Хмельницький національний університет
Факультет інформаційних систем
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА
МАГІСТРА

на тему *Метод візуалізації хмар точок «Web point cloud viewer» для прийняття контрольованих людиною критично-безпекових рішень*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Виконав:	*студент 2 курсу, група КНм-20-1*	Підпис	*Н.М.Тростинський* Ініціали, прізвище
Керівник:	*к.т.н., доцент кафедри ІПЗ*	Підпис	*О.М. Яшина* Ініціали, прізвище
Нормоконтроль:	*к.т.н., доцент кафедри КН*	Підпис	*Р.О. Багрій* Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор Підпис *О.В. Бармак* Ініціали, прізвище
3 грудня 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *магістр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

_____
(підпис)
д.т.н., професор О.В. Бармак
« 1 » вересня *2021* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

1. Тема кваліфікаційної роботи магістра: *«Метод візуалізації хмар точок «Web point cloud viewer» для прийняття контрольованих людиною критично-безпекових рішень»*

2. Завдання видано студенту *Тростинському Назару Миколайовичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *к.т.н., доцент Яшина Оксана Миколаївна*
(прізвище, ім'я, по батькові)

4. Затверджені наказом університету від «25» серпня *2021* р. № 102

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – розробка методу візуалізації хмар точок «Web point cloud viewer» для прийняття контрольованих людиною критично-безпекових рішень

Реферат

Актуальність роботи. В сучасних технологіях інформація про об’єкти, зокрема їх поверхні, у вигляді хмари точок широко застосовуються у різноманітних сферах діяльності людини - як результат дистанційного зонування Землі, в системах автоматизованого проектування, при тривимірній цифровій реконструкції будівель, до прикладу пам’яток архітектури, та приміщень, тощо.

Найбільш ефективною технологією отримання необхідної хмари точок є лазерне сканування. Це обумовлене стрімким розвитком лазерної техніки та інформаційно-комп’ютерних технологій що супроводжується покращенням технічних характеристик, з одного боку, та зниженням вартості – з іншого. Суттєве та стрімке покращення техніко-економічних показників систем тривимірного лазерного сканування обумовлює їх розширену доступність за запровадження у найрізноманітніших галузях.

Хмара точок, отримана в результаті лазерного сканування, є первинним результатом, що потребує якісного опрацювання. Подальша робота полягає у вирішенні двох головних й, іноді, взаємовиключних задач – підвищення точності відтворення поверхонь, що вирішується шляхом збільшенням точок у хмарі, та забезпечення необхідної швидкодії, з діаметрально протилежним варіантом розв’язання.

Розробка та запровадження належних ефективних інформаційних технологій з використанням відповідних сучасних засобів розробки дозволяють реалізувати системи, які одночасно забезпечують високу точність сканування у поєднанні з прийнятною швидкодією, тобто, дозволяють оперувати великими хмарами точок з прийнятними термінами виконання.

Хмара точок – це цифрове 3D-зображення фізичного об’єкта чи простору [1]. Він складається з мільйонів окремих точок вимірювання, кожна з координатами x, y і z.

Залежно від методу, який використовується для захоплення хмари – і датчиків – кожна точка може також містити дані про колір RGB або навіть інформацію про інтенсивність, яка відображає силу зворотного впливу лазерного імпульсу, який генерував точку.

Для зйомки хмари точок можна використовувати два основних інструменти: лазерні сканери та фотограмметрію.

Лазерний сканер – це система оглядового рівня, яка включає в себе ряд різних датчиків та технологій. Найважливішим є датчик лідара, який використовує швидкі лазерні імпульси для збору сотень тисяч надзвичайно точних вимірювань за секунду. Більшість лазерних сканерів також включають камеру RGB для додавання кольору до хмари точок та одиницю вимірювання інерції (IMU) [2].

Фотограмметрія – це скоріше методологія, ніж конкретний тип інструменту. Щоб створити хмару точок за допомогою фотограмметрії, потрібно було б камерами захоплювати простір з усіх ракурсів, а потім обробляти ці зображення за допомогою спеціалізованого програмного забезпечення для реконструкції простору в 3D [3].

Дослідження підтвердили, що лазерні сканери дають більш точні дані, ніж фотограмметрія.

Мета роботи. Мета роботи полягає у реалізації методу «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень з достатнім рівнем точності та швидкодії, обчислювальної здатності та доступності.

Завдання полягає у розв’язанні часткових поставлених задач:

– провести аналіз існуючих методів, технологій та рішень методів візуалізації об’єктів як хмари точок;

– удосконалити існуючі методи візуалізації об’єктів як хмари точок у напрямку покращення швидкодії;

– розробити метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень;

– виконати експериментальну перевірку методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень.

Об’єкт роботи – процес візуалізації об’єктів як хмари точок з використанням інформаційних технологій.

Предмет роботи – моделі, підходи та засоби візуалізації об’єктів як хмари точок.

Методи дослідження: Для розв’язання поставлених задач використовуються основні положення методів аналізу даних, геометричного моделювання, комп’ютерної графіки, алгоритми і методи візуалізації; для реалізації програмної складової методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень – методології розробки програмного забезпечення, об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані наступні результати:

– удосконалено існуючий метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень, що дозволило підвищити точність та швидкодію, обчислювальну здатність та доступність.

Практичне значення одержаних результатів. В результаті виконання кваліфікаційної роботи магістра розроблено відповідне експериментальне програмне забезпечення, яке підтвердило вірність запропонованих положень. Застосування методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень дає можливість здійснювати швидкий перегляд тривимірних об’єктів із використанням малопотужних технічних засобів за наявності універсального, вбудованого програмного забезпечення.

Апробація результатів кваліфікаційної роботи магістра та публікації. Основні наукові та практичні результати опубліковані в наукових виданнях МОН України:

– публікація на тему «Information technology of making controlled critically safe decisions when viewing point clouds “web point cloud viewer”» в науковому журналі “Вісник Хмельницького національного університету”.

– Доповідь на тему «Переглядач хмар точок» на XII міжнародній науково-практичній інтернет-конференції «Сучасний рух науки».

За темою кваліфікаційної роботи магістра автором виконано одну наукову публікацію [27] та доповідь [28].

Структура та обсяг роботи. Кваліфікаційна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 36 найменувань та 4 додатків. Загальний обсяг кваліфікаційної роботи магістра становить 118 сторінки, з них 79 сторінок основного тексту та 39 сторінок додатків. У роботі наведено 47 рисунків.

Ключові слова: хмара точок, системи візуалізації, лазерний сканер, фотограмметрія, 3D системи координат, Autodesk, Autodesk ReCap

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
JSON	JavaScript Object Notation
HTML	HyperText Markup Language
ASP	Active Server Pages
САПР	Система автоматизованого проектування і розрахунку
IT	Інформаційні технології
FOV	Field of view
BIM	Building information modeling

Вступ

Актуальність роботи. В сучасних технологіях інформація про об’єкти, зокрема їх поверхні, у вигляді хмари точок широко застосовуються у різноманітних сферах діяльності людини - як результат дистанційного зонування Землі, в системах автоматизованого проектування, при тривимірній цифровій реконструкції будівель, до прикладу пам’яток архітектури, та приміщень, тощо.

Найбільш ефективною технологією отримання необхідної хмари точок є лазерне сканування. Це обумовлене стрімким розвитком лазерної техніки та інформаційно-комп’ютерних технологій що супроводжується покращенням технічних характеристик, з одного боку, та зниженням вартості – з іншого. Суттєве та стрімке покращення техніко-економічних показників систем тривимірного лазерного сканування обумовлює їх розширену доступність за запровадження у найрізноманітніших галузях.

Хмара точок, отримана в результаті лазерного сканування, є первинним результатом, що потребує якісного опрацювання. Подальша робота полягає у вирішенні двох головних й, іноді, взаємовиключних задач – підвищення точності відтворення поверхонь, що вирішується шляхом збільшенням точок у хмарі, та забезпечення необхідної швидкодії, з діаметрально протилежним варіантом розв’язання.

Розробка та запровадження належних ефективних інформаційних технологій з використанням відповідних сучасних засобів розробки дозволяють реалізувати системи, які одночасно забезпечують високу точність сканування у поєднанні з прийнятною швидкодією, тобто, дозволяють оперувати великими хмарами точок з прийнятними термінами виконання.

Хмара точок – це цифрове 3D-зображення фізичного об’єкта чи простору [1]. Він складається з мільйонів окремих точок вимірювання, кожна з координатами x, y і z.

Залежно від методу, який використовується для захоплення хмари – і датчиків – кожна точка може також містити дані про колір RGB або навіть інформацію про інтенсивність, яка відображає силу зворотного впливу лазерного імпульсу, який генерував точку.

Для зйомки хмари точок можна використовувати два основних інструменти: лазерні сканери та фотограмметрію.

Лазерний сканер – це система оглядового рівня, яка включає в себе ряд різних датчиків та технологій. Найважливішим є датчик лідара, який використовує швидкі лазерні імпульси для збору сотень тисяч надзвичайно точних вимірювань за секунду. Більшість лазерних сканерів також включають камеру RGB для додавання кольору до хмари точок та одиницю вимірювання інерції (IMU) [2].

Фотограмметрія – це скоріше методологія, ніж конкретний тип інструменту. Щоб створити хмару точок за допомогою фотограмметрії, потрібно було б камерами захоплювати простір з усіх ракурсів, а потім обробляти ці зображення за допомогою спеціалізованого програмного забезпечення для реконструкції простору в 3D [3].

Дослідження підтвердили, що лазерні сканери дають більш точні дані, ніж фотограмметрія.

Мета роботи. Мета роботи полягає у реалізації методу «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень з достатнім рівнем точності та швидкодії, обчислювальної здатності та доступності.

Завдання полягає у розв’язанні часткових поставлених задач:

– провести аналіз існуючих методів, технологій та рішень методів візуалізації об’єктів як хмари точок;

– удосконалити існуючі методи візуалізації об’єктів як хмари точок у напрямку покращення швидкодії;

– розробити метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень;

– виконати експериментальну перевірку методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень.

Об’єкт роботи – процес візуалізації об’єктів як хмари точок з використанням інформаційних технологій.

Предмет роботи – моделі, підходи та засоби візуалізації об’єктів як хмари точок.

Методи дослідження: Для розв’язання поставлених задач використовуються основні положення методів аналізу даних, геометричного моделювання, комп’ютерної графіки, алгоритми і методи візуалізації; для реалізації програмної складової методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень – методології розробки програмного забезпечення, об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані наступні результати:

– удосконалено існуючий метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень, що дозволило підвищити точність та швидкодію, обчислювальну здатність та доступність.

Практичне значення одержаних результатів. В результаті виконання кваліфікаційної роботи магістра розроблено відповідне експериментальне програмне забезпечення, яке підтвердило вірність запропонованих положень. Web Point Cloud Viewer є власністю AMC Bridge. Застосування методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень дає можливість здійснювати швидкий

перегляд тривимірних об’єктів із використанням малопотужних технічних засобів за наявності універсального, вбудованого програмного забезпечення.

Апробація результатів кваліфікаційна роботи магістра та публікації. Основні наукові та практичні результати опубліковані в наукових виданнях МОН України:

– публікація на тему «Information technology of making controlled critically safe decisions when viewing point clouds “web point cloud viewer”» в науковому журналі “Вісник Хмельницького національного університету”.

– Доповідь на тему «Переглядач хмар точок» на XII міжнародній науково-практичній інтернет-конференції «Сучасний рух науки».

За темою кваліфікаційної роботи магістра автором виконано одну наукову публікацію [27] та доповідь [28].

Структура та обсяг роботи. Кваліфікаційна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 36 найменувань та 4 додатків. Загальний обсяг кваліфікаційної роботи магістра становить 118 сторінок, з них 79 сторінок основного тексту та 39 сторінок додатків. У роботі наведено 47 рисунків.

Ключові слова: хмара точок, системи візуалізації, лазерний сканер, фотограмметрія, 3D системи координат, Autodesk, Autodesk ReCap

Розділ 1
Характеристика предметної області і постановка задачі

1.1 Аналіз предметної області

У сучасному світі хмари точок використовуються досить широко через те, що ці набори даних пропонують точне та всеосяжне цифрове зображення реального простору, поверхні чи об’єкта. А це означає, що вони пропонують величезну цінність для широкого кола застосувань.

Розглянемо найбільш поширені сфери застосування для хмар точок у сучасному виробництві.

Хмари точок забезпечують можливість швидко створити план поверху для споруди [4]. Цей метод є більш швидким і точним, ніж робота з інструментами зйомки, такими як тахеометри, або ручними інструментами, такими як лазерні дисто.

Для реалізації даного підходу є два методи. Хмара точок може бути створена вручну шляхом сканування приміщення зверху вниз. Після чого буде можливо простежити план поверху вручну за допомогою САПР або програмного забезпечення для планування поверхів. Або може бути використаним програмне забезпечення на базі штучного інтелекту для автоматичного створення плану поверху.

Хмари точок використовуються для створення будівельної інформаційної моделі (BIM). BIM являє собою цифрове відтворення функціональних та фізичних характеристик об’єкта [5]. BIM – це цілісний процес створення інформації та управління інформацією про вбудований об’єкт. На основі інтелектуальної моделі та підтримки хмарної платформи, BIM інтегрує структуровані, багатопрофільні дані для створення цифрового зображення

об’єкту протягом усього його життєвого циклу, від планування та проектування до будівництва та експлуатації.

Оскільки хмара точок точно та всебічно відображає умови будівлі, це означає, що вона пропонує всі необроблені просторові дані, необхідні для створення нової інформаційної моделі будівлі або оновлення існуючої моделі. Це дає змогу порівнювати існуючі умови будівлі з «задуманими» для перевірки на наявність помилок, конфліктів та зіткнень.

Щоб створити BIM з хмари точок, необхідно використати спеціалізоване програмне забезпечення для моделювання BIM, щоб створити модель поверх даних хмари точок.

Деякі програми дозволяють це робити вручну, малюючи елементи та кодуючи таку інформацію, як матеріал точок. Інше програмне забезпечення може частково автоматизувати процес, розпізнаючи поверхні, об’єкти і навіть MEP, а також використовуючи штучний інтелект для створення моделі. Ці напівавтоматизовані процеси можуть бути дуже точними, але вони не є досконалими, оскільки програмне забезпечення працює набагато швидше, ніж люди, але може пропустити елементи або повернути хибні результати. В результаті завжди доведеться перевіряти роботу та виконувати деякі ручні виправлення.

Якщо ви регулярно збираєте хмари точок під час будівництва, ці дані можуть бути використані, щоб відстежувати, які роботи завершені, коли вони завершені та де.

Існують ручні методи відстеження ходу будівництва за допомогою хмари точок та вдосконалені автоматизовані методи. Необробленимі дані з хмарних точок можуть бути поширені із замовниками, щоб надати їм доступ до швидкого, високорівневого огляду прогресу будівлі в будь-який момент з початку будівництва. Або можнавикористовувати автоматичні інструменти для надання більш детальних оновлень, наприклад програмне забезпечення, яке автоматично аналізує хмару точок для розпізнавання конкретних об’єктів і

навіть забезпечує відстеження роботи відповідно до конкретних угод, розкладів та бюджетів.

Ви можете використовувати хмару точок для створення операційної 3D-моделі об’єкта. Ця модель, більш відома як цифровий близнюк, має симбіотичні стосунки з фізичним об’єктом, що забезпечується різноманітними датчиками Інтернету речей (IoT). В ідеальному цифровому близнюку будь-які зміни у фізичному об’єкті автоматично переносяться на цифрового близнюка і навпаки.

Цей симбіотичний зв’язок означає, що цифрові близнюки можуть використовуватися як моделі BIM, для перегляду інформації про об’єкт або для виконання складних завдань, таких як моніторинг, моделювання, аналітика та контроль.

Створення цифрового близнюка є складним багатошаровим процесом. Саме тому багато компаній пропонують послуги для їх створення. Вони захоплюють об’єкт, оцифровують його та розробляють цифрового близнюка, який відповідає необхідним специфікаціям і включає дані, які потрібні «в реальному часі» для моделі (геометричні дані, дані про температуру тощо).

Також під час аналізу предметної області, потрібно звернути увагу на ReCap [6]. Це продукт компанії Autodesk. Дана програма надає можливість користувачас працювати з хмарами точкоч, отриманими за допомогою лазерного сканування.

Лазерне сканування – це популярний метод геодезії, який дозволяє точно вимірювати та збирати дані з об’єктів, поверхонь, будівель та ландшафтів. Лазерні сканери збирають інформацію у вигляді даних хмарних точок, які складаються з мільйонів 3D-координат (координати XYZ)[7].

3D-лазерне сканування включає в себе набагато більш прогресивні технології, ніж це було в 1960-х роках. Сучасні процедури лазерного сканування використовують лазерні промені, вдосконалені датчики, глобальні системи позиціонування (GPS)[8], інерційні одиниці вимірювання (IMU)[9], електроніку

приймача та фотоприймачі. Використовуючи всі ці компоненти, лазерні сканери можуть обчислити точні координати поверхонь і структур.

Системи лазерного сканування викидають світлові хвилі, які відбиваються від поверхонь і повертаються назад до датчика. Потім датчик обчислює, на якій відстані знаходиться поверхня, вимірюючи час, необхідний для того, щоб світловий промінь завершив свою подорож. Цей процес відомий як вимірювання "часу польоту". Виміряну відстань потім використовують для обчислення координати крихітної ділянки поверхні, на яку потрапляє лазерний промінь. Все це відбувається всього за секунди, і під час одного сканування лазерний сканер збирає мільйони 3D-координат.

Коли хмари точок, отримані за допомогою лазерного сканування, оброблені, то вони формують цифрове зображення сканованих поверхонь, демонструючи розміри та просторові відносини топографічних об’єктів та структур.

ReCap об'єднує дані лазерного сканування та фотограмметрію в одне сімейство продуктів для вирішення та спрощення всього робочого процесу. У той час як традиційні хмари точок виглядають як крапки, технологія ReCap тепер може візуалізувати справді масивні хмари точок як реалістичні поверхні. Унікальним для ReCap є те, що користувачі можуть взаємодіяти з цими величезними наборами даних, виконуючи CAD-подібні операції, такі як вибір, позначення тегами, переміщення, вимірювання, виявлення зіткнення та вилучення об'єктів, все з власними точками. Лазерне сканування та фотограмметрія історично дуже затратні та потребують великої кількості даних.

Отже, проаналізувавши предметну область, можна дійти до конструктивного висновку, що обробка моделей, які зберігаються у вигляді хмар точок, є актуальним напрямком сучасних ІТ, оскільки це дозволяє зменшити час обробки геоданих та підвищити зручність їх візуалізації.

1.2 Аналіз інформаційного забезпечення предметної області

1.2.1 Аналіз існуючого програмного забезпечення предметної області

У сучасних інформаційних технологіях є досить багато десктопних додатків, що надають можливість роботи із хмарами точок, насамперед, це Autodesk Recap.

Програма ReCap проста у використанні. Для початку роботи необхідно вибрати файл точки для імпорту, і він буде доданий до нового проекту ReCap. Структура проекту дозволяє розбити сканування на керовані фрагменти і працювати лише з потрібними даними в будь -який момент часу. Наприклад, якщо у користувача було повне сканування міського блоку, він міг би розбити дані на певні дні сканування даних або навіть за типами об’єктів, наприклад, будівлі в одному наборі та дерева в іншому.

Після того як користувач обрав файли для імпорту у проект, у нього є можливість застосувати до даних фільтри. Фільтри дозволяють встановлювати зовнішні межі для даних, тому, якщо потрібно ввести лише певну область сканування, можна просто вибрати межу, яка закінчується близько до неї, і все поза рамками не імпортується. ReCap також дозволяє застосувати "фільтри шуму", які надають можливість усувати бродячі знімки, які могли бути виявлені під час сканування.

Після того, як дані будуть у ReCap, користувач можете розпочати вибір того, що він хоче очистити, переглянути, змінити тощо, використовуючи прості інструменти виділення, такі як вікно, виділення на основі кольорів і навіть площинне виділення. Останній інструмент корисний, особливо коли користувач працює з такими спорудами, як будівлі та дороги. Просто натиснувши значок «Вибір площини», а потім вибравши кілька точок на екрані, програмне забезпечення вибере всі точки на цій площині і відфільтрує всі інші, забезпечити роботу лише з необхідними даними.

ReCap надає змогу візуалізувати та редагувати великі набори даних. На робочому столі користувачі ReCap можуть переглядати та редагувати мільярди точок, щоб підготувати їх до використання у продуктах портфоліо Autodesk, щоб забезпечити реалістичну роботу з контекстного проектування

ReCap розкриває можливості всюдисущих камер для зйомки високоякісних 3D-моделей, забезпечуючи захоплення реальності в межах досяжності будь-кого, хто має камеру. ReCap підтримує об'єкти будь-якого розміру та діапазону, повну роздільну здатність для сіток високої щільності, точок огляду та багаторазового експорту файлів.

ReCap включає найкраще з фото та лазерного збору даних, тому клієнти можуть використовувати фотографії, щоб заповнити отвори або збільшити дані лазерного сканування. Користувачі можуть як збільшити точність фотозйомки за допомогою лазерних точок, так і додати фотореалістичні деталі до лазерного сканування.

Основним недоліком Autodesk Recap є необхідність його встановлення на особистий комп’ютер, а також потреба в покупці ліцензії на використання.

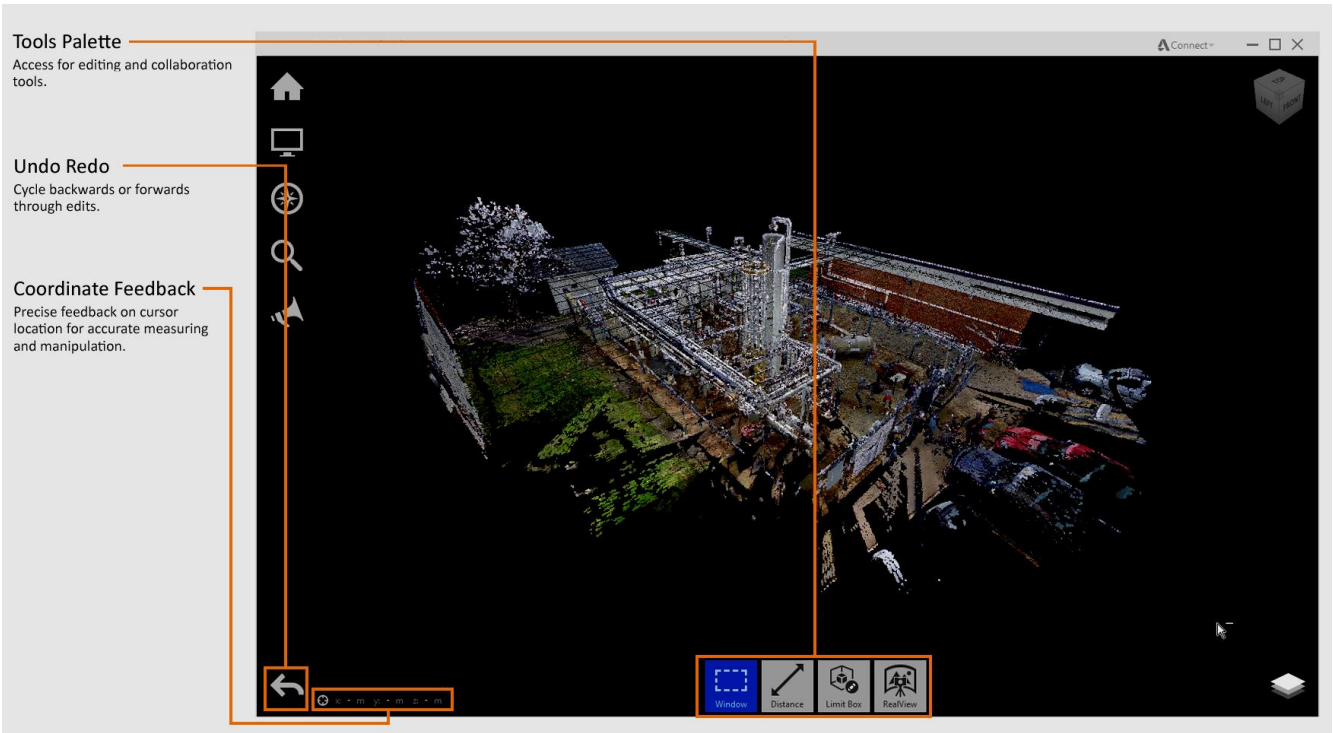


Рисунок 1.1 – Огляд головного меню ReCap

Аналізуючи предметну область необхідно звернути увагу на веб-застосунки. Достатньо відомим продуктом є розробка Інститута комп'ютерної графіки та алгоритмів TU Wien «Potree» [10]. Potree має відкритий код на надає змогу користувачам відображати хмари точок (рисунок 1.2). Також він безкоштовний.

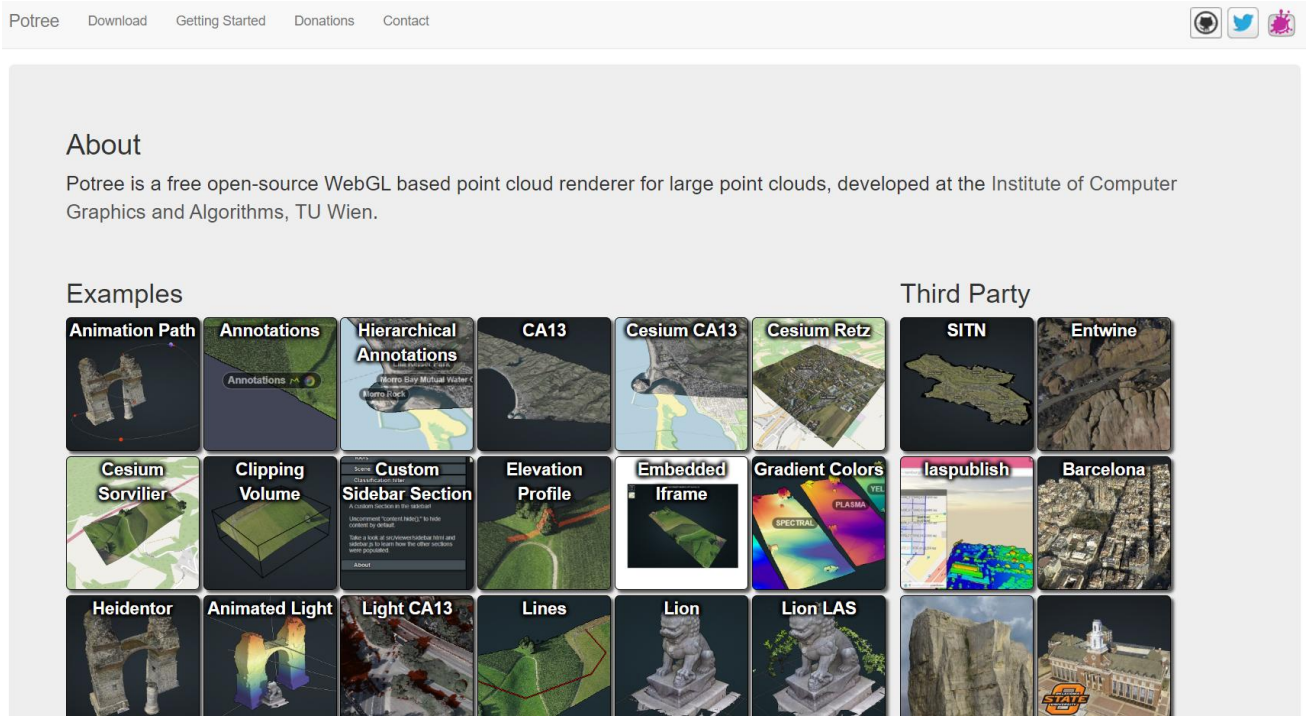


Рисунок 1.2 – Інформація про сайт «Potree»

Згаданий вище веб-застосунок має наступний набір функцій для вимірювання хмар точок:

– Angle measurement – знаходження кута між точками;

– Point measurement – пошук положення точки;

– Height measurement – вимір висоти фрагмента хмари точок;

– Circle measurement – пошук радіуса;

– Area measurement – вимір площі вибраної області;

– Volume measurement – знаходження об'єму вибраної області та інші (рисунок 1.3).

Відкритий код Potree є його основною перевагою, тобто користувач може його переглянути і при необхідності модифікувати для власних потреб.

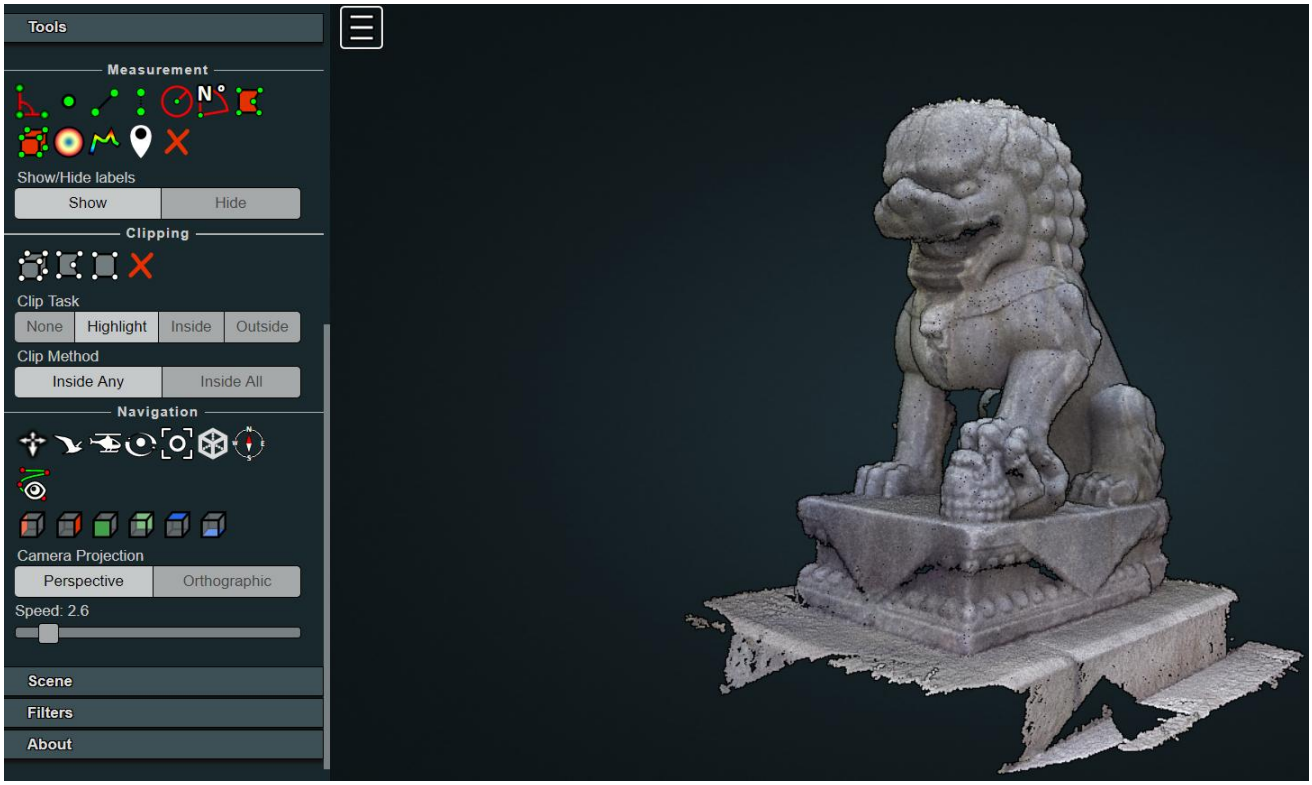


Рисунок 1.3 – Інструменти веб-застосунку «Potree»

Розглянемо ще один сайт який надає можливість користувачу переглядати хмари. Це сайт Online LIDAR point cloud viewer [11] (рисунок 1.4). Даний сайт дозволяє користувачу завантажувати і переглядати файли двох форматів LAS 1.2 та XYZ. Також він містить наступний перелік додаткових функцій:

– зчитування і обробка хедеру файлу;

– можливість зміни кольору точок хмари, у відповідності до вибраного режиму;

– налаштування додаткових параметрів відображення.

Суттєвим недоліком сайту Online LIDAR point cloud viewer є підтримка лише двох форматів даних XYZ та LAS 1.2, в яких зберігається хмара точок, а також відсутність інструментів для вимірювання хмари точок.

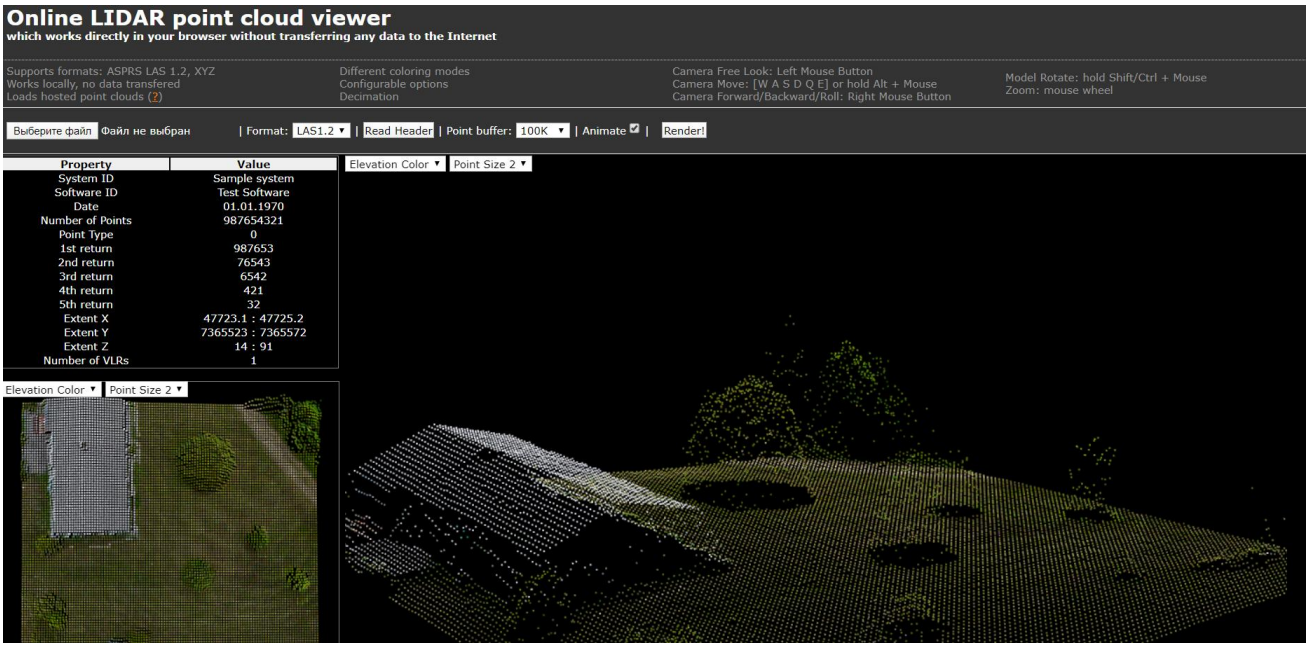


Рисунок 1.4 – Головна сторінка Online LIDAR point cloud viewer

Отже, проаналізувавши описані вище програмні застосунки, можна скласти таблицю підтримуваних форматів файлів.

Таблиця 1.1 – Підримуванні формати файлів

Формат/застосунок	Recap	Potree	Online LIDAR	Web Viewer
CL3 (Topcon)	+			
CLR (Topcon)	+			
E57	+			+
FLS (Faro)	+			
FWS (Faro)	+			
LSPROJ (Faro)	+			
LAS	+	+	+	+
PCG	+			
PRJ (Leica)	+			
PTG (Leica)	+			
PTS	+			+
PTX	+			+
RCS	+			

RDS (3D only; Riegl)	+			
TXT	+			+
XYB	+			
XYZ	+			+
ZFS (Zoller+Fröhlich)	+			
ZFPRJ (Zoller+Fröhlich)	+			
LAZ		+	+	+
EPT		+		
PCD				+

1.2.2 Аналіз існуючих форматів даних для збереження хмар точок

У сучасних інформаційних технологіях досить багато форматів файлів у яких зберігаються хмари точок. Найбільш поширені формати PCD, LAS, LAZ, PTS, PTX, XYZ, TXT та E57.

Файли формату PCD призначені для збереження 3D даних хмар точок. Даний формат даних виділяється гнучкістю і швидкістю в порівнянні з іншими форматами даних для збереження хмар точок. Одною з переваг PCD є можливість зберігання та обробки організованих наборів даних хмари точок. Версія PCD вказана номером в хедері на початку кожного файлу, наприклад 0.5, 0.6. Основною відмінністю між версіями є те, що у версії 0.6 додалось нове поле VIEWPOINT. У ньому вказана інформація про орієнтацію сенсора до набору даних. Файли PCD містять два розділи:

– кожен файл PCD містить заголовок, який ідентифікує та оголошує кількість, розмір, розмірність та тип даних хмари точок, що зберігаються у файлі. Заголовок PCD повинен кодуватися в ASCII [12].

– розділ даних, який може бути у форматі ASCII або у двійковому форматі, що не читається людиною.

PCD підтримує лише числові компоненти. Дані з файлу PCD зчитуються або записуються в геометрію хмари точок на функції FME. Ці файли використовують розширення .pcd [13].

Формати файлів PCD можуть мати різні номери версій відповідно до випуску Point Cloud Library (PCL) версії 1.0. Типова структура та вміст файлу хмари точок збереженої у форматі PCD зображено на рисунку 1.5.

```
# .PCD v.7 - Point Cloud Data file format
VERSION .7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 213
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 213
DATA ascii
0.93773 0.33763 0 4.2108e+06
0.90805 0.35641 0 4.2108e+06
0.81915 0.32 0 4.2108e+06
0.97192 0.278 0 4.2108e+06
```

Рисунок 1.5 – Структура файлу у форматі PCD

LAS - це формат файлу для обміну 3-мірними даними хмари точок [14]. Незважаючи на те, що цей формат розроблений в основному для обміну даними хмарних точок лідара, цей формат підтримує обмін будь-якими тривимірними даними-туплетами x, y, z. Lidar розшифровується як “Light Detection And Range” за аналогією з радаром, Radio Detection and Range. Lidar використовує ультрафіолетове, видиме або ближнє інфрачервоне світло для зображення об'єктів та приладів, встановлених на літаках та супутниках, використовує lidar для зйомки та картографування.

Формат LAS використовує чотири типи записів. Усі значення даних є двійковими у форматі little-endian. Версія 1.4 базується на 64-бітній структурі файлів. Типи записів перераховані тут у порядку, знайденому у файлі LAS:

– обов'язковий блок одного заголовка містить підпис файлу (LASF), основні метадані, що ідентифікують проект, кількість та тип записів даних точок, розміри включеного діапазону x, y, z та вказівники на основні розділи файлу;

– необов’язкові записи змінної довжини (VLR) містять різні типи даних, включаючи інформацію про проекцію, метадані, інформацію про пакети сигналів та дані користувацьких програм. VLR обмежуються корисним навантаженням даних 65 535 байт;

– точкові записи даних утворюють первинний вміст файлу. LAS 1.4 визначає типи від 0 до 10 для типів точкових даних. Усі точкові записи у файлі повинні бути одного типу;

– необов'язкові записи розширеної змінної довжини (EVLR) забезпечують більший корисний набір даних, ніж VLR, і їх можна зручно додати до кінця файлу LAS.

Тип файлу LAZ в першу чергу асоціюється з LASzip від Laszip. LAZ - це розширення, що використовується форматом даних для виявлення та вимірювання стисненого світла (lidar). LAZ розшифровується як LASzip Compressed Lidar Data. Lidar - це технологія дистанційного зондування. Він був розроблений з метою вимірювання відстані шляхом освітлення цілі лазером. Потім відбите світло аналізували, щоб отримати значення відстані. Файли LAZ часто використовуються для передачі величезних обсягів даних lidar [15].

Файл PTS - це простий текстовий файл, що використовується для зберігання точкових даних, як правило, зі сканерів LIDAR. Перший рядок зазначає кількість точок, за якими слід слідувати. Кожен наступний рядок має 7 значень, перші три - це (x, y, z) координати точки, четвертий - значення "інтенсивності", а останні три - оцінки кольору (r, g, b). Значення (r, g, b)

знаходяться в діапазоні від 0 до 255 (одиночний непідписаний байт). Значення інтенсивності - це оцінка частки падаючого випромінювання, відбитого поверхнею в цій точці, 0 вказує на дуже погану віддачу, тоді як 255 - дуже сильну віддачу [16]. Типова структура та вміст файлу хмари точок збереженої у форматі PTS зображено на рисунку 1.6.

```
30571
-0.037829 0.12794 0.004474
-0.044779 0.128887 0.001904
-0.068009 0.151244 0.037195
-0.002287 0.13015 0.02322
-0.022605 0.126675 0.007155
-0.025107 0.125921 0.006242
-0.03712 0.127449 0.001795
0.033213 0.112692 0.027686
-0.025508 0.112568 0.036676
-0.02453 0.112636 0.037346
0.027403 0.12156 0.02122
-0.062896 0.158419 -0.017587
```

Рисунок 1.6 – Структура файлу у форматі PTS

PTX - це формат, заснований на ASCII, для збереження даних хмарних точок, як правило, зі сканерів LIDAR. Концептуально інформація про кожну 3D-точку зберігається як 4 або 7 значень (залежно від того, зберігається інформація про колір чи ні). Координати для кожної точки зберігаються неперетвореними у власній системі координат, а матриця перетворення надається як заголовок файлу. PTX не можна використовувати на невпорядкованих або уніфікованих хмарах. Усі точки, навіть ті, що знаходяться в тіні, де координати не розраховані, зберігаються [17].

Заголовок файлу PTX містить 10 рядків. Рядки 3-6 заголовка - це положення точки початку і первинних осей (X, Y, Z) системи координат сканера. Рядки 7-10 є матрицею перетворення, де рядки 7-9 містять матрицю обертання, а лінія 10 вектор перекладу. Значення рядків 3-10 мають подвійну точність.

Кожен рядок після заголовка PTX містить інформацію про одну виміряну точку. Якщо хмара точок не включає інформацію про колір, лінія має 4 значення - (X, Y, Z, інтенсивність). Координати X, Y, Z зазвичай знаходяться в метрах. Інтенсивність знаходиться в десятковому діапазоні [0, 1]. Хмари точок, що містять інформацію про колір, мають 3 додаткові числа на рядок, де вказані значення R, G, B у діапазоні [0, 255]. Іноді колір можна зберегти як плаваючі в діапазоні [0, 1].

Файл даних хмари точок у форматі XYZ повинен містити рядки даних. Для прямокутної системи координат кожен рядок повинен складатися з X, Y та Z координат, за якими слід визначати значення поля в цій точці. Також після визначих позицій точок йде опис кольорів хмари точок у форматі R, G, B, де R – червоний, G – зелений та B – синій. Значення в кожному рядку можна розділити будь-якою комбінацією пробілів, табуляції або коми; кожен пробіл, вкладка або кома вважається окремим роздільником поля [18]. Типова структура та вміст файлу хмари точок збереженої у форматі XYZ зображено на рисунку 1.7.

```
1.282654 -11.316550 10.334572 85 76 76
1.233657 -11.240221 11.369599 173 186 200
0.786608 -11.029999 11.926729 193 206 217
1.183794 -10.997295 11.877870 193 206 217
1.983021 -10.511738 12.538944 196 210 220
1.679299 -16.814801 13.463854 224 236 245
2.092387 -16.281422 13.563091 209 222 232
2.351805 -15.808789 15.046903 224 236 245
3.129568 -14.516328 14.906748 184 199 212
2.889839 -12.874211 15.562304 183 197 208
2.815117 -11.450948 15.261078 183 198 210
1.784401 -15.738208 17.145962 224 236 245
2.269567 -14.357418 16.406036 224 236 245
1.983380 -12.920394 17.279327 196 210 221
2.237862 -11.482546 16.519947 184 199 212
```

Рисунок 1.7 – Структура файлу у форматі XYZ

У файлах з розширенням TXT зберігаються різні типи даних. Тому хмари точок, які збережені у форматі TXT не мають чіткої специфікації для формату файлу. Відповідно до цього вимоги до хмар точок збережених у форматі TXT та структура файлу припускаються схожими до формату XYZ. А саме кожен рядок файлу повинен містити опис точок у форматі X, Y, Z. Якщо хмара точок представлена з кольорами, то їх опис повинен іти після опису координат точок, у форматі R, G, B.

На високому рівні структура файлу E57 є ієрархічним деревом структура. Формат ієрархії базується на XML формат даних. Однак було б неефективно представляти надзвичайно великі масиви даних, пов'язані з лазерними сканерами в чистому вигляді формату XML. Розміри файлів були б неприпустимо великими, а введення / виведення даних було б болісно повільним. Тому на низькому рівні фактичниі точкові дані представляються за допомогою стиснутого двійкового формату. Інший великі блоки даних, такі як зображення, також ефективно представлені в двійковий. Таким чином, формат підтримує гнучкість та розширюваність використання текстового XML, одночасно забезпечуючи ефективне введення / виведення та використання сховища стиснуті потоки двійкових даних.

