9733 \cdot 2,0218 = 523.83$ грн.,

де N_C – середньо-споживча потужність приладу; $K_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0,67 = 5360$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 17280 + 6353,86 + 2300 + 460 + 523.83 + 5360 = 33883,55$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 33883,55/1706,4 = 20,68$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т$$

I. $С_М = 20,68 \cdot 1328,64 = 27476$ грн.;

II. $С_М = 20,68 \cdot 1345.52 = 27825$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

I. $С_Н = 71172,59 \cdot 0,67 = 47685,63$ грн.;

II. $С_Н = 72076,82 \cdot 0,67 = 48291,47$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП}$ = 71172,59 + 4461,6 + 27476 + 47685,63 = 150795,82 грн.;

II. $С_{ПП}$ = 72076,82 + 4461,6 + 27825 + 48291,47 = 152654,89 грн.;

4.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$К_{ТЕР1}$ = 5,325 / 150795,82 = $0,35 \cdot 10^{-4}$;

$К_{ТЕР2}$ = 6,35 / 152654,89 = $0,42 \cdot 10^{-4}$;

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0,42 \cdot 10^{-4}$.

4.6 Висновки

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих

варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0,42 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мультимедійна платформа HTML5 Canvas;

– браузер Google Chrome;

– гра, ідея якої взята з прототипу, що був зроблений до 2000-х.

Даний варіант виконання програмного комплексу дає можливість створити новий веб-додаток, який буде популярний і в майбутньому завдяки новим стандартам HTML5 та буде використовуватись на будь-яких пристроях значної частини користувачів.

ВИСНОВКИ

Результатом даної роботи є аналіз особливостей та ефективності використання HTML5 Canvas для вирішення задачі відображення різних типів мультимедіа. Продуктом даного аналізу є різні програми, реалізовані за допомогою HTML5 Canvas, що показує різноманітність використання даної технології.

Були досліджені основні аналоги, які на сьогоднішній день представлені на ринку, виділені їх основні переваги та недоліки. Оскільки жодна з існуючих систем не орієнтована на врахування індивідуальних побажань користувачів при виборі пристрою для перегляду графіки та мультимедіа, то було доведено доцільність використання саме нового стандарту HTML.

Наведено порівняльну характеристику роботи HTML5 та його аналогів у вигляді графіків. Для ілюстрації роботи з зображенням засобами HTML5 Canvas біло зроблено декілька перетворень вхідного зображення. Для ілюстрації відображення трьохвимірного простору було розроблено алгоритм повороту 3D зображення поверхні та реалізовано. Також для демонстрації ігри на HTML5 Canvas біло реалізовано гру Pac-Man.

HTML5-тег Canvas грає центральну роль при створенні додатків на базі браузерів. Він надає практичну середу малювання, що базується на технології JavaScript і обмежену лише уявою розробника. Це середовище не дуже складне для освоєння. Крім того, в Інтернеті є безліч інструментів підтримки, що полегшують навчання і підготовку, включаючи інструкції щодо використання, блоги, онлайн статті, відео-та текстові посібники, типові програми.

Canvas використовується, як правило, для малювання графіків для статей і ігрового поля в деяких браузерних іграх. Але також може використовуватися для

вбудовування відео в сторінку і створення повноцінного плеєра. Використовується в WebGL для апаратного прискорення 3D-графіки. Canvas безальтернативно слід використовувати для обробки зображень і відео, добре показує себе у складних сценах з великою кількістю об'єктів і можна хитро використовувати в веб-рекламі, так як реклама буде показуватися не буквами, а набором точок, що лиш схожі на них.

Зміст і головна функція canvas полягають в тому, що його вміст можна динамічно оновлювати, відрисовуючи нові елементи у відповідь на дії користувача. Ця його здатність реагувати на події, ініційовані відвідувачем сторінки, робить можливим створення таких інструментів і ігор, які раніше вимагали б застосування сторонніх технологій, таких як Флеш.

Можливість змінювати тексти та зображення в візуальному режимі і імітувати рух робить Canvas виключно цінним інструментом. В будь-якому випадку Canvas є найважливішим компонентом можливостей, що надаються специфікацією HTML5.

