НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут (факультет) Інститут прикладного системного аналізу
Кафедра Системного проектування

До захисту допущено:
Завідувач кафедри
_____ А.І.Петренко
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні системи та технології**»**

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Порівняння засобів server-side та client-side рендерингу 3D-моделей за допомогою Paraview, VTK**»**

Виконав (-ла):
студент (-ка) IV курсу, групи ДА-72
Коваль Сергій Сергійович _____

Керівник:
Кандидат технічних наук, доцент,
Чкалов Олексій Валерійович _____

Консультант з економічного розділу:
Кандидат наук, доцент,
Рощина Надія Василівна _____

Рецензент:
Кандидат технічних наук, доцент,

_____

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

Київ – 2021 року

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра Системного проектування

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ А.І.Петренко

«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Ковалю Сергію Сергійовичу

1. Тема роботи «Порівняння засобів server-side та client-side рендерингу 3D-моделей за допомогою Paraview, VTK», керівник роботи Чкалов Олексій Валерійович, кандидат технічних наук, доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи

- Результати порівняння server-side та client-side рендерингу
- Вказівки щодо налаштування засобів Paraview, Vtk
- Аналіз переваг та недоліків підходів до рендерингу

Програмне забезпечення, що:

- Дозволяє виконувати віддалений рендеринг за допомогою засобів Paraview та VTK.
- Дозволяє виміряти швидкість рендерингу на клієнтах/серверах.

Методології:

- Налаштування клієнт серверної системи для більш ефективного рендерину 3D моделей

Paraview, VTK, Python, Javascript

4. Зміст роботи

1. Розглянути предметну область, проаналізувати особливості програм Paraview та VTK для задач рендерингу зображень

2. Огляд набору інструментів VTK;

3. Огляд набору інструментів Paraview

4; Порівняння та вибір між client-side та server-side

5. Провести функціонально-вартісний аналіз розробленого програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Презентація до захисту роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	30.01.2021	
2	Аналіз проблем при обробці зображень	29.02.2021	
3	Ознайомлення з інструментами Paraview та VTK	15.03.2021	
4	Створення та налаштування програмного забезпечення для віддаленого рендерингу	15.04.2021	
5	Встановлення клієнтсерверної взаємодії Paraview.	30.04.2021	
6	Налаштування клієнтсерверної взаємодії Paraview.	5.05.2021	
8	Порівняння client-side та server-side	15.05.2021	

Студент Коваль С.С.

Керівник Чкалов О.В.

АНОТАЦІЯ

бакалаврської дипломної роботи Коваля Сергій Сергійовича на тему «Порівняння засобів server-side та client-side рендерингу 3D моделей за допомогою Paraview, VTK»

Метою даної роботи було порівняння засобів server-side та client-side рендерингу 3D моделей за допомогою Paraview, VTK.

При дослідженні засобів ренгерингу було проаналізовано основні проблеми існуючих засобів при рендерингу 3D моделей та обгрунтована актуальність цих проблем.

Розглянуто засоби, на основі відкритих бібліотек Paraview та VTK для мов програмування Python та JavaScript, а також засоби, що входять в комплект программ VTK, Paraview.

Було розглянуто особливості цих засобів, деталі роботи з ними, їх встановлення та налаштування.

Також було створено додатки, що реалізують віддалений рендеринг за допомогою VTK та додаток для тестування швидкості рендерингу для Paraview.

Використання Server-side може значно покращити продуктивність та швидкість роботи у багатьох задачах та зменшити витрати на апаратне забезпечення клієнтів. Також рендеринг на стороні сервера дозволяє створювати великі надійні системи для централізовані зберігання та рендерингу даних.

Результат роботи: порівняння server-side та client-side засобів рендерингу у різних ситуаціях, огляд переваг та недоліків кожного з підходів. Також було створено приклад налаштування таких засобів забезпечення.

Загальний обсяг роботи 88 с., 22 рис., 3 таблиць, 2 додатки, 7 джерел.

Ключові слова: Paraview, VTK, server-side, Client-side, рендеринг, 3D

Annotation

Bachelor diploma work by Serhii Koval on the topic

«Comparison of server-side and client-side tools for rendering 3D models using Paraview, VTK.»

The purpose of this work was to compare client-side and server-side rendering tools using Paraview and VTK.

In the study of rendering tools, the main problems of existing tools for rendering 3D models were analyzed and the relevance of these problems was substantiated.