Однією з найбільших переваг формату E57 є те, що він був розроблений для того, щоб його можна було легко розширювати для забезпечення підтримки нових апаратних інновацій. Використовуючи даний механізм розширення E57, розробники апаратного забезпечення можуть диференціювати свої продукти та додавати нові функції, розробляючи розширення, які дозволяють розробникам програмного забезпечення отримати доступ до додаткових технологічних інновацій. Ці розширення надають розробникам програмних систем E57 нові функції, що виходять за межі зазначених у стандарті E2807. Розширення E57 підтримують формат з останніми інноваціями та технологіями.

Файл E57 розділений на три частини: заголовок, набір необов’язкових двійкових розділів та розділ XML. Заголовок - це невелика, 48-байтова двійкова

структура, що містить критичний рівень файлу таку інформацію, як номер версії та розташування XML розділ (рисунок 1.8).

```
struct E57FileHeader {
    char        fileSignature[8];
    uint32_t    majorVersion;
    uint32_t    minorVersion;
    uint64_t    filePhysicalLength;
    uint64_t    xmlPhysicalOffset;
    uint64_t    xmlLogicalLength;
    uint64_t    pageSize;
}
```

Рисунок 1.8 – Структура заголовку файлу у форматі E57

Розділ XML містить ієрархічну деревоподібну структуру описати вище. Якщо файл містить точкові дані або зображення, вони на частини ієрархії посилаються на розділ XML, а фактичні дані зберігаються у двійкових розділах для кожного набору точок або зображення [19].

1.3 Висновки до розділу 1 та постановка задачі

Проведений аналіз літературних джерел засвідчує суттєву та зростаючу потребу у візуалізації тривимірних об’єктів, з високим рівнем коректності відтворення таких об’єктів у поєднанні з невисокими вимогами до технічних засобів із використанням універсального та (або) вбудованого в операційну систему програмного забезпечення.

В результаті проведеного аналізу існуючих підходів сформульовані такі завдання дослідження метою якого є:

– проведення аналізу існуючих технологій, методів та рішень для відображення хмар точок;

– удосконалення існуючих методів по роботі з хмарами точок у напрямку доступності для кінцевих користувачів;

– реалізація методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень;

– виконання експериментальної перевірки методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень.

Розділ 2
Інформаційна модель для візуалізації хмар точок

2.1 Аналіз та автоматизація обробки інформаційних потоків

Метод передбачає аналіз та автоматизацію обробки інформаційних потоків. Спеціалізоване програмне забезпечення розробляється для користувачів методу, його функціональні можливості повинні задовольнити всі необхідні умови при опрацюванні даних з облікового запису. Бізнес-процеси в роботі користувача методу можна розбити на наступні групи:

1. Бізнес-процес «Використання хмарного сховища» (рисунок 2.1):

– створення нового користувача;

– логін користувача;

– логаут користувача;

– створення нових папок;

– завантаження хмар точок у сховище;

– завантаження хмар точок із сховища.

Рисунок 2.1 – Блок-схема бізнес-процесу «Використання хмарного сховища»

2. Бізнес-процес «Робота з камерою» (рисунок 2.2):

– зміна сектору огляду камери;

– зміна виду камери;

– зміна позиції камери;

– повернення камери до початкової позиції.

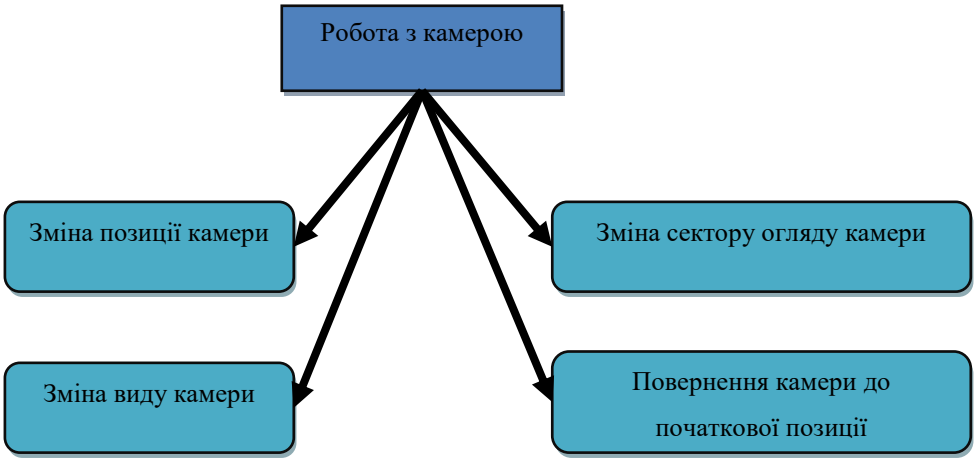


Рисунок 2.2 – Блок-схема бізнес-процесу «Робота з камерою»

3. Бізнес-процес «Робота з моделлю» (рисунок 2.3):

– модифікація розміру точок;

– зміна забарвлення точок моделі;

– повернення оригінального кольору.

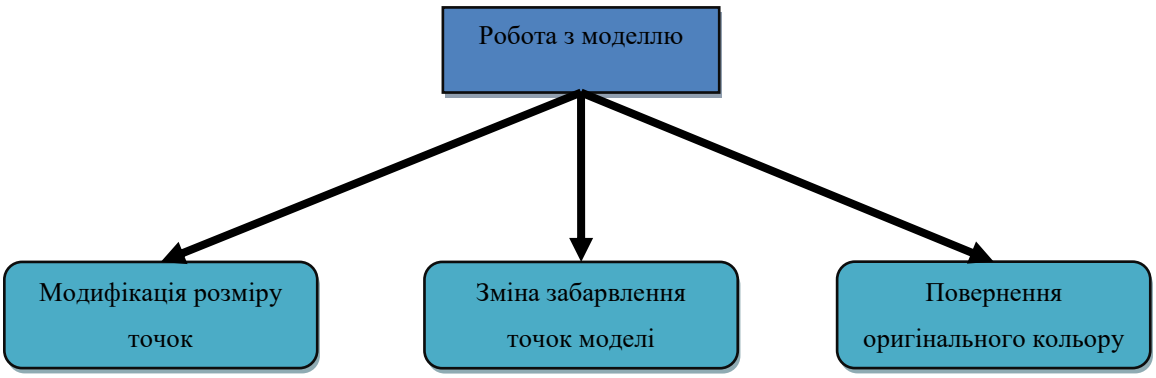


Рисунок 2.4 – Блок-схема бізнес-процесу «Робота з моделлю»

4. Бізнес-процес «Робота із сценою» (рисунок 2.4):

– показ моделі з різних сторін;

– створення знімку сцени;

– зміна кольорового фону сцени;

– підтримка списку скорочень клавіш;

– відображення глобальної системи координат моделі.

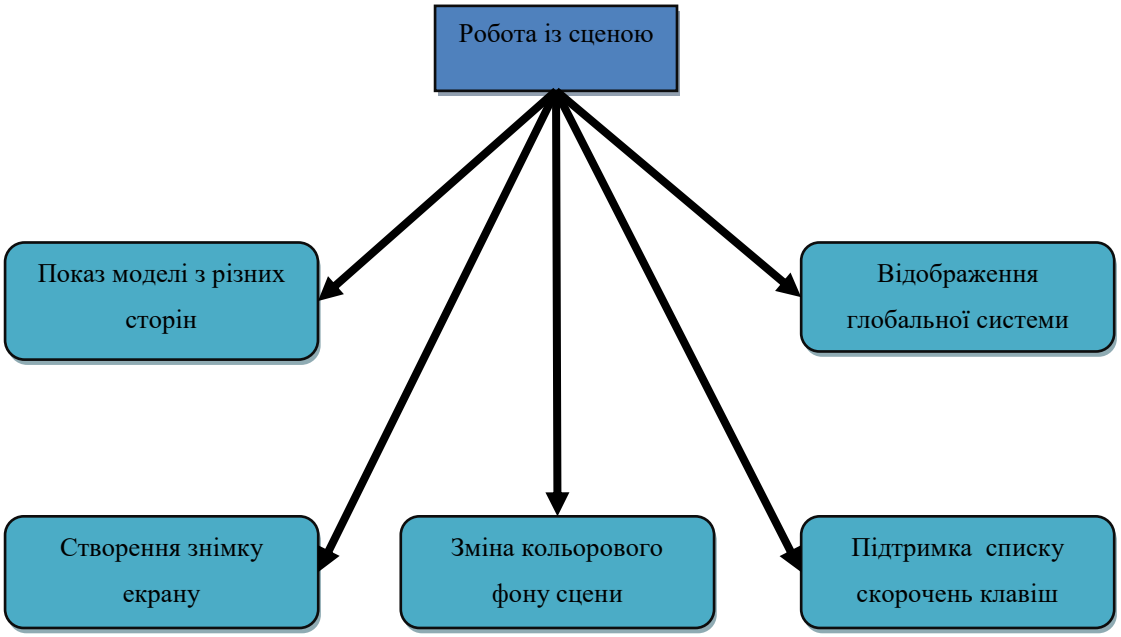


Рисунок 2.3 – Блок-схема бізнес-процесу «Робота з сценою»

5. Бізнес-процес «Робота з вимірюванням моделі» (рисунок 2.5):

– вимірювання позиції точки;

– вимірювання дистанції між двома точками;

– вимірювання дистанції між безліччю точок;

– вимірювання кута між точками;

– вимірювання площі.

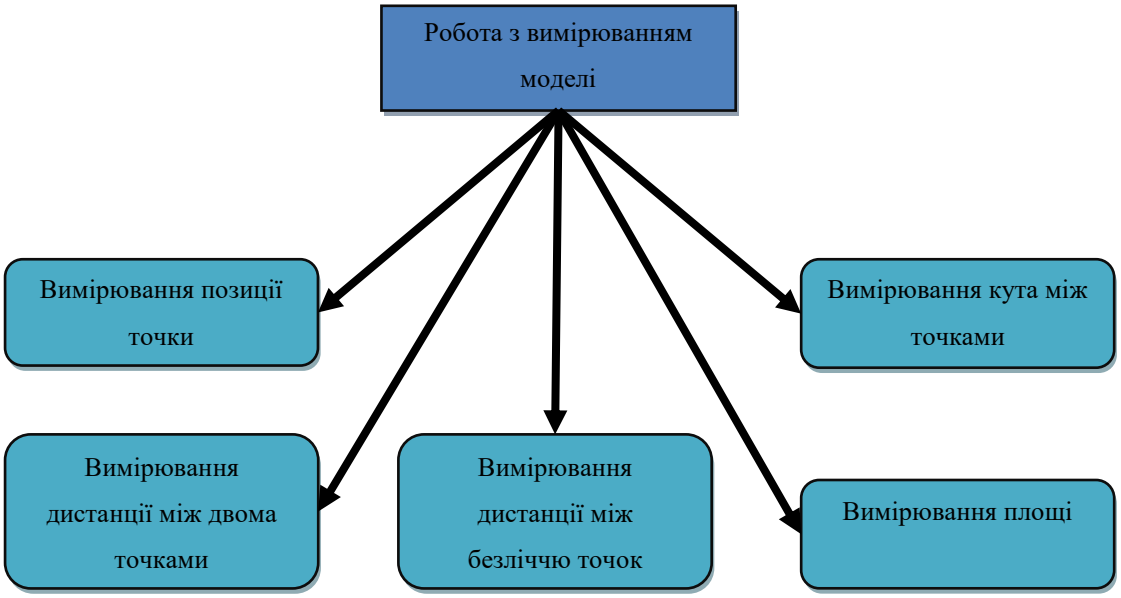


Рисунок 2.5 – Блок-схема бізнес-процесу «Робота з вимірюванням моделі»

6. Бізнес-процес «Робота з хмарами точок» (рисунок 2.6):

– розпізнавання розширення файлу;
– обробка та зчитування бінарних файлів;
– обробка та зчитування файлів у форматі ASCII;
– попередній перегляд моделі.

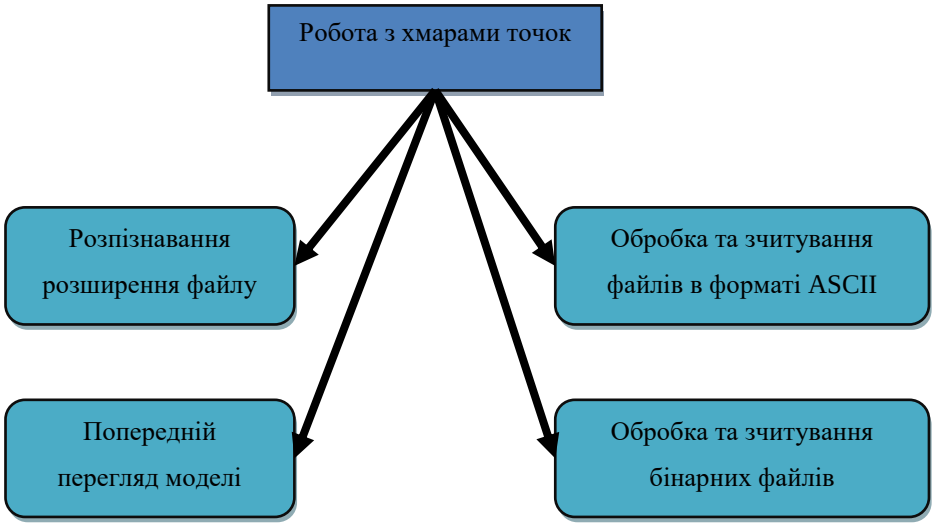


Рисунок 2.6 – Блок-схема бізнес-процесу «Робота з хмарами точок»

Бізнес-процес по роботі користувача з хмарним сховищем. Даний бізнес-процес забезпечує користувача можливістю створення облікового запису, авторизації до облікового запису, виходу із особистого облікового запису, завантажеиня хмар точок у сховище та завантажеиня хмар точок на локальний комп’ютер.

Бізнес-процес по роботі користувача з камерою. Даний бізнес-процес забезпечує користувача можливістю зміни сетору огляду камери, зміни позиції камери, зміни виду камери (перспективнаб ортогональна) та повернення камери до початкової позиції.

Бізнес-процес по роботі користувача із сценою. Даний бізнес-процес надає змогу користувачу змінювати сторону з якої відображається модель, виконувати та завантажувати знімок сцени, змінювати забарвлення заднього фону сцени, динамічно показувати направлення моделі у за допомогою глобальної системи координат. Окрім цього даний бізнес-процес надає можливість підтримки виконання команд за допомогою гарячих клавіш, а також відображає їх список.

Бізнес-процес по роботі користувача з моделлю. Даний бізнес-процес дозволяє користувачу модифікувати забарвлення та розміри точок моделі та повертати розміри та кольори точок до стану за змовчуванням.

Бізнес-процес по роботі користувача з вимірюванням моделі. Даний бізнес-процес надає змогу користувачу визначати позицію точки, вимірювати відстань та кут між точками, а також вимірювати площу.

Бізнес-процес по роботі користувача з хмарами точок. Даний бізнес-процес забезпечує користувача можливістю обробки та отримання даних з бінарних файлів, а також з файлів у форматі ASCII. Також даний бізнес-процес визначає розширення файлу і надає можливість переглянути прев’ю моделі.

Загальна схема бізнес-процесів та їх взаємозв’язків зображена на рисунку 2.7.

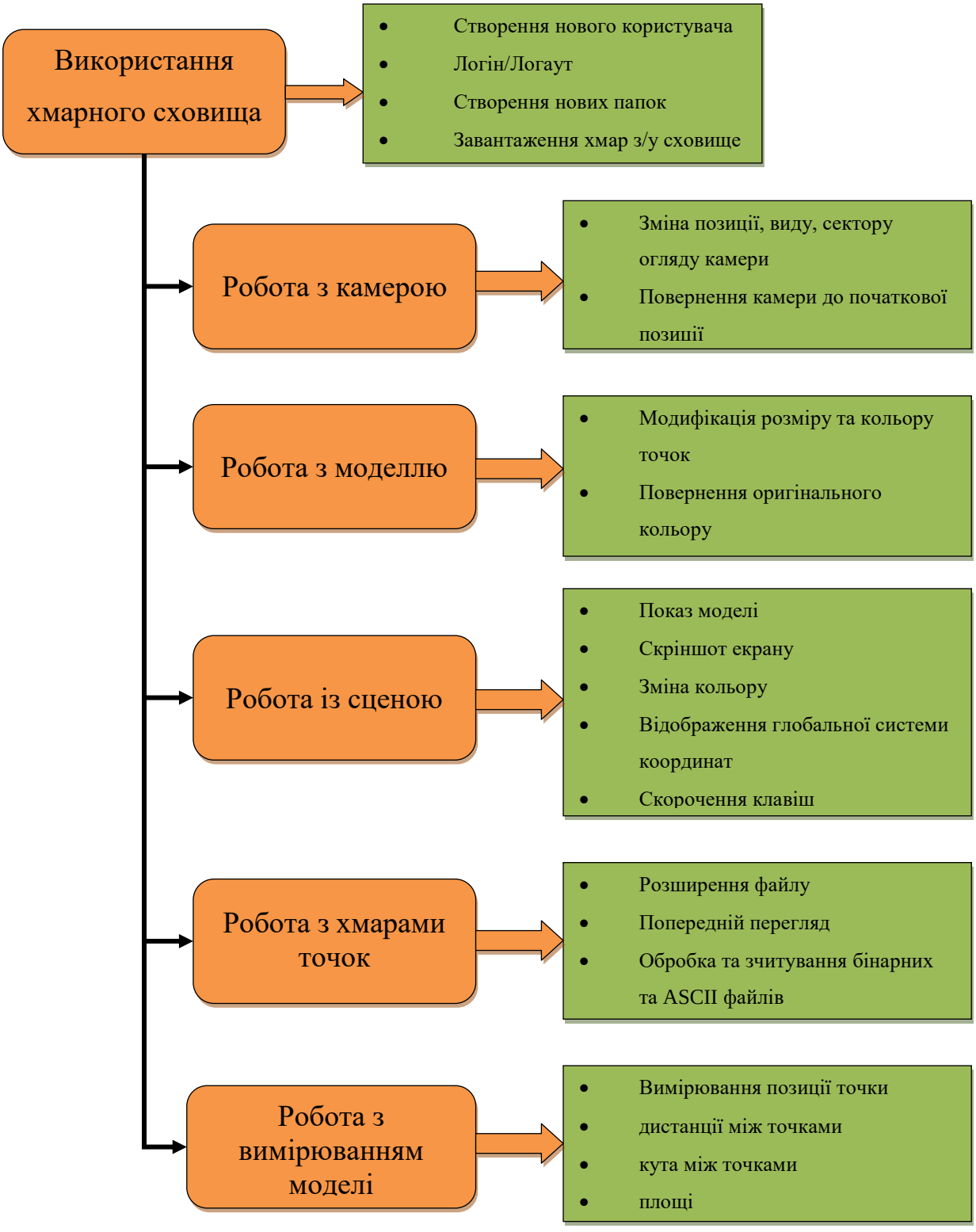


Рисунок 2.7 – Блок-схема бізнес-процесів методу візуалізації хмар точок

Для того щоб користувач мав змогу працювати з даними особистого облікового запису, він повинен виконати необхідний набір операцій для авторизації (рисунок 2.8).

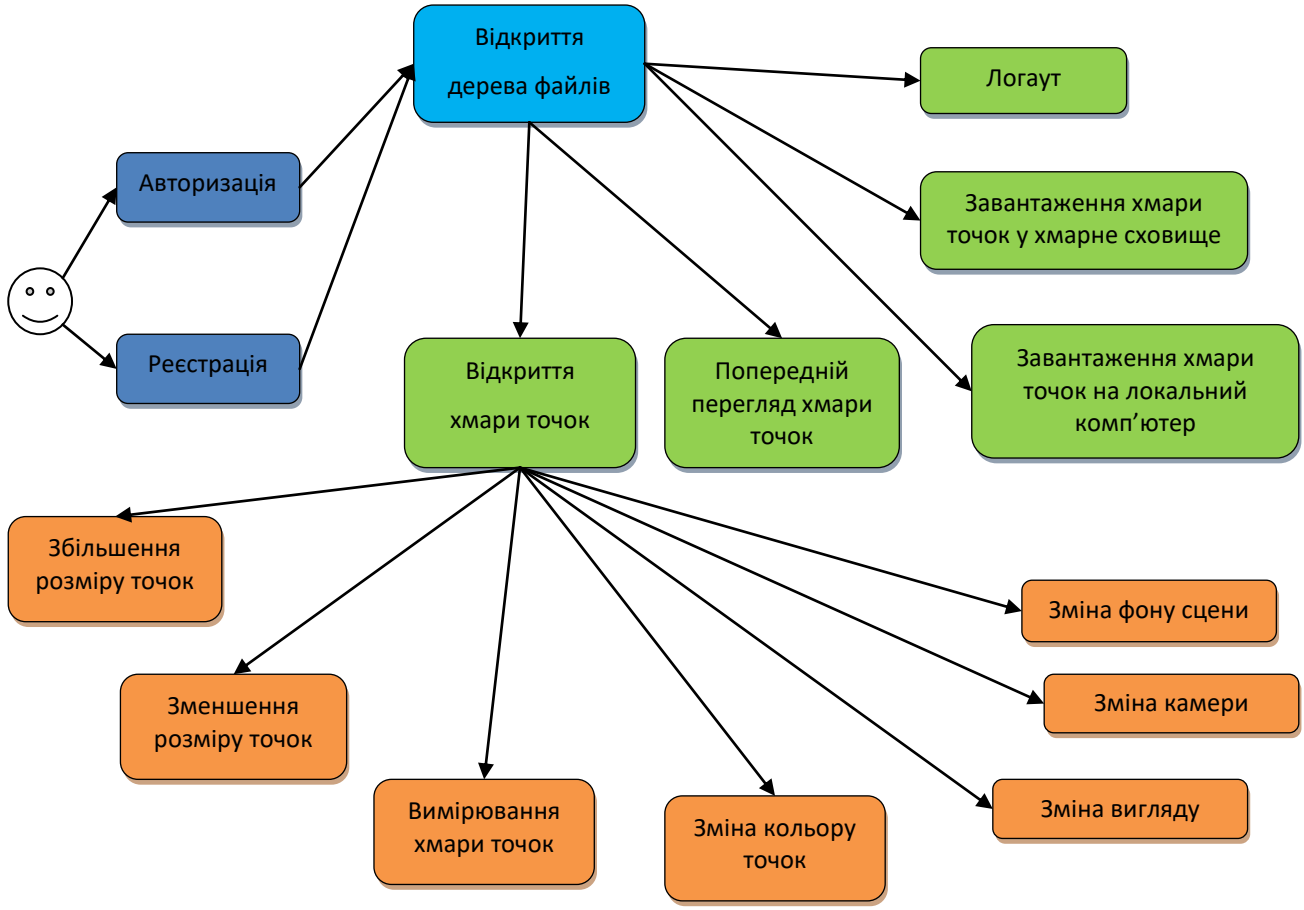


Рисунок 2.8 – Функціональна діаграма користувача

Для забезпечення коректної роботи методу візуалізації хмар точок «Web Point Cloud Viewer» та для отримання даних з хмарного сховища користувачу необхідно зайти в особистий кабінет Autodesk. У разі якщо користувач отримав помилку під час входження в обліковий запис, отже були введені невірні дані. При введені правильних логіну та паролю, програмний застосунок отримає дозвіл для обробки користувацьких даних з серверів Autodesk.

На своїй обліковій сторінці користувач має справу з наступною функціональністю: робота з хмарним сховищем (створення нового користувача,

логін користувача, логаут користувача, створення нових папок, завантаження хмар точок у сховище, завантаження хмар точок із сховища), робота з камерою (зміна сектору огляду камери, зміна виду камери, зміна позиції камери, повернення камери до початкової позиції), робота із сценою (показ моделі з різних сторін, створення знімку сцени, зміна кольору фону сцени, відображення глобальної системи координат моделі, робота з гарячими клавішами), робота з моделлю (модифікація розміру точок, забарвлення точок моделі, повернення оригінального кольору), робота з вимірюванням моделі (вимірювання позиції точки, вимірювання дистанції між двома точками, вимірювання дистанції між безліччю точок, вимірювання кута між точками, вимірювання площі) та безпосередньо робота з хмарою точок (розпізнавання розширення файлу, обробка та зчитування бінарних файлів та файлів у форматі ASCII, попередній перегляд моделі).

2.2 Розробка структури методу візуалізації хмар точок

Для користувачів методу візуалізації хмар точок «Web Point Cloud Viewer» доцільним є реалізація процедури санкціонованого доступу, відповідно до якої потрібно виконати визначену послідовність дій для реєстрації та автентифікації при безпосередній роботі з обліковим записом підключеним до хмарного сховища в особистому кабінеті Autodesk та його даними.

Зважаючи на організовану функціональну діаграму кінцевого користувача і наведені об’єктно-орієнтовані групи можливостей, можна розробити структуру методу візуалізації хмар точок «Web Point Cloud Viewer», яка буде призначена для модифікування, маніпуляції та перегляду хмар точок, а також для зберігання документів в обліковому записі Autodesk.

Для автоматизації обліку даних, необхідних для використання бізнес-процесами, має бути використано хмарне сховище. Потрібно забезпечити повну підтримку його структури, необхідну для автоматизації розроблюваного методу.

Дана модель даних має містити інформацію про файл, що зберігається у сховищі, проекти, структуру папок, тощо. Формат JSON буде використаний для обробки та передачі елементів облікового запису.

JSON представлений текстовим форматом та застосовується для обміну даними в мережі. Формат дає змогу описувати об'єкти та інші структури даних. JSON базується на тексті, що дозволяє йому бути прочитаним людиною. Цей формат використовується переважно для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією) [20].

Зважаючи на те, що важливими бізнес-процесами при використанні можливостей методу візуалізації хмар точок «Web Point Cloud Viewer» є робота з елементами хмарного сховища, то основними об’єктами для виконання автоматизації слід визначити таблиці 2.1-2.6. Структура таблиць та їх опис наведені нижче.

Таблиця «Folder Tree» (таблиця 2.1). Дані полів обмеженні типом даних, тобто поля «FolderName», «FolderId» – мають тип String. Полю «FolderTreeChild» відповідає тип даних List<FolderTree> і воно включає список підпапок. List<CloudDataFile> – тип даних для поля «Files».

Таблиця 2.1 – Таблиця «Folder Tree» («Дерево папок») та її атрибути

Атрибут	Тип даних	Опис
FolderId	String	Ідентифікатор папки
FolderName	String	Найменування папки
FolderTreeChild	List<FolderTree>	Список нащадків папки
Files	List<CloudDataFile>	Список файлів

Таблиця «Cloud Data File» (таблиця 2.2). Поля «PreviewId», «OpenInViewerLink», «DeleteLink», «Extension», «OwnerID», «ItemId», «Version», «DownloadLink», «Name», «LastModifiedTime» мають тип даних String, «HasPreview» – тип даних Boolean. Поле «StorageSize» має тип даних Int.

Таблиця 2.2 – Таблиця «Cloud Data File» («Хмарний файл даних») та її атрибути

Атрибут	Тип даних	Опис
Name	String	Найменування документа
OwnerID	String	Ідентифікатор власника документа
DeleteLink	String	Посилання для видалення
DownloadLink	String	Посилання для завантаження
OpenInViewerLink	String	Посилання для відкриття документа у переглядачі
ItemId	String	Ідентифікатор документа
Version	String	Версія документа
Extension	String	Розширення документа
PreviewId	String	Ідентифікатор попереднього перегляду документа
HasPreview	Boolean	Наявність можливості попереднього перегляду
StorageSize	Int	Розмір документа
LastModifiedTime	String	Останній час модифікування документа

Таблиця «Hub Tree» (таблиця 2.3) містить поле «Projects», якому відповідає тип даних List<ProjectTree>. Поля «HubName» та «HubId» мають тип даних String.

Таблиця 2.3 – Таблиця «Hub Tree» («Дерево хабів») та її атрибути

Атрибут	Тип даних	Опис
HubId	String	Ідентифікатор хабу
HubName	String	Найменування хабу
Projects	List<ProjectTree>	Список проектів хабу

Таблиця «Forge Models Tree View» (таблиця 2.4) містить єдине поле «Hubs», що характеризується типом даних List<HubTree>.

Таблиця 2.4 – Таблиця «Forge Models Tree View» та її атрибути

Атрибут	Тип даних	Опис
Hubs	List<HubTree>	Список хабів

Таблиця «Project Tree» (таблиця 2.5) Поле «Folders» характеризується типом даних List<FolderTree>. Поля «ProjectId», «RootFolder», «ProjectName» мають тип даних String. Поле «Files» характеризується типом даних List<CloudDataFile>.

Таблиця 2.5 – Таблиця «Project Tree» («Дерево проектів») та її атрибути

Атрибут	Тип даних	Опис
ProjectId	String	Ідентифікатор проекту
ProjectName	String	Найменування проекту
RootFolder	String	Коренева папка
Folders	List<FolderTree>	Список папок проекту
Files	List<CloudDataFile>	Список файлів проекту

Модель даних «Object response» використовується, щоб забезпечити можливість отримання даних з хмарного сховища (таблиця 2.6).

Таблиця «Point Cloud Data» (таблиця 2.6) містить поле «Name», яке має текстовий тип даних. Поля «Positions», «Colors», що мають тип даних Array.

Таблиця 2.6 – Таблиця «Point Cloud Data» та її атрибути

Атрибут	Тип даних	Опис
Positions	Array	Масив точок хмари
Colors	Array	Масив кольорів точок хмари
Name	String	Найменування хмари точок

Таблиця «Object response» (таблиця 2.7) містить поля «Location» «Objectid», «Size», «Sha1», «Objectkey», «Contenttype», «Bucketkey». Всі перечислені поля мають стрічковий тип даних.

Таблиця 2.7 – Таблиця «Object response» («Відповідь об’єкту») та її атрибути

Атрибут	Тип даних	Опис
Bucketkey	String	Ключ бакету
Objectid	String	Ідентифікатор об’єкту
Objectkey	String	Ключ об’єкту
Sha1	String	Ключ шифрування
Size	String	Розмір відповіді
Contenttype	String	Тип контенту
Location	String	Розташування

Для забезпечення взаємодії з моделями у вигляді хмар точок, було створено допоміжну модель (табл. 2.6), атрибути та опис якої наведені вище.

Отже, для автоматизації бізнес-процесів, необхідних для функціональної роботи методу візуалізації хмар точок «Web Point Cloud Viewer», було обрано формат запису та зчитування даних JSON. Розроблена структура методу дозволить забезпечити надійний взаємозв’язок між програмним забезпечення та хмарною платформою Autodesk Forge [21].

2.3 Вибір засобів розробки методу візуалізації хмар точок

Для розробки методу візуалізації хмар точок «Web Point Cloud Viewer»в рамках обраної платформи та для реалізації інтерфейсу користувача потрібно обрати фреймворки для серверної та клієнтської частин, мови програмування, програмне оточення розробки для вирішення поставленої задачі та 3D-фреймворк для відображення хмар точок.

Отже, для реалізації методу візуалізації хмар точок «Web Point Cloud Viewer» було обрано мову програмування JavaScript [22] для клієнтської частини застосунку, мову програмування C# [23] для серверної частини застосунку, фреймворки AngularJS [24] та ASP MVC 5 [25]. Для взаємодії з 3D-елементами та їх подальшим відображенням було обрано бібліотеку Three.js [26].

Висновки до розділу 2

Отже, проаналізувавши обробку інформаційних потоків, для методу візуалізації хмар точок «Web Point Cloud Viewer» було визначено повний перелік бізнес-процесів, які підлягають алгоритмам автоматизації. Розроблено структуру методу візуалізації хмар точок та сформовано функціональну діаграму користувача. Обрано технології для реалізації методу візуалізації хмар точок «Web Point Cloud Viewer», а саме мови програмування, фреймворки та бібліотеку для роботи з 3D-елементами.

Розділ 3

Проектування програмної системи методу візуалізації хмар точок

3.1 Структура і функціональне призначення модулів методу

Програмна архітектура методу візуалізації хмар точок «Web Point Cloud Viewer» випливає з основних компонентів методу. Архітектурна схема методу наведена на рисунку 3.1.

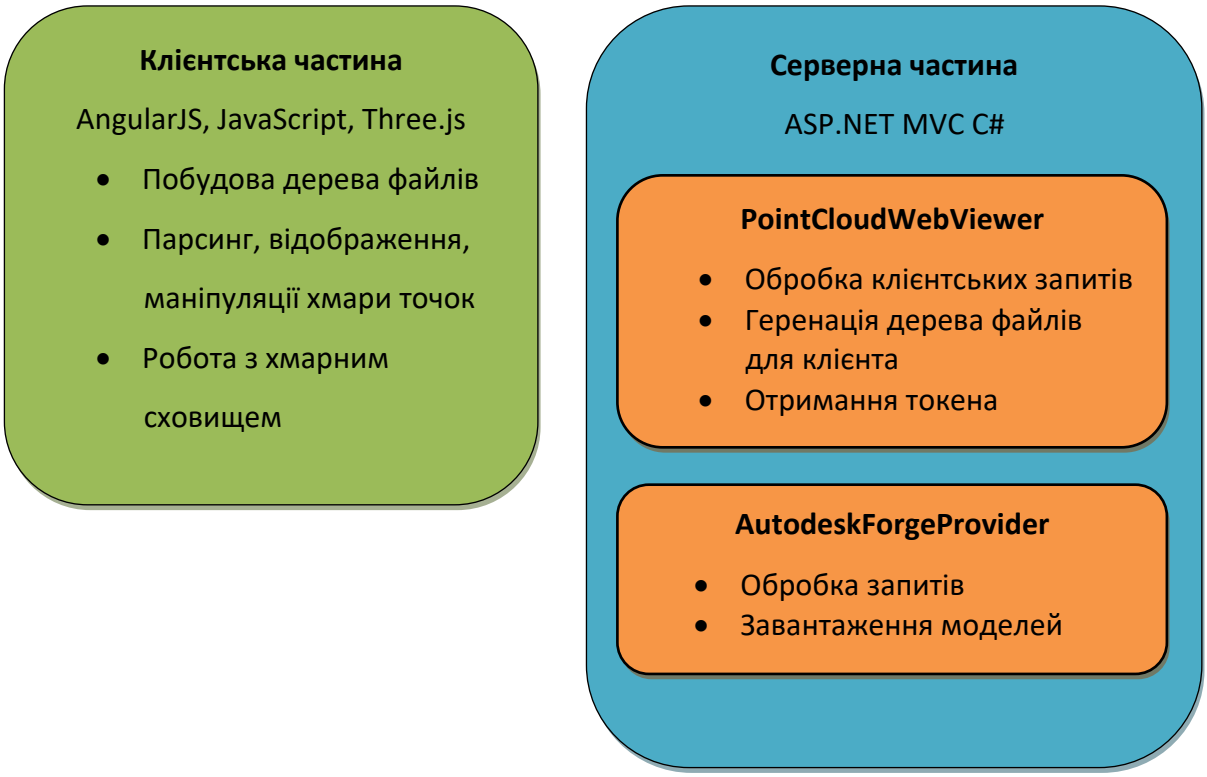


Рисунок 3.1 – Архітектурна схема методу

В рамках створення практичної реалізації було обрано шаблон Model-View-Controller (MVC), який реалізує структуру програмно-логістичних зв’язків методу візуалізації хмар точок «Web Point Cloud Viewer».Тому варто окреслити призначення кожного окремого модулю та описати їх взаємодію один з одинм на рівні функціональних взаємозв’язків.

Model (модель) описує використовувані в додатку дані, а також логіку, яка пов'язана безпосередньо з даними. Нижче наведена діаграма класів

серверного модуля моделі, що необхідна для практичної реалізації додатку (рисунок 3.2).

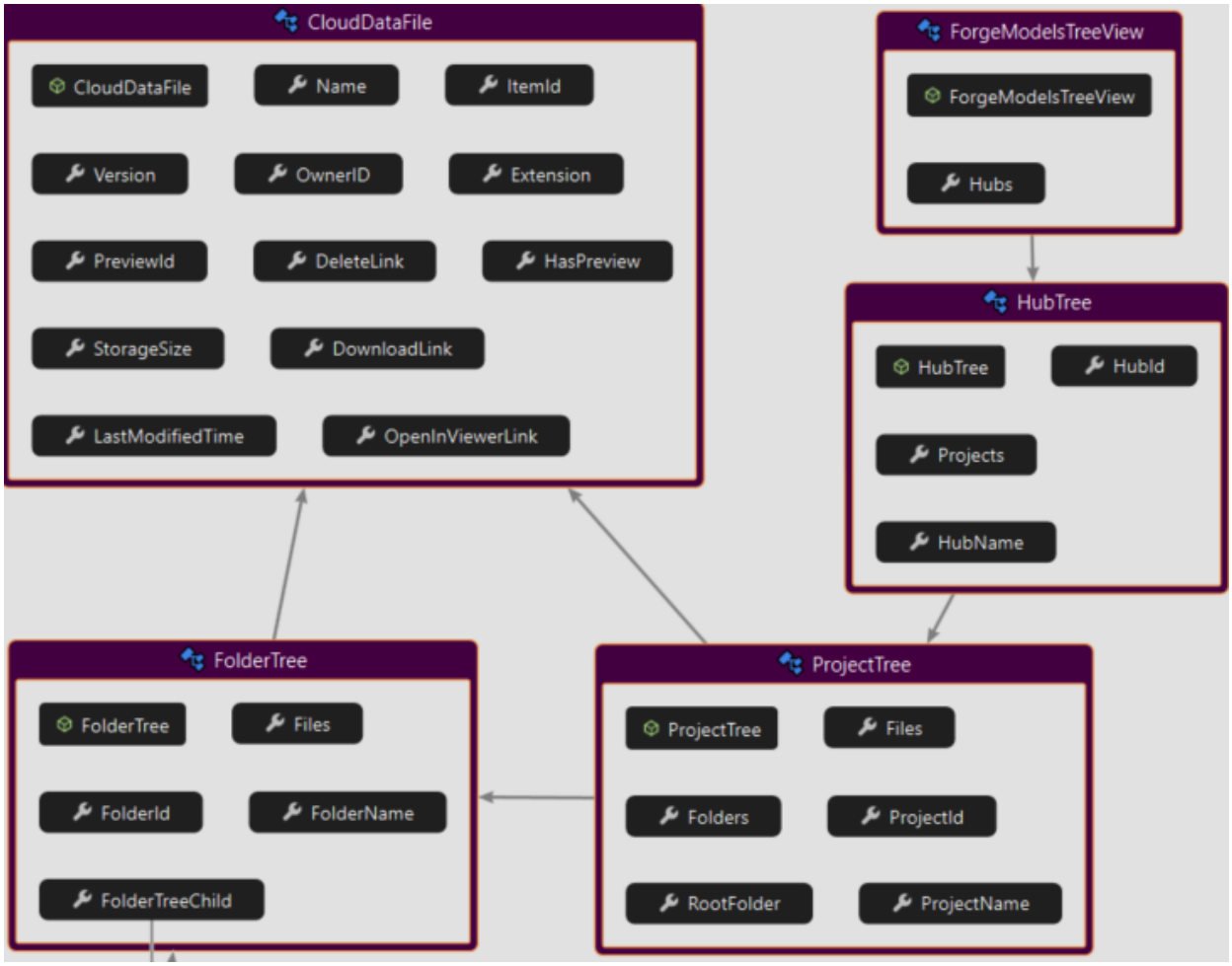


Рисунок 3.2 – Діаграма класів серверного компонента Model

Описані класи зберігають дані про документи та їх елементи. Також вони необхідні для зберігання інформації у текстовому форматі JSON. Модель використовується представленнями для відображення і передачі даних.

Структура модуля моделі клієнтської частини методу представлена на рисунку 3.3.

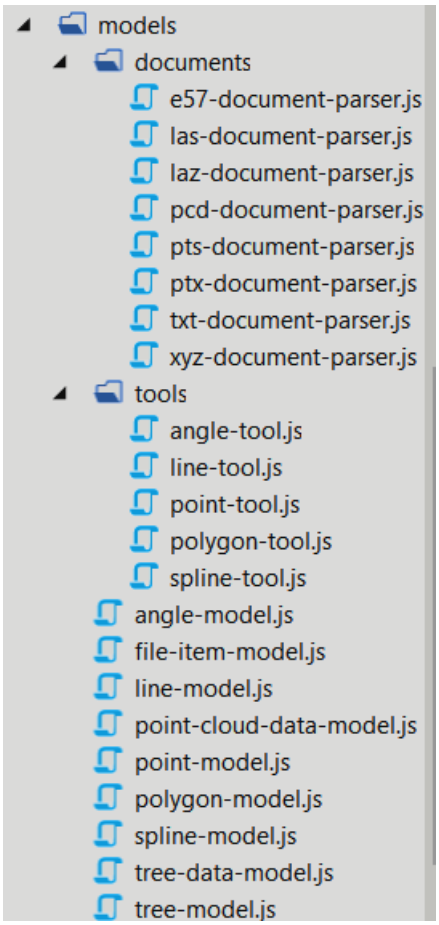


Рисунок 3.3 – Структура клієнтського компонента компонента Model

Описані класи із закінченням “document-parser” зберігають дані про аналізатори хмар точок кожного формату даних. Класи із закінченням “tool” призначені для зберігання даних пов’язаних із вимірюванням моделі, а саме позиції точок, довжину ліній, тощо.

Controller (контролер) представляє центральний компонент MVC, який забезпечує зв'язок між користувачем та програмою, представленням і сховищем даних. Він містить логіку обробки запиту користувача. Контролер отримує дані користувача і обробляє їх. Нижче наведена діаграма класів компонента Controller (рисунок 3.4).

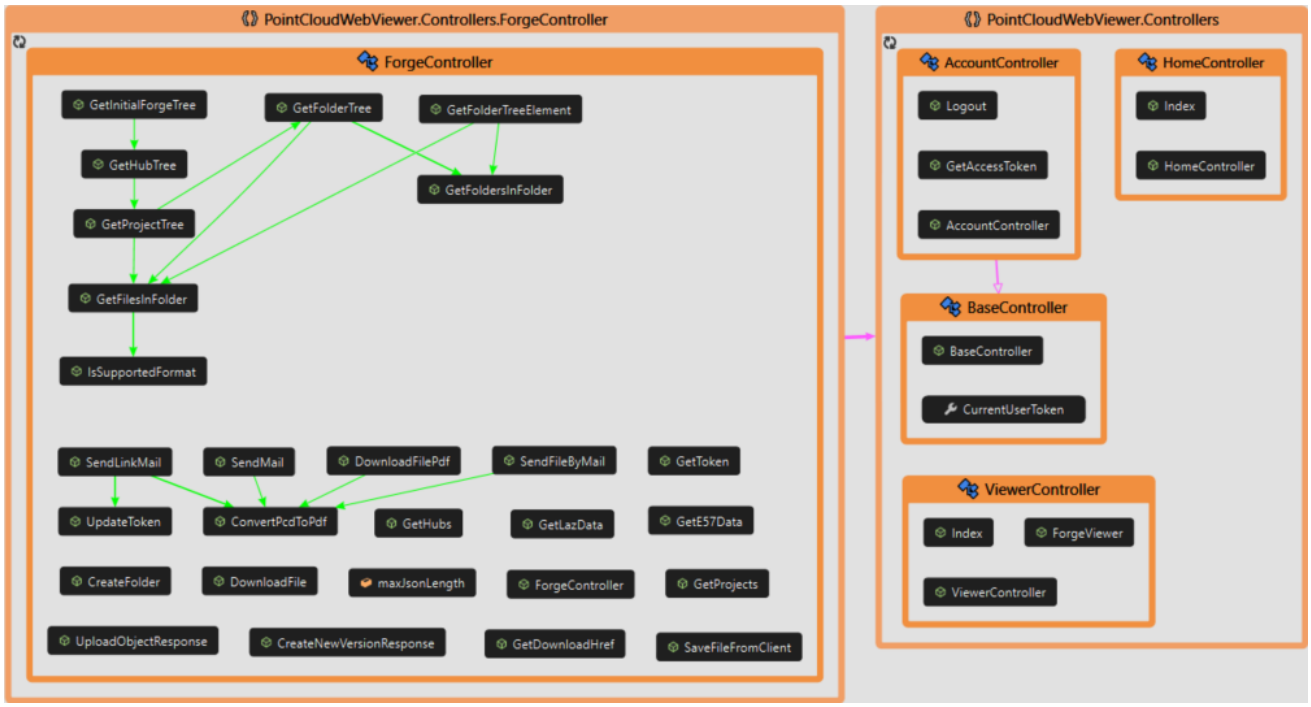


Рисунок 3.4 – Діаграма класів компонента Controller

Розглянемо класи більш детально.

AccountControler – це клас, який забезпечує можливості авторизації та отримання токена. Діаграма класу зображена на рисунку 3.5.