ПЕРЕЛІК ПОСИЛАНЬ

1. Журавель И.М. Краткий курс теории обработки изображений Режим доступу: http://matlab.exponenta.ru/imageprocess/book2/index.php Дата доступу – 5.05.2016
2. RIP Flash: Why HTML5 will finally take over video and the Web this year, Режим доступу: http://thenextweb.com/dd/2014/04/19/rip-flash-html5-will-take-video-web-year/ Дата доступу – 5.05.2016
3. Flash vs HTML5, Режим доступу: http://flashvhtml.com/ Дата доступу – 5.05.2016
4. Top 12 browser versions from 2015 to 2016 Режим доступу: http://gs.statcounter.com/#desktop-browser_version-ww-monthly-201502-201602-bar Дата доступу – 5.05.2016
5. Worldwide PC Shipments Declined 6.9 Percent in Fourth Quarter of 2013, Режим доступу: http://www.gartner.com/newsroom/id/2647517 Дата доступу – 5.05.2016
6. Infographic: 2013 Mobile Growth Statistics, Режим доступу: http://www.digitalbuzzblog.com/infographic-2013-mobile-growth-statistics Дата доступу – 5.05.2016
7. HTML5 Canvas and the Canvas Shadow DOM, Режим доступу: https://msdn.microsoft.com/ru-ru/library/hh968259(v=vs.85).aspx Дата доступу – 5.06.2016

8. Будник С.А. Який вибір ми зробимо: HTML 5 чи Flash? / Будник С.А. // Системний аналіз та інформаційні технології : 18-а Міжнародна науково-технічна конференція «SAIT 2016», Київ, 30 травня – 2 червня 2016 р. / ННК “ІПСА” НТУУ “КПІ”. – К.: ННК “ІПСА” НТУУ “КПІ”, 2016. – С. 330

9. Медведський А.М. Застосування WhitestormJS для оптимізації 3Д графіки в онлайн іграх / Медведський А.М. // Системний аналіз та інформаційні технології : 18-а Міжнародна науково- технічна конференція «SAIT 2016», Київ, 30 травня – 2 червня 2016 р. / ННК “ІПСА” НТУУ “КПІ”. – К.: ННК “ІПСА” НТУУ “КПІ”, 2016. – С. 390

10. Все про HTML 5 Режим доступу: http://htmlbook.ru/html5 Дата доступу – 5.06.2016

11. Б. Лоусон, Р. Шарп - Изучаем HTML5. Библиотека специалиста/ . – Санкт-Петербург : «Питер», 2011. -С. , 200, 235

12. Шмитт К. - CSS. Рецепты программирования Режим доступу: http://static.ozone.ru/multimedia/book_file/1005871779.pdf Дата доступу – 5.06.2016

13. Клиентская оптимизация в действии: прикладные методы и приемы Режим доступу: http://www.cap-design.ru/rw/6-6-5.htm Дата доступу – 14.06.2016

14. GUIMARK 2: THE RISE OF HTML5 Режим доступу: http://www.craftymind.com/guimark2/ Дата доступу – 14.06.2016

15. Веб - ресурс Moodle НТУУ КПІ Режим доступу: http://moodle.kpi.ua Дата доступу – 5.06.2016

Додаток А

Лабораторна робота

З дисципліни Комп’ютерна графіка

Тема: "Навички роботи з Canvas: зміна кольору фотографії".

Мета: Ознайомитися з нововведенням HTML5 Canvas на прикладі роботи з кольорами малюнків.

Методика виконання роботи

Роботу треба виконувати у такій послідовності:

1. Дослідити властивості та синтаксис Canvas. Для вибору завдання необхідно використати таблицю 1. У таблиці 1 наведено назву кольору, яку необхідно виключити з малюнку.

2. Ознайомитися з фрагментами коду, що містяться в лабораторній роботі.

3. Для отримання нового малюнку без відповідного кольору треба змінити текст завдання (наведено в прикладі)

Протокол повинен містити завдання на виконання, код завдання, скріншот екрану зі зміненим малюнком та висновки.

Теоретичні відомості

Елемент Canvas в HTML5 використовує пікселі для створення графічних зображень на екрані. За допомогою цих методів є можливість розробляти унікальні зображення для ефектного художнього оформлення веб-сайтів, блогів, відеоігор, рекламних оголошень, ілюстрацій і багатьох інших елементів. Елемент Canvas підтримує immediate mode для відображення пікселів безпосередньо на екрані, що робить його ефективним засобом розробки спеціальних фотоефектів. Підтримка миттєвого режиму дозволяє зчитувати і записувати кожен піксель

елемента Canvas, що дає можливість створювати складні візуальні ефекти, змінюючи характеристики пікселів. При імпорті і відображенні фотографії в елементі Canvas зображення перетворюється в масив кольорових пікселів, кожен з яких має значення червоного, синього, зеленого кольорів, а також альфа-фактора (визначає прозорість). Оскільки елемент Canvas дозволяє змінювати кольори окремих пікселів, а також малювати в масиві пікселів точки, лінії і фігури, з його допомогою можна з легкістю створювати привабливі фотоефекти. Приклади коду написані з використанням JavaScript і елемента Canvas в HTML5. Приклади мають найпростішу структуру і призначені для демонстрації основних принципів використання функцій елемента Canvas. При необхідності можна з легкістю розширити цей код, реалізувавши власні фотоефекти відповідно до вимог конкретних веб-додатків.

У тезі body програми використовується функція onload для виклику функції displayImage при завантаженні сторінки. Початкове зображення завантажується в тег <body>, що дозволяє порівнювати з ним зображення, яке буде змінено за допомогою Canvas. Подія onerror додається в елемент img і визначає, завантажено чи зображення. Елемент canvas входить до складу тега <body>. У цьому тезі вказується вихідна ширина і висота елемента Canvas, а також атрибут id. Він є необхідним для додавання елемента Canvas в об'єктну модель сторінки.

код сценарію

Код сценарію містить чотири функції: displayImage, getColorData, putColorData і noPhoto. Функція displayImage викликається при завантаженні сторінки. Функції getColorData і putColorData викликаються з функції displayImage. Функція noPhoto викликається, якщо при завантаженні фотографії виникає помилка. На початку розділу сценаріїв в коді створюються глобальні змінні для створення порожнього файлу зображення, який буде використовуватися пізніше. Також створюються змінні, що визначають висоту і ширину елемента canvas. Крім того,

обчислюється довжина масиву зображення на основі числа 4-байтових значень в масиві.

функція displayImage

Ця функція викликається при завантаженні сторінки. Вона отримує елемент canvas за допомогою ID елемента canvas в коді тега <body>. Потім виходить об'єкт CanvasRenderingContext2D Object елемента canvas, що дозволяє підготуватися до отрисовке; для завантаження зображення в контекст використовується функція DrawImage. Після ініціалізації контексту як двовимірного елемента canvas можна виконати отрисовку.

Далі в функції вказується шлях до джерела зображення. Оскільки завантаження зображення може займати деякий час, визначається подія, яка викликає функцію по завершенні завантаження. Після того, як зображення було завантажене воно відображається, і викликаються функції getColorData і putColorData, які виконують отрисовку.

Зображення корабля має зберігатися на вашому комп'ютері. Виклик зображення з іншого домену не підтримується. Якщо джерело пікселів для елемента canvas знаходиться в іншому домені, зображення розглядається як загроза безпеці і не підлягає зміні. Додаткові відомості про загрози безпеки між доменами см. В документації по canvas. Для виконання сценарію можна використовувати будь-який файл з ім'ям kestral.png.

функція getColorData

Зображення копіюється за допомогою функції getImageData в об'єкт myImage і обробляється перебором значень в масиві даних зображення і присвоєння байтам, відповідним червоному кольору, нульових значень. Масив для зображення розміром 200 на 200 пікселів містить 40 000 осередків. Кожна осередок містить 4 байта даних, в результаті чого загальний розмір масиву становить 160 000 байт. У програмі цикл виконується 160 000 разів. Кожен раз

зчитується 4 байта. У кожному проході циклу першому байту присвоюється нульове значення. В результаті з палітри зображення виключається червоний колір. Байти зеленого і синього кольорів не змінюються.