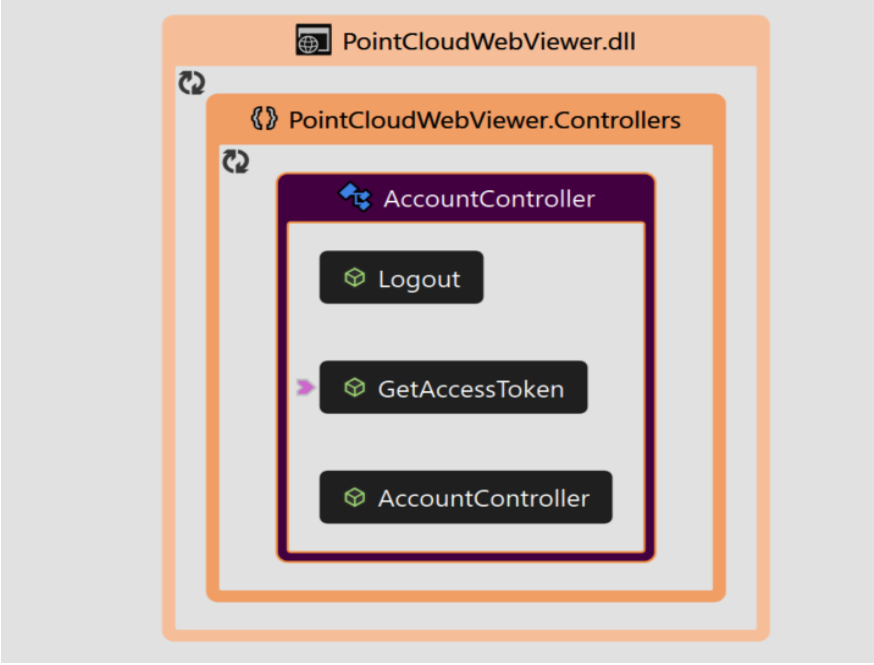


Рисунок 3.5 – Діаграма класу AccountController

ViewerController – це клас, який забезпечує можливість отримання прев’ю моделі з хмарного сховища. Його діаграма зображена на рисунку 3.6.

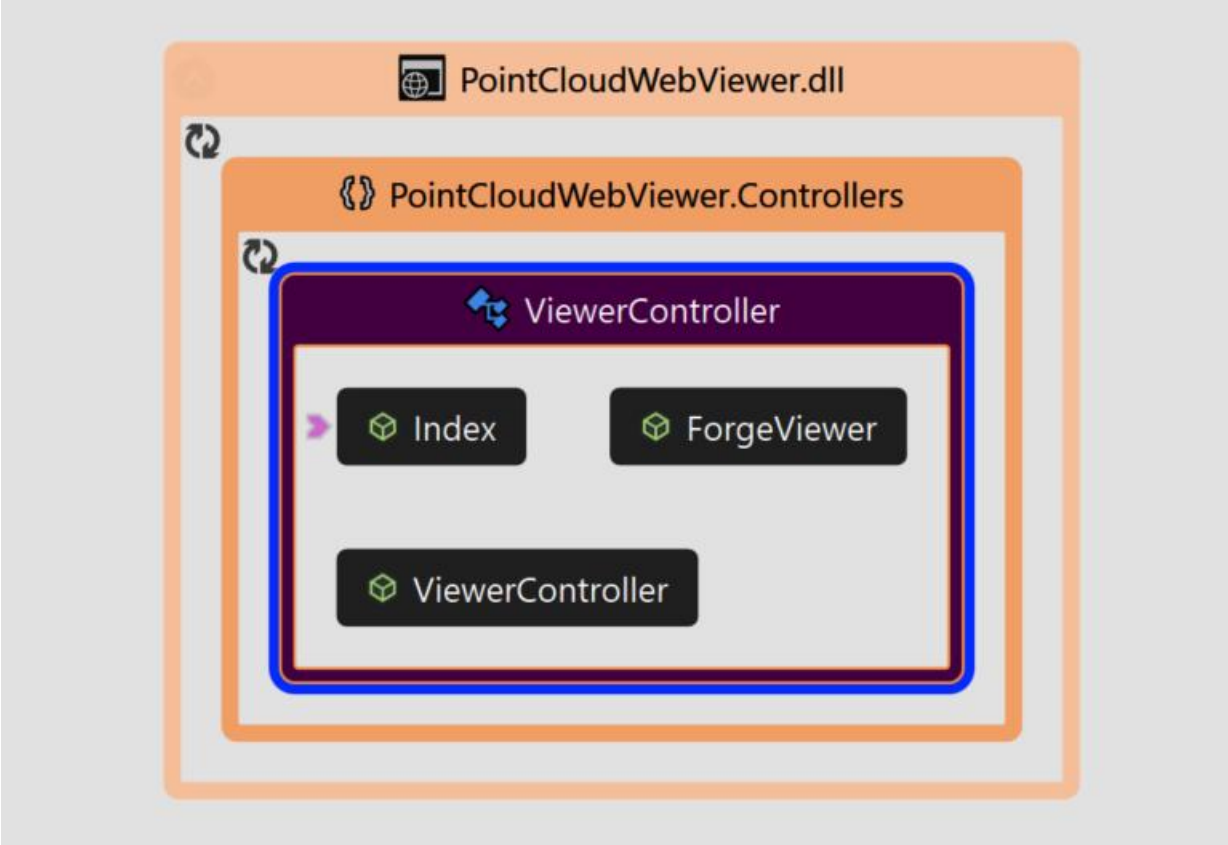


Рисунок 3.6 – Діаграма класу ViewerController

ForgeController – це клас, який відповідає за реалізацію взаємозв’язку з хмарним сховищем, а саме він забезпечує наступні функції:

– отримання дерева файлів (структуру папок і файлів поточного користувача);

– завантаження файлів у сховище;

– створення нових папок;

– отримання файлів з сховища.

Діаграма класу ForgeController наведена на рисунку 3.7.

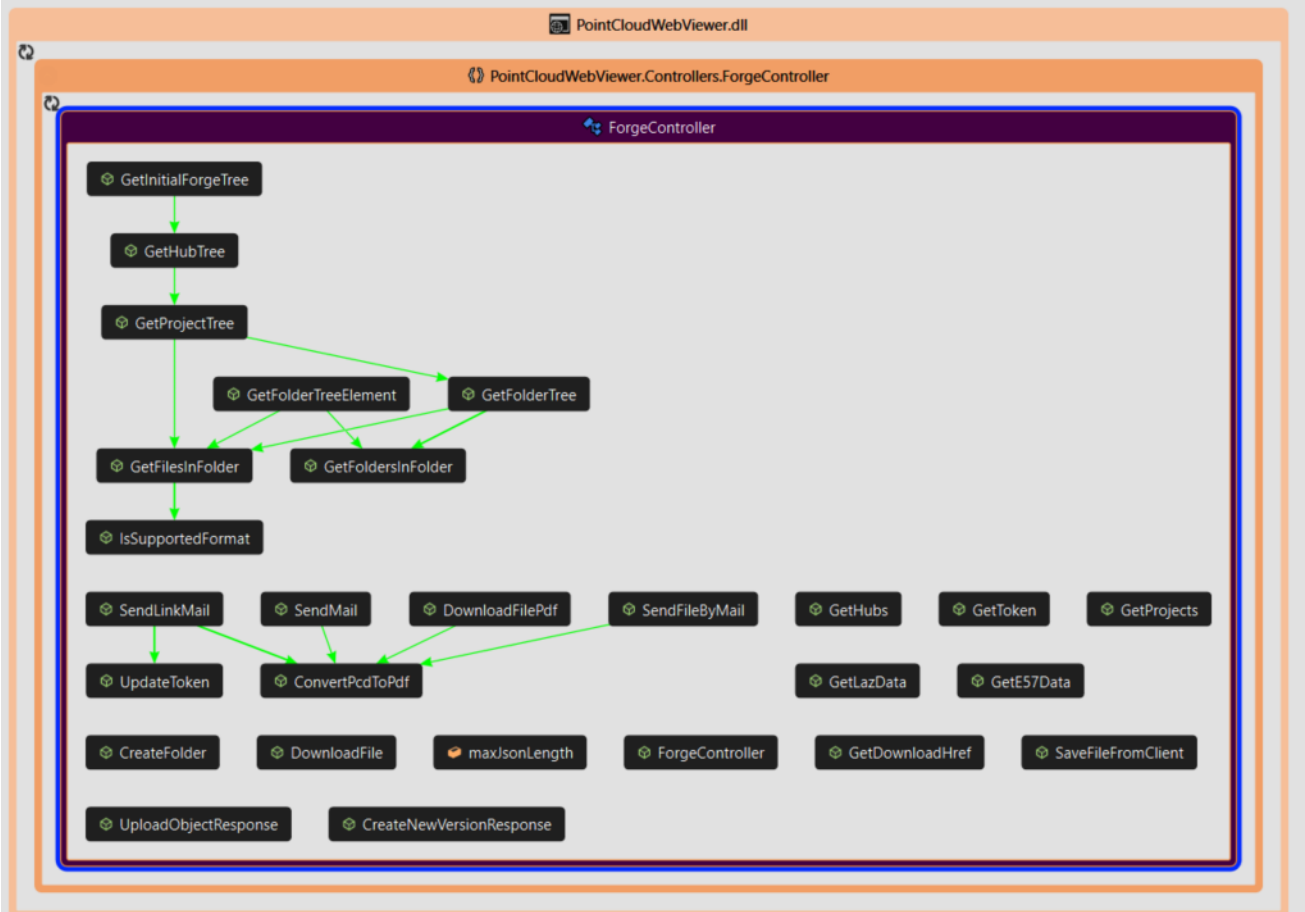


Рисунок 3.7 – Діаграма класу ForgeController

Модуль Controller клієнтської частини методу представлений чотирма класами, а саме ask-access-modal-ctrl, previewer-ctrl, tree-ctrl та viewer-ctrl.

Ask-access-modal-ctrl контролер забезпечує можливість роботи з модальним вікном та містить такі методи як onClose, onDisagreeButton, onOkButton, askAccessModal.

Previewer-ctrl контролер відповідає за забезпечення попереднього перегляду моделі. Він безпосередньо надає можливість для створення невеликої 3д сцени, де можна переглянути обрану модель хмари точок.

Tree-ctrl контролер забезпечує формування дерева папок та файлів на клієнтській стороні. Це є можливим через зв’язок даного контролера із ForgeController, що представлений на серверній частині методу.

Viewer-ctrl є одним з найважливіших класів, так як він забезпечує підтримку взаємодії із сценою, моделлю та камерою. Даний контролер надає

можливість маніпуляції з моделлю хмари точок, що включає в себе її відображення на сцені, зміну розміру та кольору точок. Також він забезпечує можливості маніпуляцій зі сценою та камерою.

Структура компонента Controller клієнтської частини методу представлена на рисунку 3.8.

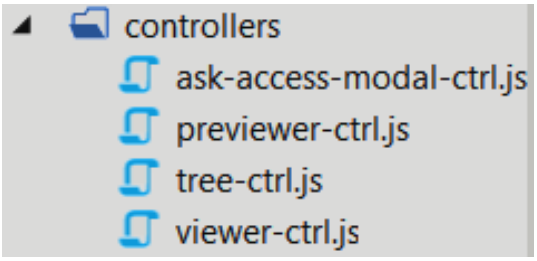


Рисунок 3.8 – Структура клієнтської частини модуля Controller

Функціональний зв’язок між хмарним сховищем та методу візуалізації хмар точок повністю забезпечений методами програмного інтерфейсу, що містяться у зазначених вище класах. Схему зв’язку методу з хмарним сховищем наведено на рисунку 3.9.

Контролер перехоплює подію ззовні і відповідно до закладеної в нього логіки, реагує на цю подію змінюючи модель, за допомогою виклику відповідного методу. Після зміни модель використовує подію про те що вона змінилася, і всі підписані на цю подію представлення, отримавши його, звертаються до моделі за оновленими даними, після чого їх і відображають.

View (представлення) відповідають за візуальну частину або користувацький інтерфейс, нерідко html-сторінка, через який користувач взаємодіє з додатком. Також представлення може містити логіку, пов'язану з відображенням даних. У той же час представлення не повинно містити логіку обробки запиту користувача або управління даними. Логічну схему представлень наведено нижче (рисунок 3.10).

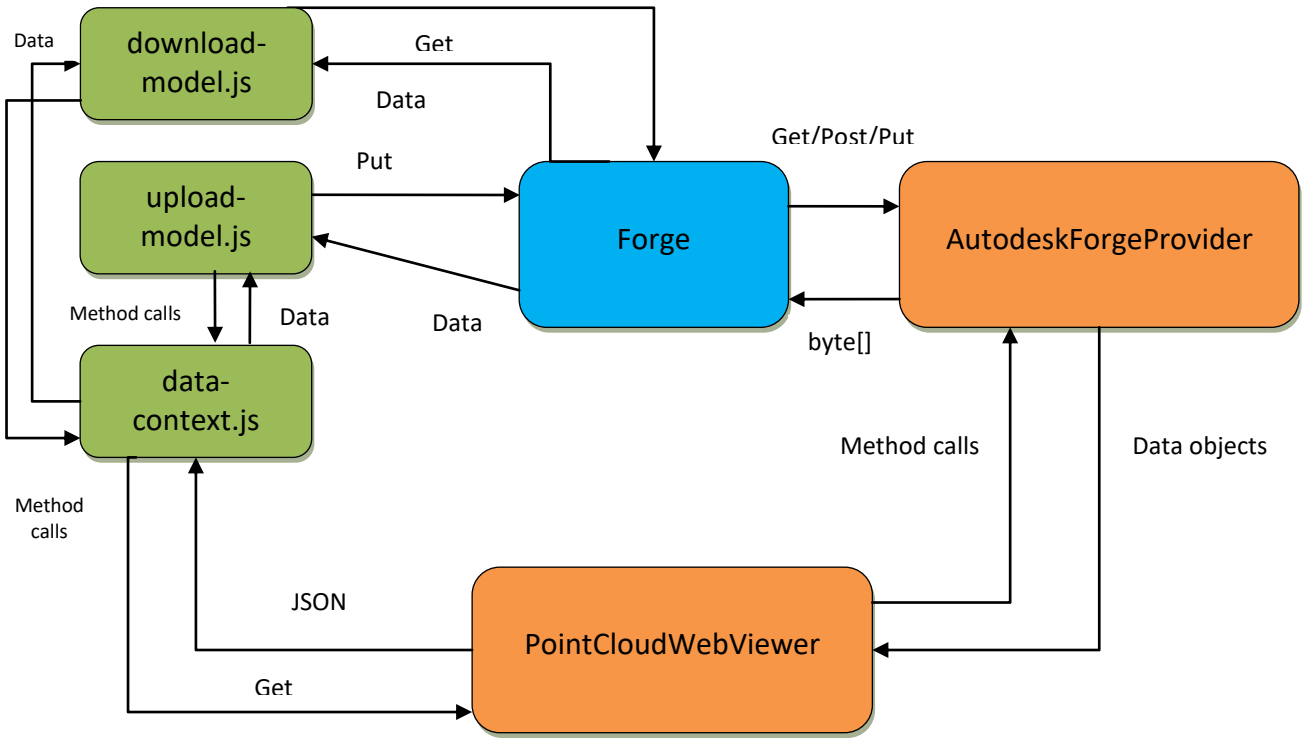


Рисунок 3.9 – Схема зв’язку методу із хмарним сховищем

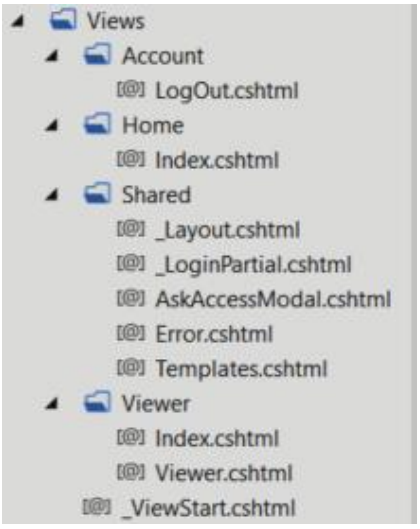


Рисунок 3.10 – Структура компонента View

За створення дизайну програмного продукту відповідає CSS3. Це спеціальний веб-інструмент, що використовується для стилізації веб-сторінок, що використовують HTML.

Для забезпечення коректної роботи методу функціональна структура передбачає, що кінцевий користувач буде зареєстрований в системі Autodesk.

Для розробки цієї структури буде використано шаблон MVC 5 та мову програмування C#. Даний метод виконується на стороні сервера IIS. Робота зазначених модулів дозволяє реалізувати обробку важливих запитів на сторону сервера Autodesk Forge та отримання відповіді в форматі JSON.

Таким чином, для формування функціональної моделі методу візуалізації хмар точок «Web Point Cloud Viewer» потрібно розробити задані програмні модулі та реалізувати роботу з клієнтськими файлами, що являють собою хмари точок.

3.2 Розробка програмних модулів

Для забезпечення валідної роботи методу було впроваджено та розроблено описані нижче програмні модулі, які забезпечують виконання всіх необхідних функцій.

Ініціалізація методу починається з етапу авторизації. Основний потік використання OAuth на платформі такий:

– додаток здійснює HTTP -запит до кінцевої точки OAuth REST і надає свої облікові дані;

– у відповідь на запит у додаток повертається токен;

– здійснюючи наступні виклики HTTP до різних API на платформі, додаток включає маркер у заголовок запиту.

Розглянемо процес авторизації детальніше. Типовий робочий процес такий:

– клієнт використовує кінцеву точку автентифікації POST для надсилання ідентифікатора клієнта та секрету разом з усіма необхідними областями;

– припускаючи успішну перевірку облікових даних, токет доступу повертається.

– метод може здійснювати виклики до будь-якої кінцевої точки для якої маркер має необхідні області, включивши заголовок запиту Authorization: Bearer <token> (де <token>- 28-знаковий маркер доступу);

– коли термін дії маркера закінчується, програмі потрібно буде отримати новий токен, повторивши всі ці кроки ще раз.

Якщо процес авторизації буде виконано успішно, то користувач отримає можливість керувати даними облікового запису на рівні перегляду файлової структури та її подальшої модифікації, тобто можливість завантаження файлів, їх видалення, створення нових папок та інше (рисунок 3.11).

За умови, що користувач не має існуючого облікового запису Autodesk, йому необхідно його створити. Щоб створити обліковий запис Autodesk користувачу потрібно виконати процедуру створення облікового запису (рисунок 3.12).

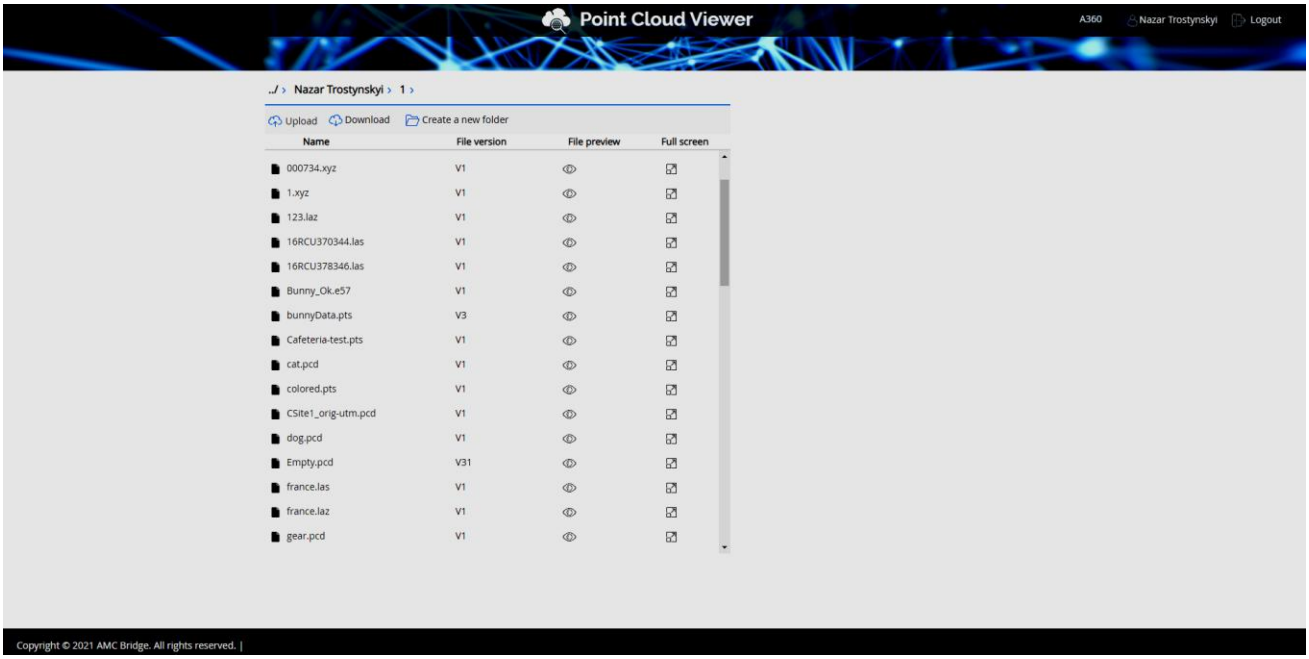


Рисунок 3.11 – Результат ініціалізації методу

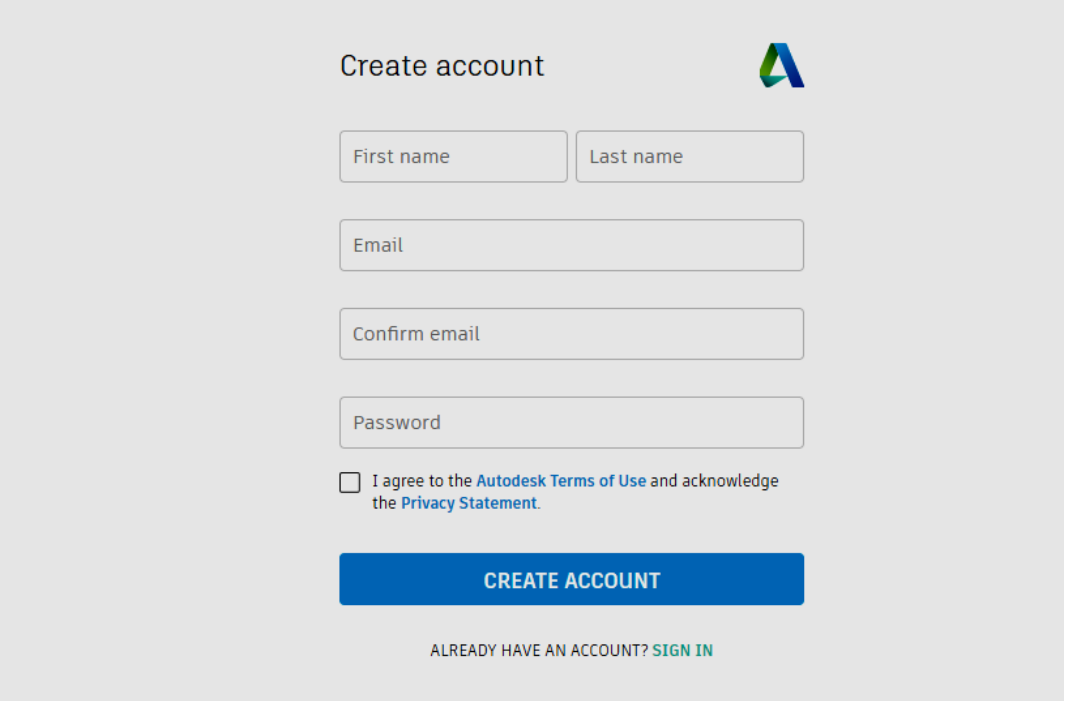


Рисунок 3.12 – Форма створення облікового запису Autodesk

Для забезпечення використання файлової системи хмарного сховища, методу візуалізації хмар точок надає наступні можливості (рисунок 3.11):

1) створення нових папок, для забезпечення повноцінної роботи користувача з хмарним сховищем;
2) завантаження хмар точок у сховище, для їх подальшого використання та обробки на різних машинах;
3) завантаження хмар точок з сховища на локальний комп’ютер, у разі необхідності.

Щоб переглянути хмару точок або отримати її прев’ю необхідно проаналізувати даний файл. Так як метод забезпечує мультиформатний перегляд хмар точок, то він підтримує можливість аналізу та розбору хмар точок збережених у різних форматах даних. Це забезпечується парсерами даних на серверній та на клієнтській сторонах. Розглянемо аналіз хмари точок на прикладі файлу збереженого у форматі LAS. Для початку потрібно зчитати дані з хедеру файлу. Він містить наступну інформацію про хмару точок:

– відступ в байтах від початку файлу до початку опису координат точок;

– загальну кількість точок у хмарі;

– формат у якому точки збережені;

– коефіцієнти масштабу точок та інше.

Коли всі необхідні дані зчитані можна приступати до парсингу хмари точок (рисунок 3.13).

```
//Offset to points always on 96 byte, count from 0
                var offsetToPointData = dataview.getUint32(96, true);
                //Data Format always set on 104 byte, count from 0
                var dataformat = dataview.getUint8(104, true);
                //Point data record length in bytes always set on 105 byte
                var dataRecordLength = dataview.getUint16(105, true);
                //Number of points always set on 107 byte, count from 0
                var numberOfPoints = dataview.getUint32(107, true);
                //scalesAndOffsets contains values of the scaleX, scaleY, scaleZ,
offsetX, offsetY, offsetZ
                var scalesAndOffsets = new Float64Array(6);
                //The scale and offset factor fields contains a double floating point
value (8 bytes) and always start on 131 byte
                for (var i = 0, j = 131; i < scalesAndOffsets.length; i++ , j += 8) {
                    scalesAndOffsets[i] = dataview.getFloat64(j, true);
                }
        values = factory.GetLasPointsWithColor(data, numberOfPoints, dataRecordLength,
offsetToPointData, 20, scalesAndOffsets);
```

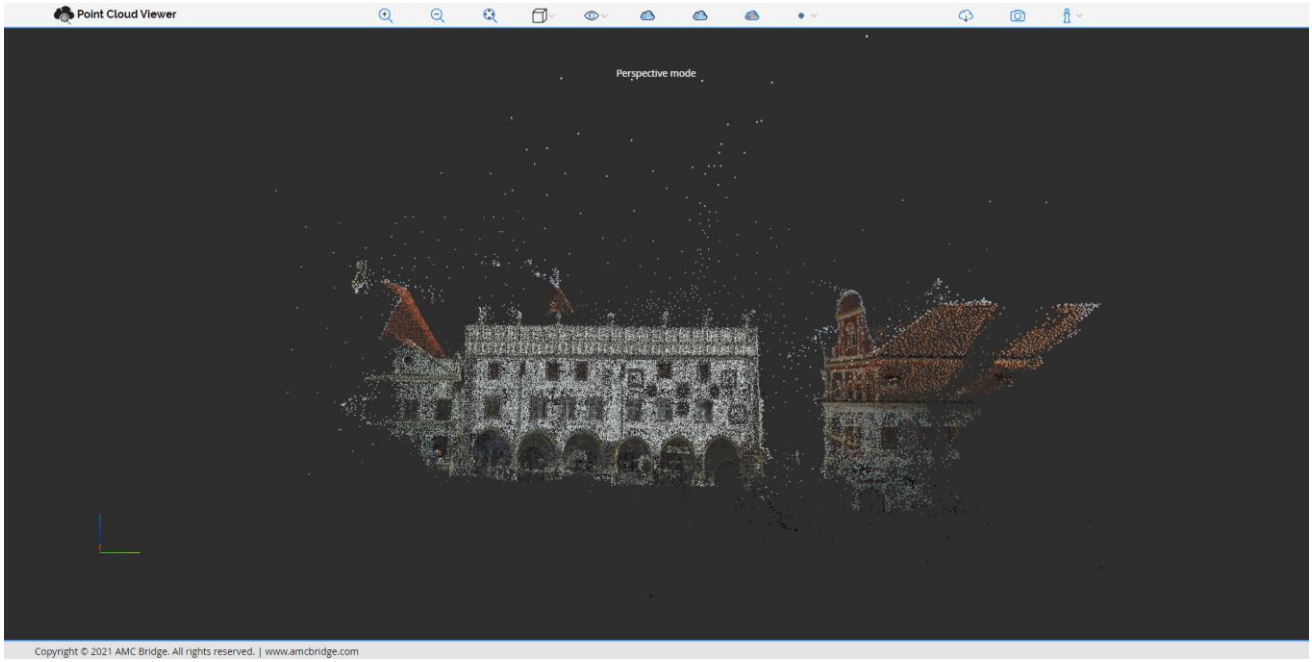


Рисунок 3.13 – Відкрита хмара точок

Розглянемо завантаження хмари точок в хмарне сховище та подальше відкриття моделі більш детально. Для завантаження потрібно спочатку обрати

локальний файл, після чого за допомогою Forge API відбудеться завантаження файлу у хмарне сховище. У разі, якщо операція пройшла успішно хмара точок з’явиться у дереві файлів.

Для того, щоб відкрити хмару точок для перегляду потрібно спочатку визначити формат даних у якому вона збережена. Після цього хмара точок подається у відповідний до її формату парсер для визначення ключових параметрів, насамперед, координат точок та їх кольори. У випадку якщо для точки не визначено кольору, то буде застосовано колір за змовчуванням. Більш детально розбір хмари точок описано вище на прикладі формату даних LAS.

Для побудови 3D відображення потрібно імпортувати бібліотеку three.js у за стосунок. Щоб насправді мати можливість відобразити що-небудь за допомогою three.js, потрібні три речі: сцена, камера та рендерер (візуалізатор), щоб ми могли відтворити сцену за допомогою камери.

Ініціалізація сцени відбується найпростіше у порівнянні з камерами чи візуалізатором. Створення сцени відбувається за допомогою конструктора Scene. Scene надає можливість налаштувати, що і де має відображатися за допомогою three.js. На сцену додаються хмари точок для відображення, освітлення та камери.

Для забезпечення валідної роботи методу візуалізації хмар точок «Web Point Cloud Viewer» необхідно реалізувати підтримку двох камер three.js:

– PerspectiveCamera;

– OrthographicCamera.

Кожна з цих двох камер унаслідується від базового класу Camera який реалізовує бібліотека three.js. Camera є абстрактним базовим класом для камер. Цей клас завжди слід успадковувати, коли створюється нова камера.

Розглянемо налаштування камери PerspectiveCamera. Це камера, яка використовує перспективну проекцію.

Цей режим проекції створений, щоб імітувати те, як бачить людське око. Це найпоширеніший режим проекції, який використовується для візуалізації 3D-сцени за допомогою three.js.

Розглянемо параметри, які потрібно подати в конструктор PerspectiveCamera для ініціалізації камери.

Її першим параметром є поле зору. FOV являє собою масштаб сцени, який видно на дисплеї в будь-який момент. Його значення зазначене в градусах.

Другий параметр це співвідношення сторін. Його значення дорівнює ширині екрану, поділену на висоту.

Наступні два параметри це near та far площини відсікання. Це означає те, що об’єкти, розташовані далі від камери, ніж значення far чи ближче, ніж near, не відображатимуться.

У контексті PerspectiveCamera також варто звернути на параметр zoom. За допомоги цього параметру можна отримати чи встановити фактор зуму камери. За змовчуванням він дорівнює 1.

Розглянемо налаштування камери OrthographicCamera. Це камера, яка використовує орфографічну проекцію.

У цьому режимі проекції розмір об’єкта у відтвореному зображенні залишається незмінним незалежно від його відстані від камери.

Це може бути корисно для візуалізації 2D-сцен та елементів інтерфейсу. Для ініціалізації OrthographicCamera потрібно вказати наступний набір параметрів:

- left - зріз лівої площини камери;
- right - зріз в правій площині камери;
- top - зріз верхньої площини камери;
- bottom - зріз нижньої площини камери;
- near – зріз ближньої площини камери;
- far - зріз дальньої площини камери.

Разом вони визначають кут огляду камери.

Оскільки камери забезпечують можливість відображення хмар точок на сцені, але не надають можливості маніпулювати зі сценою, то необхідно підключити спеціальні інструменти які надає бібліотека three.js для забезпечення такого функціоналу, так звані «контроли». three.js має наступний набір контролів:

– ArcballControls;
– DragControls;
– FirstPersonControls;
– FlyControls;
– OrbitControls;
– PointerLockControls;
– TrackballControls;
– TransformControls.

Для забезпечення маніпуляцій зі сценою у нашому випадку найкраще підійде TrackballControls. Вони схожі на OrbitControls. Однак не підтримують постійний вектор камери вгору. Це означає, що якщо камера обертається над «північним» і «південним» полюсами, вона не перевертається, щоб залишатися «правою стороною вгору». Тобто вони дозволяють камері обертатись навколо цілі, у нашому випадку це хмара точок.

Для ініціалізації TrackballControls у конструктор потрібно подати наступні параметри:

– camera, камера відтвореної сцени.

– domElement, елемент HTML, який використовується для прослуховування подій.

Для того, щоб користувач методу візуалізації хмар точок «Web Point Cloud Viewer» мав можливість взаємодіяти зі сценою за допомогою TrackballControls потрібно забезпечити підтримку подій. TrackballControls підтримує наступні події:

– change, спрацьовує, коли камера змінена елементами керування;

– start, спрацьовує, коли ініціюється взаємодія користувача зі сценою. Наприклад, дотик чи натискання кнопки мишки;

– end, спрацьовує після завершення взаємодії.

Також важливими властивостями класу TrackballControls є:

– panSpeed, швидкість пану камери;

– rotateSpeed, швидкість обертання;

– zoomSpeed, швидкість зміни зуму.

Всі ці три параметри важливі для забезпечення користувача методу візуалізації хмар точок «Web Point Cloud Viewer» комфортною маніпуляцією зі сценою. Встановимо їх значення відповідно 0.3, 1.0 та 1.2.

Далі рендерер. У даній реалізації використовується WebGLRenderer, але на додаток до нього, three.js постачається з декількома іншими, які часто використовуються як резервні копії для користувачів із старими браузерами або для тих, хто не має підтримки WebGL з якихось причин.

При ініціалізації WebGLRenderer необхідно встановити колір, розмір та співвідношення пікселів. Це забезпечують наступні фунції класу:

– setClearColor, встановлює чіткий колір і непрозорість;

– setSize, встановлює співвідношення пікселів пристрою. Зазвичай це використовується для пристрою HiDPI, щоб запобігти розмиванню вихідного полотна;

– setPixelRatio, змінює розмір вихідного полотна до поданих значень з урахуванням співвідношення пікселів пристрою, а також налаштовує область перегляду відповідно до цього розміру, починаючи з (0, 0). Установлення для updateStyle значення false запобігає будь-яким змінам стилю вихідного полотна.

На додаток до створення екземпляра візуалізації, потрібно встановити розмір для відображення моделі. У даному випадку було використано ширину та висоту вікна браузера.

І останнє, але не менш важливе, потрібно додати елемент візуалізації в наш документ HTML. Це елемент <canvas>, який рендерер використовує для відображення сцени.

Щоб бачити зміни які будуть відбуватись на сцені необхідно запустити цикл рендерингу.

Це створить цикл, який змушує візуалізатор малювати сцену кожного разу, коли екран оновлюється (на типовому екрані це означає 60 разів на секунду).

Після підготовки всіх необхідних модулів перейдемо, безпосередньо, до відображення на сцені хмар точок. Для відображення моделі, наостанок, потрібно додати підготовані точки на сцену за допомогою THREE.Points.

Для ініціалізації даного класу необхідно в конструктор передати два параметри:

– geometry, геометрія, яка описує та визначає структуру хмари точок;

– material, матеріал, що визначає зовнішній вигляд об’єкта.

Для визначення геометрії використаємо клас BufferGeometry, так як даний клас є продуктивнішим аніж Geometry та потребує набагато менше машиних ресурсів. Він є поданням геометрії сітки, лінії або точки. Включає позиції вершин, індекси граней, нормалі, кольори, UV та користувацькі атрибути в буферах, зменшуючи витрати на передачу всіх цих даних до GPU.

Для опису матеріалу хмари точок використаємо клас PointsMaterial. Він завжди використовується при роботі з точками. Для його ініціалізації потрібно подати об'єкт з однією або кількома властивостями, що визначають зовнішній вигляд матеріалу. Сюди можна передати будь-яку властивість матеріалу. Подамо в нього наступні параметри:

– size, даний параметр встановлює розмір точок. Встановимо його на рівні 0.005;

– vertexColor, даний параметр визначає чи використовується забарвлення точок. Якщо хмара точок немає кольорів для вершин, то буде використано колір за змовчуванням.

Розглянемо роботу модулю, що забезпечує можливість вимірювання хмар точок.

Базовою вимогою до нього є можливість забезпечення визначення позиції точки, що відноситься до хмари точок. Для підтримки даного функціоналу потрібно використати метод рейкастингу. У бібліотеці three.js для цього реалізовано клас Raycaster. Рейкастинг використовується для визначення, які об’єкти у 3д просторі знаходяться безпосередньо під мишкою. Для отримання точки використаємо метод класу Raycaster intersectObject. Даний метод перевіряє всі перетини між променем і об’єктом з його нащадками або без них. Як результат метод повертає масив перетинів відсортованих за відстанню, спочатку найближчі. Першим елементом масиву буде найближча точка, з якої буде отримано позицію для відображення на екрані.

Для побудови лінії або сплайну між точками хмари необхідно визначити позиції точок (двох у випадку лінії та безлічі у випадку сплайну) описаним вище методом. Після знаходження позицій точок, пошук відстані між ними забезпечується класом Vector3 реалізацію якого надає бібліотека three.js. Даний клас представляє тривимірний вектор, який є впорядкованою трійкою чисел (позначених x, y і z), яку можна використовувати для представлення ряду речей, в даному випадку точки в 3д просторі. За допомогою методу distanceTo класу Vector3 можна легко обчислити відстань між векторами.

Для знаходження кута між точками використаємо метод класу Vector3 angleTo. Даний метод повертає значення кута між двома векторами у радіанах. Після чого отримане значення буде передене в градуси.

Блок-схему роботи методу візуалізації хмар точок «Web Point Cloud Viewer» можна побачити на рисунку 3.14.

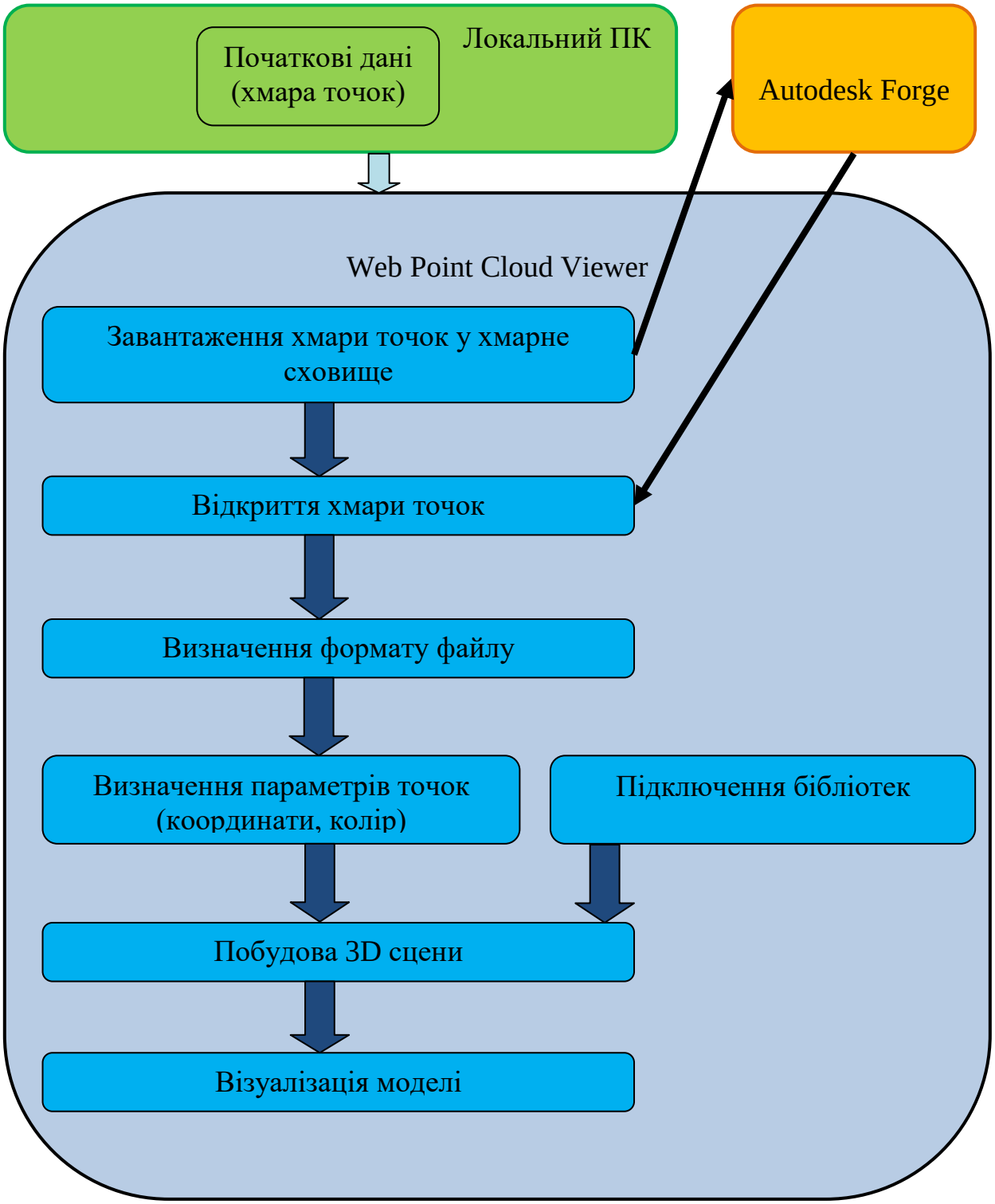


Рисунок 3.14 – Блок-схема роботи методу

Висновки до розділу 3

В даному розділі було розроблено програмні модулі та детально описані алгоритмічні конструкції методу візуалізації хмар точок «Web Point Cloud Viewer». Описано функціональне призначення програмних модулів та їх взаємозв’язок між собою. Розроблено механізм авторизації користувача методу візуалізації хмар точок «Web Point Cloud Viewer». Сформовано алгоритми для аналізу та відображення хмар точок.

Отже, після аналізу та розробки програмних модулів, виконання детального опису алгоритмічних конструкцій методу візуалізації хмар точок, можна дійти до конструктивного висновку, що метод забезпечує виконання та підримку:

1. Вище зазначених бізнес-процесів для автоматизації інформаційних потоків;
2. Базових вимог використання користувачем;
3. Можливості переходу до апробації методу.

Розділ 4
Апробація методу візуалізації хмар точок

4.1 Апробація швидкодії методу візуалізації хмар точок

Для перевірки швидкодії методу візуалізації хмар точок «Web Point Cloud Viewer» потрібно провести порівняння з іншими переглядачами точок. Насамперед необхідно проаналізувати час за який хмара точок відобразиться на екрані. Так як багато переглядачів підтримують лише один чи декілька форматів даних у яких зберігаються хмари точок, то необхідно провести порівняння з кількома з них.

Для початку на прикладі моделей france.las, pump_000004_big.xyz та Townhall_Ok.xyz порівняємо час відкриття хмари точок з веб-застосунком Online LIDAR point cloud viewer. У таблиці 4.1 наведено результати проведеного аналізу. Як бачимо хмари точок france.las та pump_000004_big.xyz відкриваються швидше у методі візуалізації хмар точок «Web Point Cloud Viewer», при цьому хмару точок Townhall_Ok.xyz неможливо відкрити за допомогою Online LIDAR point cloud viewer (рисунок 4.1).

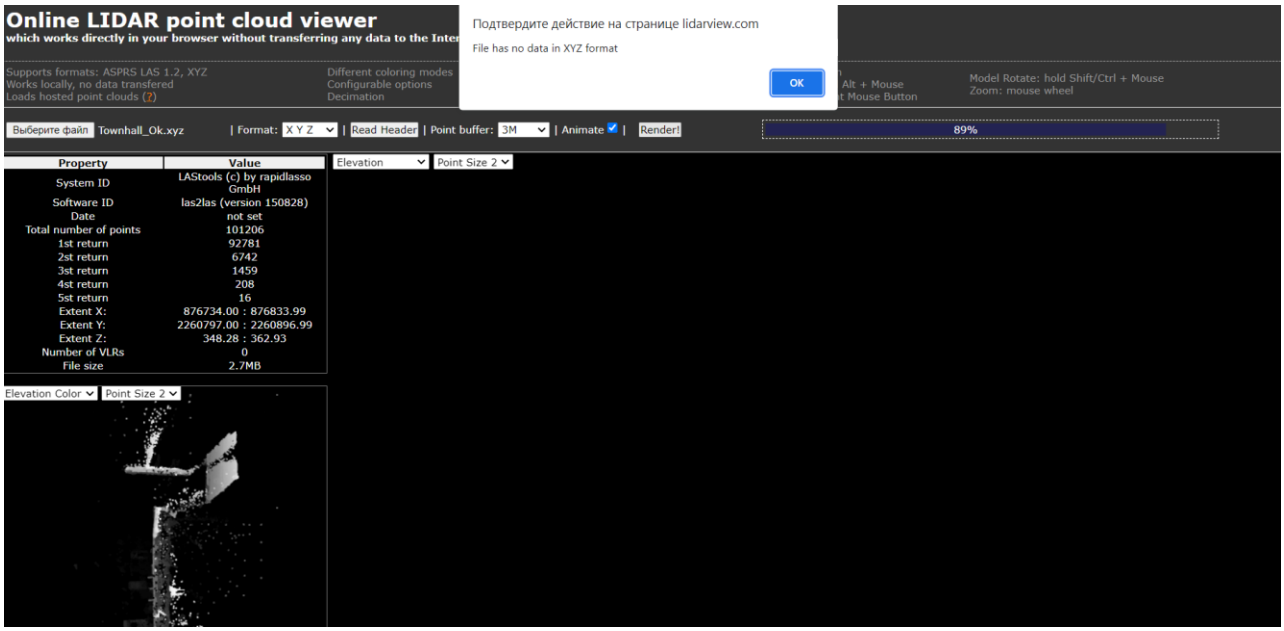


Рисунок 4.1 – Помилка відкриття хмари точок

Таблиця 4.1 – Таблиця порівняння швидкодії відображення

Час	Web Point Cloud Viewer	Online LIDAR
france.las	1.75 с	2.65 с
pump_000004_big.xyz	10.85 с	11.51 с
Townhall_Ok.xyz	1.93 с	-

Після проведеного порівняння швидкодії відкриття хмар точок з веб-застосунком Online LIDAR point cloud viewer, можна дійти до конструктивного висновку, що метод візуалізації хмар точок «Web Point Cloud Viewer» є на порядок швидшим. Хмара точок pump_000004_big.xyz зображена на рисунку 4.2.

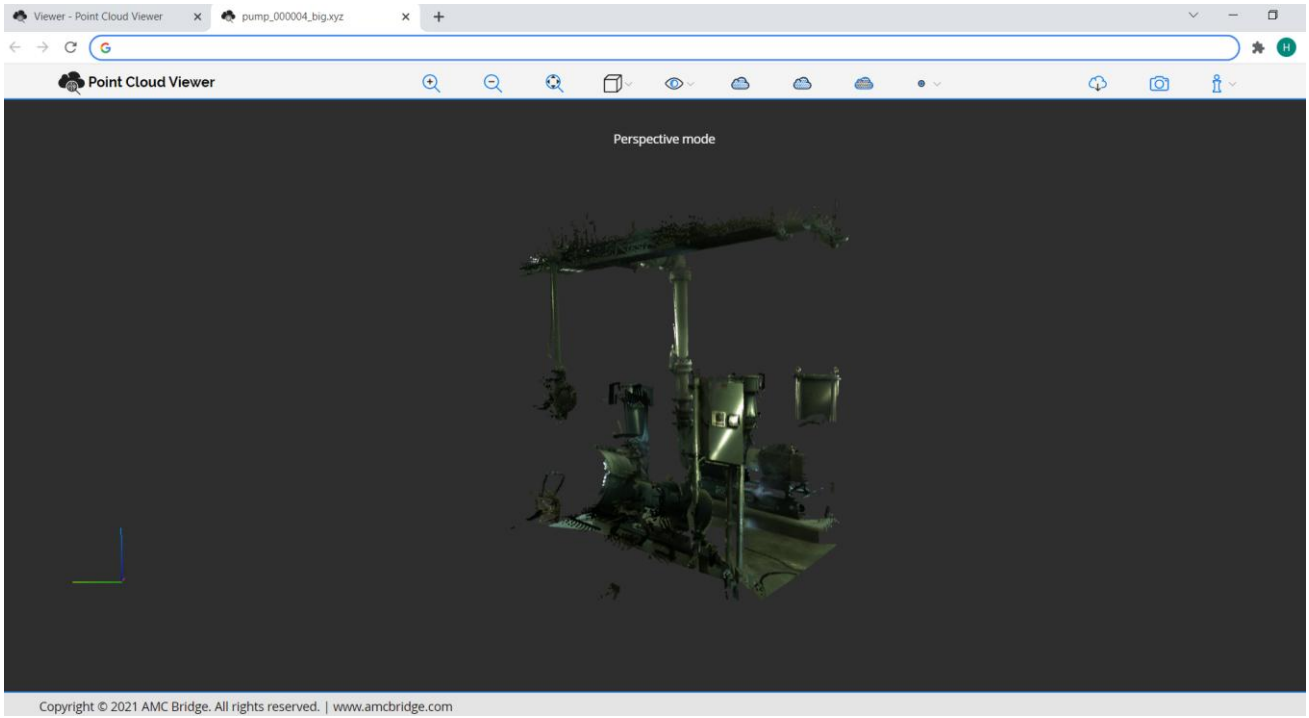


Рисунок 4.2 – Хмара точок pump_000004_big.xyz

Під час апробації методу візуалізації хмар точок «Web Point Cloud Viewer» необхідно перевірити швидкість відклику переглядача на маніпуляції, проведені користувачем із сценою на якій відображена хмара точок. Хмара точок mountain.laz зображена на рисунку 4.3.

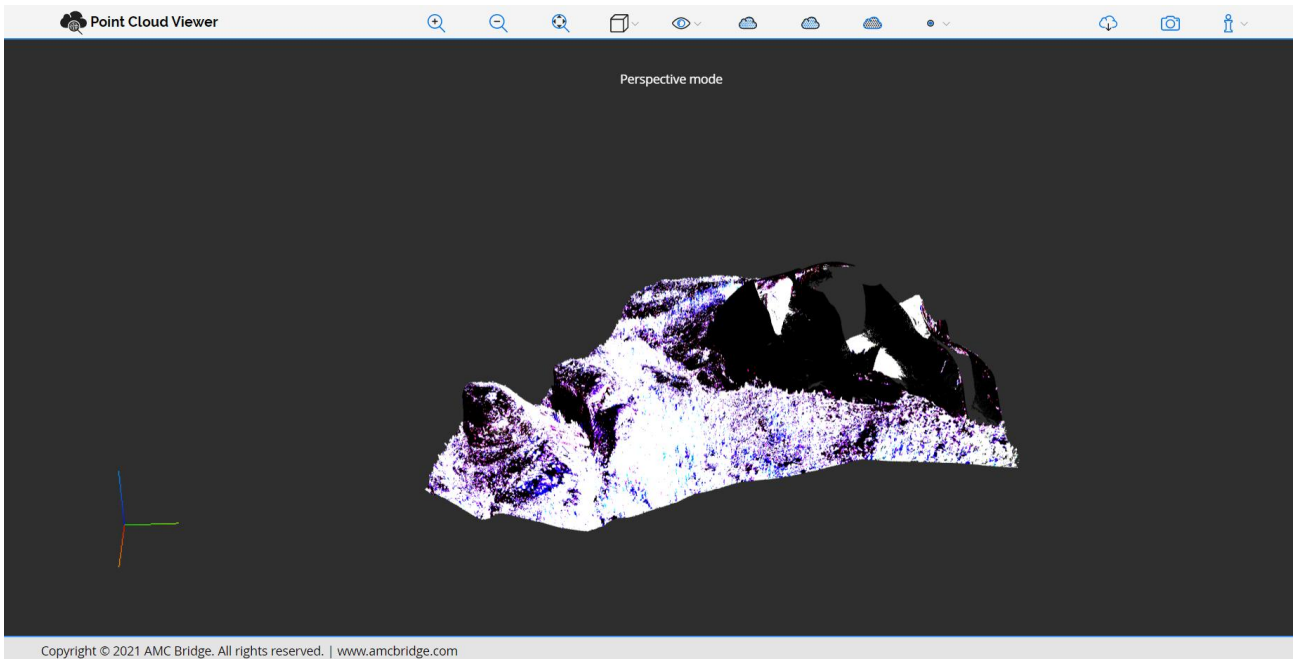


Рисунок 4.3 – Хмара точок mountain.laz

Для ефективної перевірки буде використано велику хмару точок, яка містить близько восьми мільйонів точок. Порівняння буде проведено на прикладі хмари точок mountain.laz. Також цього разу для перевірки швидкодії відклику буде використано веб-застосунок plas.io [29]. На рисунку 4.4 приведене зображення хмари точок france.laz.

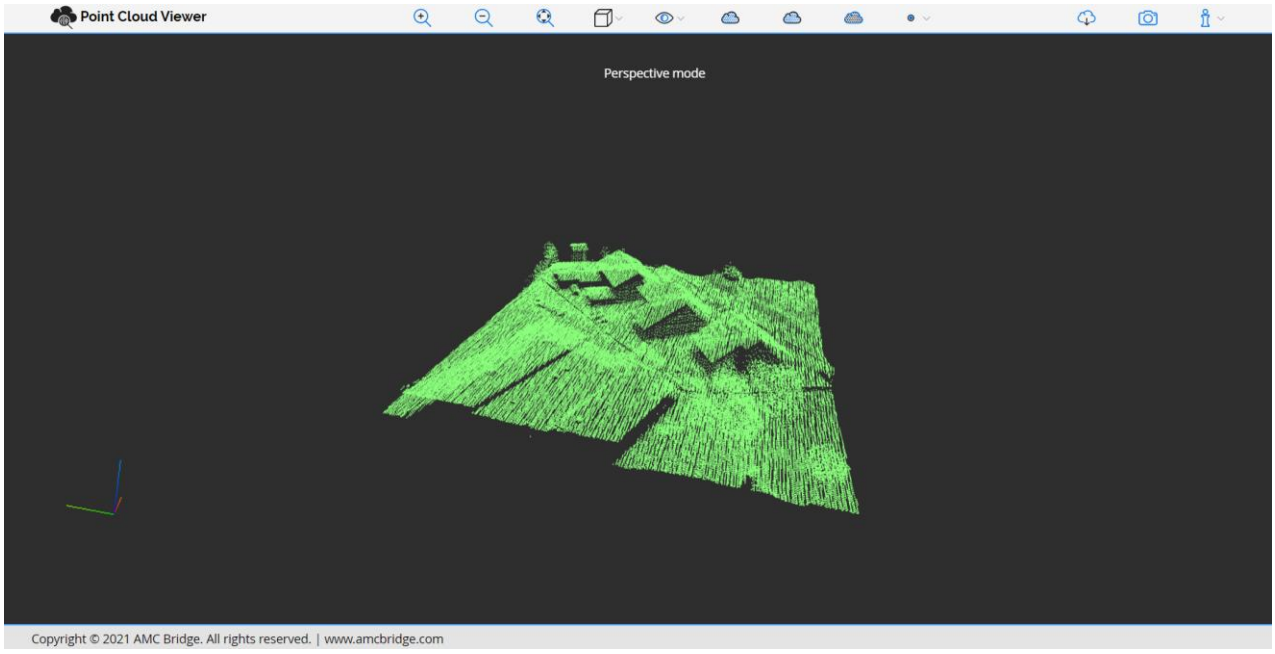


Рисунок 4.4 – Хмара точок france.laz

Таблиця 4.2 – Таблиця порівняння швидкодії відклику

Час	Web Point Cloud Viewer	plas.io
Зум камери	0.92 с	0.87 с
Зміна розміру точок	0.12 с	0.17 с
Зміна положення камери	0.34 с	0.61 с

Отже, як можна побачити з таблиці 4.2 у якій наведено результати апробації швидкодії відклику переглядачів хмар точок, можна зробити конструктивний висновок, що метод візуалізації хмар точок «Web Point Cloud Viewer» у двох із трьох перевірок виявився швидшим. А саме зміна розміру точок та положення камери є більш ефективними у розробленому методі. При цьому варто зауважити, що для великих хмар точок зум камери відбувається швидше у веб-застосунку plas.io. Хмара точок Townhall_Ok.laz зображена на рисунку 4.5.

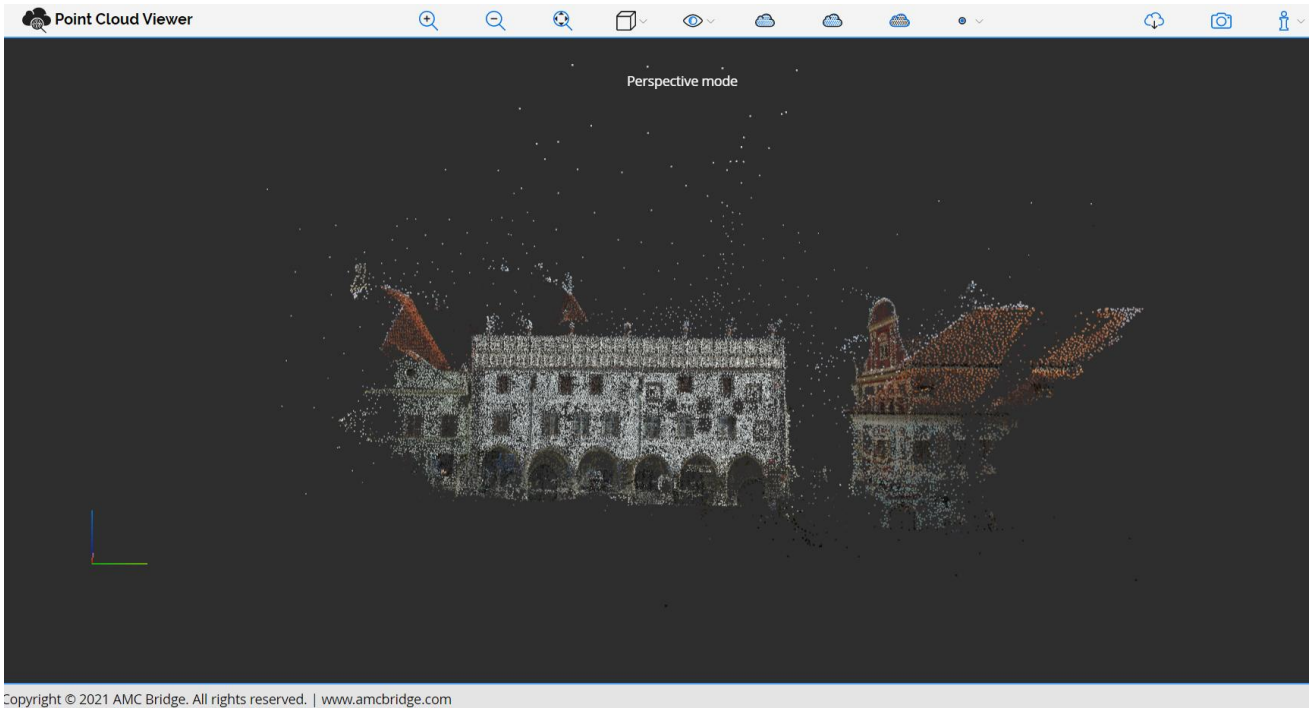


Рисунок 4.5 – Хмара точок Townhall_Ok.laz

Під час перевірки швидкодії відклику були дотримані наступні умови:

– однаковий розмір точок моделі;

– однакова віддаленість камери від хмари точок;

– схоже положення камери.

Варто зауважити, що при збільшені розміру точок зменшується продуктивність переглядачів, тим самим збільшується час відклику на дії користувача. Схоже поведінка спостерігається при зумі камери. Чим більше точок зображено на моніторі, тим повільнішими стають переглядачі хмар точок. Така поведінка є очікуваною.

Після проведення апробації швидкодії відображення хмар точок та швидкодії відклику на дії користувача, можна зробити висновок, що метод візуалізації хмар точок «Web Point Cloud Viewer» є суттєво швидшим за системи з якими було проведено порівняння.

4.1 Основні функції програми

Для перевірки можливостей методу візуалізації хмар точок «Web Point Cloud Viewer» була виконана його функціональна перевірка. У відповідності до поставленого завдання, метод повинен забезпечувати виконання наступних груп функцій:

– робота із хмарним сховищем (реєстрація, вхід та вихід користувача, створення нових папок, завантаження хмар точок у сховище, завантаження хмар точок із сховища);

– робота з камерою (зміна сектору огляду камери, зміна виду камери, зміна позиції камери, повернення камери до початкової позиції);

– робота із сценою (показ моделі з різних сторін, створення знімку сцени, зміна кольорового фону сцени, підтримка списку скорочень клавіш, відображення глобальної системи координат моделі);

– робота з моделлю (модифікація розміру точок, зміна забарвлення точок моделі, повернення оригінального кольору);

– робота з вимірюванням моделі (вимірювання позиції точки, вимірювання дистанції між двома точками, вимірювання дистанції між безліччю точок, вимірювання кута між точками, вимірювання площі);

– робота з хмарами точок (розпізнавання розширення файлу, обробка та зчитування бінарних файлів, обробка та зчитування файлів у форматі ASCII, попередній перегляд моделі).

Виконання бізнес-процесу «Використання хмарного сховища» продемонстровано на рисунках 4.5-4.7. Для реєстрації користувачу необхідно натиснути кнопку Register, після чого ввести необхідні дані у форму реєстрації (рисунок 3.10). Якщо реєстрація пройшла успішно, то для користувача буде створено обліковий запис із забезпеченням взаємодії з хмарним сховищем.

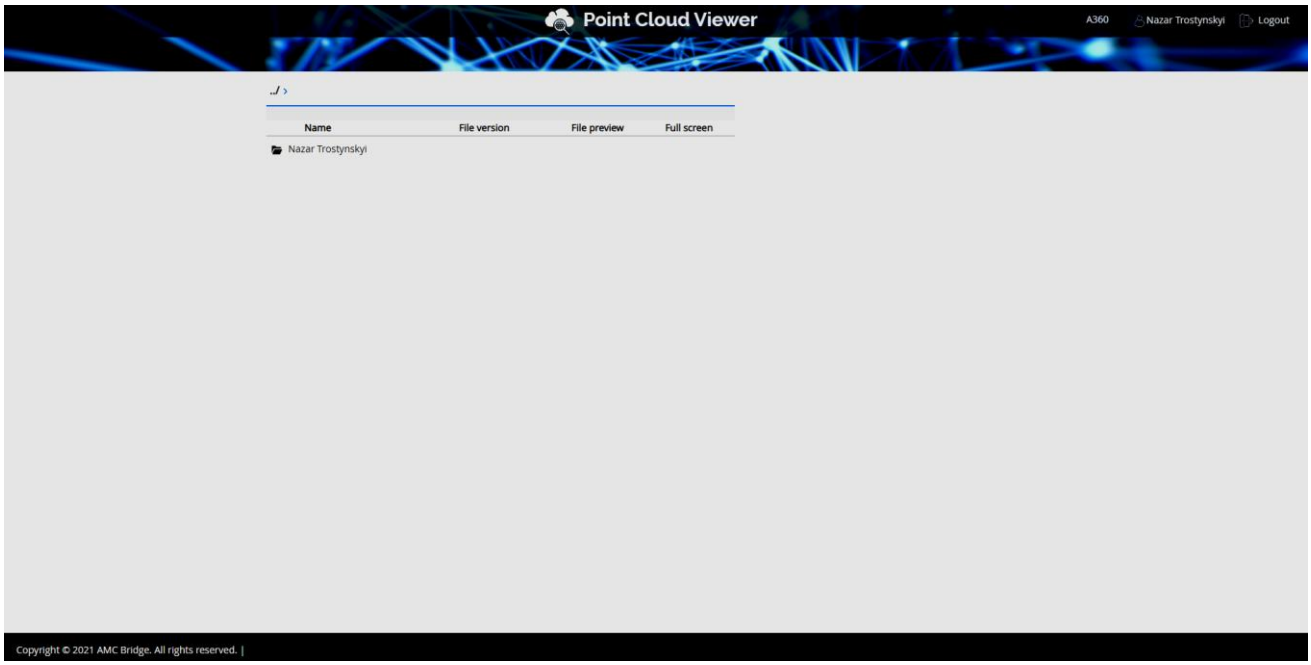


Рисунок 4.5 – Результат кнопки Register

Кнопка «Create a new folder» повинна забезпечувати виконання наступних функцій:

– показ модального вікна для вводу назви папки;

– безпосереднє створення папки у хмарному сховищі.

Як результат виконання даної функції нова папка повинна бути створена (рисунок 4.6).

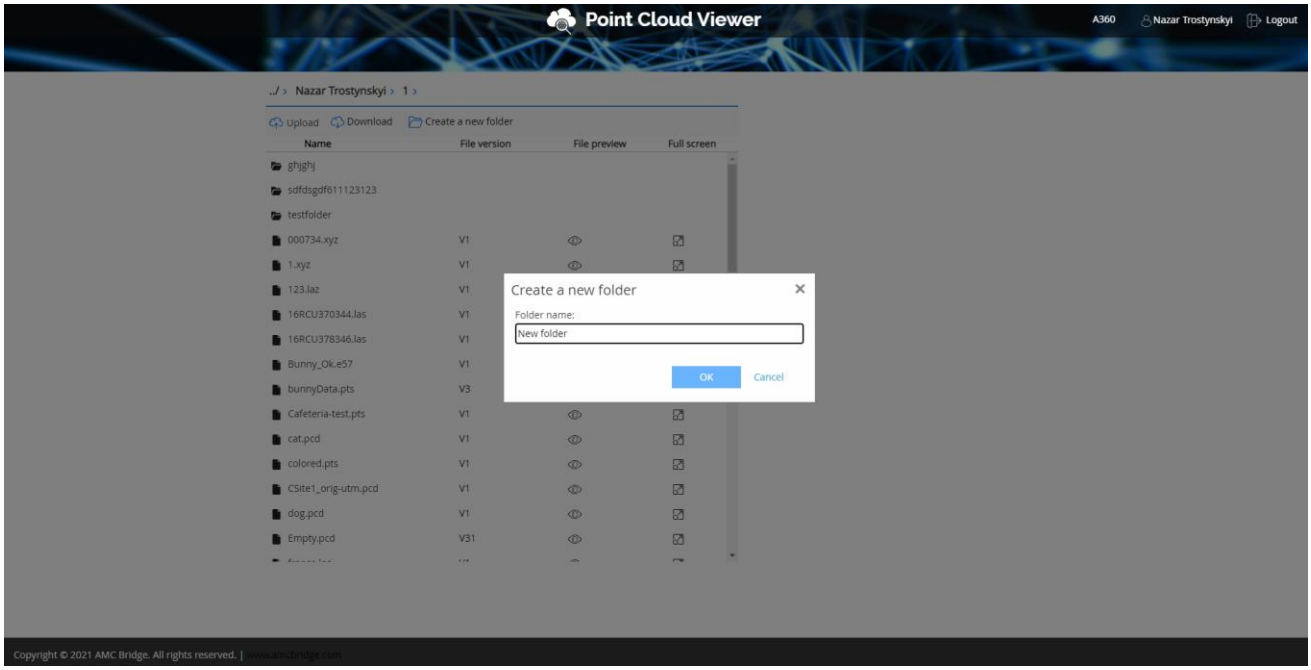


Рисунок 4.6 – Створення нової папки

Завантаження файлів з локального комп’ютера повинно бути забезпечено кнопкою «Upload». Після її натискання повинен бути показаний системний провідник, щоб користувач міг обрати необхідні файли. Як результат виконання даної функції обрані файли мають бути завантаженні до хмарного сховища та відображені у методі (рисунок 4.7).

Завантаження файлів на локальну машину забезпечується кнопкою «Download». Результат виконання даної функції зображено на рисунку 4.7.

Виконання бізнес-процесу «Робота з хмарою точок», що забезпечує виконання наступних функцій:

– розпізнавання розширення файлу;

– обробка та зчитування бінарних файлів;

– обробка та зчитування файлів у форматі ASCII;

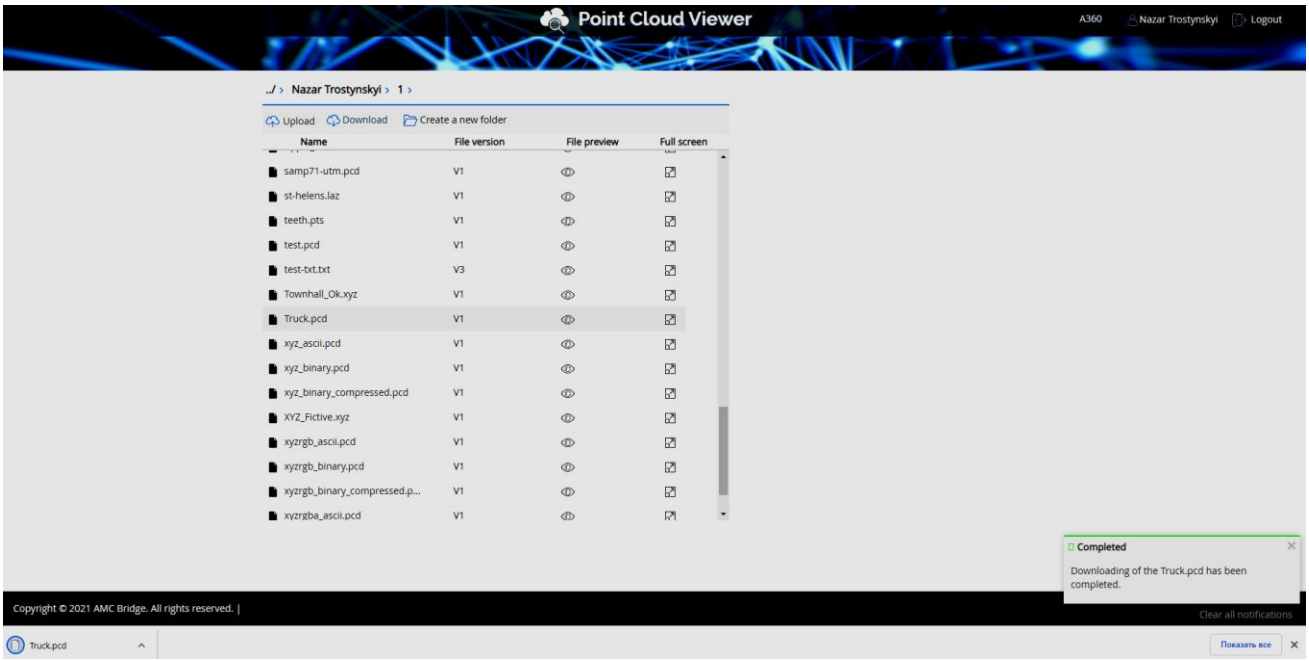


Рисунок 4.7 – Завантаження файлів

Дані функції можна перевірити якщо відкрити хмару точок для перегляду, оскільки для цього потрібно визначити розширення файлу і після цього відповідно проаналізувати хмару точок для отримання всіх необхідних даних для її відображення (рисунок 4.8).

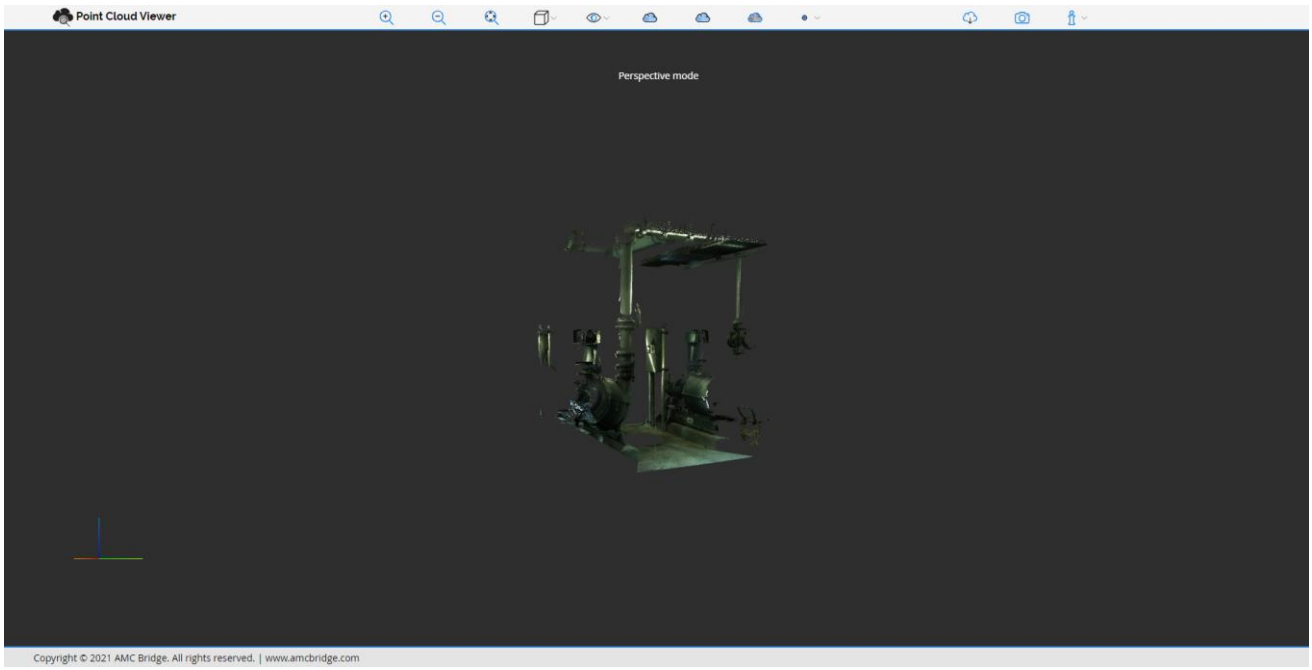


Рисунок 4.8 – Початкове положення моделі

Виконання бізнес-процесу «Робота з камерою» продемонстровано на рисунках 4.8-4.9. Відповідно до умови автоматизації інформаційних потоків метод візуалізації хмар точок в поданому пункті повинен забезпечувати організацію поданих функцій:

– зміна сектору огляду камери;
– зміна виду камери;
– зміна позиції камери;
– повернення камери до початкової позиції.

Зміна сектору огляду камери забезпечується прокручуванням колеса мишки (рисунок 4.9). Можливість зміни позиції камери забезпечена наступним способом: при натисненій правій кнопці мишки змінювати позицію мишки (рисунок 4.9). Коли користувач змінює тип камери, вона повинна змінитись у відповідності до обраного варіанту (рисунок 4.9). Метод повинен підтримувати два типи камер:

– ортогональну;
– перспективну.

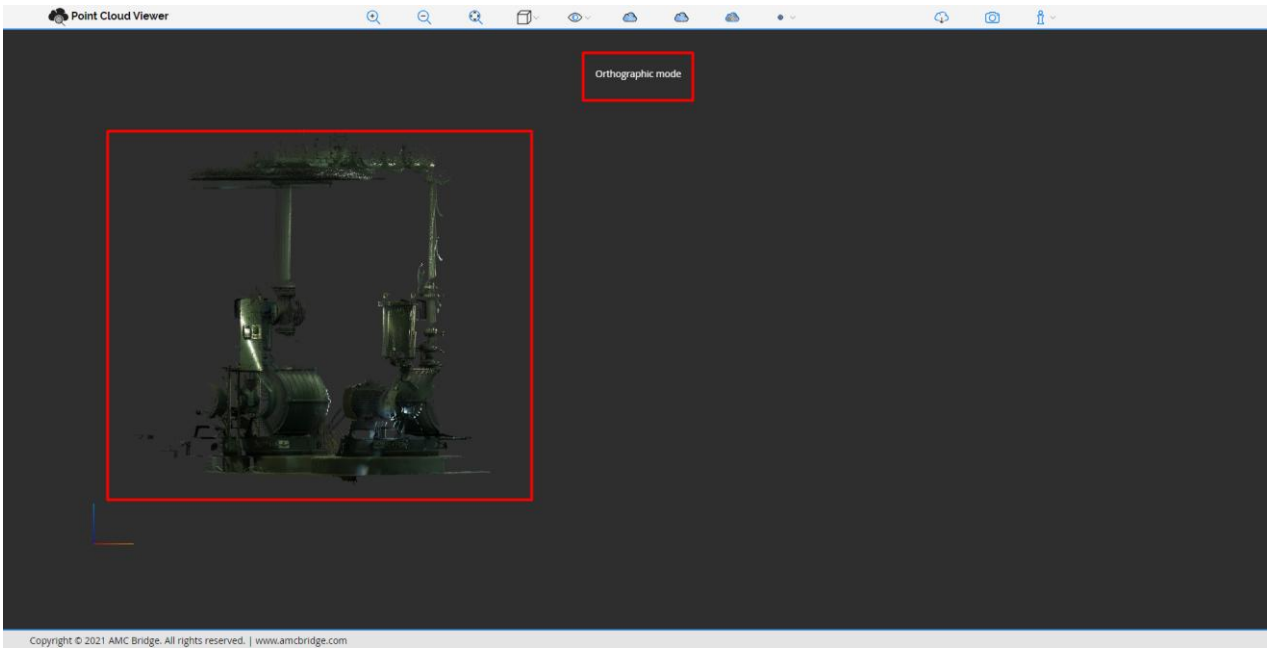


Рисунок 4.9 – Перевірка скролу, пану та зміни типу камери

Після натискання на кнопку «Zoom to fit» камера повинна зайняту початкову позицію по відношенню до моделі (рисунок 4.8).

Виконання бізнес-процесу «Робота із сценою» продемонстровано на рисунках 4.10-4.12. Відповідно до умови автоматизації інформаційних потоків метод візуалізації хмар точок в поданому пункті повинна забезпечувати організацію поданих функцій:

– показ моделі з різних сторін;
– створення знімку сцени;
– зміна кольорового фону сцени;
– підтримка списку скорочень клавіш;
– відображення глобальної системи координат моделі.

Кнопка «Set background color» повинна забезпечувати зміну кольору сцени. Після її натискання повинно з’явитись модальне вікно для вибору кольору, як результат виконання даного функціоналу колір фону сцени повинен змінитись (рисунок 4.10).

Кнопка «Set camera position» повинна забезпечувати появу випадаючого списку з наступними варіантами положення камери:

– Top;
– Bottom;
– Left;
– Right;
– Front
– Back.

Коли положення обрано камера повинна відповідно змінити своє положення по відношенню до моделі (хмари точок). Система координат у лівому нижньому куті повинна змінитись відповідно (рисунок 4.10).

Кнопка «Take a photo of model» повинна забезпечувати можливість знімку екрану. Після її натискання отримане фото повинно завантажитись на локальний

комп’ютер, а також повинне відкритись нове вікно у браузері з отриманим знімком (рисунок 4.11).

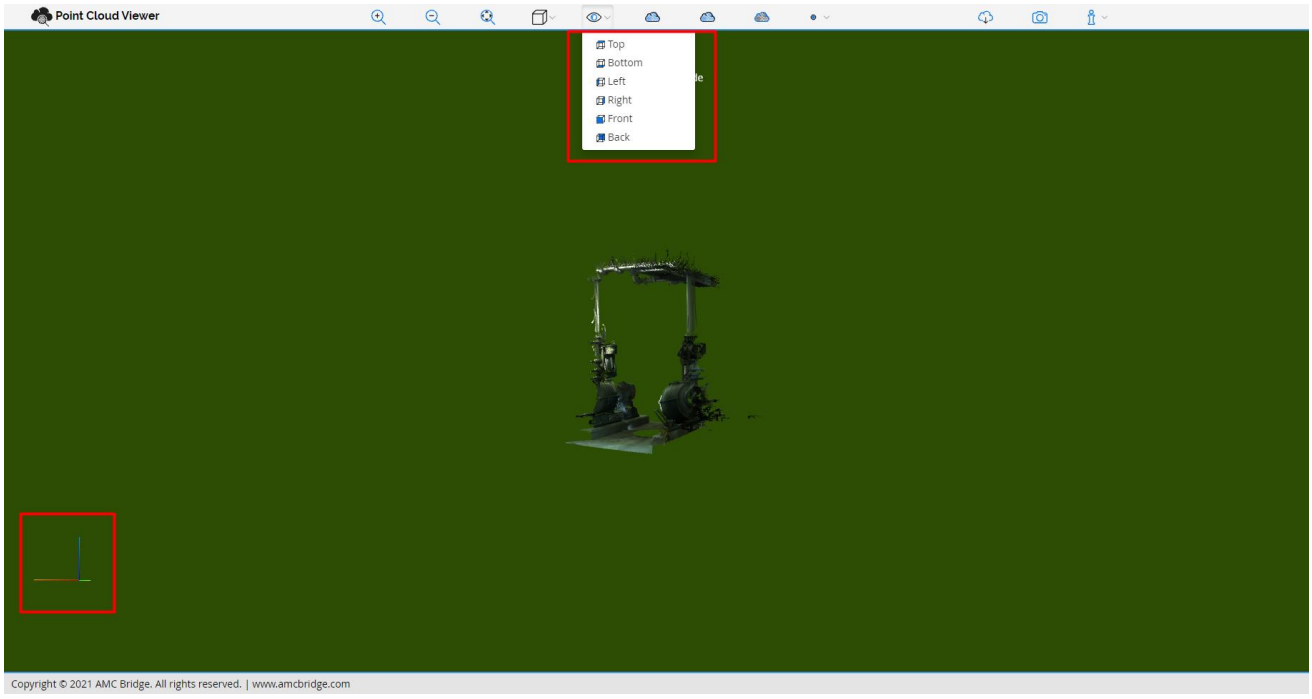


Рисунок 4.10 – Перевірка функцій роботи із сценою

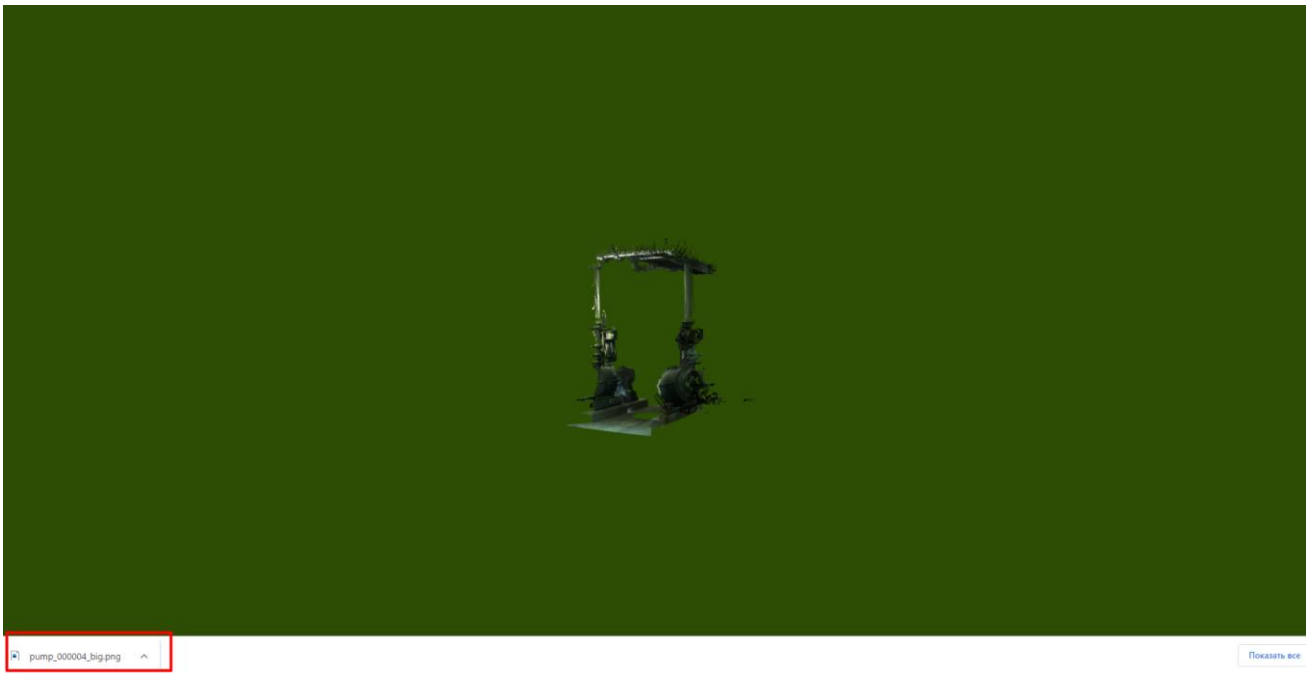


Рисунок 4.11 – Перевірка скріншоту моделі

Кнопка «Information about hotkeys» повинна забезпечувати показ випадаючого меню із списком гарячих клавіш, що підтримуються методом (рисунок 4.12).

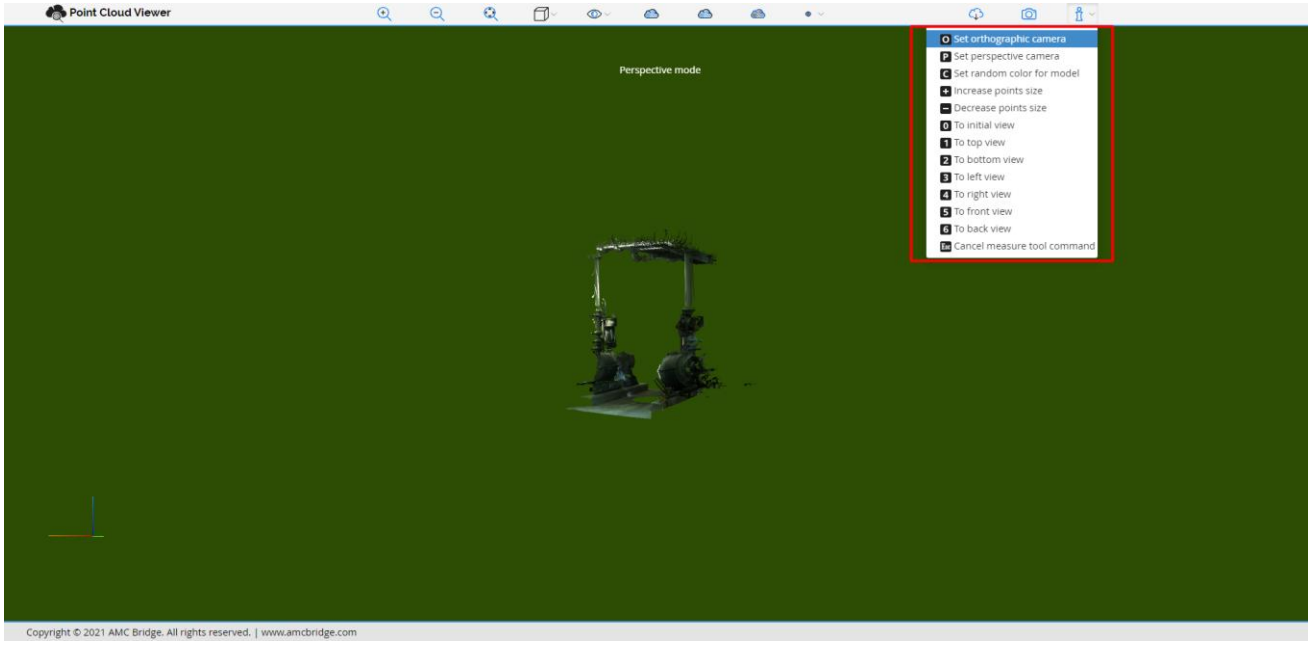


Рисунок 4.12 – Перевірка гарячих клавіш

Список повинен містити наступні пункти:

– Set orthographic camera;
– Set perspective camera;
– Set random color for model;
– Increase point size;
– Decrease point size;
– To initial view;
– To top view;
– To bottom view;
– To left view;
– To right view;
– To front view;
– To back view;

– Cancel measure tool command.

При натисканні на будь-яку кнопку із списку повинна виконатись відповідна їй команда.

Виконання бізнес-процесу «Робота з моделлю» продемонстровано на рисунках 4.13-4.14. Відповідно до умови автоматизації інформаційних потоків метод візуалізації хмар точок в даному пункті повинен забезпечувати організацію поданих функцій:

– модифікацію розміру точок;

– зміна забарвлення точок моделі;

– повернення оригінального кольору.

Кнопка «Set points color» повинна забезпечувати появу модального вікна для вибору кольору точок моделі. Коли користувач обрав необхідний колір, то забарвлення точок моделі повинно змінитись у відповідності до вибору (рисунок 4.13).

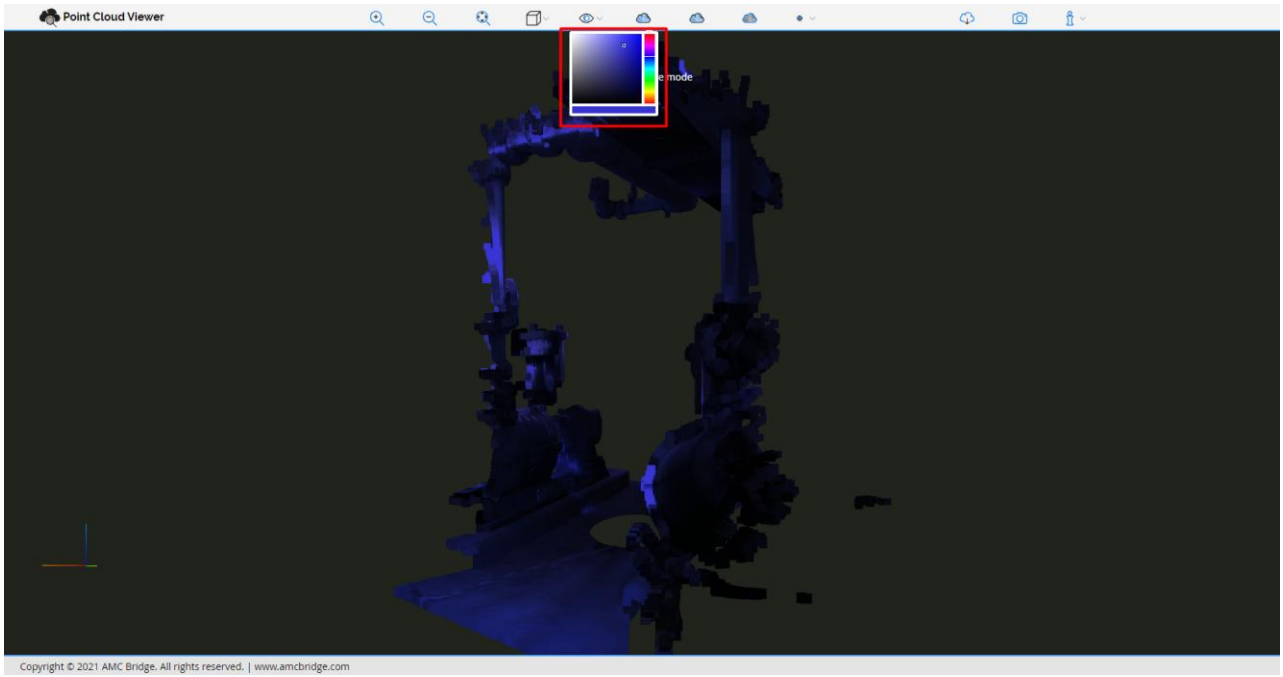


Рисунок 4.13 – Перевірка зміни кольору та розміру точок моделі

Кнопка «Increase points size» повинна забезпечувати збільшення розміру точок у два рази. Відповідно кнопка «Decrease points size» повинна забезпечувати зменшення розміру точок у два рази (рисунок 4.14).