Для читання і запису пікселів на екрані кожного пікселя необхідно привласнити колір. Значення кольору складається з чотирьох компонентів: червоний, зелений, синій і альфа-канал. Наприклад, білий піксель складається з 100% червоного, 100% зеленого, 100% синього і 100% альфа-каналу. (Альфа-канал відображає ступінь прозорості кольору.) Значення кольору для кожного пікселя в елементі canvas зберігається у вигляді масиву розміром 4 байта. Всі пікселі елемента canvas зберігаються в одному масиві. Відлік починається з верхнього лівого і закінчується в правому нижньому кутку. Розмір масиву визначається перемножением висоти елемента canvas на його ширину і на число 4 (розмір масиву для одного пікселя).

Чотири байта відповідають відсотковим значенням червоного, зеленого, синього кольору і альфа-каналу для кожного пікселя. (Значення байтів знаходяться в діапазоні від 0 до 255 для десяткових чисел і від # 0 до #FF для шістнадцятирічних.) У коді, наведеному в даному розділі, показано, як в миттєвому режимі Canvas виконується запис кольору пікселів на екрані.

функція putColorData

Ця функція записує змінені дані зображення в елемент canvas за допомогою методу putImageData. Найчастіше рекомендується виконувати фактичне опрацювання другого зображення і відображати результат тільки після завершення обробки.

Приклад виконання лабораторної роботи

Колір, що потрібно видалити з таблиці 1		
червоний	**зелений**	**синій**
1	1	1

Лістинг програми на мові HTML

```
<!DOCTYPE html>
<html>
 <meta charset="utf-8">
  <head>
    <script type="text/javascript">

      //Global variables
      var picWidth = 200; // width of the canvas
      var picHeight = 200; // height of the canvas
      var picLength = picWidth * picHeight; // number of chunks
      var myImage = new Image(); // Create a new blank image.

      // Load the image and display it.
      function displayImage() {

        // Get the canvas element.
        canvas = document.getElementById("myCanvas");

        // Make sure you got it.
        if (canvas.getContext) {

          // Specify 2d canvas type.
          ctx = canvas.getContext("2d");

          // When the image is loaded, draw it.
          myImage.onload = function() {
            // Load the image into the context.
            ctx.drawImage(myImage, 0, 0);

            // Get and modify the image data.
```

```
      getColorData();

      // Put the modified image back on the canvas.
      putColorData();
    }

    // Define the source of the image.
    // This file must be on your machine in the same folder as this web page.
    myImage.src = "1.jpg";
  }
}

function getColorData() {

  myImage = ctx.getImageData(0, 0, 200, 200);

  // Loop through data.
  for (var i = 0; i < picLength * 4; i += 4) {

    // First bytes are red bytes.
    // Get red value.
    var myRed = myImage.data[i];

    // Second bytes are green bytes.
    // Get green value.
    var myGreen = myImage.data[i + 1];

    // Third bytes are blue bytes.
    // Get blue value.
    var myBlue = myImage.data[i + 2];

    // Fourth bytes are alpha bytes
    // We don't care about alpha here.
    // Add the three values and divide by three.
    // Make it an integer.
    myGray = parseInt((myRed + myGreen + myBlue) / 3);

    // Assign average to red, green, and blue.
    myImage.data[i] = myGray;
```

```
        myImage.data[i + 1] = myGray;
        myImage.data[i + 2] = myGray;
      }
    }

    function putColorData() {

      ctx.putImageData(myImage, 0, 0);
    }

    function noPhoto() {

      alert("Please put a .png file in this folder and name it kestral.png.");

      }

  </script>
</head>

<body onload="displayImage()">
  <h1>
    Чорно-біла фотографія
  </h1>
  <p>
    Оригінал зліва, модифіковане зображення - справа:
  </p>
  <img id="myPhoto" src="1.jpg" onerror="noPhoto()">
  <canvas id="myCanvas" width="200" height="200">
  </canvas>

</body>
```

Результати

Чорно-біла фотографія

Оригінал зліва, модифіковане зображення - справа:

Рис 1. Модифіковане зображення

Таблиця 1. Варіанти завдань до лабораторної роботи

Варіант	**Колір, що потрібно видалити**		
	червоний	**зелений**	**синій**
1	1	1	0
2	1	1	1
3	1	0	1
4	0	1	1
5	0	1	0
6	0	0	1
7	0	1	1
8	1	1	1
9	1	0	0
10	0	1	0
11	1	0	1
12	1	0	0

13	1	1	0
14	1	1	1

***Для тих варіантів, де для всіх кольорів поставлена 1 мається на увазі зробити чорно-біле зображення**

Контрольні питання:

1. Що таке HTML? Що таке показники Canvas?