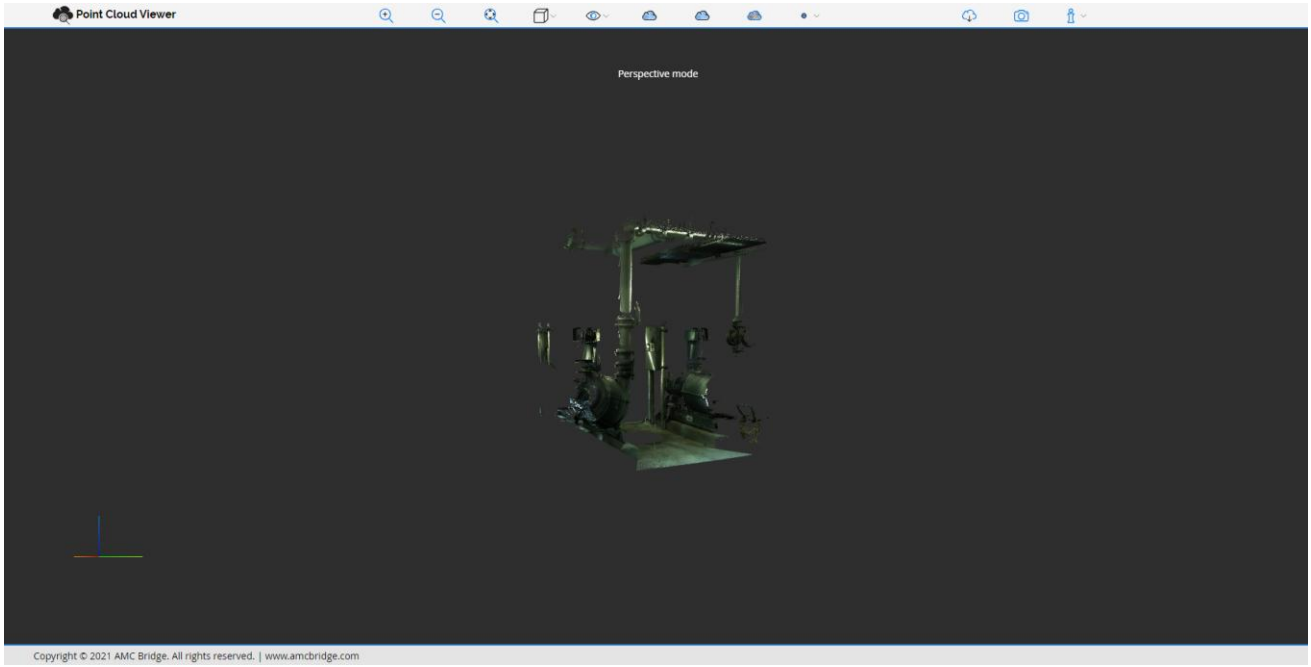


Рисунок 4.14 – Перевірка повернення початкового кольору

Кнопка «Set points color to default» повинна забезпечувати повернення початкового кольору точок моделі (рисунок 4.14).

Для перевірки реалізації методом візуалізації хмар точок можливості вимірювання моделей, що представленні у вигляді хмари точок необхідно послідовно провести вимірювання позиції точки моделі, відстанні між точками моделі, кута між точками моделі та площі між обраними точками.

Для забезпечення точної перевірки було створено спеціальну хмару точок, що містить 8 точок розміщених на відстані в один метр один від одного. Як можна побачити з рисунка 4.15 вимірювання відстані між точками та позиції точки є ідеально точним, так як довжина лінії склала 4 метри та отримані позиції точок відповідають реальності.

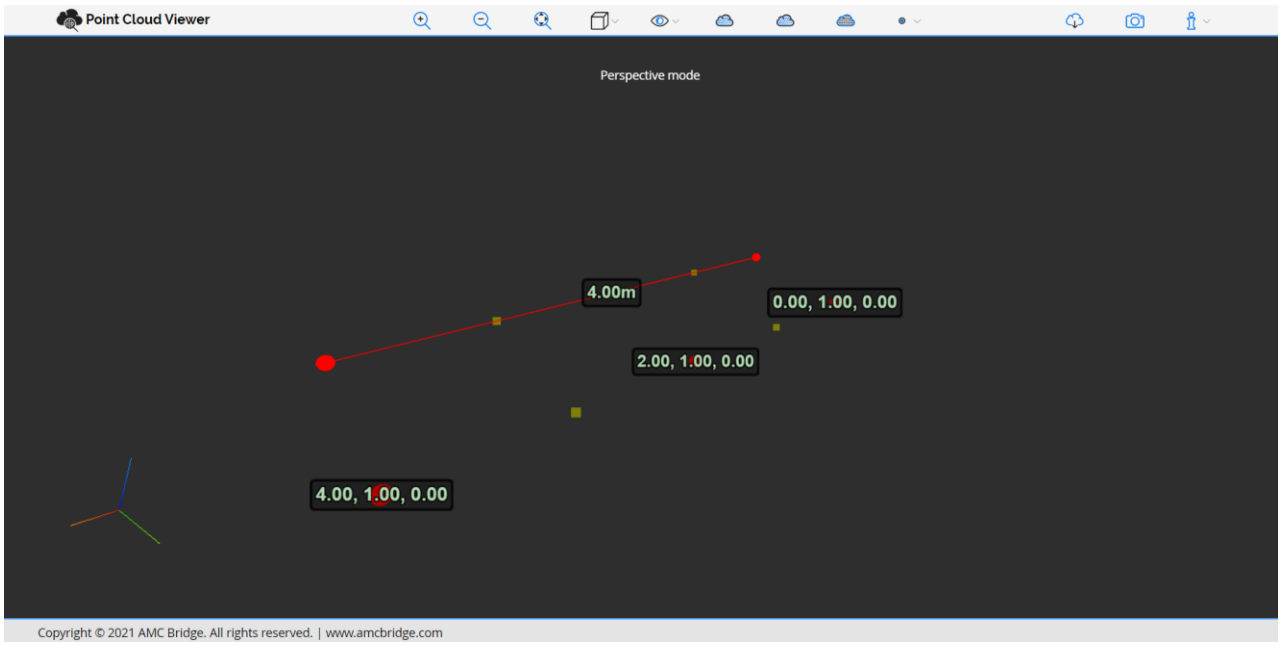


Рисунок 4.15 – Перевірка вимірювання позиції та відстані

На рисунку 4.16 відображені результати перевірки вимірювання площі фігури та кута між точками. Як можна побачити площа квадрати зі сторонами в 1м дорівнює 1 квадратному метру, що в свою чергу відповідає дійсності. В даному випадку було перевірено кут між точками, що відповідає прямокутному трикутнику, при цьому кути вимірено правильно, тобто 45, 45 та 90 градусів.

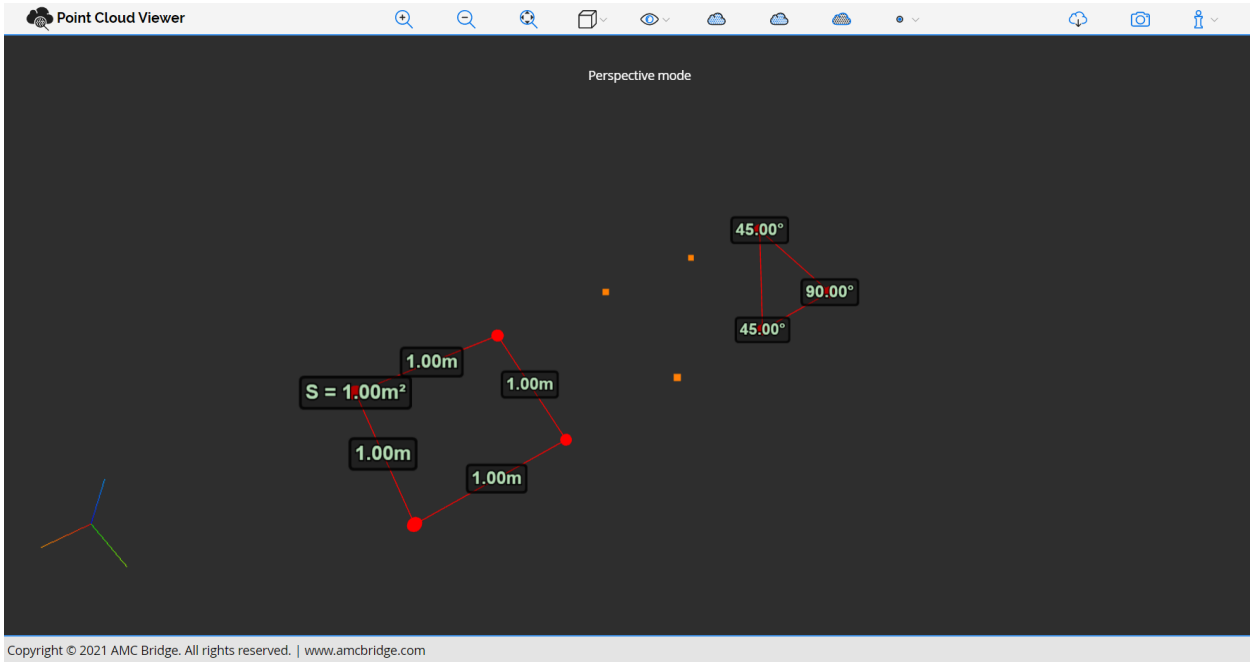


Рисунок 4.16 – Перевірка вимірювання моделі

Висновки до розділу 4

В рамках четвертого розділу було проведено апробацію реалізованого методу візуалізації хмар точок «Web Point Cloud Viewer». Для прикладу було проведено апробацію швидкодії методу, а саме швидкодії відображення хмари точок та швидкодію відклику методу на дії користувача. Для ефективної перевірки швидкодії відображення хмари точок тестування було проведено на декількох хмарах точок різних форматів та з різною кількістю точок. В результаті було отримано висновок, що метод візуалізації хмар точок «Web Point Cloud Viewer» є швидшим ніж інші системи предметної області. Аналогічний результат було отримано при апробації швидкодії відклику методу на дії користувача.

Також було перевірено основні функції методу візуалізації хмар точок «Web Point Cloud Viewer». В результаті даного тестування може дійти до конструктивного висновку, що розроблений метод забезпечує виконання усіх описаних функцій та повністю відповідає поставленим вимогам.

Загальні висновки

В рамках кваліфікаційної роботи магістра було проведено аналіз літературних джерел, що засвідчило суттєву та зростаючу потребу у візуалізації тривимірних об’єктів, з високим рівнем коректності відтворення таких об’єктів у поєднанні з невисокими вимогами до технічних засобів із використанням універсального та (або) вбудованого в операційну систему програмного забезпечення.

В результаті проведеного аналізу існуючих підходів сформульовано та вирішено такі завдання як:

1. Проведено аналіз існуючих технологій, методів та рішень для відображення хмар точок;

2. Удосконалено існуючі методи по роботі з хмарами точок у напрямку покращення швидкодії та доступності для кінцевих користувачів;

3. Реалізовано метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень;

4. Виконано експериментальну перевірку методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень.

Було визначено повний перелік бізнес-процесів, які підлягають алгоритмам автоматизації, для методу візуалізації хмар точок «Web Point Cloud Viewer». Розроблено структуру методу візуалізації хмар точок та сформовано функціональну діаграму користувача.

При створенні методу візуалізації хмар точок «Web Point Cloud Viewer» було розроблено програмні модулі та детально описані алгоритмічні конструкції. Описано функціональне призначення програмних модулів та їх взаємозв’язок між собою. Розроблено механізм авторизації користувача методу візуалізації хмар точок «Web Point Cloud Viewer». Сформовано алгоритми для аналізу та відображення хмар точок.

Проведено експериментальну апробацію результатів роботи реалізованого методу візуалізації хмар точок «Web Point Cloud Viewer». Для прикладу було проведено апробацію швидкодії методу, а саме швидкодії відображення хмари точок та швидкодію відклику методу на дії користувача. Для ефективної перевірки швидкодії відображення хмари точок апробація була проведена на декількох хмарах точок різних форматів та з різною кількістю точок. В результаті було отримано висновок, що метод візуалізації хмар точок «Web Point Cloud Viewer» є швидшим ніж інші системи предметної області.

Було перевірено основні функції методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень. Результати перевірки засвідчили, що розроблений метод забезпечує виконання усіх описаних функцій та повністю відповідає поставленим вимогам, що доводить його спроможність розв’язувати поставлені задачі.

Перелік посилань

1. Хмара точок [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Хмара_точок
2. 3D-сканер [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/3D-сканер
3. Фотограметрія [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/photogrammetry
4. Блог NavVis [Електронний ресурс]. – Режим доступу: https://www.navvis.com/blog/everything-you-need-to-know-about-point-clouds-navvis
5. BIM [Електронний ресурс]. – Режим доступу: https://www.autodesk.com/solutions/bim
6. ReCap [Електронний ресурс]. – Режим доступу: https://www.autodesk.com/products/recap/overview
7. Laser scanning [Електронний ресурс]. – Режим доступу: https://new.certainty3d.com/blog/what-is-laser-scanning-and-how-can-it-be-used/
8. GPS [Електронний ресурс]. – Режим доступу: https://www.geotab.com/blog/what-is-gps/
9. IMU [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/what-is-imu-9565e55b44c
10. Potree [Електронний ресурс]. – Режим доступу: http://potree.org/
11. Lidarview [Електронний ресурс]. – Режим доступу: http://lidarview.com/
12. PCD file format [Електронний ресурс]. – Режим доступу: http://pointclouds.org/documentation/tutorials/pcd_file_format.html

13. PCD [Електронний ресурс]. – Режим доступу: https://docs.safe.com/fme/html/FME_Desktop_Documentation/FME_ReadersWriters/pcd/pcd.htm
14. Las specification [Електронний ресурс]. – Режим доступу: https://www.loc.gov/preservation/digital/formats/fdd/fdd000418.shtml
15. LAZ specification [Електронний ресурс]. – Режим доступу: https://filext.com/file-extension/LAZ
16. PTS specification [Електронний ресурс]. – Режим доступу: http://paulbourke.net/dataformats/pts
17. PTX file format [Електронний ресурс]. – Режим доступу: https://wiki.photoneo.com/index.php/PTX_file_format
18. XYZ specification [Електронний ресурс]. – Режим доступу: https://abaqus-docs.mit.edu/2017/English/SIMACAECAERefMap/simacae-c-fldusingmappointxyz.htm
19. E57 specification [Електронний ресурс]. – Режим доступу: http://paulbourke.net/dataformats/e57/2011-huber-e57-v3.pdf
20. JSON [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/js/js_json_intro.asp
21. Autodesk Forge [Електронний ресурс]. – Режим доступу: https://forge.autodesk.com/
22. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language 7th Edition by David Flanagan. – 3-4с.
23. C# 10 and .NET 6 – Modern Cross-Platform Development: Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code, 6th Edition 6th Edition, Kindle Edition. – 5-6с.
24. Angular Up and Running: Learning Angular, Step by Step Taschenbuch – 22. Juni 2018 von Shyam Seshadri. – 7с.

25. ASP.NET Core 5 Secure Coding Cookbook: Practical recipes for tackling vulnerabilities in your ASP.NET web applications by Roman Canlas, Ed Price (Foreword). – 8p.
26. Three.js [Електронний ресурс]. – Режим доступу: https://threejs.org/
27. INFORMATION TECHNOLOGY OF MAKING CONTROLLED CRITICALLY SAFE DECISIONS WHEN VIEWING POINT CLOUDS “WEB POINT CLOUD VIEWER” Nazar Mykolayovych Trostynskyi, Oleksandr Anatoliiovych Pasichnyk, Tetiana Kazymyrivna Skrypnyk, Eduard Andriiovych Manziuk [Електронний ресурс]. – Режим доступу: http://journals.khnu.km.ua/vestnik/wp-content/uploads/2021/10/299-text_2021_4_t-11-17.pdf
28. Сучасний рух науки: тези доп. XII міжнародної науково-практичної інтернет-конференції, 1-2 квітня 2021 р. – Дніпро, Україна, 2021. – Т.2. – 512 с.
29. Plas.io [Електронний ресурс]. – Режим доступу: https://plas.io/
30. C# 8.0 in a Nutshell: The Definitive Reference 1st Edition by Joseph Albahari, Eric Johannsen, 2020. – 55p.
31. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code, 4th Edition, 2019. – 113p.
32. Clean Architecture: A Craftsman's Guide to Software Structure and Design: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series) Taschenbuch – Illustriert, 17. September 2017 Englisch Ausgabe von Robert C. Martin. – 238 p.
33. Javascript: The Definitive Guide: Master the World's Most-Used Programming Language Taschenbuch – 9. Juni 2020, Englisch Ausgabe von David Flanagan. – 81 p.

34. Eloquent JavaScript, 3rd Edition: A Modern Introduction to Programming Taschenbuch – Illustreited, 4. December 2018by Marijn Haverbeke. - 221 p.
35. M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, “The Digital Michelangelo Project: 3D Scanning of Large Statues,” in Proceedings of ACM SIGGRAPH, 2000, pp. 131-144.
36. Visual Studio 2019 In Depth: Discover and make use of the powerful features of the Visual Studio 2019 IDE to develop better and faster mobile, web, and desktop applications by Ockert J. du Preez. – 369-385 p.

Додатки

Додаток А

Ксерокопії наукових публікацій, виконаних при роботі над кваліфікаційною роботою магістра

КОМП'ЮТЕРНІ НАУКИ, ІНФОРМАЦІЙНІ ТЕХНОЛОГІ, СИСТЕМНИЙ АНАЛІЗ ТА КІБЕРБЕЗПЕКА

DOI 10.31891/2307-5732-2021-297-3-11-**Помилка! Закладку не визначено.**
УДК 004.5.02

NAZAR MYKOLAYOVYCH TROSTYNSKYI
Khmelnytskyi National University
ORCID ID: 0000-0002-2271-7526
e-mail: trost1999@ukr.net
OLEKSANDR ANATOLIIOVYCH PASICHNYK
Khmelnytskyi National University
ORCID ID: 0000-0002-8760-4688
e-mail: o.a.pasichnyk@gmail.com
TETIANA KAZYMYRIVNA SKRYPNYK
Khmelnytskyi National University
ORCID ID: 0000-0002-8531-5348
e-mail: marine_1996@ukr.net
EDUARD ANDRIIOVYCH MANZIUK
Khmelnytskyi National University
ORCID ID: 0000-0002-7310-2126
e-mail: eduard.em.km@gmail.com

INFORMATION TECHNOLOGY OF MAKING CONTROLLED CRITICALLY SAFE DECISIONS WHEN VIEWING POINT CLOUDS "WEB POINT CLOUD VIEWER"

The developed software product performs the following functions: work with cloud storage, work with camera, work with stage, work with model measurement, work with models and work with point clouds. The application of such a system is designed to automate and improve the end user's work with point clouds, namely their storage in the Autodesk cloud storage, display and manipulation. Interaction with such a system does not require high system requirements of hardware and software. From the obtained results it is possible to draw a constructive conclusion that the functional direction of the complex algorithmic model fully meets the requirements of the task.

Keywords: point clouds, visualization systems, laser scanner, photogrammetry, 3D coordinate systems, Autodesk, Autodesk ReCap.

Н.М. ТРОСТИНСЬКИЙ, О.А. ПАСІЧНИК, Т.К. СКРИПНИК, Е.А. МАНЗЮК
Хмельницький національний університет

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОПТИМІЗАЦІЇ КОНТРОЛЬОВАНИХ ЛЮДИНОЮ КРИТИЧНО-БЕЗПЕКОВИХ РІШЕНЬ ПРИ ПЕРЕГЛЯДІ ХМАР ТОЧОК "WEB POINT CLOUD VIEWER"

Розроблений програмний продукт виконує такі функції: робота з хмарним сховищем, робота з камерою, робота зі сценою, робота з вимірюванням моделей, робота з моделями та робота з хмарами точок. Застосування такої системи призначене для автоматизації та покращення роботи кінцевого користувача з хмарами точок, а саме їх зберігання у хмарному сховищі, відображенні та маніпулюванні Autodesk. Взаємодія з такою системою не вимагає високих системних вимог до апаратного та програмного забезпечення. З отриманих результатів можна зробити конструктивний висновок, що функціональний напрямок складної алгоритмічної моделі повністю відповідає вимогам завдання.

Ключові слова: точкові хмари, системи візуалізації, лазерний сканер, фотограмметрія, 3D системи координат, Autodesk, Autodesk ReCap.

FORMULATION OF THE PROBLEM

The cloud of points obtained as a result of laser scanning is the primary result that requires high-quality processing. Further work consists in solving two main and sometimes mutually exclusive problems - increasing the accuracy of surface reproduction, which is solved by increasing the points in the cloud, and providing the necessary speed, with a diametrically opposite solution.

The development and implementation of appropriate efficient information technologies using appropriate modern development tools allow the implementation of systems that simultaneously provide high scanning accuracy combined with acceptable speed, allow you to operate large clouds of points with acceptable lead times.

ANALYSIS OF RECENT RESEARCH

Currently, there are quite a few desktop applications for working with point clouds, first of all, it's Autodesk ReCap (Figure 1). This application allows you to create 3D models from photos or laser scanning with excellent accuracy and efficiency, as well as accurately create 3D clouds or grids for further design in Autodesk tools [2], such as Revit, Civil 3D, AutoCAD, Navisworks and more. Recap provides an opportunity before indexing the project and its completion in a single cloud of points, to apply various settings using a set of filters.

Figure 1. Autodesk ReCap

ReCap determines how to exclude aggressively rejected points from an imported scan file, that is, points that do not belong directly to the object. The ReCap toolkit allows you to measure, mark point clouds and share them with collaborators.

In addition, because points are generated by your own Autodesk product, points can be imported into all other Autodesk products. You can use the ReCap dot file to clean up a scan of an existing building and then import it into Revit to start the exact 3D BIM design [3]. You can also import a cleaned ReCap cloud in Civil 3D [4].

The main disadvantage of ReCap is its desktop and paid implementation.

The current global trend in the development of modern information technology is their Internet orientation, which consists in the implementation of software in the form of web applications with obvious conveniences and benefits.

The Potree web application [4] is a free open source cloud point rendering created at the Institute of Computer Graphics and Algorithms, TU Wien [6] (Figure 2).

This technology is easily integrated into the mechanical engineering and manufacturing industries. It is possible to capture the reality of any existing part, such as a pipe flange, to which you want to connect, but has no design parameters. With this technology, you can impose a new part according to the size, placement of holes in the bolts with tolerances.

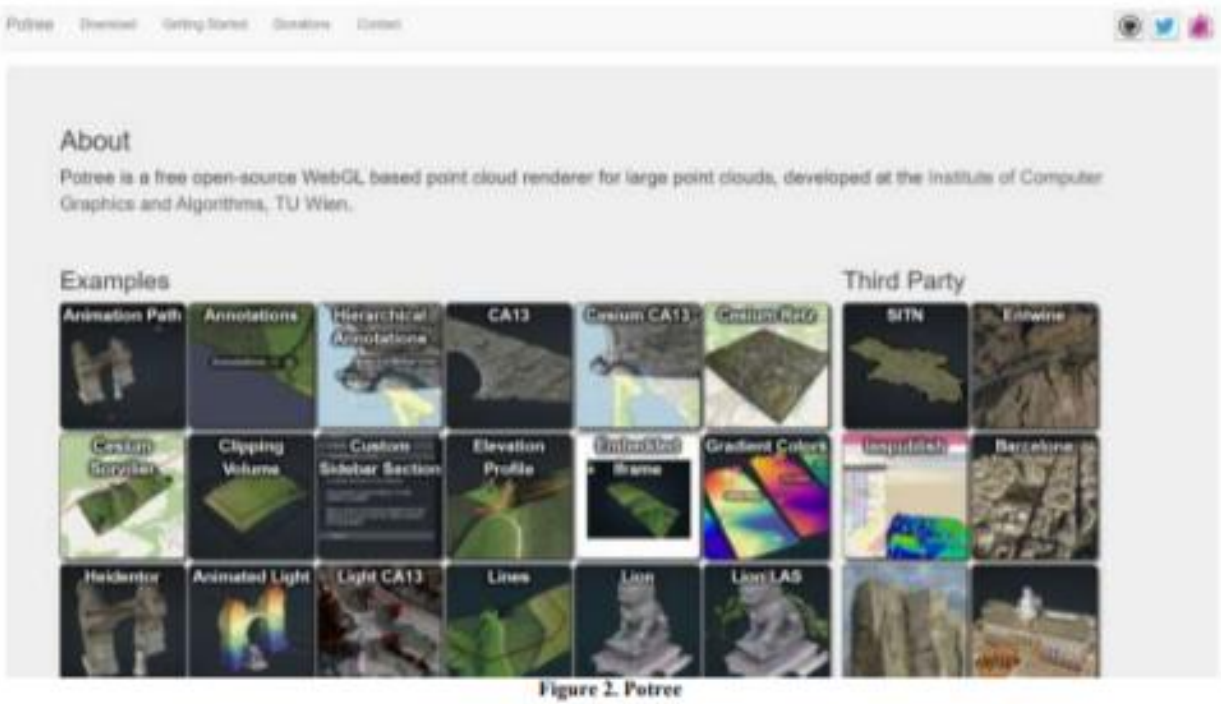


Figure 2. Potree

This web application provides opportunities directly for point cloud rendering, as well as additional features for measuring geometric parameters.

The main advantage of this web application is its open source.

Another example of specialized sites for viewing point clouds is the site Online LIDAR point cloud viewer [7] (Figure 3). This site also provides the ability to render a cloud of points with a set of additional features.

The main disadvantage of this site is the support of only two data formats LAS 1.2 [8] and XYZ [9].

Figure 3. Online Lidar point cloud viewer

The analysis of literature sources shows a significant and growing need for means of identification of remote users. This is due to the rapid development and widespread introduction of information technology in various spheres of human life. The process of informatization and digitalization of society creates opportunities for raising the standard of living of people to a qualitatively new level, and creates a whole range of challenges and threats associated with various types of misuse of data. A prerequisite for abuses in the information sphere is the acquisition of unauthorized access by attackers, which determines the issue of identification of remote users as a priority.

In modern systems, various access control systems are implemented from hardware, hardware-software, software-hardware to purely software that implement various technical solutions and software algorithms.

In the case of information systems and information technologies, the basic tools of these issues are the use of appropriate algorithms for identifying remote users. These algorithms should provide reliable protection against malicious intrusion, and on the other - to provide comfortable access to appropriate users. One of the most effective are technologies that implement the concept of "zero knowledge". However, the application of this technology requires a solution to the question of the level of computing power.

The purpose of the research: as a result of the analysis of existing approaches the following tasks of research which purpose is realization of the viewer of clouds of points "Web Point Cloud Viewer" with a sufficient level of accuracy and speed of computing ability and availability are formulated:

1. To analyze the existing methods, technologies and solutions of methods of information system of visualization of objects as clouds of points;
2. Improving existing methods of visualization of objects as a cloud of points in the direction of improving accuracy and speed;
3. To develop information system for visualization of objects as a cloud of user identification points using the obtained models and methods;
4. Perform an experimental test of the information system for visualization of objects as a cloud of points.

PRESENTATION OF THE MAIN MATERIAL

In modern production, the wide expansion contains models that are stored in the form of a cloud point. A point cloud is a set of data about points in a real coordinate system [10].

In a three-dimensional coordinate system, points define the coordinates X, Y, and Z, and are most often defined to represent the outer zone of an object.

The point cloud can be created by a 3D scanner [11]. Such devices form a large number of points in the objects with the subsequent presentation of information about the cloud of points in the data files.

Capturing individual unrelated points is the key to using a cloud point. Recovered point clouds are easy to edit, display and filter. The computer should not worry about the scale or ask for points, limit the position and capabilities, color. This makes them a convenient way to store large amounts of detailed data.

Although point clouds can be used for rendering and validation, they do not need to be supplemented for

use in the context of 3D applications. Therefore, they are usually retold in models with a polygonal or triangular grid, in the NURBS model or CAD model through the process of so-called system reconstruction. It is not uncommon for a scan of a single area, such as a city block or airfield, to contain billions of data points.

There are many methods of converting a cloud of points into a 3D surface. In particular, in [12] a comparative table with 35 methods of converting information from a cloud of points is given. Some approximations, such as the Delaunay triangulation, the Alpha form, and the rotary crop method, will construct a triangular or polygonal grid for already existing vertices of the point cloud, while other approximations will construct long-range volumetric tables or reconstruct uncertain confidentiality using a cropping algorithm.

One of the industries where point clouds are used is industrial metrology and quality control using industrial computed tomography. The cloud of points obtained as a result of three-dimensional scanning of the finished industrial production is compared with the corresponding CAD-model of this production or other (reference) chromatography, which provides a difference between the projects and the actual parameters. These differences can be reflected in the form of color maps, in those places and areas that deviate between the actual surface and the formal model can be automatically highlighted by certain indicators. Geometric dimensions and additional dimensions can also be used using a point cloud.

Dot clouds can be used to represent and visualize volumetric data, for example, in the field of medical imaging. Due to the use of cloud points in such tasks, data animation and stimulation are achieved.

In geographic information systems, point clouds are one of the sources used to create a digital terrain model. Point clouds are also used to create digital models of urban areas.

To design the structure of the web application "Web Point Cloud Viewer" provides analysis and automation of information flow processing. Specialized software is developed for users of the system; its functionality must meet all the necessary conditions for processing account data. Business processes in the work of the user of the system can be divided into the following groups:

1. Business process "Using cloud storage" (Figure 4):
- creating a new user;
- user login;
- user logout;
- creation of new folders;
- loading point clouds into the storage;
- loading point clouds from storage.

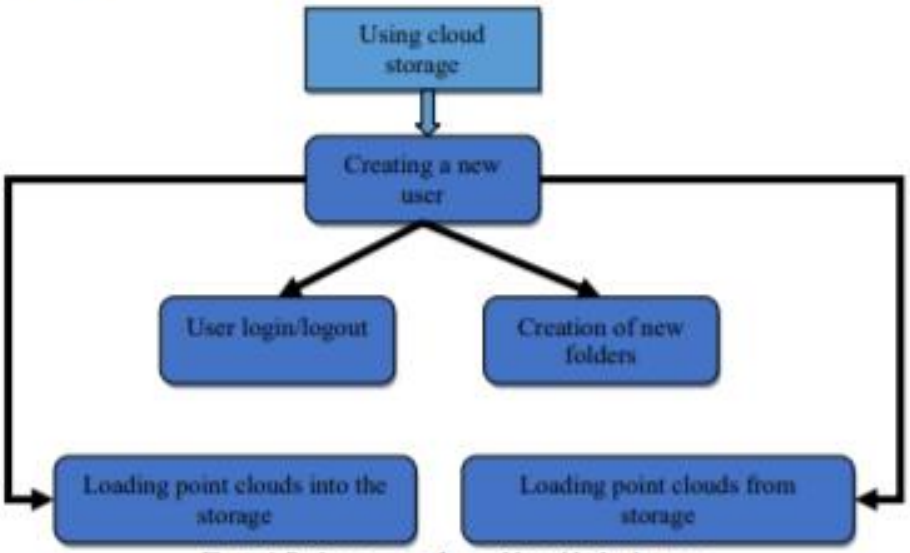


Figure 4. Business process for working with cloud storage

2. Business process "Working with the camera" (Figure 5):
- change of the camera inspection sector;
- change the type of camera;
- change the position of the camera;
- return the camera to its original position.

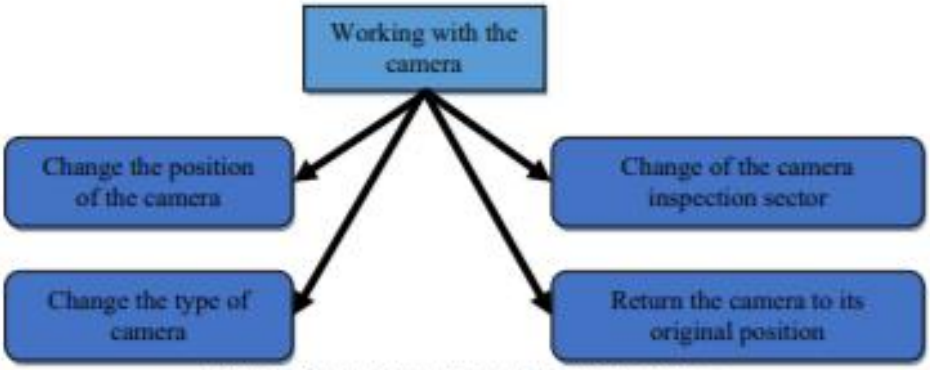


Figure 5. Business process for working with the camera

3. Business process "Working with the scene" (Figure 6):
- showing the model from different angles;
- taking a picture of the scene;
- change the color background of the scene;
- support for the list of keyboard shortcuts;
- display of the global coordinate system of the model.

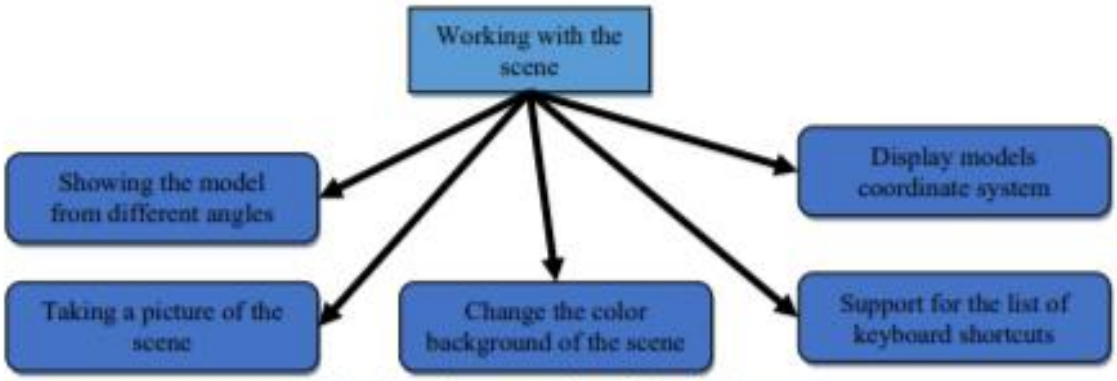


Figure 6. Business process for working with the scene

4. Business process "Working with the model" (Figure 7):
- modification of the size of points;
- color change of model points;
- return to the original color.

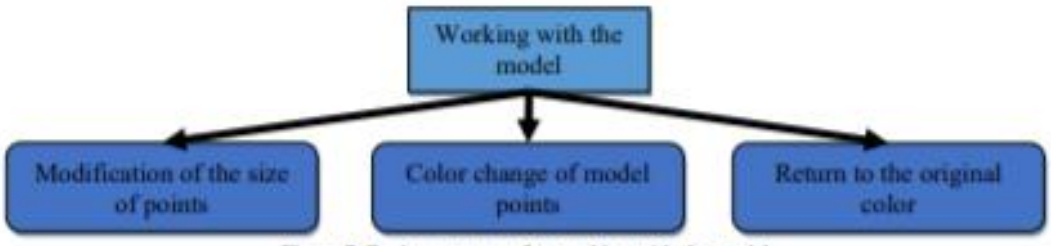


Figure 7. Business process for working with the model

5. Business process "Working with model measurement" (Figure 8):
- measuring the position of the point;
- measuring the distance between two points;
- measuring the distance between many points;
- measuring the angle between points;
- area measurement.

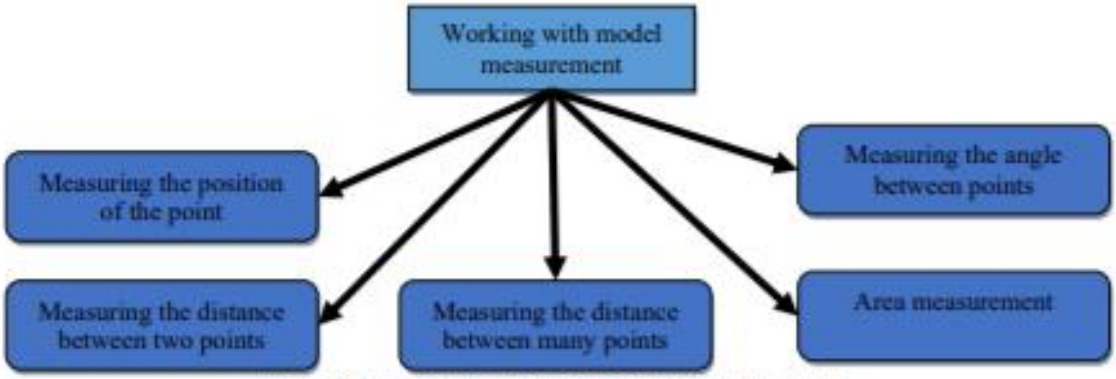


Figure 8. Business process for working with model measurement

6. Business process "Working with point clouds" (Figure 9):
- file extension recognition;
- processing and reading binary files;
- processing and reading files in ASCII format;
- model preview.

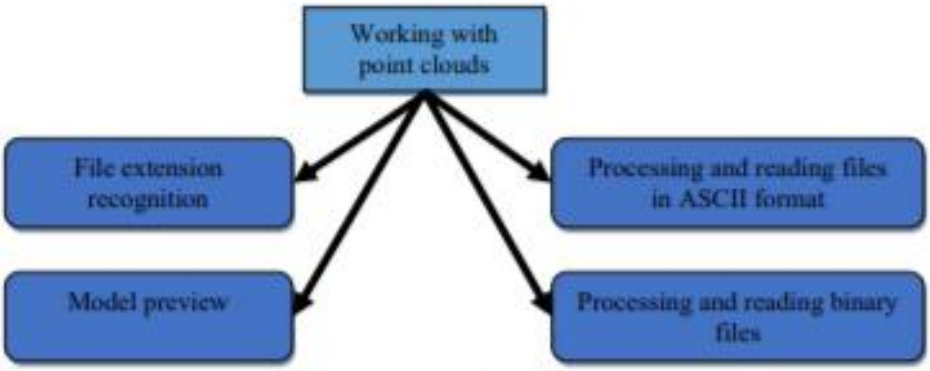


Figure 9. Business process for working with point clouds

Business process for user work with cloud storage. This business process provides the user with the ability to create an account, log in to an account, log out of a personal account, upload point clouds to the repository, and upload point clouds to a local computer.

Business process for working with the camera. This business process provides the user of the IP with the ability to change the viewing sector of the camera, change the position of the camera, change the type of camera (perspective orthogonal) and return the camera to its original position.

Business process for the user's work with the scene. This business process allows the user of the IP to change the side from which the model is displayed, take and load a scene image, change the background color of the scene, dynamically show the direction of the model using a global coordinate system. In addition, this business process provides the ability to support the execution of commands using hotkeys, as well as displays a list of them.

Business process for user work with the model. This business process allows the user of the IP to modify the colors and sizes of the model points and return the sizes and colors of the points to the default state.

Business process for user work with model measurement. This business process allows the user of the IP to determine the position of the point, measure the distance and angle between the points, as well as measure the area.

Business process for user work with point clouds. This business process provides the user of the IP the ability to process and retrieve data from binary files as well as files in ASCII format. This business process also determines the file extension and allows you to view a preview of the model.

In order for a web application user to be able to work with personal account data, he must perform the required set of operations for authorization.

To start working with the connected cloud storage, the user must log in to the personal account of Autodesk. If an error message was received while trying to log in to your account, it means that the IP user entered incorrect data. If the login and password were entered correctly, the software application will have to grant permission to access user data from Autodesk servers.

On its account page, the user deals with the following functionality: working with the cloud storage (creating a new user, user login, user login, creating new folders, loading point clouds into the repository, loading point clouds from the repository), working with the camera, change camera view, change camera position, return camera to starting position), work with scene (show model from different sides, create scene snapshot, change scene background color, display global model coordinate system, work with hot keys), work with model (modification of point size, coloring of model points, return of original color), work with model measurement (measuring point

position, measuring distance between two points, measuring distance between many points, measuring angle between points, measuring area) and working directly with point cloud (file extension recognition, processing and reading binaries and files in for mother ASCII, model preview).

Thus, after analyzing the processing of information flows, for the web application "Web Point Cloud Viewer" was defined a complete list of business processes that are subject to automation algorithms. With an organized functional diagram of the end user and object-oriented feature groups, you can develop an information system structure that will be designed to view, manipulate, and modify point clouds, as well as to store documents from an Autodesk account.

CONCLUSION

The developed software product perfomns the following functions: work with cloud storage, work with camera, work with stage, work with model measurement, work with models and work with point clouds.

The application of such a system is designed to automate and improve the end user's work with point clouds, namely their storage in the Autodesk cloud storage, display and manipulation. Interaction with such a system does not require high system requirements of hardware and software.

From the obtained results it is possible to draw a constructive conclusion that the functional direction of the complex algorithmic model fully meets the requirements of the task.

References

1. Everything you need to know about point clouds. URL: https://www.navvis.com/blog/everything-you-need-to-know-about-point-clouds-navvis

2. Autodesk. URL: https://www.autodesk.com/

3. Autodesk BIM. URL: https://www.autodesk.com/solutions/bim

4. Autodesk Civil 3D. URL: https://www.autodesk.ru/products/civil-3d/overview

5. Potree. URL: http://potree.org/

6. Tuwien. URL: https://www.tuwien.at/

7. Lidarview. URL: http://lidarview.com/

8. LAS specification The American Society for Photogrammetry & Remote Sensing. 2013. P. 10–13. URL: https://www.asprs.org/wp-content/uploads/2010/12/LAS_1_4_r13.pdf

9. XYZ specification. URL: https://www.adoclib.com/blog/point-cloud-xyz-format-specification.html

10. Point clouds. URL: https://uk.wikipedia.org/wiki/Хмара_точок

11. 3D laser scanner. URL: https://uk.wikipedia.org/wiki/3D-сканер

12. Marcel Berger articles. URL: https://en.wikipedia.org/wiki/Marcel_Berger

Додаток Б

Ксерокопії наукових доповідей, виконаних при роботі над кваліфікаційною роботою магістра

Editorial board of International Electronic Scientific and Practical Journal «WayScience»

Сучасний рух науки: тези доп. XII міжнародної науково-практичної інтернет-конференції, 1-2 квітня 2021 р. – Дніпро, Україна, 2021. – Т.2. – 512 с.

(Modern Movement of Science: abstracts of the 12th International Scientific and Practical Internet Conference, April 1-2, 2021. – Dnipro, Ukraine, 2021. – P.2. – 512 p.)

12th International Scientific and Practical Internet Conference "Modern Movement of Science" is devoted to the main mission of the International Electronic Scientific and Practical Journal "WayScience" - to pave the way for development of modern science from idea to result.

Topics cover all sections of the International Electronic Scientific and Practical Journal "WayScience", namely:

- public administration;
- philosophical sciences;
- economic sciences;
- historical sciences;
- legal sciences;
- agricultural sciences;
- geographic sciences;
- pedagogical sciences;
- psychological sciences;
- sociological sciences;
- political sciences;
- philological sciences;
- technical sciences;
- medical sciences;
- chemical sciences;
- biological sciences;
- physical and mathematical sciences;
- other professional sciences.