2. Як видалити з фотографії жовтий колір?

3. Який байт у масиві відповідає за синій колір?

4. Для чого в масиві байтів 4-й байт?

Додаток Б

Код програми, що створює рамки для фотографії:

```
<!DOCTYPE html>
<html>
  <meta charset="utf-8">
  <head>
    <script type="text/javascript">

      //Global variables
      var myImage = new Image(); // Create a new blank image.

      // Load the image and display it.
      function displayImage() {

        // Get the canvas element.
        canvas = document.getElementById("myCanvas");

        // Make sure you got it.
        if (canvas.getContext) {

          // Specify 2d canvas type.
          ctx = canvas.getContext("2d");

          // When the image is loaded, draw it.
          myImage.onload = function() {

            // Load the image into the context.
            ctx.drawImage(myImage, 0, 0);

            // Get and modify the image data.
            changeImage();
          }
```

```
          // Define the source of the image.
          myImage.src = "1.jpg";
        }
      }

      function changeImage() {

        ctx.strokeStyle = "white";
        ctx.lineWidth = "100";
        ctx.beginPath();
        ctx.arc(100, 100, 150, 0, Math.PI * 2, true);
        ctx.closePath();
        ctx.stroke();
      }
    </script>
  </head>

  <body onload="displayImage()">
    <h1>
      Закруглені кути

    </h1>
    <p>
      Оригінал зліва, модифіковане зображення - справа:
    </p>
    <img id="myPhoto" src="1.jpg">
    <canvas id="myCanvas" width="200" height="200">
    </canvas>

  </body>

</html>
```

Код програми, що перетворює зображення на чорно-біле:

```
<!DOCTYPE html>
<html>
```

```
<meta charset="utf-8">
 <head>
   <script type="text/javascript">

     //Global variables
     var picWidth = 200; // width of the canvas
     var picHeight = 200; // height of the canvas
     var picLength = picWidth * picHeight; // number of chunks
     var myImage = new Image(); // Create a new blank image.

     // Load the image and display it.
     function displayImage() {

       // Get the canvas element.
       canvas = document.getElementById("myCanvas");

       // Make sure you got it.
       if (canvas.getContext) {

         // Specify 2d canvas type.
         ctx = canvas.getContext("2d");

         // When the image is loaded, draw it.
         myImage.onload = function() {
           // Load the image into the context.
           ctx.drawImage(myImage, 0, 0);

           // Get and modify the image data.
           getColorData();

           // Put the modified image back on the canvas.
           putColorData();
         }

         // Define the source of the image.
         // This file must be on your machine in the same folder as this web
page.
```

```
    myImage.src = "1.jpg";
  }
}

function getColorData() {

  myImage = ctx.getImageData(0, 0, 200, 200);

  // Loop through data.
  for (var i = 0; i < picLength * 4; i += 4) {

    // First bytes are red bytes.
    // Get red value.
    var myRed = myImage.data[i];

    // Second bytes are green bytes.
    // Get green value.
    var myGreen = myImage.data[i + 1];

    // Third bytes are blue bytes.
    // Get blue value.
    var myBlue = myImage.data[i + 2];

    // Fourth bytes are alpha bytes
    // We don't care about alpha here.
    // Add the three values and divide by three.
    // Make it an integer.
    myGray = parseInt((myRed + myGreen + myBlue) / 3);

    // Assign average to red, green, and blue.
    myImage.data[i] = myGray;
    myImage.data[i + 1] = myGray;
    myImage.data[i + 2] = myGray;
  }
}
```

```
      function putColorData() {

        ctx.putImageData(myImage, 0, 0);
      }

      function noPhoto() {

        alert("Please put a .png file in this folder and name it kestral.png.");

        }

    </script>
  </head>

  <body onload="displayImage()">
    <h1>
      Чорно-біла фотографія
    </h1>
    <p>
      Оригінал зліва, модифіковане зображення - справа:
    </p>
    <img id="myPhoto" src="1.jpg" onerror="noPhoto()">
    <canvas id="myCanvas" width="200" height="200">
    </canvas>

  </body>

</html>
```

Код програми, що модифікує колір зображення:

```
<!DOCTYPE html>
<html>
<meta charset="utf-8">
  <head>
    <script type="text/javascript">

      //Global variables
```

```
var picWidth = 200; // width of the canvas
var picHeight = 200; // height of the canvas
var picLength = picWidth * picHeight; // number of chunks
var myImage = new Image(); // Create a new blank image.

// Load the image and display it.
function displayImage() {

  // Get the canvas element.
  canvas = document.getElementById("myCanvas");

  // Make sure you got it.
  if (canvas.getContext) {

    // Specify 2d canvas type.
    ctx = canvas.getContext("2d");

    // When the image is loaded, draw it.
    myImage.onload = function() {
      // Load the image into the context.
      ctx.drawImage(myImage, 0, 0);

      // Get and modify the image data.
      getColorData();

      // Put the modified image back on the canvas.
      putColorData();
    }

    // Define the source of the image.
    // This file must be on your machine in the same folder as this web
page.
    myImage.src = "1.jpg";
  }
}

function getColorData() {
```

```
      myImage = ctx.getImageData(0, 0, 200, 200);

      // Loop through data.
      for (var i = 0; i < picLength * 4; i += 4) {

        // First bytes are red bytes.
        // Remove all red.
        myImage.data[i] = 0;
            myImage.data[i+1] = 0;
            myImage.data[i+2] = 0;
        // Second bytes are green bytes.
        // Third bytes are blue bytes.
        // Fourth bytes are alpha bytes
      }
    }

    function putColorData() {

      ctx.putImageData(myImage, 0, 0);
    }

    function noPhoto() {

      alert("Please put a .png file in this folder and name it kestral.png.");

      }

  </script>
</head>

<body onload="displayImage()">
  <h1>
    Зміна кольору на фотографії
  </h1>
  <p>
```

```
    Оригінал зліва, модифіковане зображення - справа:
  </p>
  <img id="myPhoto" src="1.jpg" onerror="noPhoto()">
  <canvas id="myCanvas" width="200" height="200">
  </canvas>

</body>

</html>
```

Код веб-додатку (гра Пак-мен):

Index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>HTML5 Pacman</title>

    <style type="text/css">
      @font-face {
          font-family: 'BDCartoonShoutRegular';
        src: url('BD_Cartoon_Shout-webfont.ttf') format('truetype');
          font-weight: normal;
          font-style: normal;
      }
      #pacman {
        height:450px;
        width:342px;
        margin:20px auto;
      }
      #shim {
        font-family: BDCartoonShoutRegular;
        position:absolute;
        visibility:hidden
      }
      h1 { font-family: BDCartoonShoutRegular; text-align:center; }
      body { width:342px; margin:0px auto; font-family:sans-serif; }
      a { text-decoration:none; }
    </style>

  </head>

  <body>

    <div id="shim">shim for font face</div>

    <h1>HTML5 Pacman</h1>



    <div id="pacman"></div>
    <script src="pacman.js"></script>
    <script src="modernizr-1.5.min.js"></script>
```

```
  <script>

    var el = document.getElementById("pacman");

    if (Modernizr.canvas && Modernizr.localstorage &&
        Modernizr.audio && (Modernizr.audio.ogg || Modernizr.audio.mp3)) {
      window.setTimeout(function () { PACMAN.init(el, "./"); }, 0);
    } else {
      el.innerHTML = "Sorry, needs a decent browser<br /><small>" +
        "(firefox 3.6+, Chrome 4+, Opera 10+ and Safari 4+)</small>";
    }
  </script>

</body>
</html>
};
```