Dnipro, Ukraine – 2021

401

Technical sciences

ПЕРЕГЛЯДАЧ ХМАР ТОЧОК

Тростинський Назар Миколайович
магістр
Пасічник Олександр Анатолійович
к.т.н., доцент (o.a.pasichnyk@gmail.com)
Хмельницький національний університет, м. Хмельницький, Україна

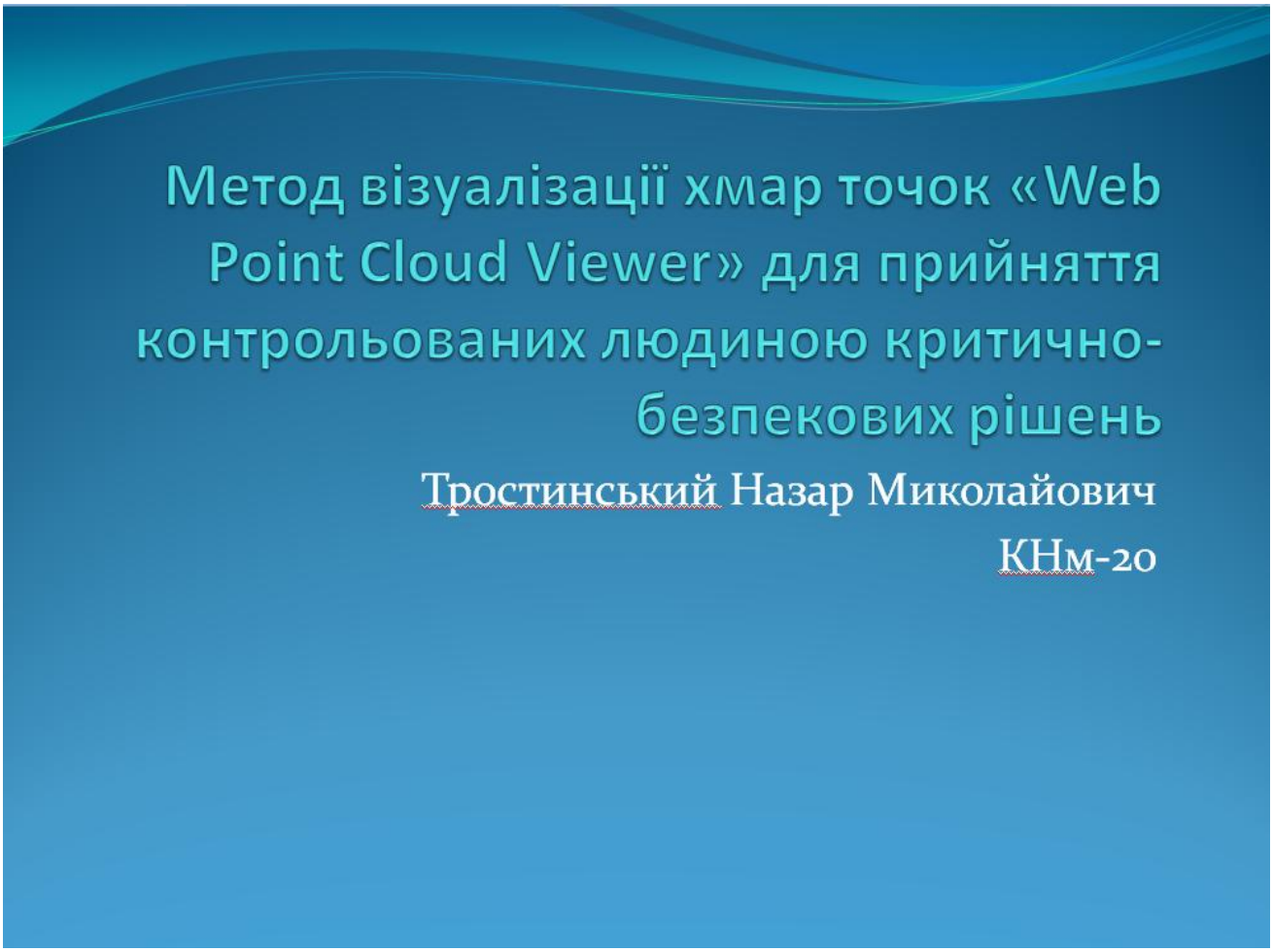
В сучасних технологіях інформація про об'єкти, зокрема їх поверхні, у вигляді хмари точок широко застосовуються у різноманітних сферах діяльності людини - як результат дистанційного зондування Землі, в системах автоматизованого проектування, при тривимірній цифровій реконструкції будівель, до прикладу пам'яток архітектури, та приміщень, тощо. Найбільш ефективною технологією отримання необхідної хмари точок є лазерне сканування. Це обумовлене стрімким розвитком лазерної техніки та інформаційно-комп'ютерних технологій що супроводжується покращенням технічних характеристик, з одного боку, та зниженням вартості – з іншого. Суттєве та стрімке покращення техніко-економічних показників систем тривимірного лазерного сканування обумовлює їх розширену доступність за запровадження у найрізноманітніших галузях. Хмара точок, отримана в результаті лазерного сканування, є первинним результатом, що потребує якісного опрацювання. Подальша робота полягає у вирішенні двох головних й, іноді, взаємовиключних задач – підвищення точності відтворення поверхонь, що вирішується шляхом збільшенням точок у хмарі, та забезпечення необхідної швидкодії, з діаметрально протилежним варіантом розв'язання. Використання для зберігання даних хмарних сховищ зменшує витрати кінцевого користувача на зберігання даних, а відмова від реалізації десктопного додатку на користь веб-застосунку сприяє покращенню доступності. Реалізація підтримки мультиформатного режиму роботи з вхідними даним створює можливості роботи з різноманітними програмними додатками широкого кола розробників програмного забезпечення та технічними пристроями різної конструкції та принципів дії як джерелами початкових даних.

Мета роботи полягає у реалізації переглядача хмар точок «3D Viewer for point clouds» з підтримкою мультиформатного режиму роботи з покращеним рівнем доступності із вирішенням часткових поставлених задач: - провести аналіз існуючих методів, технологій та рішень методів інформаційної системи візуаліції об'єктів як хмари точок з підтримкою мультиформатного режиму роботи з даними; удосконалення існуючих методів візуаліції об'єктів як хмари точок; розробити інформаційне та програмне забезпечення інформаційної системи візуаліції об'єктів як хмари із забезпеченням покращеного рівня доступності за допомогою отриманих моделей та методів; виконати експериментальну перевірку інформаційної системи візуаліції об'єктів як хмари точок.

В роботі удосконалено інформаційну технологію роботи з хмарами точок як представленням тривимірних об'єктів шляхом реалізації підтримки мультиформатного режиму роботи з вхідними даними та у поєднанні хмарних технологій з реалізацією системи як веб-застосунку. Практичне значення отриманих результатів полягає у програмній реалізації переглядача хмар точок «3D Viewer for point clouds» який підтримує мультиформатний режим доступу до даних із покращеним рівнем доступності для кінцевого користувача з використанням хмарних сховищ для зберігання даних, з покращеним рівнем доступності для кінцевого користувача за рахунок відмови від реалізації десктопного додатку на користь веб-застосунку, а реалізація підтримки мультиформатного режиму роботи з вхідними даним забезпечила можливість роботи з різноманітними програмними додатками широкого кола розробників програмного забезпечення та технічними пристроями різної конструкції та принципів дії як джерелами початкових даних.

Додаток В

Презентація



Вступ

- В сучасних технологіях інформація про об'єкти, зокрема їх поверхні, у вигляді хмари точок широко застосовуються у різноманітних сферах діяльності людини - як результат дистанційного зонування Землі, в системах автоматизованого проектування, при тривимірній цифровій реконструкції будівель, до прикладу пам'яток архітектури, та приміщень, тощо.
- У сучасному світі хмари точок використовуються досить широко через те, що ці набори даних пропонують точне та всеосяжне цифрове зображення реального простору, поверхні чи об'єкта. А це означає, що вони пропонують величезну цінність для широкого кола застосувань.

Мета роботи

- Мета роботи полягає у реалізації методу «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень з достатнім рівнем точності та швидкодії, обчислювальної здатності та доступності.

Завдання

- провести аналіз існуючих методів, технологій та рішень візуалізації об'єктів як хмари точок;
- удосконалити існуючі методи візуалізації об'єктів як хмари точок у напрямку покращення швидкодії;
- розробити метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень;
- виконати експериментальну перевірку методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень.

Об'єкт та предмет

- **Об'єктом роботи** є процес візуалізації об'єктів як хмари точок з використанням інформаційних технологій.
- **Предметом роботи** є моделі, підходи та засоби візуалізації об'єктів як хмари точок.

Наукова новизна одержаних результатів

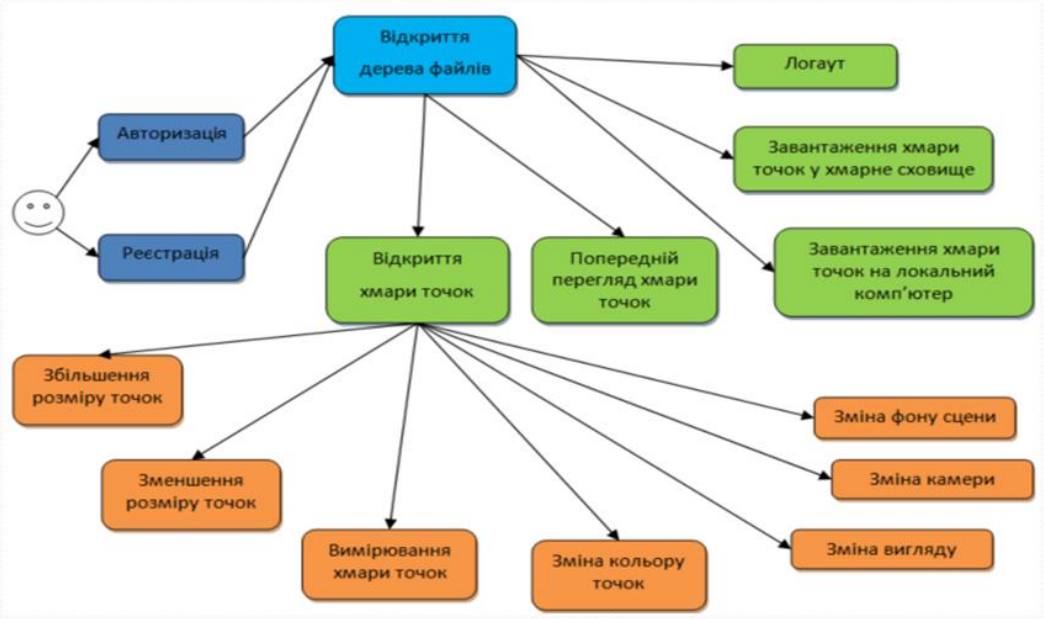
В результаті проведеної роботи були отримані наступні результати:

- Удосконалено існуючий метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень, що дозволило підвищити швидкодію, обчислювальну здатність та доступність.

Практичне значення одержаних результатів

- В результаті виконання дипломної роботи магістра розроблено відповідне експериментальне програмне забезпечення, яке підтвердило вірність запропонованих положень. Застосування методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень дає можливість здійснювати швидкий перегляд тривимірних об'єктів із використанням малопотужних технічних засобів за наявності універсального, вбудованого програмного забезпечення.

Функціональна діаграма користувача



Архітектура методу

Клієнтська частина

AngularJS, JavaScript, Three.js

- Побудова дерева файлів
- Парсинг, відображення, маніпуляції хмари точок
- Робота з хмарним сховищем

Серверна частина

ASP.NET MVC C#

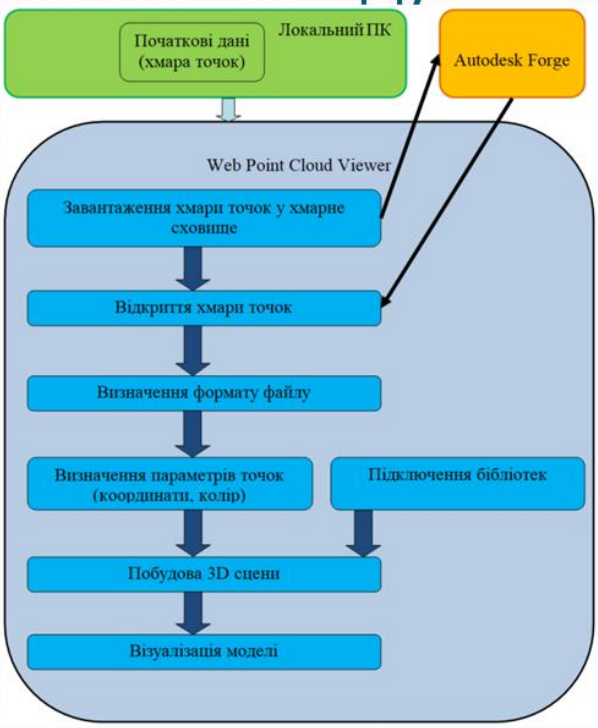
PointCloudWebViewer

- Обробка клієнтських запитів
- Геренація дерева файлів для клієнта
- Отримання токена

AutodeskForgeProvider

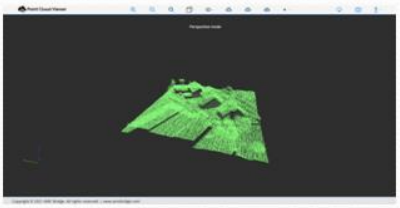
- Обробка запитів
- Завантаження моделей

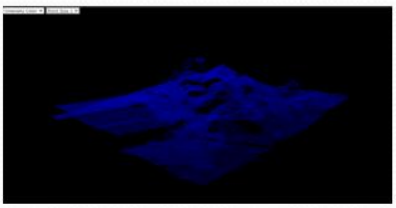
Схема роботи методу



Апробація швидкодії відображення

На прикладі моделей france.las , pump_000004_big.xyz та Townhall_Ok.xyz було порівняно час відкриття хмар точок з веб-застосунком Online LIDAR point cloud viewer.
Хмару точок Townhall_Ok.xyz неможливо відкрити за допомогою Online LIDAR point cloud viewer.

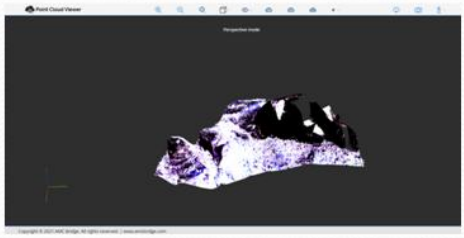




Час	Web Point Cloud Viewer	Online LIDAR
france.las	1.75 с	2.65 с
pump_000004_big.xyz	10.85 с	11.51 с
Townhall_Ok.xyz	1.93 с	-

Апробація швидкодії відклику

Для ефективної перевірки швидкодії відклику методу візуалізації хмар точок «Web Point Cloud Viewer» на дії користувача буде використано велику хмару точок, яка містить близько восьми мільйонів точок. Порівняння буде проведено на прикладі хмари точок mountain.laz та із застосунком plas.io.



Час	Web Point Cloud Viewer	plas.io
Зум камери	0.92 с	0.87 с
Зміна розміру точок	0.12 с	0.17 с
Зміна положення камери	0.34 с	0.61 с

Висновки

- В рамках кваліфікаційної роботи магістра було проведено аналіз літературних джерел, що засвідчило суттєву та зростаючу потребу у візуалізації тривимірних об'єктів, з високим рівнем коректності відтворення таких об'єктів у поєднанні з невисокими вимогами до технічних засобів із використанням універсального вбудованого в операційну систему програмного забезпечення.

  В результаті проведеного аналізу існуючих підходів сформульовано та вирішено такі завдання як:
- Проведено аналіз існуючих технологій, методів та рішень для відображення хмар точок;
- Удосконалено існуючі методи по роботі з хмарами точок у напрямку покращення швидкодії та доступності для кінцевих користувачів;
- Реалізовано метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень;
- Виконано експериментальну перевірку методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень.

Додаток Г

Програмні коди

Програмний код є власністю AMC Bridge

Лістинг Request.cs:

```
using AutodeskForgeProvider.Authentication.Models;
using System;
using System.Net;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;

namespace AutodeskForgeProvider.Helpers.Network
{
    public class Request
    {
        private readonly Uri RequestUri;
        private readonly WebClient webClient = new WebClient();

        public Request(string requestUri)
        {
            RequestUri = new Uri(requestUri);
        }

        public Request(string requestUri, params string[] args) : this(string.Format(requestUri, args))
        {
        }

        public Request(Token token, string requestUri)
        {
            RequestUri = new Uri(requestUri);
            AddHeader(HttpRequestHeader.Authorization, String.Format("{0} {1}", token.TokenType, token.AccessToken));
        }

        public Request(Token token, string requestUri, params string[] args) : this(token, string.Format(requestUri, args))
        {
        }

        public void AddHeader(HttpRequestHeader type, string value)
        {
            switch (type)
            {
               case HttpRequestHeader.ContentLength:
                    {
                        break;
                    }
                case HttpRequestHeader.ContentType:
                default:
                    {
                        webClient.Headers.Add(type, value);
                        break;
                    }
            }
        }

        public async Task<byte[]> Get()
        {
            try
            {
                return await webClient.DownloadDataTaskAsync(RequestUri);
            }
            catch (WebException webException)
            {
                return GetExceptionInfo(webException);
            }
        }

        public async Task<byte[]> Post(byte[] data)
        {
            try
            {
                return await webClient.UploadDataTaskAsync(RequestUri, "POST", data);
            }
            catch (WebException webException)
            {
                return GetExceptionInfo(webException);
            }

        }

        public async Task<byte[]> Post(string data)
        {
            try
            {
                return await this.Post(Encoding.UTF8.GetBytes(data));
            }
            catch (WebException webException)
            {
                return GetExceptionInfo(webException);
            }
        }


        public async Task<byte[]> Put(byte[] data)
        {
            try
            {
                return await webClient.UploadDataTaskAsync(RequestUri, "PUT", data);
            }
            catch (WebException webException)
            {
                return GetExceptionInfo(webException);
            }
        }

        public async Task<byte[]> Patch(byte[] data)
        {
            try
            {
                return await webClient.UploadDataTaskAsync(RequestUri, "PATCH", data);
            }
            catch (WebException webException)
            {
                return GetExceptionInfo(webException);
            }
        }

        public async Task<byte[]> Patch(string data)
        {
            try
            {
                return await Patch(Encoding.UTF8.GetBytes(data));
            }
            catch (WebException webException)
            {
                return GetExceptionInfo(webException);
            }
        }


        private byte[] GetExceptionInfo(WebException webException)
        {
            if (webException.Status == WebExceptionStatus.ProtocolError)
            {
                var response = webException.Response as HttpWebResponse;
                var stream = response.GetResponseStream();
                byte[] buffer = new byte[response.ContentLength];
                stream.Read(buffer, 0, (int)response.ContentLength);
                return buffer;
            }
            else
            {
                return Encoding.Default.GetBytes(webException.Message);
            }
        }

        private HttpStatusCode GetStatusCode()
        {
            FieldInfo responseField = webClient.GetType().GetField("m_WebResponse", BindingFlags.Instance | BindingFlags.NonPublic);

            if (responseField != null)
            {
                HttpWebResponse response = responseField.GetValue(webClient) as HttpWebResponse;
                if (response != null)
                {
                    return response.StatusCode;
                }
            }
            return 0;
        }
    }
}
```

Лістинг E57Converter.cpp:

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>
#include "E57Foundation.h"
#include "E57Converter.h"

using namespace e57;
using namespace std;

//outputFullFileName should be without extention
void E57Converter::ConvertToPts(const char *inputFullFileName, const char *outputFullFileName)
{
        try {
                string inputFullFileName(inputFullFileName);
```

```
                string outputFullFileName(outputFullFileName);

                ImageFile inf(inputFullFileName, "r");
                StructureNode root = inf.root();

                if (!root.isDefined("/data3D")) {
                        cout << "File does not contain 3D images" << endl;
                }
                Node n = root.get("/data3D");
                if (n.type() != E57_VECTOR) {
                        cout << "bad file" << endl;
                }

                VectorNode data3D(n);

                int64_t scanCount = data3D.childCount();
                cout << "Number of scans in file:" << scanCount << endl;

                ofstream outfile(outputFullFileName + ".pts");
                stringstream scanPoints;

                int totalPointCount = 0;

                for (int scanIndex = 0; scanIndex < scanCount; scanIndex++) {
                        StructureNode scan(data3D.get(scanIndex));
                        CompressedVectorNode points(scan.get("points"));
                        StructureNode proto(points.prototype());
                        if (proto.isDefined("cartesianX") && proto.isDefined("cartesianY") && proto.isDefined("cartesianZ")) {
                                totalPointCount += points.childCount();
                        }
                }

                outfile << totalPointCount << endl;

                for (int scanIndex = 0; scanIndex < scanCount; scanIndex++) {
                        StructureNode scan(data3D.get(scanIndex));
                        cout << "got:" << scan.pathName() << endl;

                        CompressedVectorNode points(scan.get("points"));
                        cout << "got:" << points.pathName() << endl;

                        StructureNode proto(points.prototype());

                        if (proto.isDefined("cartesianX") && proto.isDefined("cartesianY") && proto.isDefined("cartesianZ")) {
                                int64_t count = points.childCount();

                                vector<SourceDestBuffer> destBuffers;
                                vector<double> x(count), y(count), z(count), a(count), r(count), g(count), b(count);

        destBuffers.push_back(SourceDestBuffer(inf, "cartesianX", &x[0], count, true));
        destBuffers.push_back(SourceDestBuffer(inf, "cartesianY", &y[0], count, true));
        destBuffers.push_back(SourceDestBuffer(inf, "cartesianZ", &z[0], count, true));
                                if (proto.isDefined("colorRed") && proto.isDefined("colorGreen") && proto.isDefined("colorBlue")) {
        destBuffers.push_back(SourceDestBuffer(inf, "colorRed", &r[0], count, true));
        destBuffers.push_back(SourceDestBuffer(inf, "colorGreen", &g[0], count, true));
        destBuffers.push_back(SourceDestBuffer(inf, "colorBlue", &b[0], count, true));
                                }
                                CompressedVectorReader reader = points.reader(destBuffers);

                                unsigned gotCount = reader.read();

                                //2048 - max intensity value
                                for (unsigned i = 0; i < gotCount; i++) {
                                        outfile << x[i] << " " << y[i] << " " << z[i] << " 2048 " << r[i] << " " << g[i] << " " << b[i] << endl;
                                }
                                reader.close();
                        }
                        else {
                                cout << "Error: could not find either Cartesian or spherical points in scan" << endl;
                        }
                }
                outfile.close();
                inf.close();
        }
        catch (E57Exception& ex) {
                ex.report(__FILE__, __LINE__, __FUNCTION__);
        }
        catch (exception& ex) {
                cerr << ex.what() << endl;
        }
}
```

Листинг AccountController.cs:

```
using System.Web.Mvc;
using AutodeskForgeProvider.Authentication;
using AutodeskForgeProvider.Authentication.Models;
using System.Web.Configuration;
using System.Threading.Tasks;
using System.Net.Http;
using System.Net;

namespace PointCloudWebViewer.Controllers
{
        public class AccountController : BaseController
        {

                [HttpGet]
                public async Task<ActionResult> GetAccessToken(string code)
                {
                        string callBackUrl = WebConfigurationManager.AppSettings["callBackUrl"];
                        string clientId = WebConfigurationManager.AppSettings["forgeClientId"];
                        string clientSecret = WebConfigurationManager.AppSettings["forgeClientSecret"];

            if (code == null)
            {
                Session.Clear();
                ViewBag.logOutUrl = System.Configuration.ConfigurationManager.AppSettings["logOutUrl"];
                return View("LogOut");
            }
                        OAuth auth = new OAuth();
                        Token token = await auth.GetToken(clientId, clientSecret, code, callBackUrl);

                        if (token != null)
                        {
                                Profile profile = await auth.GetProfileInformation(token);
                                Session["LogIn"] = profile.FirstName + " " + profile.LastName;
                                Session["UserID"] = profile.UserId;
                                CurrentUserToken = token;
                                return RedirectToAction("../Viewer");
                        }
                        else
                        {
                                var errorResp = new HttpResponseMessage(HttpStatusCode.Unauthorized)
                                {
                                        ReasonPhrase = "Error while loging!"
                                };
                                throw new System.Web.Http.HttpResponseException(errorResp);       //ToDo: Return Error page!
                        }
                }

                public ActionResult Logout()
                {

                        if (Session["PathToDelete"] != null)
                        {
                                string folderToDelete = Session["PathToDelete"].ToString();
                System.IO.Directory.Delete(folderToDelete, true);
                        }
                        Session.Clear();
                        ViewBag.logOutUrl = System.Configuration.ConfigurationManager.AppSettings["logOutUrl"];
                        return View("LogOut");
                }

        }
}
```

Листинг BaseController.cs:

```
using AutodeskForgeProvider.Authentication.Models;
using System.Web.Mvc;

namespace PointCloudWebViewer.Controllers
{
    public class BaseController:Controller
    {
        public Token CurrentUserToken
```

```
        {
            get { return (Token)Session["3LToken"]; }
            set { Session["3LToken"] = value; }
        }
    }
}
```

Лістинг ForgeController.cs:

```
using AutodeskForgeProvider.Authentication;
using AutodeskForgeProvider.Authentication.Models;
using AutodeskForgeProvider.DataManagement;
using AutodeskForgeProvider.DataManagement.Models;
using AutodeskForgeProvider.DataManagement.Models.Child.Items;
using AutodeskForgeProvider.Models;
using PointCloudWebViewer.Filters;
using PointCloudWebViewer.Helper.PcdToPdfConverter;
using PointCloudWebViewer.Helpers;
using PointCloudWebViewer.Services.Utils;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Net.Mail;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Web.Configuration;
using System.Web.Mvc;

namespace PointCloudWebViewer.Controllers.ForgeController
{
        [AutodeskForgeAuthorize]
        [SessionState(System.Web.SessionState.SessionStateBehavior.ReadOnly)]
        public class ForgeController : BaseController
        {
                private const int maxJsonLength = int.MaxValue;

                [HttpGet]
                public async Task<JsonResult> SaveFileFromClient(string filename, string projectId, string folderId)
                {
                        ForgeHelper forgeHelper = new ForgeHelper();
                        var fileId = await forgeHelper.CheckFileExistanceInFolder(projectId, folderId, filename);

                        if (fileId != null)
                        {
                                try
                                {
                                        DataManagement dataManagment = new DataManagement(CurrentUserToken);
                                        StorageResponse storageLocation = await dataManagment.CreateStorage(projectId, filename, folderId);

                                        string storageId = storageLocation.Data.Id;
                                        string itemFileName = storageId.Substring(58);
                                        string bucketKey = storageId.Substring(22, 11);


        Dictionary<string, string> createNewVersionResponseParameters = new Dictionary<string, string>();

        createNewVersionResponseParameters["responseType"] = "Upd";

        createNewVersionResponseParameters["bucketKey"] = bucketKey;

        createNewVersionResponseParameters["itemFileName"] = itemFileName;

        createNewVersionResponseParameters["storageId"] = storageId;

        createNewVersionResponseParameters["fileId"] = fileId;

        createNewVersionResponseParameters["token"] = CurrentUserToken.AccessToken;

                        return Json(createNewVersionResponseParameters, JsonRequestBehavior.AllowGet);
                                }
                                catch (Exception)
                                {
                                        var errorResp = new HttpResponseMessage(HttpStatusCode.InternalServerError)
                                        {

        ReasonPhrase = "Error while model updating!"
                                        };
                                        throw new System.Web.Http.HttpResponseException(errorResp);
                                }
                        }
                        else
                        {
                                try
                                {
                                        DataManagement dataManagment = new DataManagement(CurrentUserToken);
                                        StorageResponse storageLocation = await dataManagment.CreateStorage(projectId, filename, folderId);

                                        string storageId = storageLocation.Data.Id;
                                        string itemFileName = storageId.Substring(58);
                                        string bucketKey = storageId.Substring(22, 11);


        Dictionary<string, string> createFileResponseParameters = new Dictionary<string, string>();

        createFileResponseParameters["responseType"] = "Crt";

        createFileResponseParameters["bucketKey"] = bucketKey;

        createFileResponseParameters["itemFileName"] = itemFileName;

        createFileResponseParameters["storageId"] = storageId;

        createFileResponseParameters["token"] = CurrentUserToken.AccessToken;

                                        return Json(createFileResponseParameters, JsonRequestBehavior.AllowGet);
                                }
                                catch (Exception)
                                {
                                        var errorResp = new HttpResponseMessage(HttpStatusCode.InternalServerError)
                                        {

        ReasonPhrase = "Error while model uploading!"
                                        };
                                        throw new System.Web.Http.HttpResponseException(errorResp);
                                }
                        }
                }

                [HttpGet]
                public async Task<int> CreateNewVersionResponse(string projectId, string fileName, string fileId, string storageId)
                {
                        DataManagement dataManagment = new DataManagement(CurrentUserToken);
                        NewVersionResponse createNewVersionResponse = await dataManagment.CreateNewVersion(projectId, fileName, fileId, storageId);
                        return Int32.Parse(createNewVersionResponse.Data.Attributes.VersionNumber);
                }

                [HttpGet]
                public async Task<string> UploadObjectResponse(string projectId, string fileName, string folderId, string storageId)
                {
                        DataManagement dataManagment = new DataManagement(CurrentUserToken);
                        ItemResponse createItemResponse = await dataManagment.CreateItem(projectId, folderId, storageId, fileName);
                        return createItemResponse.Data.Id;
                }

                #region Token

                [HttpGet]
                public async Task<Token> UpdateToken()
                {
                        OAuth oauth = new OAuth();
                        Token token = CurrentUserToken;
                        string clientId = WebConfigurationManager.AppSettings["ForgeClientId"];
                        string clientSecret = WebConfigurationManager.AppSettings["ForgeClientSecret"];
                        CurrentUserToken = await oauth.RefreshToken(clientId, clientSecret, token.RefreshToken);
                        return CurrentUserToken;
                }

                [HttpGet]
                public string GetToken()
                {
                        return CurrentUserToken.AccessToken;
                }

                #endregion

                #region Hubs

                [HttpGet]
                public async Task<JsonResult> GetHubs()
                {
```

```
                DataManagement dataManagement = new
DataManagement(currentUserUrn);
                Hub hubs = await
dataManagement.GetHubs();
                return Json(hubs,
JsonRequestBehavior.AllowGet);
            }

            #endregion

            #region Projects

            [HttpGet]
            public async Task<JsonResult>
GetProjects(string projectId)
            {
                DataManagement dataManagement = new
DataManagement(currentUserUrn);
                Project projects = await
dataManagement.GetProjects(projectId);
                return Json(projects,
JsonRequestBehavior.AllowGet);
            }

            #endregion

            #region Folders

            public async Task<Folder[]>
GetFoldersInFolder(string projectId, string folderId)
            {
                DataManagement dataManagement = new
DataManagement(currentUserUrn);
                Item<Data[]> items = await
dataManagement.GetItems(projectId, folderId);
        List<Folder> folder = new List<Folder>();
                foreach (var _item in items.Data)
                {
                    if (_item.Type ==
"folders")
                    {
                        folder.Add(await
dataManagement.GetFolder(projectId, _item.Id));
                    }
                }
        folder.Sort((x, y) =>
string.Compare(x.Data.Attributes.Name, y.Data.Attributes.Name,
true));
        return folder.ToArray();
            }

            #endregion

            #region Files
            [HttpGet]
            public async Task<JsonResult>
GetE57Data(string url)
            {
                const string fileName =
"processedFile";
                const string newFileName =
"resultFile";

                DataManagement dataManagement = new
DataManagement(currentUserUrn);

                DirectoryHelper directoryHelper =
new DirectoryHelper();

        directoryHelper.CreateTempDirectory(Session["UserID
"].ToString(), fileName);

                string inputFullFileName =
Path.Combine(directoryHelper.FileFolderPath, fileName);
                string outputFullFileName =
Path.Combine(directoryHelper.FileFolderPath, newFileName);

                byte[] fileItem = await
dataManagement.DownloadObject(url);

                using (FileStream fs = new
FileStream(inputFullFileName, FileMode.OpenOrCreate))
                {
                    fs.Write(fileItem, 0,
fileItem.Length);
                }

                E57Parser e57Parser = new
E57Parser(inputFullFileName);
                e57Parser.Parse();

                var data = new { e57Parser.Points,
e57Parser.Colors };

        directoryHelper.DeleteTempDirectory();

                JsonResult result = Json(data,
JsonRequestBehavior.AllowGet);
                result.MaxJsonLength =
maxJsonLength;
                return result;
            }

        #region Not used because parsing of the *.laz files
occurs on the client side
        [HttpGet]
        public async Task<JsonResult> GetLazData(string url)
        {
            const string fileName = "processedFile";
            const string newFileName = "resultFile";

            DataManagement dataManagement =
new DataManagement(currentUserUrn);

            DirectoryHelper directoryHelper =
new DirectoryHelper();

directoryHelper.CreateTempDirectory(Session["UserID"].ToStri
ng(), fileName);

            string inputFullFileName =
Path.Combine(directoryHelper.FileFolderPath, fileName);
            string outputFullFileName =
Path.Combine(directoryHelper.FileFolderPath, newFileName);

            byte[] fileItem =
await dataManagement.DownloadObject(url);

            using (FileStream fs =
new FileStream(inputFullFileName, FileMode.OpenOrCreate))
            {
                fs.Write(fileItem, 0, fileItem.Length);
            }

            LazParser lazParser =
new LazParser(inputFullFileName);
            lazParser.Parse();

            var data = new { lazParser.Points,
lazParser.Colors };

            directoryHelper.DeleteTempDirectory();

            JsonResult result =
Json(data, JsonRequestBehavior.AllowGet);
            result.MaxJsonLength = maxJsonLength;
            return result;
        }
        #endregion

        [HttpGet]
        public async Task<String>
GetDownloadHref(string projectId, string fileId)
        {
            DataManagement dataManagement = new
DataManagement(currentUserUrn);
            Item<Data> item;
            if (Regex.IsMatch(fileId,
@".+\?version=\d+"))
                item =
await dataManagement.GetItemSpecificVersion(projectId, fileId);
            else
                item =
await dataManagement.GetItem(projectId, fileId);

            return
item.Included[0].Relationships.Storage.Meta.Link.Href;
        }

        [HttpGet]
        public async Task<FileStreamResult>
DownloadFilePdf(string projectId, string fileId, string
fileName)
        {
            if (Session[fileName] == null)
            {
                PDFExporter exporter =
await ConvertPcdToPdf(projectId, fileId, fileName);
                Session["PathToDelete"] =
exporter.UserFolderPath;
                Session[fileName] = exporter.PathToPDF;
            }

            byte[] pdfByte =
System.IO.File.ReadAllBytes(Session[fileName].ToString());
            string file_type = "application/pdf";
            MemoryStream ms = new MemoryStream(pdfByte);

            return File(ms, file_type,
Path.GetFileName(Session[fileName].ToString()));
        }

        private async Task<PDFExporter> ConvertPcdToPdf(string
projectId, string fileId, string fileName)
        {
            PDFExporter exporter =
new PDFExporter(fileName,
Session["UserID"].ToString());
            string descFullFileName =
Path.Combine(exporter.PathToTempOutputDir, fileName);

            if
(System.IO.File.Exists(descFullFileName))
                System.IO.File.Delete(descFullFileName);

            DataManagement dataManagement =
new DataManagement(currentUserUrn);
            Item<Data> item =
await dataManagement.GetItem(projectId, fileId);
            byte[] fileItem =
await dataManagement.DownloadObject(item.Included[0].Relationships.St
orage.Meta.Link.Href);
            using (FileStream fs =
new FileStream(descFullFileName, FileMode.OpenOrCreate))
            {
                fs.Write(fileItem, 0,
fileItem.Length);
            }
            exporter.Convert();

            return exporter;
        }
```

```
        //Returns generated pdf from point cloud model.
        [HttpGet]
        public async Task<HttpStatusCodeResult> SendMail(string projectId, string fileId, string fileName, string emailTo, string msgText)
        {
            if (fileName != "undefined")
            {
                if (Session[fileName] == null)
                {
                    PDFExporter exporter = await ConvertPcdToPdf(projectId, fileId, fileName);
                    Session[fileName] = exporter.PathToPDF;
                }
                var body = System.IO.File.ReadAllText(Server.MapPath("~/Content/mailtemplate.html"));
                var message = new MailMessage();
                message.To.Add(new MailAddress(emailTo));
                message.Subject = WebConfigurationManager.AppSettings["mailSubject"];
                message.Body = string.Format(body, Session["Login"], fileName, msgText);
                message.IsBodyHtml = true;

                byte[] pdfByte = System.IO.File.ReadAllBytes(Session[fileName].ToString());
                Stream stream = new MemoryStream(pdfByte);

                if (Session[fileName] != null && pdfByte.Length > 0)
                {
                    message.Attachments.Add(new Attachment(stream, Path.GetFileName(Session[fileName].ToString())));
                }
                using (var smtp = new SmtpClient())
                {
                    await smtp.SendMailAsync(message);
                    return new HttpStatusCodeResult(HttpStatusCode.OK);
                }
            }
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }

        [HttpPost]
        public async Task<HttpStatusCodeResult> SendFileByMail(string data)
        {
            string[] splitData = data.Split("$");
            string projectId = splitData[0];
            string fileId = splitData[1];
            string fileName = splitData[2];
            string emailTo = splitData[3];
            if (fileName != "undefined")
            {
                if (Session[fileName] == null)
                {
                    PDFExporter exporter = await ConvertPcdToPdf(projectId, fileId, fileName);
                    Session[fileName] = exporter.PathToPDF;
                }
                string body = System.IO.File.ReadAllText(Server.MapPath("~/Content/mailtemplateForDownload.html"));
                var message = new MailMessage();
                message.To.Add(new MailAddress(emailTo));
                message.Subject = fileName;
                message.Body = string.Format(body, fileName);
                message.IsBodyHtml = true;

                byte[] pdfByte = System.IO.File.ReadAllBytes(Session[fileName].ToString());
                Stream stream = new MemoryStream(pdfByte);

                if (Session[fileName] != null && pdfByte.Length > 0)
                {
                    message.Attachments.Add(new Attachment(stream, Path.GetFileName(Session[fileName].ToString())));
                }
                using (var smtp = new SmtpClient())
                {
                    await smtp.SendMailAsync(message);
                    return new HttpStatusCodeResult(HttpStatusCode.OK);
                }
            }
            return new HttpStatusCodeResult(HttpStatusCode.OK);
        }

        [HttpPost]
        public async Task<HttpStatusCodeResult> SendLinkMail(string data)
        {
            string[] splitData = data.Split("$");
            string projectId = splitData[0];
            string folderId = splitData[1];
            string fileId = splitData[2];
            string fileName = splitData[3];
            string emailTo = splitData[4];
            string pdfFileName = Path.GetFileNameWithoutExtension(fileName) + ".pdf";
            if (fileName != "undefined")
            {

                ForgeHelper forgeHelper = new ForgeHelper();
                var pdfFileId = await forgeHelper.CheckFileExistanceInFolder(projectId, folderId, pdfFileName);

                if (pdfFileId != null)
                {
                    try
                    {
                        DataManagement dataManagment = new DataManagement(currentUserToken);
                        DateTime lastModifiedFile = new DateTime();
                        DateTime lastModifiedPdf = new DateTime();
                        Item<Data[]> items = await dataManagment.GetItems(projectId, folderId);
                        int includCount = 0;
                        foreach(var item in items.Data)
                        {
                            if (item.Type != "folders")
                            {
                                if (item.Attributes.DisplayName == fileName && item.Type == "items")
                                {
                                    lastModifiedFile = Convert.ToDateTime(items.Included[includCount].Attributes.LastModifiedTime);
                                }
                                if (item.Attributes.DisplayName == pdfFileName && item.Type == "items")
                                {
                                    lastModifiedPdf = Convert.ToDateTime(items.Included[includCount].Attributes.LastModifiedTime);
                                }
                                includCount++;
                            }
                        }
                        if (DateTime.Compare(lastModifiedFile,lastModifiedPdf) > 0)
                        {
                            if (Session[fileName] == null)
                            {
                                PDFExporter exporter = await ConvertPcdToPdf(projectId, fileId, fileName);
                                Session[fileName] = exporter.PathToPDF;
                            }
                            dataManagment = new DataManagement(await UpdateToken());
                            StorageResponse storageLocation = await dataManagment.CreateStorage(projectId, pdfFileName, folderId);

                            string storageId = storageLocation.Data.id;
                            string itemFileName = storageId.Substring(59);
                            string bucketKey = storageId.Substring(27, 11);
                            byte[] pdfByte = System.IO.File.ReadAllBytes(Session[fileName].ToString());

                            await dataManagment.UploadObject(bucketKey, itemFileName, pdfByte);
                            NewVersionResponse createNewVersionResponse = await dataManagment.CreateNewVersion(projectId, pdfFileName, pdfFileId, storageId);
                        }
                    }
                    catch (Exception)
                    {
                        var errorResp = new HttpResponseMessage(HttpStatusCode.InternalServerError)
                        {
                            ReasonPhrase = "Error while model updating!"
                        };
                        throw new System.Web.Http.HttpResponseException(errorResp);
                    }
                }
                else
                {
                    try
                    {
                        if (Session[fileName] == null)
                        {
                            PDFExporter exporter = await ConvertPcdToPdf(projectId, fileId, fileName);
                            Session[fileName] = exporter.PathToPDF;
                        }
                        DataManagement dataManagment = new DataManagement(await UpdateToken());
                        StorageResponse storageLocation = await dataManagment.CreateStorage(projectId, pdfFileName, folderId);
```

```
                    string storageId =
storageLocation.Data.Id;
                    string itemFileName =
storageId.Substring(39);
                    string bucketKey =
storageId.Substring(27, 11);
                    byte[] pdfByte =
System.IO.File.ReadAllBytes(Session[fileName].ToString());

                    await
dataManagment.UploadObject(bucketKey, itemFileName, pdfByte);
                    await
dataManagment.CreateItem(projectId, folderId, storageId,
pdfFileName);
                }
                catch (Exception)
                {
                    var errorResp = new
HttpResponseMessage(HttpStatusCode.InternalServerError)
                    {
                        ReasonPhrase = "Error while model
uploading!"
                    };
                    throw new
System.Web.Http.HttpResponseException(errorResp);
                }
            }

            string body =
System.IO.File.ReadAllText(Server.MapPath("~/Content/mailTempl
ateForLink.html"));
            var message = new MailMessage();
            message.To.Add(new MailAddress(emailTo));
            message.Subject = fileName;
            message.Body = string.Format(body, fileName);
            message.IsBodyHtml = true;

            using (var smtp = new SmtpClient())
            {
                await smtp.SendMailAsync(message);
                return new
HttpStatusCodeResult(HttpStatusCode.OK);
            }
        }
        return new
HttpStatusCodeResult(HttpStatusCode.OK);
    }

    public async Task<HttpStatusCodeResult>
CreateFolder(string projectId, string parentFolderId, string
folderName)
    {
        DataManagement dataManagment = new
DataManagement(CurrentUserToken);
        var json = await
dataManagment.CreateFolder(projectId, parentFolderId,
folderName);
        return new
HttpStatusCodeResult(HttpStatusCode.OK);
    }

    [HttpGet]
    public async Task<FileStreamResult>
DownloadFile(string projectId, string fileId)
    {
        DataManagement dataManagment = new
DataManagement(CurrentUserToken);
        Item<Data> item = await
dataManagment.GetItem(projectId, fileId);
        byte[] fileItem = await
dataManagment.DownloadObject(item.Included[0].Relationships.St
orage.Meta.Link.Href);
        MemoryStream ms = new
MemoryStream(fileItem);
        return File(ms, "application/pcd",
item.Data.Attributes.DisplayName);
    }
    #endregion

    #region Tree building
    // Returns JSON: Hubs>Projects(with files
information)>Folders(with files information).
    // Example: /Forge/GetInitialForgeTree/
    [HttpGet]
    public async Task<JsonResult>
GetInitialForgeTree()
    {
        ForgeModelsTreeView forgeTree = new
ForgeModelsTreeView();
        // Getting Hubs
        forgeTree.Hubs = await
GetHubTree();

        return Json(forgeTree,
JsonRequestBehavior.AllowGet);
    }

    // Returns HubTree List, including Projects
tree
    private async Task<List<HubTree>>
GetHubTree()
    {
        DataManagement dataManagment = new
DataManagement(CurrentUserToken);
        Hub hubs = await
dataManagment.GetHubs();
        List<HubTree> hubTreeList = new
List<HubTree> { };
        foreach (var hub in hubs.Data)
        {
            HubTree hubTree = new
HubTree();
            hubTree.HubId = hub.Id;
            hubTree.HubName =
hub.Attributes.Name;
            // Getting Projects
            hubTree.Projects = await
GetProjectTree(hub.Id);

            hubTreeList.Add(hubTree);
        }
        return hubTreeList;
    }

    // Returns ProjectTree List, including
Folders tree and files information
    private async Task<List<ProjectTree>>
GetProjectTree(string hubId)
    {
        DataManagement dataManagment = new
DataManagement(CurrentUserToken);
        Project projects = await
dataManagment.GetProjects(hubId);
        List<ProjectTree> projectTreeList =
new List<ProjectTree> { };
        foreach (var project in
projects.Data)
        {
            ProjectTree projectTree =
new ProjectTree();
            projectTree.ProjectId =
project.Id;
            projectTree.ProjectName =
project.Attributes.Name;
            projectTree.RootFolder =
project.Relationships.RootFolder.Data.Id;
            // Getting Files
            projectTree.Files = await
GetFilesInFolder(projectTree.ProjectId,
projectTree.RootFolder);
            //Getting Folders
            projectTree.Folders =
await GetFolderTree(projectTree.ProjectId,
projectTree.RootFolder);

        projectTreeList.Add(projectTree);
        }
        return projectTreeList;
    }

    // Returns FolderTree List, including files
information
    private async Task<List<FolderTree>>
GetFolderTree(string projectId, string folderId)
    {
        Folder[] folders = await
GetFoldersInFolder(projectId, folderId);
        List<FolderTree> folderTreeList =
new List<FolderTree>();

        foreach (var folder in folders)
        {
            FolderTree folderTree =
new FolderTree();
            folderTree.FolderId =
folder.Data.Id;
            folderTree.FolderName =
folder.Data.Attributes.Name;
            // Getting Files
            folderTree.Files = await
GetFilesInFolder(projectId, folder.Data.Id);
            List<FolderTree>
childFoldersList = new List<FolderTree> { };
            Folder[] childFolders =
await GetFoldersInFolder(projectId, folder.Data.Id);
            foreach (var childFolder
in childFolders)
            {
                FolderTree
childfolderTree = new FolderTree();

        childfolderTree.FolderId = childFolder.Data.Id;

        childfolderTree.FolderName =
childFolder.Data.Attributes.Name;

        childFoldersList.Add(childfolderTree);
            }

        folderTree.FolderTreeChild = childFoldersList;


        folderTreeList.Add(folderTree);
        }
        return folderTreeList;
    }

    // Returns List of CloudDataFile in
folder(with .pcd extention).
    private async Task<List<CloudDataFile>>
GetFilesInFolder(string projectId, string folderId)
    {
        DataManagement dataManagment = new
DataManagement(CurrentUserToken);
        List<CloudDataFile>
cloudDataFileList = new List<CloudDataFile> { };
        Item<Data[]> items = await
dataManagment.GetItems(projectId, folderId);
        int includedCounter = 0;

        foreach (var item in items.Data)
        {
```

```
                                if      (item.Type      !=
"folders")
                                {
                                    string   filename
= item.Attributes.DisplayName;
                                    string
fileExtension = Path.GetExtension(filename).ToLower();

                                    if
(IsSupportedFormat(fileExtension))
                                    {

        CloudDataFile file = new CloudDataFile();

        file.OwnerID = Session["UserID"].ToString();

        file.ItemId = item.Id;

        file.Version                                    =
items.Included[includedCounter].Attributes.VersionNumber.ToStr
ing();

        file.Name = item.Attributes.DisplayName;

        file.DeleteLink      =      Url.Content("~/")      +
"Forge/DeleteObject/ToDo";

        file.OpenInViewerLink                           =
String.Format("{0}Viewer/ForgeViewer/{1}/{2}/",
Url.Content("~/"), projectId, item.Id);

        file.DownloadLink                               =
String.Format("{0}Forge/DownloadFile?ProjectId={1}&FileId={2}"
, Url.Content("~/"), projectId, item.Id);

        file.Extension = fileExtension;
                        file.StorageSize                =
int.Parse(items.Included[includedCounter].Attributes.StorageSi
ze.ToString());


        cloudDataFileList.Add(file);
                                    }

        includedCounter++;
                                }
                            }

                            foreach (var item in items.Data)
                            {
                                string      filename      =
item.Attributes.DisplayName;
                                string   fileExtension   =
Path.GetFileNameWithoutExtension(filename);
                                if      ((item.Type      !=
"folders") && IsSupportedFormat(fileExtension))
                                {
                                    int      i      =
cloudDataFileList.FindIndex(fileItem    =>    fileItem.Name    ==
fileExtension);

                                    if (i > -1)
                                    {
                                        var
itemVersions   =   await   dataManagment.GetVersions(projectId,
item.Id);
                                        var
preview      =      itemVersions.Data.FirstOrDefault(a        =>
a.Attributes.Version == cloudDataFileList[i].Version);
                                        if
(preview == null)

        preview   =   itemVersions.Data.OrderByDescending(a   =>
a.Attributes.Version).First();


        cloudDataFileList[i].HasPreview = true;

        cloudDataFileList[i].PreviewId = preview.Id;
                                    }
                                }
                            }

            cloudDataFileList.Sort((x,          y)          =>
string.Compare(x.Name, y.Name, true));
                            return cloudDataFileList;
                        }

                        [HttpGet]
                        // Returns json tree with folders and files
in specified folder
                        public        async        Task<JsonResult>
GetFolderTreeElement(string projectId, string folderId)
                        {
                            Folder[]     folders     =     await
GetFoldersInFolder(projectId, folderId);
                            List<CloudDataFile> files = await
GetFilesInFolder(projectId, folderId);

                            Dictionary<string, object> data =
new Dictionary<string, object>();
                            data.Add("folders", folders);
                            data.Add("files", files);
                            data.Add("projectId", projectId);

                            return                      Json(data,
JsonRequestBehavior.AllowGet);
                        }
                        #endregion

                        public      bool      IsSupportedFormat(string
extension)
                        {
                            switch (extension)
                            {
                                case ".pcd":
                                case ".pts":
                                case ".ptx":
                                case ".e57":
                                case ".xyz":
                                case ".las":
                                case ".laz":
                                case ".txt":
                                    //case ".ptg":
                                    return true;
                                default:
                                    return false;
                            }
                        }
                    }
}
```

Лістинг HomeController.cs:

```
using Newtonsoft.Json;
using PointCloudWebViewer.Filters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace PointCloudWebViewer.Controllers
{
    public class HomeController : Controller
    {

        public ActionResult Index()
        {
            return View();
        }

    }
}
```

Лістинг ViewerController.cs:

```
using AutodeskForgeProvider.DataManagement;
using Newtonsoft.Json;
using System.Threading.Tasks;
using System.Web.Mvc;

namespace PointCloudWebViewer.Controllers
{
    public class ViewerController : Controller
    {
        public ActionResult Index()
        {
            if (Session["3LToken"] != null)
            {
                return View();
            }
            else
            {
                string              authorizeLink             =
"https://developer.api.autodesk.com/authentication/v1/authoriz
e?response_type=code&client_id=" +

@System.Configuration.ConfigurationManager.AppSettings["forgeC
lientId"] + "&redirect_uri=" +

@System.Configuration.ConfigurationManager.AppSettings["callBa
ckUrl"] +

"&scope=data:read%20bucket:create%20bucket:read%20bucket:delet
e%20data:write";

                return Redirect(authorizeLink);
            }
        }

        public  async  Task<ActionResult>  ForgeViewer(string
projectId,string folderId, string itemId, string hasPreview)
        {

            if (Session["3LToken"] != null && projectId !=
null && itemId != null)
            {
                DataManagement     dataManagment     =     new
DataManagement((AutodeskForgeProvider.Authentication.Models.To
ken)Session["3LToken"]);

AutodeskForgeProvider.DataManagement.Models.Item<AutodeskForge
Provider.DataManagement.Models.Child.Items.Data> item =  await
dataManagment.GetItem(projectId, itemId);
                ViewBag.Path = JsonConvert.SerializeObject(new
string[]                                                    {
item.Included[0].Relationships.Storage.Meta.Link.Href,
item.Data.Attributes.DisplayName,     folderId,     projectId,
hasPreview.ToString() });
                ViewBag.Name                                =
item.Data.Attributes.DisplayName;
            }
            else
            {
                return RedirectToAction("Index");
            }
            return View("Viewer");
        }


    }
```

Лістинг E57ToPtsConverter.cs:

```
using System;
using System.IO;
using System.Runtime.InteropServices;
using System.Web;

namespace PointCloudWebViewer.Helpers.E57ToPtsConverter
{
	/// <summary>
	/// Describes the e57 to pts conversion
	/// </summary>
	public class E57ToPtsConverter
	{
		private const string e57ToPtsConverterDllPath = @"E57ToPtsConverter\E57Converter.dll";
		private const string DllEntryPoint = "?ConvertToPts@E57Converter@@SAXPBD0@Z";
		private readonly string inputFullFileName;
		private readonly string outputFileName;

		/// <summary>
		/// Constructor
		/// </summary>
		/// <param name="inputFullFileName">Expected the full file name with file expansion</param>
		/// <param name="outputFileName">Expected the full file name without file expansion. The output file will be with *.pts expansion.</param>
		public E57ToPtsConverter(string inputFullFileName, string outputFileName)
		{
			this.inputFullFileName = inputFullFileName;
			this.outputFileName = outputFileName;
		}

		[DllImport(dllName: e57ToPtsConverterDllPath, CallingConvention = CallingConvention.Cdecl, EntryPoint = DllEntryPoint)]
		private static extern void Converter(string inputFullFileName, string outputFileName);

		public void Convert()
		{
			Converter(inputFullFileName, outputFileName);
		}
	}
}
```

Лістинг ColorARGB.cs:

```
namespace PointCloudWebViewer.Helpers
{
	public struct ColorARGB
    {
        public double A { get; set; }
        public double R { get; set; }
        public double G { get; set; }
        public double B { get; set; }
    }
}
```

Лістинг Directoryhelper.cs:

```
using System;
using System.IO;

namespace PointCloudWebViewer.Helpers
{
	public class DirectoryHelper
	{
		public string UserFolderPath { get; set; }
		public string FileFolderPath { get; set; }

		/// <summary>
		/// Describes work with the directories for a temporary files.
		/// </summary>
		/// <param name="userFolderName">The name of a directory for the definite user</param>
		/// <param name="fileFolderName">Name of a directory for a temporary files</param>
		public void CreateTempDirrectory(string userFolderName, string fileFolderName)
		{
			UserFolderPath = Path.Combine(Path.GetTempPath(), userFolderName);
			if (!Directory.Exists(UserFolderPath))
			{
				Directory.CreateDirectory(UserFolderPath);
			}

			int i = 1;
			FileFolderPath = Path.Combine(UserFolderPath, fileFolderName);
			while (Directory.Exists(FileFolderPath))
			{
				FileFolderPath = Path.Combine(UserFolderPath, fileFolderName + (i++).ToString());
			}
		}

		public void DeleteTempDirrectory()
		{
			Directory.Delete(FileFolderPath, true);
		}
	}
}
```

Лістинг E57Parser.cs:

```
using System.IO;
using System.Windows.Media.Media3D;
using E57 = PointCloudWebViewer.Helpers.E57ToPtsConverter;

namespace PointCloudWebViewer.Helpers
{
	public class E57Parser
	{
		private readonly string fullFileName;

		public E57Parser(string fullFileName)
		{
			this.fullFileName = fullFileName;
		}

		public Point3D[] Points { get; set; }
		public ColorARGB[] Colors { get; set; }

		public void Parse()
		{
			string outputFullFileName = Path.GetFullPath(fullFileName);

			E57.E57ToPtsConverter e57ToPtsConverter = new E57.E57ToPtsConverter(fullFileName, outputFullFileName);

			e57ToPtsConverter.Convert();

			PtsParser ptsParser = new PtsParser(fullFileName + ".pts");
			ptsParser.Parse();

			Points = ptsParser.Points;
			Colors = ptsParser.Colors;
		}
	}
}
```

Лістинг LasParser.cs:

```
using System;
using System.IO;
using System.Windows.Media.Media3D;

namespace PointCloudWebViewer.Helpers
{
    public class LasParser
    {
        public string FilePath { get; set; }
        public Point3D[] Points { get; set; }
        public ColorARGB[] Colors { get; set; }
        public double offsetX, offsetY, offsetZ, scaleX, scaleY, scaleZ;
        public LasParser(string filePath)
        {
            FilePath = filePath;
        }
        public void Parse()
        {
            using (BinaryReader binaryReader = new BinaryReader(File.Open(FilePath, FileMode.Open)))
            {
                int lenght = (int)binaryReader.BaseStream.Length;
                byte[] lasData = binaryReader.ReadBytes(lenght);

                //Las format always set on 24 and 25 bytes, count from 0
                //string fileType = a[24].ToString() + a[25].ToString();

                //Offset to points always on 96 byte, count from 0
                uint offsetToPointData = BitConverter.ToUInt32(lasData, 96);

                //Data format always set on 104 byte, count from 0
                string dataFormat = lasData[104].ToString();

                //Point data record length in bytes always set on 105 byte
                ushort dataRecordLength = BitConverter.ToUInt16(lasData, 105);

                //Number of points always on 107 byte, count from 0
                uint numberOfPoints = BitConverter.ToUInt32(lasData, 107);

                Points = new Point3D[numberOfPoints];
                Colors = new ColorARGB[numberOfPoints];

                //Necessary variables for calculating coordinates
                scaleX = BitConverter.ToDouble(lasData, 131);
```

```
                scaleY = BitConverter.ToDouble(lasData, 139);
                scaleZ = BitConverter.ToDouble(lasData, 147);
                offsetX = BitConverter.ToDouble(lasData, 155);
                offsetY = BitConverter.ToDouble(lasData, 163);
                offsetZ = BitConverter.ToDouble(lasData, 171);

                byte[] dataPoints = new byte[dataRecordLength
* numberOfPoints];
                Array.Copy(lasData,          offsetToPointData,
dataPoints, 0, dataRecordLength * numberOfPoints);

                //*.las format has different point cloud data
formats (0-10)
                //each format has own items and point data
record length
                //follow this link for more information
https://www.asprs.org/wp-
content/uploads/2010/12/LAS_1_4_r13.pdf
                switch (dataFormat)
                {
                    case "0":
                    case "1":
                    case "4":
                    case "6":
                    case "9":
                        GetPoints(dataPoints,  numberOfPoints,
dataRecordLength);
                        break;
                    case "2":
                        //fourth argument (20) is the offset
in bytes to color data of the point
                        GetPointsWithColor(dataPoints,
numberOfPoints, dataRecordLength, 20);
                        break;
                    case "3":
                    case "5":
                        //fourth argument (28) is the offset
in bytes to color data of the point
                        GetPointsWithColor(dataPoints,
numberOfPoints, dataRecordLength, 28);
                        break;
                    case "7":
                    case "8":
                    case "10":
                        //fourth argument (30) is the offset
in bytes to color data of the point
                        GetPointsWithColor(dataPoints,
numberOfPoints, dataRecordLength, 30);
                        break;
                }

            }
        }
        private    void    GetPoints(byte[]    data,    uint
numberOfPoints, int dataFormatLength)
        {
            Int32[] points = new Int32[numberOfPoints * 3];

            //variable j contains information about data
record length in bytes
            for (int i = 0, j = 0; i < points.Length; i += 3,
j += dataFormatLength)
            {
                //The X, Y, and Z values are stored as long
integers (4 bytes), therefore step is 4
                points[i + 0] = BitConverter.ToInt32(data, 0 +
j);
                points[i + 1] = BitConverter.ToInt32(data, 0 +
4 + j);
                points[i + 2] = BitConverter.ToInt32(data, 0 +
8 + j);
            }
            for (int i = 0, ii = 0; i < points.Length; i+=3,
ii++)
            {
                Points[ii] = new Point3D()
                {
                    X = (points[i + 0] * scaleX) + offsetX,
                    Y = (points[i + 1] * scaleY) + offsetY,
                    Z = (points[i + 2] * scaleZ) + offsetZ
                };
            }
        }
        private  void  GetPointsWithColor(byte[]  data,  uint
numberOfPoints, int dataFormatLength, int offsetToColor)
        {
            Int32[] points = new Int32[numberOfPoints * 3];
            Int32[] colors = new Int32[numberOfPoints * 3];

            //variable j contains information about data
record length in bytes
            for (int i = 0, j = 0; i < points.Length; i += 3,
j += dataFormatLength)
            {
                //The X, Y, and Z values are stored as long
integers (4 bytes), therefore step is 4
                points[i + 0] = BitConverter.ToInt32(data, 0 +
j);
                points[i + 1] = BitConverter.ToInt32(data, 0 +
4 + j);
                points[i + 2] = BitConverter.ToInt32(data, 0 +
8 + j);

                //The red, green, blue values are stored as
unsigned short (2 bytes), therefore step is 2
                colors[i  +  0]  =  BitConverter.ToInt16(data,
offsetToColor + 0 + j);
                colors[i  +  1]  =  BitConverter.ToInt16(data,
offsetToColor + 2 + j);
                colors[i  +  2]  =  BitConverter.ToInt16(data,
offsetToColor + 4 + j);
            }
            for (int i = 0, ii = 0; i < points.Length; i += 3,
ii++)
            {
                Points[ii] = new Point3D()
                {
                    X = (points[i + 0] * scaleX) + offsetX,
                    Y = (points[i + 1] * scaleY) + offsetY,
                    Z = (points[i + 2] * scaleZ) + offsetZ
                };

                Colors[ii] = new ColorARGB()
                {
                    A = 1,
                    R = colors[i + 0] / 255f,
                    G = colors[i + 1] / 255f,
                    B = colors[i + 2] / 255f,
                };
            }
        }
    }
}
```

Листинг LazParser.cs:

```
using System;
using System.Windows.Media.Media3D;
//link              to              this              library
https://github.com/shintadono/laszip.net
using laszip.net;

namespace PointCloudWebViewer.Helpers
{
    public class LazParser
    {
        public string FilePath { get; set; }
        public Point3D[] Points { get; set; }
        public ColorARGB[] Colors { get; set; }
        public LazParser(string filePath)
        {
            FilePath = filePath;
        }
        public void Parse()
        {
            var lazReader = new laszip_dll();
            var compressed = true;
            lazReader.laszip_open_reader(FilePath,          ref
compressed);

            //Reading quantity of the points
            var                numberOfPoints                =
lazReader.header.number_of_point_records;
            Points = new Point3D[numberOfPoints];
            Colors = new ColorARGB[numberOfPoints];
            var coordinates = new double[3];
            var colors = new ushort[4];

            // Loop through number of points indicated
            for   (int   pointIndex   =   0;   pointIndex   <
numberOfPoints; pointIndex++)
            {
                // Read the point
                lazReader.laszip_read_point();

                // Get precision coordinates
                lazReader.laszip_get_coordinates(coordinates);
                Points[pointIndex] = new Point3D()
                {
                    X = coordinates[0],
                    Y = coordinates[1],
                    Z = coordinates[2]
                };

                //Get color
                colors = lazReader.point.rgb;
                Colors[pointIndex] = new ColorARGB()
                {
                    //A = 1,
                    R = colors[0] / 255f,
                    G = colors[1] / 255f,
                    B = colors[2] / 255f,
                };
            }

            // Close the reader
            lazReader.laszip_close_reader();

        }
    }
}
```

Листинг PcdParser.cs:

```
using System;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows.Media.Media3D;

namespace PointCloudWebViewer.Helpers
{
        public class PcdParser
        {
                public string filePath { get; set; }
                public Point3D[] Points3D { get; set; }
                public string PointCount { get; set; }

                public PcdParser(string filePath)
                {
                        this.filePath = filePath;
                }
```

```
                public void Parse()
                {
                        using (BinaryReader binaryReader = new BinaryReader(File.Open(filePath, FileMode.Open)))
                        {
                            int lenght = (int)binaryReader.BaseStream.Length;
                            string headerString = string.Empty;
                            string[] header;
                            int byteNumber = 0;

                            int pointsCount = 0;
                            string[] fields = null;
                            string[] types = null;
                            string data = string.Empty;

                            while (byteNumber < lenght)
                            {
                                headerString += binaryReader.ReadChar();
                                byteNumber++;
                                if (Regex.IsMatch(headerString, @"DATA \w+(\r\n|\n|\r\n|\n)"))
                                    break;
                            }

                            header = Regex.Split(headerString.Trim(), "\r\n|\n|\r\n|\n");
                #region Parse header
                for (int i = 0; i < header.Length; i++)
                {
                    string[] line = Regex.Split(header[i].Trim(), "\\s+");
                    if (line[0] == "FIELDS")
                    {
                        fields = new string[line.Length - 1];
                        for (int j = 1; j < line.Length; j++)
                        {
                            fields[j - 1] = line[j];
                        }
                        continue;
                    }
                    if (line[0] == "TYPE")
                    {
                        types = new string[line.Length - 1];
                        for (int j = 1; j < line.Length; j++)
                        {
                            types[j - 1] = line[j];
                        }
                        continue;
                    }
                    if (line[0] == "POINTS")
                    {
                        pointsCount = int.Parse(line[1]);
                        continue;
                    }
                    if (line[0] == "DATA")
                    {
                        data = line[1];
                        continue;
                    }
                }
                #endregion

                Points3D = new Point3D[pointsCount];

                            switch (data.ToLower())
                            {
                                case "ascii":
                                    {
        StringBuilder stringBuilder = new StringBuilder();

        while (byteNumber < lenght)
        {

        stringBuilder.Append(binaryReader.ReadChar());

            byteNumber++;

        }

        string[] pointsLines = Regex.Split(stringBuilder.ToString().Trim(), "\\r\\n|\\n|\r\n|\n");

        for (int i = 0; i < pointsLines.Length; i++)

        {

                string[] curentLine = Regex.Split(pointsLines[i], "\\s+");
                                        if (curentLine.Length >= 3)
                                        {
                                            Points3D[i] = new Point3D()

                                {

                                            X = double.Parse(curentLine[0]),

                                            Y = double.Parse(curentLine[1]),

                                            Z = double.Parse(curentLine[2])

                                        };
                                    }

        }
                                    }
                                    break;
                                case "binary":
                                    {
        float[] points = new float[pointsCount * fields.Length];
        int index = 0;

                                        while (index < points.Length)
                                        {
                                            for (int i = 0; i < fields.Length; i++)
                                            {
                                                if (types[i] == "F")
                                                {
                                                    if (byteNumber + 4 > lenght)
                                                    {
                                                        points[index++] = 0;
                                                    }
                                                    else
                                                    {
                                                        byte[] floatBytes = binaryReader.ReadBytes(sizeof(float));
                                                        points[index++] = BitConverter.ToSingle(floatBytes, 0);
                                                        byteNumber += sizeof(float);
                                                    }
                                                }
                                                if (types[i] == "I")
                                                {
                                                    if (byteNumber + 4 > lenght)
                                                    {
                                                        points[index++] = 0;
                                                    }
                                                    else
                                                    {
                                                        byte[] floatBytes = binaryReader.ReadBytes(sizeof(Int32));
                                                        points[index++] = BitConverter.ToInt32(floatBytes, 0);
                                                        byteNumber += sizeof(Int32);
                                                    }
                                                }
                                                if (types[i] == "U")
                                                {
                                                    if (byteNumber + 4 > lenght)
                                                    {
                                                        points[index++] = 0;
                                                    }
                                                    else
                                                    {
                                                        byte[] floatBytes = binaryReader.ReadBytes(sizeof(UInt32));
                                                        points[index++] = BitConverter.ToUInt32(floatBytes, 0);
                                                        byteNumber += sizeof(UInt32);
                                                    }
                                                }
                                            }
                                            int b = 0;
                                            int positionX = 0, positionY = 0, positionZ = 0;
                                            for (int j = 0; j < fields.Length; j++)
                                            {
                                                if (fields[j] == "x")
                                                {
                                                    positionX = j;
                                                }
                                                if (fields[j] == "y")
                                                {
                                                    positionY = j;
                                                }
                                                if (fields[j] == "z")
                                                {
                                                    positionZ = j;
                                                }
                                            }

        for (int i = 0; i < points.Length; i += fields.Length, b++)
        {

                    Points3D[b] = new Point3D()

                    {

                                X = points[i + positionX], //position points of X in description string
                                Y = points[i + positionY], //position points of Y in description string
```

```
                                Z = points[i + positionZ]
//position points of Z in description string
                                };

        }
                                }
                                break;
                            case
"binary_compressed":
                                {

        byte[]              decompressedData              =
Decompress(binaryReader.ReadBytes(lenght - byteNumber));

        float[] pointsX = new float[pointsCount];

        float[] pointsY = new float[pointsCount];

        float[] pointsZ = new float[pointsCount];

        if (types[0] == "F")

        {
                            int indexOfX = 0, indexOfY =
pointsCount * sizeof(float), indexOfZ = 2 * pointsCount *
sizeof(float);
                            for (int i = 0; i <
pointsCount; i++, indexOfX += sizeof(float), indexOfY +=
sizeof(float), indexOfZ += sizeof(float))
                            {
                                float       xValue      =
BitConverter.ToSingle(decompressedData, indexOfX);
                                float       yValue      =
BitConverter.ToSingle(decompressedData, indexOfY);
                                float       zValue      =
BitConverter.ToSingle(decompressedData, indexOfZ);
                                if
(xValue.Equals(float.NaN)   ||   yValue.Equals(float.NaN)   ||
zValue.Equals(float.NaN))
                                {
                                    pointsX[i] = 0;
                                    pointsY[i] = 0;
                                    pointsZ[i] = 0;
                                }
                                else
                                {
                                    pointsX[i] = xValue;
                                    pointsY[i] = yValue;
                                    pointsZ[i] = zValue;
                                }

                            }

        }

        if (types[0] == "I")

        {

                int b = 0;

                for (int i = 0; i < pointsCount; i++, b +=
sizeof(Int32))

                            pointsX[i]                    =
BitConverter.ToInt32(decompressedData, b);

                for (int i = 0; i < pointsCount; i++, b +=
sizeof(Int32))

                            pointsY[i]                    =
BitConverter.ToInt32(decompressedData, b);

                for (int i = 0; i < pointsCount; i++, b +=
sizeof(Int32))

                            pointsZ[i]                    =
BitConverter.ToInt32(decompressedData, b);

        }

        if (types[0] == "U")

        {

                int b = 0;

                for (int i = 0; i < pointsCount; i++, b +=
sizeof(UInt32))

                            pointsX[i]                    =
BitConverter.ToUInt32(decompressedData, b);

                for (int i = 0; i < pointsCount; i++, b +=
sizeof(UInt32))

                            pointsY[i]                    =
BitConverter.ToUInt32(decompressedData, b);

                for (int i = 0; i < pointsCount; i++, b +=
sizeof(UInt32))

                            pointsZ[i]                    =
BitConverter.ToUInt32(decompressedData, b);

        }

        for (int i = 0; i < pointsCount; i++)
        {
                Points3D[i] = new Point3D()

                {
                        X = pointsX[i],

                        Y = pointsY[i],

                        Z = pointsZ[i]
                };

        }
                                }
                                break;
                        }
                }
        }

        private byte[] Decompress(byte[] byteData)
        {
                UInt32[] sizes = new UInt32[2];

                Buffer.BlockCopy(byteData,      0,
sizes, 0, 8);
                int       compressedLength      =
(int)sizes[0];
                int       decompressedLength    =
(int)sizes[1];

                byte[]    compressedData    =    new
byte[compressedLength];
                byte[]    decompressedData    =    new
byte[decompressedLength];

                Buffer.BlockCopy(byteData,      8,
compressedData, 0, (int)sizes[0]);

                int inPtr = 0, outPtr = 0, ctrl,
len, reff;
                do
                {
                        ctrl                  =
compressedData[inPtr++];
                        if (ctrl < (1 << 5))
                        {
                                ctrl++;
                                if   (outPtr   +
ctrl > decompressedLength)
                                        throw
new Exception("Output buffer is not large enough");
                                if (inPtr + ctrl
> compressedLength)
                                        throw
new Exception("Invalid compressed data");

                                do
                                {
        decompressedData[outPtr++] = compressedData[inPtr++];
                                } while (--ctrl
> 0);
                        }
                        else
                        {
                                len = ctrl >> 5;
                                reff = outPtr -
((ctrl & 0x1f) << 8) - 1;
                                if   (inPtr   >=
compressedLength)
                                        throw
new Exception("Invalid compressed data");

                                if (len == 7)
                                {
                                        len   +=
compressedData[inPtr++];
                                        if
(inPtr >= compressedLength)
        throw new Exception("Invalid compressed data");
                                }

                                reff          -=
compressedData[inPtr++];

                                if (outPtr + len
+ 2 > decompressedLength)
                                        throw
new Exception("Output buffer is not large enough");
                                if (reff < 0)
                                        throw
new Exception("Invalid compressed data");
                                if    (reff    >=
outPtr)
                                        throw
new Exception("Invalid compressed data");

                                do
                                {
        decompressedData[outPtr++]                    =
decompressedData[reff++];
                                } while (--len +
2 > 0);
                        }
                } while (inPtr < compressedLength);
```

```
                    return decompressedData;
                }
        }
}
```

Лістинг PtsParser.cs:

```
using System.IO;
using System.Text.RegularExpressions;
using System.Windows.Media.Media3D;

namespace PointCloudWebViewer.Helpers
{
        public class PtsParser
        {
                private readonly string fullFileName;
                public Point3D[] Points { get; set; }
                public ColorARGB[] Colors { get; set; }
                public string PointCount { get; set; }

                public PtsParser(string fullFileName)
                {
                        this.fullFileName = fullFileName;
                }

                public void Parse()
                {
                        string[] ptsFile = File.ReadAllLines(fullFileName);
                        int pointCount = ptsFile.Length -1; //First row is not a point description
                        Points = new Point3D[pointCount];
                        Colors = new ColorARGB[pointCount];
                        string[] currentLine;

                        for (int i = 0; i < pointCount; i++)
                        {
                                currentLine = Regex.Split(ptsFile[i + 1], "\\s+");

                                Points[i] = new Point3D()
                                {
                                        X = double.Parse(currentLine[0]),
                                        Y = double.Parse(currentLine[1]),
                                        Z = double.Parse(currentLine[2])
                                };

                                if (currentLine.Length > 3)
                                {
                                        Colors[i] = new ColorARGB()
                                        {
        //Convert [-2048, 2048] intensity format to [0, 1]
                                                A = (double.Parse(currentLine[3]) + 2048) / 4096,
        //Convert [0, 255] color format to [0, 1]
                                                R = double.Parse(currentLine[4]) / 255f,
                                                G = double.Parse(currentLine[5]) / 255f,
                                                B = double.Parse(currentLine[6]) / 255f,
                                        };
                                }
                        }
                }
        }
}
```

Лістинг PtxParser.cs:

```
using System;
using System.IO;
using System.Text.RegularExpressions;
using System.Windows.Media.Media3D;

namespace PointCloudWebViewer.Helpers
{
        public class PtxParser
        {
                private readonly string filePath;
                public Point3D[] Points { get; set; }
                public ColorARGB[] Colors { get; set; }
                public string PointCount { get; set; }

                public PtxParser(string filePath)
                {
                        this.filePath = filePath;
                }

                public void Parse()
                {
                        string[] ptxFile = File.ReadAllLines(filePath);
            int pointCount = ptxFile.Length - 10; //First 10 rows is a header of file

            Points = new Point3D[pointCount];
            Colors = new ColorARGB[pointCount];

                        for (int i = 0; i < pointCount; i++)
                        {
                                string[] currentLine = ptxFile[i + 10].Split(new[] { " " }, StringSplitOptions.RemoveEmptyEntries); //first 10 rows is a header of file
                if (currentLine.Length >= 3)
                {
                    Points[i] = new Point3D()
                                        {
                                                X = double.Parse(currentLine[0]),
                                                Y = double.Parse(currentLine[1]),
                                                Z = double.Parse(currentLine[2])
                                        };
                }
                else
                {
                    Points[i] = new Point3D()
                    {
                        X = 0,
                        Y = 0,
                        Z = 0
                    };
                }

                if (currentLine.Length == 7)       //The color transmission format is argb. In the case where the point has a color, we have 7 fields: x, y, z, a, r, g, b
                {
                    Colors[i] = new ColorARGB()
                    {
                        A = double.Parse(currentLine[3]),
                        //Convert [0, 255] color format to [0, 1]
                        R = double.Parse(currentLine[4]) / 255f,
                        G = double.Parse(currentLine[5]) / 255f,
                        B = double.Parse(currentLine[6]) / 255f
                    };
                }
            }
        }
}
```

Лістинг TxtParser.cs:

```
using System.IO;
using System.Text.RegularExpressions;
using System.Windows.Media.Media3D;

namespace PointCloudWebViewer.Helpers
{
    public class TxtParser
    {
        private readonly string FullFileName;
        public Point3D[] Points { get; set; }
        public ColorARGB[] Colors { get; set; }
        public string PointCount { get; set; }

        public TxtParser(string fullFileName)
        {
            FullFileName = fullFileName;
        }

        public void Parse()
        {
            string[] txtFile = File.ReadAllLines(FullFileName);
            int pointCount = txtFile.Length;
            Points = new Point3D[pointCount];
            Colors = new ColorARGB[pointCount];
            string[] currentLine;

            for (int i = 0; i < pointCount; i++)
            {
                currentLine = Regex.Split(txtFile[i], "\\s+");

                Points[i] = new Point3D()
                {
                    X = double.Parse(currentLine[0]),
                    Y = double.Parse(currentLine[1]),
                    Z = double.Parse(currentLine[2])
                };

                if (currentLine.Length == 7)
                {
                    Colors[i] = new ColorARGB()
                    {
                        //Convert [-2048, 2048] intensity format to [0, 1]
                        A = (double.Parse(currentLine[3]) + 2048) / 4096,
                        //Convert [0, 255] color format to [0, 1]
                        R = double.Parse(currentLine[4]) / 255f,
                        G = double.Parse(currentLine[5]) / 255f,
                        B = double.Parse(currentLine[6]) / 255f,
                    };
                }
                if (currentLine.Length == 6)
                {
                    Colors[i] = new ColorARGB()
                    {
                        A = 1,
```

```
                    //Convert [0, 255] color format to [0,
1]
                    R   =   double.Parse(currentLine[3])   /
255f,
                    G   =   double.Parse(currentLine[4])   /
255f,
                    B   =   double.Parse(currentLine[5])   /
255f,
                };
            }
        }
    }
}
```

Лістинг XyzParser.cs:

```
using System;
using System.IO;
using System.Text.RegularExpressions;
using System.Windows.Media.Media3D;

namespace PointCloudWebViewer.Helpers
{
    public class XyzParser
    {
        private readonly string fullFileName;
        public Point3D[] Points { get; set; }
        public ColorARGB[] Colors { get; set; }
        public string PointCount { get; set; }

        public XyzParser(string fullFileName)
        {
            this.fullFileName = fullFileName;
        }

        public void Parse()
        {
            string[]          xyzFile          =
File.ReadAllLines(fullFileName);
            Points            =            new
Point3D[xyzFile.Length];
            Colors            =            new
ColorARGB[xyzFile.Length];

            for (int i = 0; i < xyzFile.Length;
i++)
            {
                MatchCollection
currentLine = Regex.Matches(xyzFile[i], @"\-?\d+(\.\d+)?");
                if (currentLine.Count > 3)
                {
                    Points[i] = new Point3D()
                    {
                        X           =
double.Parse(currentLine[0].Value),
                        Y           =
double.Parse(currentLine[1].Value),
                        Z           =
double.Parse(currentLine[2].Value)
                    };
                }
                else
                {
                    Points[i] = new Point3D()
                    {
                        X = 0,
                        Y = 0,
                        Z = 0
                    };
                }


                if (currentLine.Count == 6)
                    {
                        Colors[i]  =  new
ColorARGB()
                        {
                            A = 1,
                            R           =
double.Parse(currentLine[3].Value) / 255f,
                            G           =
double.Parse(currentLine[4].Value) / 255f,
                            B           =
double.Parse(currentLine[5].Value) / 255f,
                        };
                    }
                }
            }
        }
    }
}
```

Лістинг previewer-ctrl.js:

```
(function (THREE, angular) {
    'use strict';
    angular.module('pointCloudWebViewer').controller("Pre
viewerController", [
        '$http',  '$rootScope',  '$scope',  'enums',
'cameraOptions',  'settings',  'dataContext',  'Documentloader',
'uploadModel', 'downloadModel',
        function    ($http,    $rootScope,    $scope,    enums,
cameraOptions,    settings,    dataContext,    Documentloader,
uploadModel, downloadModel) {
            var    filename,    canvasContainer,
imageContainer,  previewContainer,  spinner,  camera,  controls,
scene,    renderer,    sphereRadius,    center,    windowWidth,
windowHeight, closePreview, activeCamera = { camera: camera };
            $scope.showPreviewContainer       =
false;
            $scope.showCanvasPreview = false;
            $scope.showImagePreview = false;
            $scope.previewContainerStyle;
            $scope.ImageSrc = "";

            $scope.setPreviewVisibility       =
function (visibility) {
        $rootScope.safeApply(function () {
        $scope.showPreviewContainer = visibility;
                });
            };
            $scope.setImagePreviewVisibility  =
function (visibility) {
        $rootScope.safeApply(function () {
        $scope.showImagePreview = visibility;
                });
            };
            $scope.setCanvasPreviewVisibility =
function (visibility) {
        $rootScope.safeApply(function () {
        $scope.showCanvasPreview = visibility;
                });
            };

            $scope.clousePreview = function ()
{
        $scope.setPreviewVisibility(false);
        $scope.setImagePreviewVisibility(false);
        $scope.setCanvasPreviewVisibility(false);
            };

            initSubscribers();
            moveElement();

            function moveElement() {
                var pos1 = 0, pos2 = 0,
pos3 = 0, pos4 = 0,
                    header          =
document.getElementsByClassName("preview-header")[0],
                    preview         =
document.getElementById("preview-container");
                if (header)
        header.onmousedown = dragMouseDown;
                else
        preview.onmousedown = dragMouseDown;

                function  dragMouseDown(e)
{
                    e    =    e    ||
window.event;
                    pos3            =
e.clientX;
                    pos4            =
e.clientY;
        document.onmouseup = closeDragElement;
        document.onmousemove = elementMove;
                }
                function elementMove(e) {
                    e    =    e    ||
window.event;
                    pos1  =  pos3  -
e.clientX;
                    pos2  =  pos4  -
e.clientY;
                    pos3            =
e.clientX;
                    pos4            =
e.clientY;

        $rootScope.safeApply(function () {
        $scope.previewContainerStyle = {
        "top": (preview.offsetTop - pos2) + "px",
        "left": (preview.offsetLeft - pos1) + "px"
                        };
                    });
                }
                function
closeDragElement() {
        document.onmouseup = null;
        document.onmousemove = null;
                }
            }

            function initSubscribers() {

        $rootScope.$on(settings.pointCloudViewerEvents.showPo
intCloudPreview, function (event, data) {
        $rootScope.safeApply(function () {
                    if
(spinner)
```

```
        spinner.stop();

        openModelInPreviewWindow(data.data.projectId,
data.data.id,        data.data.folderId,        data.label,
data.data.hasPreview, data.data.previewId);

        $scope.showPreviewContainer = true;

        $scope.showCanvasPreview = true;

        $scope.showImagePreview = false;

        previewContainer  =  document.getElementById("preview-
container");

        canvasContainer  =  document.getElementById("preview-
canvas");

        ImageContainer  =  document.getElementById("preview-
image");

        removeOldPreview();
                                                    spinner
= new Spinner(opts).spin(previewContainer);
                                                });
                                    });

        $rootScope.$on(settings.pointCloudViewerEvents.showPr
eviewImage, function (event, url) {
                                                $scope.ImageSrc
= url;
                                                spinner.stop();

        $scope.setImagePreviewVisibility(true);

        $scope.setCanvasPreviewVisibility(false);
                                                });
                            };

                            function
openModelInPreviewWindow(projectId,      fileId,      folderId,
fileName, hasPreview, previewId) {
                                $rootScope.filename      =
fileId;
                                if (!hasPreview) {

        checkNewPreview(projectId,      folderId,      fileId,
fileName);
                                }
                                else {

        showPreviewPicture(projectId, previewId);
                                }
                            }

                            function checkNewPreview(projectId,
folderId, fileId, fileName) {
                                var          url          =
settings.forgeProjectsURL + projectId + "/folders/" + folderId
+ "/contents";

        $http.defaults.headers.get = {};

        dataContext.getToken().then(function (token) {
                                                $http.get(url, {

        headers: {

        "Authorization": 'Bearer ' + token
                                                }

        }).success(function (data) {
                                                var
previewIsFound = false;
                                                for
(var i = 0; i < data.data.length; i++) {

        if (data.data[i].type == "items") {

                if  (data.data[i].attributes.displayName  ==
fileName + settings.previewImageFormat) {

                            showPreviewPicture(projectId,
data.data[i].id);

                            previewIsFound = true;

                            break;

                }

        }
                                                }
                                                if
(!previewIsFound) {

        //Show the accessToSavePreview modal window

        var access = checkAccessToSavePreview(projectId);

        if (access == null) {

                $rootScope.askAccessModal.show = true;


        $rootScope.$on(settings.pointCloudViewerEvents.modalW
indowResult, function (event, data) {

                            addToLocalStorage(projectId, data)

                            showPreview(projectId,      fileId,
folderId, fileName, data);

                            animate();

                    });
        }
        else {

                showPreview(projectId,   fileId,   folderId,
fileName, access);

                animate();
        }
                                                    }
                                                })
                                    });
                            }

                            function
checkAccessToSavePreview(projectId) {
                                var     projectAccess     =
localStorage.getItem(settings.localStoragePrefixes.projectAcce
ssToSavePreview + projectId);

                                if    (projectAccess    ===
null)
                                    return null;
                                else
                                    return
projectAccess === "true";
                            }

                            function     showPreview(projectId,
fileId, folderId, fileName, accessToSavePreview) {
                                initScene();


        $scope.setImagePreviewVisibility(false);

        $scope.setCanvasPreviewVisibility(true);

            var loader = new Documentloader();

        dataContext.getDownloadHref(projectId,
fileId).then(function (modelURL) {

        dataContext.getToken().then(function (token) {

        loader.load(modelURL, token, function (mesh) {

        if (settings.errors.indexOf(mesh) == -1) {

                mesh.name = fileName;

                scene.remove(scene.children[1]);//the  first
element is a points array

                scene.add(mesh);


                sphereRadius                                =
mesh.geometry.boundingSphere.radius;

                activeCamera.camera                         =
enums.mode.perspectiveCamera;

                cameraOptions.toView(camera,
enums.views.Initial, activeCamera, sphereRadius);

                center                                      =
mesh.geometry.boundingSphere.center;

                controls.target.set(center.x,      center.y,
center.z);

                controls.update();

                spinner.stop();

        }

        else {

                try {


        canvasContainer.append(settings.errorMessages.get(mes
h));

                    spinner.stop();

                } catch (e) {

                    spinner.stop();


        alert(settings.errorMessages.get(mesh));

                }

        }


        if (accessToSavePreview) {

                sendScreen(fileName, projectId, folderId);
```

```
        }

        else {


        $rootScope.$$listeners[settings.pointCloudViewerEvent
s.modalWindowResult] = [];

        }
                                                    });
                                                });
                                    });
                                }

                                function
showPreviewPicture(projectId, previewId) {
                                    removeOldPreview();

        downloadModel.setPreviewImage(imageContainer,
projectId, previewId);
                                }

                                function removeOldPreview() {
                                    while
(canvasContainer.firstChild) {

        canvasContainer.removeChild(canvasContainer.firstChil
d);
                                    }
                                    $scope.imageSrc = "";
                                }

                                function       sendScreen(fileName,
projectId, folderId) {

        renderer.setSize(settings.previewWight,
settings.previewHeight);
                                    renderer.render(scene,
camera);


        renderer.domElement.toBlob(function (blob) {
                                                var file = new
File([blob], fileName + settings.previewImageFormat);

        uploadModel.uploadModel(file, projectId, folderId);
                                                var    store    =
URL.createObjectURL(blob);
                                    });
                                }

                                function initScene() {
                                    scene          =          new
THREE.Scene();
                                    $rootScope.scene = scene;
                                    camera         =          new
THREE.PerspectiveCamera(settings.fovPerspective,
(settings.previewWight)       /       (settings.previewHeight),
settings.cameraNear, settings.cameraFar);
                                    controls       =       new
THREE.TrackballControls(camera, canvasContainer);

        cameraOptions.setDefaultControlsOptions(controls);
                                    scene.add(camera);
                                    renderer       =       new
THREE.WebGLRenderer({ antialias: true, preserveDrawingBuffer:
true });

        renderer.setClearColor(settings.defaultRenderBgColor)
;

        renderer.setPixelRatio(window.devicePixelRatio);

        renderer.setSize(settings.previewWight,
settings.previewHeight);
                                    while
(canvasContainer.firstChild)

        canvasContainer.removeChild(canvasContainer.firstChil
d);

        canvasContainer.appendChild(renderer.domElement);
                                }

                                function animate() {
                                    controls.update();
                                    renderer.render(scene,
camera);
                                    if
($scope.showPreviewContainer)

        requestAnimationFrame(animate);
                                    };

                                function
addToLocalStorage(projectId, access) {

        localStorage.setItem(settings.localStorageePrefixes.pr
ojectAccessToSavePreview + projectId, access)
                                }
                    ]]);
})(THREE, angular));
```

Лістинг tree-ctrl.js:

```
(function (angular) {
    'use strict';

angular.module("pointCloudWebViewer").controller("TreeControll
er",
        [
            "$rootScope",    "$scope",    "$window",    "$http",
"dataContext",     "dataProcessor",     "$timeout",     "enums",
"settings", "uploadModel", "downloadModel", "$log",
            function   ($rootScope,   $scope,   $window,   $http,
dataContext,    dataProcessor,    $timeout,    enums,    settings,
uploadModel, downloadModel, $log) {
                loadData();
                startEventListeners();
                $scope.modal = {};
                $scope.DisplayModalWindow = "EmptyModal";
                $scope.dataType = enums.dataType;
                $scope.type = enums.dataType.hub;
                $scope.pathToFolder = [];
                $rootScope.pathToFolder = $scope.pathToFolder;
                $scope.isClearUploadsShown = false;
                var selectedItemId;
                var isFolderNotLoading = true;
                $scope.folderOpened = false;
                $scope.isSorted = true;


window.mediator.subscribe(settings.pointCloudViewerEvents.star
tLoading, function (file) {

        $rootScope.$broadcast(settings.pointCloudViewerEvents
.addNotification, {                                             type:
settings.notification.types.process,

        processName: "uploading",

        fileName: file.name
                                                    });
                });

window.mediator.subscribe(settings.pointCloudViewerEvents.fini
shLoading, function (file) {

        $rootScope.$broadcast(settings.pointCloudViewerEvents
.removeNotificationNotification,

        settings.notification.types.process.eventNamePrefix +
file.name);


$rootScope.$broadcast(settings.pointCloudViewerEvents.addNotif
ication, {
                        type:
settings.notification.types.complete,
                        processName: "Uploading",
                        fileName: file.name
                    });
                    setTimeout(function () {

$rootScope.$broadcast(settings.pointCloudViewerEvents.removeNo
tificationNotification,

settings.notification.types.complete.eventNamePrefix          +
file.name);
                        $scope.$apply();
                    }, settings.notificationTime);
                });

                $scope.showPointCloudPreview = function (data)
{
        $rootScope.$broadcast(settings.pointCloudViewerEvents
.showPointCloudPreview, data);
                };

                /*
                 * @description Loads the object selected in
the file tree
                 * @typedef {TreeModel} selectedItem Selected
item in the file tree
                 * @param {boolean} isFromPath Indicates is
function called by clicking on item in files tree or in path
line
                 */
                $scope.load    =    function    (selectedItem,
isFromPath) {
                    $scope.folderOpened = false;
                    if (isFolderNotLoading) {
                        $scope.previousSelectedItem          =
selectedItem;
                        $scope.previousIsFromPath            =
isFromPath;
                        isFolderNotLoading = false;
                        selectedItem.loading = isFromPath;
                        $timeout(function () {
                            if    (selectedItem.data.type    ===
enums.dataType.hub) {
                                selectedItemId               =
selectedItem.data.id;
                                $scope.selectedItemType      =
selectedItem.data.type;

dataContext.getProjectTreeElement(selectedItem.data.id).then(f
unction (element) {
                                    if
($scope.pathToFolder.length == 0) {

$scope.pathToFolder.push({  name:  "../",  id:  "Init",  item:
$scope.tree });
                                    }
                                    $scope.tree.some(function
(item) {
                                        if   (item.data.id   ==
selectedItemId) {
                                            item.children    =
dataProcessor.processProjectData(element);
```

```
item.parentFolderId = selectedItem.data.folderId;

$scope.pathToFolder.push({ name: selectedItem.label, item:
selectedItem });
                                    $scope.isHubsLevel
= false;
                                } else {
                                    item.isItemShown =
true;
                                }
                            });
                            selectedItem.isHubHiden =
true;
                            $scope.isPathLoading =
false;
                            selectedItem.loading =
false;
                            isFolderNotLoading = true;
                        })
                    } else {
dataContext.getFolderTreeElement(selectedItem.data.projectId,
selectedItem.data.folderId).then(function (element) {
                            var folderId =
selectedItem.data.folderId || selectedItem.data.id;
                            $scope.folderId =
folderId;
                            $scope.projectId =
selectedItem.data.projectId
                            $rootScope.isFolderActive
= true;
                            $scope.folderOpened =
true;
                            $scope.currentProjectId =
selectedItem.data.projectId;
                            $scope.currentFolderId =
folderId;
                            $scope.selectedItemType =
selectedItem.data.type;
$scope.selectedItemFolderId = selectedItem.uid;
                            $scope.tree.some(function
(item) {
                                if (item.data.id ==
selectedItemId) {
                                    item.children =
dataProcessor.processFolderData(element, $scope.folderId);
$scope.lastLoadedFolders = [];
                                    for (var i = 0; i
< element.folders.length; i++) {
$scope.lastLoadedFolders.push(element.folders[i]['Name']);
                                    }
item.parentFolderId = selectedItem.data.folderId;

        if($scope.previousIsfromPath)

        $scope.pathToFolder.push({ name: selectedItem.label,
item: selectedItem });
                                    $scope.isHubsLevel
= false;
                                    return true;
                                }
                            });
                            selectedItem.loading =
false;
                            $scope.isPathLoading =
false;
                            isFolderNotLoading = true;
                        });
                    }
                });
                selectedItem.expanded = true;
            }
        }

        /*
         * @description Loads folder selected in path
line
         * @typedef {object} selectedItem Selected
folder in path line
         */
        $scope.onPathSelect = function (selectedItem)
{
            if (isFolderNotLoading) {
                isFolderNotLoading = false;
                $scope.isPathLoading = true;
                $timeout(function () {
                    if (selectedItem.id == "Init") {
                        isFolderNotLoading = true;
                        loadData();
                        $scope.selectedItemType =
enums.dataType.hub;
                        $scope.pathToFolder.splice(1,
$scope.pathToFolder.length);
                    } else {
                        isFolderNotLoading = true;
                        $scope.load(selectedItem.item,
true);
$scope.pathToFolder.splice(getPathIndex($scope.pathToFolder,
selectedItem.name), $scope.pathToFolder.length);
                    }
                });
            }
        }

        $scope.sort = function () {
            if ($scope.isSorted) {
                $scope.tree[0].children.sort(function
(a, b) {
                    var x = a.label.toLowerCase();
                    var y = b.label.toLowerCase();
                    if (x < y) { return 1; }
                    if (x > y) { return -1; }
                    return 0;
                });
                $scope.tree[0].children.sort(function
(a, b) {
                    var x = a.data.type.toLowerCase();
                    var y = b.data.type.toLowerCase();
                    if (x < y) { return 1; }
                    if (x > y) { return -1; }
                    return 0;
                });
                $scope.isSorted = false;
            }
            else {
                $scope.tree[0].children.sort(function
(a, b) {
                    var x = a.label.toLowerCase();
                    var y = b.label.toLowerCase();
                    if (x < y) { return -1; }
                    if (x > y) { return 1; }
                    return 0;
                });
                $scope.tree[0].children.sort(function
(a, b) {
                    var x = a.data.type.toLowerCase();
                    var y = b.data.type.toLowerCase();
                    if (x < y) { return 1; }
                    if (x > y) { return -1; }
                    return 0;
                });
                $scope.isSorted = true;
            }

        }

        /*
         * @description Assigns values of selected
item params to $scope variables.
         * @typedef {TreeModel} selectedItem Selected
item in files tree
         */
        $scope.select = function (selectedItem) {
            if (selectedItem.data.type !=
enums.dataType.file) {
                $scope.previousSelectedItem =
selectedItem;
                $scope.type = enums.dataType.folder;
                if (selectedItem.data.type ==
enums.dataType.hub) {
                    $scope.type = enums.dataType.hub;
                }
                else {
                    $scope.selectedItemType =
enums.dataType.folder;
                }
                $rootScope.isFolderActive = false;
            } else {
                $scope.selectedItemType =
selectedItem.data.type;
                $scope.type = enums.dataType.file;
                $scope.downloadLink =
selectedItem.data.downloadLink;
                $scope.projectId =
selectedItem.data.projectId;
                $scope.fileId = selectedItem.data.id;
                $scope.fileName = selectedItem.label;
                $scope.fileSize =
selectedItem.data.fileSize;
            }
            selectedItem.selected = true;
            if ($scope.prevItem != null &&
selectedItem != $scope.prevItem) {
                $scope.prevItem.selected = false;
            }
            $scope.prevItem = selectedItem;
        }

        /*
         * @description Initializes hubs data in the
files tree
         */
        function loadData() {
            if (isFolderNotLoading ||
isFolderNotLoading === undefined) {
                isFolderNotLoading = false;
dataContext.getInitialHubs().then(function (hubs) {
                    if (hubs.data.length === 0)
$scope.isHubsEmpty = true;
                    $scope.tree =
dataProcessor.processHubsData(hubs);
                    $scope.selectedItemType =
enums.dataType.hub;
                    $scope.isPathLoading = false;
                    $scope.hubsLoading = true;
                    $scope.isHubsLevel = true;
                    isFolderNotLoading = true;
                });
            }
        };

        /*
```

```
                 * @description Returns index of selected path
from path array
                 * @param {Array} items Array of paths
                 * @param {string} label Name of selected path
                 * @returns {number} index
                 */
                function getPathIndex(items, label) {
                    var index = items.length;
                    items.some(function (item, i) {
                        if (item.name == label) {
                            index = i;
                            return true;
                        }
                    });
                    return index;
                }

                /*
                 * @typedef {object} file
                 */
                $scope.addToBranch = function (file) {
                    dataProcessor.addFileToBranch(file,
$scope.tree);
                }

                /*
                 * @typedef {object} file
                 */
                $scope.updateFileInBranch = function (file) {

dataProcessor.updateFileInBranch(file.fileId,     file.version,
$scope.tree);
                }

                $scope.downloadModal = function () {
                    if       ($scope.selectedItemType       ==
enums.dataType.file) {

        downloadModal.download($scope.projectId,
$scope.fileId, $scope.fileName);
                    }
                }

                /**
                 *  @description  Make  request  for  creating
mail, if inputed data is valid
                 */
                                        $scope.submitMailModal   =
function () {
                    if ($scope.modal.mail.$valid) {

$scope.openProcessMailModal($scope.fileName);

dataContext.getSendMail($scope.projectId,       $scope.fileId,
$scope.fileName,           $scope.modal.mail.email.$viewValue,
$scope.modal.mail.name.$viewValue).then(
                            function (data) {

$scope.openSuccesMailModal($scope.fileName);
                            },
                            function () {

$scope.openFailMailModal($scope.fileName);
                            })
                        $scope.closeModalForm();
                        if          ($scope.fileSize          >
settings.maxDownloadingFileSize) {
                            $scope.DisplayModalWindow        =
"SendLargeMailModal";
                            console.log("Send large mail");
                        }
                    } else {
                        console.log("not valid");
                    }
                };

                $scope.submitDownloadMailModal = function () {
                    if ($scope.modal.mail.$valid) {

dataContext.getSendFileByMail($scope.projectId,  $scope.fileId,
$scope.fileName, $scope.modal.mail.email.$viewValue);
                        $scope.closeModalForm();

$rootScope.$broadcast(settings.pointCloudViewerEvents.addNotif
ication, {
                            type:
settings.notification.types.downloadMail,
                            processName: "sent",
                            fileName: $scope.fileName
                        });
                        setTimeout(function () {

$rootScope.$broadcast(settings.pointCloudViewerEvents.removeNo
tificationNotification,

settings.notification.types.downloadMail.eventNamePrefix      +
$scope.fileName);
                            $scope.$apply();
                        }, settings.notificationTime);
                    } else {
                        console.log("not valid");
                    }
                };

                $scope.submitLinkMailModal = function () {
                    if ($scope.modal.mail.$valid) {

dataContext.getLinkSendMail($scope.projectId,  $scope.folderId,
$scope.fileId,                          $scope.fileName,
$scope.modal.mail.email.$viewValue);
                        $scope.closeModalForm();

$rootScope.$broadcast(settings.pointCloudViewerEvents.addNotif
ication, {
                            type:
settings.notification.types.downloadMail,
                            processName: "sent",
                            fileName: $scope.fileName
                        });
                        setTimeout(function () {

$rootScope.$broadcast(settings.pointCloudViewerEvents.removeNo
tificationNotification,

settings.notification.types.downloadMail.eventNamePrefix      +
$scope.fileName);
                            $scope.$apply();
                        }, settings.notificationTime);
                    } else {
                        console.log("not valid");
                    }
                };

                /**
                 *  @description  Make  request  for  creating
folder
                 */
                $scope.submitFolderModal = function (name) {
                    if (!$scope.modal.Folder.$invalid) {

dataContext.createFolder($scope.projectId,     $scope.folderId,
$scope.modal.Folder.name.$viewValue).then(

        function (data) {

                $scope.load($scope.previousSelectedItem,
false);


        $rootScope.$broadcast(settings.pointCloudViewerEvents
.addNotification, {
                            type:
settings.notification.types.complete,
                            processName: "Creating",
                            fileName: name
                        });
                        setTimeout(function () {

$rootScope.$broadcast(settings.pointCloudViewerEvents.removeNo
tificationNotification,

settings.notification.types.complete.eventNamePrefix + name);
                            $scope.$apply();
                        }, settings.notificationTime);
                        },
                        function () {


        $rootScope.$broadcast(settings.pointCloudViewerEvents
.addNotification, {
                            type:
settings.notification.types.fail,
                            processName: "Creating",
                            fileName: name

                    });
                        })
                    $scope.closeModalForm();
                }
                 else   $scope.modal.mail.email.$invalid   =
true;
            }

            $scope.modal.Folder = {};
            $scope.modal.Folder.validName = "valid";

            $scope.validationFolderModal     =     function
(folderName) {
                if (folderName == undefined || folderName
== '' || folderName.length > 18) {
        $scope.modal.Folder.validName = "char";
        $scope.modal.Folder.$invalid = true;
                                                return;
                                            }
                                            if
($scope.lastLoadedFolders.indexOf($scope.modal.Folder.name.$vi
ewValue) > -1) {
        $scope.modal.Folder.validName = "exist";
        $scope.modal.Folder.$invalid = true;
                                                return;
                                            }
                    $scope.modal.Folder.validName = "valid";
                    $scope.modal.Folder.$invalid = false;
                    return folderName.replace(/ {1,}/g, " ");
                }

                //MODAL WINDOWS
                                        $scope.openMailModal     =
function () {                                   if
($scope.selectedItemType == 'file')
```

```
        $scope.DisplayModalWindow = 'SendMailModal';
            }
            $scope.openFolderModal = function () {
                $scope.DisplayModalWindow = 'CreateFolder';
            }
            $scope.openProcessMailModal = function (fileName) {
                $rootScope.$broadcast(settings.pointCloudViewerEvents.addNotification, {
                    type: settings.notification.types.mailProcess,
                    processName: "sending",
                    fileName: fileName
                });
            }
            $scope.openSuccesMailModal = function (fileName) {
                $rootScope.$broadcast(settings.pointCloudViewerEvents.removeNotificationNotification,
                    'mailProcessNotififcation-' + fileName);

                $rootScope.$broadcast(settings.pointCloudViewerEvents.addNotification, {
                    type: settings.notification.types.complete,
                    processName: "Sending",
                    fileName: fileName
                });
            }
            $scope.openFailMailModal = function (fileName) {
                $rootScope.$broadcast(settings.pointCloudViewerEvents.removeNotificationNotification,
                    'mailProcessNotififcation-' + fileName);

                $rootScope.$broadcast(settings.pointCloudViewerEvents.addNotification, {
                    type: settings.notification.types.fail,
                    processName: "Sending",
                    fileName: fileName
                });
            }

            /**
             *  @description  EmptyModal  is  empty  ng-template for closing modals.
             */
            $scope.closeModalForm = function () {
                $scope.DisplayModalWindow = $scope.EmptyModal;
            };

            /**
             * @description Make get request for creating pdf from model and download it.
             */
            $scope.downloadPDF = function () {
                if ($scope.fileName != undefined) {
                    if ($scope.fileSize < settings.maxDownloadingFileSize) {
                        $rootScope.$broadcast(settings.pointCloudViewerEvents.addNotification, {
                            type: settings.notification.types.process,
                            processName: "Exporting",
                            fileName: $scope.fileName
                        });
                        downloadModel.createPDF($scope.projectId, $scope.fileId, $scope.fileName);
                    }
                    else if ($scope.fileSize > settings.maxDownloadingFileSize && $scope.fileSize < settings.maxMailFileSize) {
                        if ($scope.selectedItemType == 'file') {
                            $scope.DisplayModalWindow = 'SendDownloadMail';
                        }
                    }
                    else if ($scope.fileSize > settings.maxMailFileSize) {
                        if ($scope.selectedItemType == 'file') {
                            $scope.DisplayModalWindow = 'SendLinkMail';
                        }
                    }
                }
            }

            /*
             * @description Returns boolean for ng-show that indicates should the icon for this item be shown or not
             * @param {string} iconName
             * @typedef {TreeModel} data
             * @returns {boolean}
             */
            $scope.isShown = function (iconName, data) {
                switch (iconName) {
                    case enums.treeIcons.plus:
                        return (data.children != null) && (!data.expanded);
                    case enums.treeIcons.minus:
                        return (data.children != null) && (data.expanded);
                    case enums.treeIcons.close:
                        return !data.loading && data.data.type != enums.dataType.file && data.children == null;
                }
            }

            /*
             * @typedef {TreeModel} data
             */
            $scope.loadFolder = function (data) {
                if (data.data.type != enums.dataType.file) {
                    $scope.load(data, true);
                }
            }

            // pointCloudViewerEvents

            /*
             *  @description  pointCloudViewerEvents  for highlighting drop aera
             */
            function startEventListeners() {
                var dropzoneId = "filesTree";
                $window.addEventListener("dragenter", function (e) {
                    if (e.target.id != dropzoneId) {
                        e.preventDefault();
                        e.dataTransfer.effectAllowed = "none";
                        e.dataTransfer.dropEffect = "none";
                    }
                }, false);
                $window.addEventListener("dragover", function (e) {
                    angular.element("#filesTree").addClass("borderDropZone");
                    if (e.target.id != dropzoneId) {
                        e.preventDefault();
                        e.dataTransfer.effectAllowed = "none";
                        e.dataTransfer.dropEffect = "none";
                    }
                });
                $window.addEventListener("drop", function (e) {
                    if (e.target.id != dropzoneId) {
                        e.preventDefault();
                        e.dataTransfer.effectAllowed = "none";
                        e.dataTransfer.dropEffect = "none";
                    }
                });
                $window.addEventListener("dragleave", function () {
                    angular.element("#filesTree").removeClass("borderDropZone");
                });
            }

            /*
             * @typedef {File} file
             * @param {object} responseData
             */
            $rootScope.$on(settings.pointCloudViewerEvents.updateFileInBranch, function (event, file, responseData) {
                angular.element(file.previewElement.children[2]).remove();
                $scope.updateFileInBranch(responseData);
                file.previewElement.classList.add("dz-success");
            });

            /*
             * @typedef {File} file
             * @typedef {object} responseData
             */
            $rootScope.$on(settings.pointCloudViewerEvents.addToBranch, function (event, file, responseData) {
                //it doesn't work
                if (file.name.slice(-4).toLowerCase() != settings.previewImageFormat) {
                    angular.element(file.previewElement.children[2]).remove();
                    $scope.addToBranch(responseData);
                    file.previewElement.classList.add("dz-success");
                }
            });

            /*
             * @param {string} error
             */
```

```
        $rootScope.$on(settings.pointCloudViewerEvents.errorWhileUploading, function (event, error) {

alert(settings.errorMessages.get('errorWhileFileUpl') + error);
                });

                /*
                 * @description Clears all uploads in UI
                 */
                $scope.clearUploads = function () {
                    $("#previews").empty();
                    $scope.isClearUploadsShown = false;
                }

                /*
                 *  @description  Updates  progress  of  file upload in UI
                 * @typedef {File} file
                 * @param {number} progress
                 * @returns {Array} _results
                 */

$rootScope.$on(settings.pointCloudViewerEvents.updateProgress, function (event, file, progress) {
                    if (progress >= 100)

window.mediator.publish(settings.pointCloudViewerEvents.finishLoading, file);

                    var node, _i, _len, _ref, _results;
                    if (file.previewElement) {
                        _ref = file.previewElement.querySelectorAll("[data-dz-uploadprogress]");
                        _results = [];
                        for (_i = 0, _len = _ref.length; _i < _len; _i++) {
                            node = _ref[_i];
                            if (node.nodeName === 'PROGRESS') {
                                _results.push(node.value = progress);
                            } else {
                                _results.push(node.style.width = "" + progress + "%");
                            }
                        }
                        return _results;
                    }
                });

                //DROPZONE
                $scope.dropzoneConfigAera = getDropzoneConfig(false);
                $scope.dropzoneConfigButton = getDropzoneConfig(true);

                function getDropzoneConfig(clickable) {
                    return {
                        'options': {
                            url: ' ',
                            method: settings.dropzoneMethod,
                            acceptedFiles: settings.dropzoneAcceptedFilesFilter,
                            parallelUploads: settings.dropzoneParallelUploads,
                            maxFilesize: settings.dropzonemaxFilesize,
                            dictDefaultMessage: "Upload",
                            previewsContainer: "#previews",
                            clickable: clickable,
                            autoProcessQueue: false
                        },
                        'eventHandlers': {
                            'addedfile': function (file) {
                                if ($scope.selectedItemType == enums.dataType.hub) {
                                    if ($scope.tree.length == 0) {

alert(settings.errorMessages.get("noHubs"));
                                    } else {

alert(settings.errorMessages.get('ImpossibleToUpload'));
                                    }
                                    this.removeAllFiles(true);
                                    $scope.isClearUploadsShown = false;
                                    $scope.$apply();
                                    return false;
                                } else {
                                    $scope.isClearUploadsShown = true;
                                    $scope.$apply();
                                    var fileExtension = file.name.slice(-4).toLowerCase();
                                    if (settings.supportedFormats.indexOf(fileExtension) != -1) {

window.mediator.publish(window.pointCloudViewerEvents.startLoading, file);

uploadModel.uploadModel(file, $scope.currentProjectId, $scope.currentFolderId);
                                    }
                                };
                            }
                        }
                    }
                }
            ]);
})(angular);
```

Лістинг document-loader.js:

```
(function (THREE, angular) {
        'use strict';

angular.module('pointCloudWebViewer').factory('Documentloader', ['$rootScope', 'settings', 'dataContext', 'lasDocumentParser', 'lazDocumentParser', 'xyzDocumentParser', 'ptsDocumentParser', 'ptxDocumentParser', 'pcdDocumentParser', 'txtDocumentParser', 'e57DocumentParser',
        function ($rootScope, settings, dataContext, lasDocumentParser, lazDocumentParser, xyzDocumentParser, ptsDocumentParser, ptxDocumentParser, pcdDocumentParser, txtDocumentParser, e57DocumentParser) {
        var model;
        THREE.Documentloader = function (manager) {
                        this.manager = (manager !== undefined) ? manager : THREE.DefaultLoadingManager;
                        this.littleEndian = true;
                };

        Object.assign(THREE.Documentloader.prototype, THREE.EventDispatcher.prototype, {
                load: function (url, token, onLoad) {
                    var scope = this;
                    var extension = url.substring(url.lastIndexOf("."), url.lastIndexOf(".") + 4).toLowerCase();
                                if (extension == '.e57') {

                        dataContext.getE57Data(url, function (data, url) {
                                onLoad(scope.parseE57(data.Points, data.Colors, url));
                                });
                    }
                                else {
                            dataContext.downloadFile(url, token).then(function (data) {
                                if (data.byteLength > settings.warningFileSize) {

$rootScope.$broadcast(settings.pointCloudViewerEvents.enableWarning);
                                }
                                setTimeout(function () {

                                    switch (extension) {
                                        case ".pcd":

onLoad(scope.parsePcd(data, url));
                                            break;
                                        case ".ptx":

onLoad(scope.parsePtx(data, url));
                                            break;
                                        case ".pts":

onLoad(scope.parsePts(data, url));
                                            break;
                                        case ".xyz":

onLoad(scope.parseXyz(data, url));
                                            break;
                                        case ".ptg":
                                            alert('Not implemented!')

//onLoad(scope.parsePtg(data, url));
                                            break;
                                        case ".las":

onLoad(scope.parseLas(data, url));
                                            break;
                                        case ".laz":
                                            scope.parseLaz(data, url).then(function (data) {
                                                onLoad(data);
                                            });
                                            break;
                                        case ".txt":

onLoad(scope.parseTxt(data, url));
                                            break;
                                    }
                                }, 5);
                            });
                    }
                },

                parsePcd: function (data, url) {
                    model = pcdDocumentParser.Parse(data, url);
                    return createMesh(model.positions, model.colors, model.name);
                },

                parsePtx: function (data, url) {
                    model = ptxDocumentParser.Parse(data, url);
                    return createMesh(model.positions, model.colors, model.name);
                },

            parsePts: function (data, url) {
                model = ptsDocumentParser.Parse(data, url);
                return createMesh(model.positions, model.colors, model.name);
            },
```

```
                parseE57:    function    (points,
inputColors, url) {
                    model    =    e57DocumentParser.Parse(points,
inputColors, url);
                    return              createMesh(model.positions,
model.colors, model.name);
                },

                parsePtg: function (data, url) {
                    //not implemented
                    alert("not implemented");
                    var      textData      =
this.binarryToStr(data);
                    var      headerEnd      =
"header_end";
                    var     headerLenght     =
textData.indexOf(headerEnd) + headerEnd.length;
                    var      header      =
textData.substring(0, headerLenght);
                    var     pointsCount     =
+header.match(/\d+\s+points/g)[0].match(/\d+/g)[0];
                    var  textDataWithoutHeader
= textData.substring(headerLenght)
                },

                parseXyz: function (data, url) {
                    model = xyzDocumentParser.Parse(data, url);
                    return              createMesh(model.positions,
model.colors, model.name);
                },

                parseLas: function (data, url) {
                    model = lasDocumentParser.Parse(data, url);
                    return              createMesh(model.positions,
model.colors, model.name);
                },

                parseTxt: function (data, url) {
                    model = txtDocumentParser.Parse(data, url);
                    return              createMesh(model.positions,
model.colors, model.name);
                },

                parseLaz: function (data, url) {
                    model = lazDocumentParser.Parse(data, url);
                    return model.then(function (data) {
                        return          createMesh(data.positions,
data.colors, data.name);
                    });
                }
            });

            function createMesh(positions, colors, name,
PCDheader) {
                var      geometry      =      new
THREE.BufferGeometry();
                if (positions)
                    geometry.addAttribute('position',                new
THREE.BufferAttribute(positions, 3));
                if (colors)
                    geometry.addAttribute('color',                   new
THREE.BufferAttribute(colors, 3));

                geometry.computeBoundingSphere();

                var      material      =      new
THREE.PointsMaterial({
                    size: 0.005,
                    vertexColors: !(colors ==
false)
                });
                if (!colors)
                    material.color.setHex(0x7fff7f);

                var       mesh       =       new
THREE.Points(geometry, material);
                mesh.name = name;
                mesh.PCDheader = PCDheader;
                if (mesh.geometry.boundingSphere.radius > 50) {
                    mesh.material.size      =
0.35;
                }
                else                                    if
(mesh.geometry.boundingSphere.radius > 100) {
                    mesh.material.size      =
0.55;
                }
                return mesh;
            }

            return THREE.DocumentLoader;
        }]);
}(THREE, angular));
```

Лістинг mediator.js:

```
;(function () {
    var subscribe = function (event, fn) {
        if (!mediator.events[event])
            mediator.events[event] = [];
        mediator.events[event].push({ context: this, callback:
fn });
        return this;
    },
        publish = function (event) {
            if (!mediator.events[event]) return false;
            var  args  =  Array.prototype.slice.call(arguments,
1);
            for (var i = 0, l = mediator.events[event].length;
i < l; i++) {
                var subscription = mediator.events[event][i];
subscription.callback.apply(subscription.context, args);
            }
            return this;
        };

        window.mediator = {
        events: {},
        publish: publish,
        subscribe: subscribe,
        installTo: function (obj) {
            obj.subscribe = subscribe;
            obj.publish = publish;
        }
        };

        return window.mediator;
}());
```

Лістинг download-model.js:

```
(function (angular) {
        'use strict';
        angular.module("pointCloudWebViewer").factory("downlo
adModel", ["$http", "dataContext", "settings", "$rootScope",
function ($http, dataContext, settings, $rootScope) {
                var factory = [];

                factory.download  =  function  (projectId,
fileId, fileName) {
            $http.defaults.headers.get = {};
            if (fileName != undefined && fileId != undefined)
{
                return  dataContext.getDownloadHref(projectId,
fileId).then(function (url) {
                    dataContext.getToken().then(function
(token) {
                        $http.get(url, {
                            headers: {
                                "Authorization": "Bearer " +
token
                            },
                            responseType: "arraybuffer"
                        }).success(function (data) {
                            setCacheHeaders();
                            var  model  =  new  Blob([data],  {
type: "application/pcd" });
                            saveAs(model, fileName);

$rootScope.$broadcast(settings.pointCloudViewerEvents.addNotif
ication, {
                                type:
settings.notification.types.complete,
                                processName: "Downloading",
                                fileName: fileName
                            });
                            setTimeout(function () {

$rootScope.$broadcast(settings.pointCloudViewerEvents.removeNo
tificationNotification,

settings.notification.types.complete.eventNamePrefix          +
fileName);
                                $scope.$apply();
                            }, settings.notificationTime);
                        }).error(function () {

alert(settings.errorMessages.get("errorWhileFileDownld"));
                        });
                    });
                });
            }
                        //return
dataContext.getDownloadHref(projectId,    fileId).then(function
(url) {
                        //
        dataContext.getToken().then(function (token) {
                        //                  $http.get(url, {
                        //
        headers: {
                        //
        "Authorization": "Bearer " + token
                        //                              },
                        //
        responseType: "arraybuffer"
                        //
        }).success(function (data) {
                        //
        setCacheHeaders();
                        //                              var
model = new Blob([data], { type: "application/pcd" });
                        //
        saveAs(model, fileName);
                        //
        }).error(function () {
                        //
        alert(settings.errorMessages.get("errorWhileFileDownld
"));
                        //                  });
                        //      });
                        //});
                }

                factory.setPreviewImage      =      function
(previewContainer, projectId, previewId) {
                        $http.defaults.headers.get = {};
```

```
                                return
dataContext.getDownloadHref(projectId,
previewId).then(function (url) {

            dataContext.getToken().then(function (token) {
                                                $http.get(url, {

            headers: {

            "Authorization": "Bearer " + token
                                                },

            responseType: "arraybuffer"

            }).success(function (data) {

            setCacheHeaders();

                                                var
model = new Blob([data], { type: "image/png" });
                                                var
urlCreator = window.URL || window.webkitURL;
                                                var
imageUrl = urlCreator.createObjectURL(model);


            $rootScope.$broadcast(settings.pointCloudViewerEvents
.showPreviewImage, imageUrl);

            }).error(function () {

            alert(settings.errorMessages.get('errorWhileFileDownld
'));
                                                });
                                    });
                                });
                    }

            factory.createPDF = function (projectId, fileId,
fileName) {
                var nameOfFile = fileName.substring(0,
fileName.lastIndexOf("."));
                if (fileName != undefined && fileId != undefined)
{
                    return dataContext.getDownloadPdf(projectId,
fileId, fileName).then(
                        function (data) {
                            var file = new Blob([data], { type:
'application/pdf' });
                            saveAs(file, nameOfFile + '.pdf');

                            setTimeout(function () {
                                $rootScope.$broadcast(

settings.pointCloudViewerEvents.removeNotificationNotification
,

settings.notification.types.process.eventNamePrefix +
fileName);


$rootScope.$broadcast(settings.pointCloudViewerEvents.addNotif
ication, {
                                    type:
settings.notification.types.complete,
                                    processName: "Exporting",
                                    fileName: fileName
                                });
                            }, 500);

                        }, function () {
                            setTimeout(function () {
                                $rootScope.$broadcast(

settings.pointCloudViewerEvents.removeNotificationNotification
,

settings.notification.types.process.eventNamePrefix +
fileName);


$rootScope.$broadcast(settings.pointCloudViewerEvents.addNotif
ication, {
                                    type:
settings.notification.types.fail,
                                    processName: "Exporting",
                                    fileName: fileName
                                });
                            }, 500);
                        });
                }

                            //return
dataContext.getDownloadPdf(projectId, fileId, fileName).then(
                            //        function (data) {
                            //            var file = new
Blob([data], { type: 'application/pdf' });
                            //            saveAs(file,
fileName + '.pdf');

                            //
            setTimeout(function () {
                            //
            $rootScope.$broadcast(
                            //
            settings.pointCloudViewerEvents.removeNotificationNot
ification,
                            //
            settings.notification.types.process.eventNamePrefix +
fileName);
                            //
            $rootScope.$broadcast(settings.pointCloudViewerEvents
.addNotification, {
                            //
            type: settings.notification.types.complete,
                            //
            processName: "Exporting",
                            //
            fileName: fileName
                            //                });
                            //            }, 500);

                            //        }, function () {
                            //
            setTimeout(function () {
                            //
            $rootScope.$broadcast(
                            //
            settings.pointCloudViewerEvents.removeNotificationNot
ification,
                            //
            settings.notification.types.process.eventNamePrefix +
fileName);
                            //
            $rootScope.$broadcast(settings.pointCloudViewerEvents
.addNotification, {
                            //
            type: settings.notification.types.fail,
                            //
            processName: "Exporting",
                            //
            fileName: fileName
                            //                });
                            //            }, 500);
                            //        });
                }

                factory.downloadByURL = function (url,
fileName) {
                    $http.defaults.headers.get = {};
                    return
dataContext.getToken().then(function (token) {
                                $http.get(url, {
                                    headers: {

            "Authorization": "Bearer " + token
                                    },
                                    responseType:
'arraybuffer'
                                }).success(function
(data) {

            setCacheHeaders();
                                    var model = new
Blob([data], { type: 'application/pcd' });
                                    saveAs(model,
fileName);
                                }).error(function () {

            alert(settings.errorMessages.get('errorWhileFileDownld
'));
                                });
                    });
                }

                function setCacheHeaders() {
                    $http.defaults.headers.get['If-
Modified-Since'] = 'Mon, 26 Jul 1997 05:00:00 GMT';
                }
                return factory;
            }]);
})(angular);
```

Tue Nov 23 19:23:59 EET 2021, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 34.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 14%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 97110 Название: Метод візуалізації хмар точок «Web point cloud viewer» для прийняття контрольованих людиною критично-безпекових рішень Добавлено в БД: 2021-11-23 Авторы: Н.М. Тростинський Руководители: О.А. Пасічник Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	75976	655	26866 (35%)	216 (33%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы
95891	Название: ЗВІТ з науково-дослідної практики Добавлено в БД: 2021-09-29 Авторы: Тростинський Н.М Руководители: Скрипник Т.К. Консультанты: Опоненты:	25502 (34.0%)	220 (34.0%)

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1009494276

Дата перевірки:
03.12.2021 08:46:20 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.12.2021 08:50:26 EET

ID користувача:
100005671

Назва документа: **Тростинський_плагіат 02**

Кількість сторінок: **87** Кількість слів: **12481** Кількість символів: **96801** Розмір файлу: **9.45 MB** ID файлу: **1009507062**

2.36% Схожість

Найбільша схожість: **1.83%** з джерелом з Бібліотеки (ID файлу: **1009480547**)

Не знайдено джерел з Інтернету

2.36% Джерела з Бібліотеки **9** ..... Сторінка 89

0.23% Цитат

Цитати **2** ..... Сторінка 90

Не знайдено жодних посилань

88.2% Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за **8 слів** та **0%**)

5.75% Вилучення з Інтернету **101** ..... Сторінка 91

88.2% Вилученого тексту з Бібліотеки **71** ..... Сторінка 92

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи **10**

РІШЕННЯ ЕКСПЕРНОІ КОМІСІІ

КАФЕДРИ КОМП'ЮТЕРНИХ НАУК

ПРО ДОПУСК КВАЛІФІКАЦИНОІ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод візуалізації хмар точок «Web point cloud viewer» для прийняття контрольованих людиною критично-безпекових рішень

Автор: Тростинський Назар Миколайович

Спеціальність: 122 – Комп'ютерні науки

Освітня програма: освітньо-професійна

Науковий керівник: к.т.н., доцент Яшина Оксана Миколаївна

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) за програмою Anti-Plagiarism виявлені 34%, схожість виявлена зі звітом автора з науково-дослідної практики.

2) за програмою UNICHECK виявлені 2.36%.; Найбільша схожість: 1.83% з джерелом з Бібліотеки (ID файлу: 1009480547)) яке містить статтю автора; інші схожості є фрагментарними – містять поширені конструкції, загальновідомі терміни, скорочення та визначення.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості, складає 34% і 2.36% відповідно, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ О.М. Яшина

Гарант ОП _____ Р. О. Багрій

Завідувач кафедри КН _____ О. В. Бармак

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ
кафедра комп'ютерних наук

ВІДГУК ОПОНЕНТА

на кваліфікаційну роботу магістра

гр. КНм-20-1 Тростинського Назара Миколайовича за темою: *Метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень*

1. Актуальність обраної теми

У сучасному виробництві широке поширення одержали моделі, що зберігаються у вигляді хмар точок. Використання окремих непов'язаних точок є ключем до їх корисності. Хмари точок відносно легко показувати, редагувати та фільтрувати. Це робить їх чудовим способом для зберігання величезної кількості детальних даних. Очікується, що точність та доступність даних хмар точок виросте впродовж найближчого часу. Тобто, обрана тема кваліфікаційної роботи магістра є актуальною.

2. Відповідність роботи предметній області спеціальності 122 Комп'ютерні науки та загальним вимогам до наукових робіт

Зважаючи на той факт, що для розв'язання поставлених задач у кваліфікаційній роботі магістра використовуються основні положення методів аналізу даних, геометричного моделювання, комп'ютерної графіки, алгоритми і методи візуалізації можна дійти до конструктивного висновку, що робота повністю відповідає предметній області спеціальності 122 Комп'ютерні науки.

3. Повнота розкриття мети та завдань дослідження

У кваліфікаційній роботі магістра було повністю розкрито мету, яка полягає у реалізації методу «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень з достатнім рівнем точності та швидкодії, обчислювальної здатності та доступності. Завдання дослідження в повній мірі охоплюють мету дослідження та є коректними, а також вичерпно розкривають мету дослідження.

4. Наявність наукової новизни

Наукова новизна одержаних результатів полягає в удосконаленні існуючого методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень, що дозволило підвищити точність та швидкодію, обчислювальну здатність та доступність.

Матеріали кваліфікаційної роботи пройшли належну апробацію. За результатами виконання кваліфікаційної роботи магістра автором опубліковано одну наукову статтю у фаховому науковому виданні, включеному до переліку МОН України.

5. Зміст кожного розділу роботи

В рамках першого розділу кваліфікаційної роботи магістра було проведено аналіз предметної області і окреслено широку сферу застосування хмар точок в сучасних інформаційних технологіях, так як вони пропонують всеосяжне відображення реального простору та поверхонь об'єктів. Тим самим це засвідчило їх цінність для широкого кола застосувань. Також було проведено аналіз існуючого програмного забезпечення предметної області, а саме було описано такі системи як: Autodesk Recap, Online LIDAR point cloud viewer та Potree. Було проаналізовано формати даних для збереження хмар точок, а саме формати PCD, LAS, LAZ, PTS, PTX, XYZ, TXT та E57. Як результат першого розділу було сформульовано завдання дослідження.

У другому розділі кваліфікаційної роботи магістра було побудовано інформаційну модель методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень. Автоматизовано обробку інформаційних потоків, в результаті чого було сформовано наступний список бізнес-процесів для задоволення потреб користувача: використання хмарного сховища, робота з камерою, робота з моделлю, робота із сценою, робота з вимірюванням моделі, робота з хмарами точок. Розроблено структуру методу, що дозволило забезпечити надійний взаємозв'язок між програмним забезпечення та хмарною платформою Autodesk Forge.

В рамках третього розділу кваліфікаційної роботи магістра було розроблено програмні модулі та детально описані алгоритмічні конструкції методу. Також було описано функціональне призначення програмних модулів та їх взаємозв'язок між собою.

В рамках четвертого розділу було проведено апробацію реалізованого методу візуалізації хмар точок. В рамках даного дослідження було перевірено швидкодію відображення хмари точок на екрані та швидкодію відклику методу на дії користувача при роботі з великими хмарами точок. Також було перевірено основні функції методу візуалізації хмар точок.

6. Ступінь розкриття теми роботи

Основною темою кваліфікаційної роботи магістра є візуалізація хмар точок. В рамках роботи було дано визначення хмари точок, окреслено способи формування хмари точок та сферу їх застосування. Також було проаналізовано предметну область, програмні системи безпосередньо пов'язані з візуалізацією хмар точок. Таким чином, в кваліфікаційній роботі магістра тему розкрито повністю.

7. Якість оформлення кваліфікаційної роботи

Кваліфікаційна робота магістра відповідає всім вимогам до оформлення таких робіт. Стиль подання інформації є фаховим та зрозумілим. Робота не містить стилістичних відхилень та відповідає всім нормам граматики.

8. Недоліки кваліфікаційної роботи

До недоліків кваліфікаційної роботи магістра можна віднести, що не було наведено формул за допомогою яких здійснюється вимірювання хмари точок.

9. Загальний висновок (допускається чи не допускається до захисту), якої оцінки заслуговує кваліфікаційна робота.

Беручи до уваги новизну, актуальність, важливість отриманих результатів, їх достовірність та обґрунтованість, вважаю, що кваліфікаційна робота магістра Тростинського Назара Миколайовича «Метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень» є оригінальним та завершеним науковим дослідженням.

Кваліфікаційна робота Магістра Тростинського Н.М. рекомендується до захисту, рекомендована оцінка «відмінно».

Опонент _____ _____ к.т.н., доцент, доцент кафедри комп'ютерної інженерії та інформаційних систем Гнатчук Є.Г.

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу магістра

гр. КНм-20-1 Тростинського Назара Миколайовича за темою: *Метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень*

1. Актуальність теми

Представлена кваліфікаційна робота магістра присвячена реалізації методу «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень. Існуючі на сьогоднішній день програмні системи призначені для відображення хмар точок часто є незручними у використанні для користувачів та забезпечують перегляд лише одного чи кількох форматів даних, при цьому не надаючи можливості інтеграції з зовнішніми системами. З огляду на вище сказане, кваліфікаційна робота магістра характеризується актуальністю та своєчасністю.

2. Відповідність роботи предметній області спеціальності 122 Комп'ютерні науки та загальним вимогам до наукових робіт

Кваліфікаційна робота магістра Тростинського Н.М. «Метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень» є оригінальним та завершеним науковим дослідженням» за ступенем обґрунтованості наукових положень, новизни, а також обсягом, структурою та змістом викладеного матеріалу відповідає вимогам щодо наукових робіт.

У роботі використані методи комп'ютерної графіки, аналізу та представлення даних, а також методи та алгоритми візуалізації трьохвимірних об'єктів, що повністю відповідає предметній області спеціальності 122 Комп'ютерні науки.

3. Професійні та особистісні якості магістранта

Кваліфікаційна робота магістра Тростинського Н.М. є результатом багаторічної роботи по вивченню методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень на його основі інформаційних технологій. Ним запропоновані і реалізовані основні ідеї, що дозволили отримати достатньо просту та завершену методологію візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень для широкого кола тривимірних об'єктів.

Під час роботи над кваліфікаційною роботою магістра Тростинський Н.М.. виявив себе старанним, працьовитим, сумлінним, кваліфікованим спеціалістом здатним генерувати і реалізовувати нові наукові та інженерні ідеї, виявив здатність самостійно приймати складні технічні рішення та проводити науково-дослідну діяльність.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Кваліфікаційна робота виконана магістрантом самостійно. Магістрантом особисто виконано аналіз предметної області, прведено теоретичні напрацювання у поєднанні із практичною реалізацією та апробацією отриманих даних.

Спільно із науковим керівником: сформульовано мету роботи та завдання дослідження; проведено обговорення отриманих результатів.

5. Наукова новизна та оригінальність запропонованих підходів

Наукова новизна одержаних результатів полягає в тому, що у кваліфікаційній роботі магістра було удосконалено існуючий метод візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень, що дозволило підвищити доступність, точність, швидкодію та обчислювальну здатність.

За темою кваліфікаційної роботи магістра автором виконано одну наукову публікацію та доповідь.

6. Ступінь оволодіння методами дослідження

Під час роботи над кваліфікаційною роботою магістра студент Тростинський Н.М. продемонстрував здатність формалізувати предметну область запропонованої теми для побудови відповідної інформаційної моделі. Також студентом було опановано методи збору, подання, обробки, зберігання, передачі та доступу до інформації в комп'ютерних системах.

7. Повнота та якість розкриття теми роботи

Студентом було проведено порівняння та аналіз можливих методів розв'язання поставленої задачі та обрано оптимальний варіант. Також було досліджено існуючі реалізації розв'язання подібних задач. В рамках кваліфікаційної роботи магістра було проведено аналіз предметної області, побудовано інформаційну модель методу візуалізації хмар точок, а також описано алгоритмічні конструкції. Виходячи з наведених положень, тема роботи є повністю розкритою.

8. Логічність, послідовність, аргументованість, літературна грамотність викладу матеріалу

Позитивними рисами кваліфікаційної роботи є системність та послідовність викладення матеріалу. Студент продемонстрував здатність збирати і аналізувати дані, для забезпечення якості прийняття рішень. У кваліфікаційній роботі магістра чудово

формалізовані та систематизовані вимоги до розробленої комп'ютерної системи. Робота відповідає всім граматичним нормам та демонструє зрозумілий стиль подання інформації.

9. Можливість практичного застосування кваліфікаційної роботи, окремих її частин

Практичне використання розробленого методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень, забезпечує користувачів зручним способом для роботи із хмарами точок, а саме їх перегляд, зберігання, вимірювання із використанням малопотужних технічних засобів за наявності універсального, вбудованого програмного забезпечення.

10. Висновок про можливість допуску кваліфікаційної роботи до захисту, на яку оцінку заслуговує робота

Кваліфікаційна робота магістра Тростинського Н.М виконана повністю у відповідності із представленими вимогами та є завершеною науковою працею. Вона містить рішення наукової задачі, яка по суті полягає у реалізації методу візуалізації хмар точок «Web Point Cloud Viewer» для прийняття контрольованих людиною критично-безпекових рішень. З огляду на вище сказане, робота рекомендується до захисту та заслуговує на оцінку «відмінно».

Науковий керівник _____ к.т.н., доцент Яшина О.М.