During the study, several toolsbased on open source libraries were examined: Paraview and VTK for Python and Javascript languages and also tools that are a part of Paraview and VTK.

Also applications were created that implement remote rendering using VTK and application for testing render speed using Paraview.

Usage of server-side rendering can drastically improve performance and work speed in many tasks and decrease expenses on client's hardware. Rendering on server side also enables to create big reliable systems for centralized data storage and rendering.

The result: detailed comparison between server-side and client-side rendering paradigm, overview of advantages and drawback of each one. Example of configuration of these tools created.

The total volume of 104 p., 22fig. 3 Table., 2 additions 15 sources.

Keywords: Paraview, VTK, Server-side, Client-side, Rendering, 3D

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Рендеринг – це процес отримання 2D зображення з 3D моделі за допомогою комп’ютерної програми.

Сервер(апаратний) – виділений комп’ютер, або спецілізований пристрій для виконання сервісу прогрманого забезпечення або інших задач.

Сервер(програмний) – програмний компонент обчислювальної системи, який виконує функції по запиту клієнта, надаючи йому доступ до ресурсів або послуг.

Клієнт – апаратний або програмнй компонент, який надсилає запити до серверу.

SSH – Secure Shell мережевий протокол прикладного рівня, що дозволяє проводити віддалене управління операційною системою та тунелювання за допомогою TCP з’єлнань.

LOD – скорочення від Level Of Detail, - рівень деталізації зображення.

ВСТУП

Платформи для візуалізації та моделювання продовжують розвиватися та змінюватися, і разом з цим постають нові проблеми. Швидкість рендерингу зображення колись була обмежуючим фактором візуалізації, але зараз швидкість візуалізації часто є другорядним питанням.

Нові проблеми візуалізації полягають у завантаженні, управлінні та обробці великих масивів даних.

Більшості з нас не пощастило мати суперкомп'ютер або потужний ресурс візуалізації в офісах та навчальних установах, але нам все ще потрібен доступ до цих ресурсів. Можливість віддаленого рендерингу, або візуалізації, дозволяє нам забезпечити найширший доступ до наших ресурсів. За наявності мережевих можливостей можна виконувати великомасштабну візуалізацію зі свого персонального комп’ютера за допомогою віддаленої візуалізації на сервері. Відстань між науковцем та обчислювальними візуалізаційними ресурсами має тенденцію зростати. Зі збільшенням відстані між користувачем та ресурсом візуалізації віддалена візуалізація стає важливішою та складнішою.

Пакети моделювання генерують величезні обсяги даних, які як правило, неможливо або недоцільно передавати на великі відстані. Розмір цих наборів даних знаходиться в діапазонах від гігабайт до терабайт. Такі дані зазвичай знаходяться у віддаленому місці, часто в окремій кімнаті або навіть у географічно віддаленій установі чи віддаленому сервері. Server-side візуалізація цих великих наборів даних є складною проблемою.

1. АКТУАЛЬНІСТЬ ПРОБЛЕМИ ВИБОРУ SERVER-SIDE ЧИ CLIENT-SIDE РЕНДЕРИНГУ

За останнє десятиліття обробка, аналіз та візуалізація дуже стрімко розвиваються. Спочатку такі системи в основному використовували підхід з обробкою на клієнті, що вимагало встановлення1050 плагінів браузера для певної платформи. Однією з технологій, яка зазвичай використовувалася для цієї мети, був Adobe Flex, який дозволив інтегрувати різні географічні функціональні можливості. Також були інші технології плагінів браузера, такі як клієнт Cortona 3B, який дозволяв робити тривимірне моделювання а також Java Web Аплети, які використовувались, серед іншого, для побудови Інтернет-сервісів для 3D-навігації містом із використанням даних Java3D та OSM. Пізніші дослідження були зосереджені на незалежних від платформи рішеннях для легких клієнтів, які мали на меті перемістити критичну функціональність системи на сторону сервера.

Швидке зростання програмного забезпечення забезпечує величезні переваги для користувачів, а також для корпорацій, особливо для зменшення витрат часу та витрат розвитку при створенні.

Багато видів бізнесу робить ці технології основною вимогою побудови системи для забезпечення ефективної роботи системи.

Еволюція мережевих можливостей надала можливість для швидкого та надійного підключення між сервером та клієнтом.

Саме тому багато програмного забезпечення використовує метод візуалізації на стороні сервера, який є процесом, повністю запущеним на стороні сервера, поки клієнт чекає у відповідь лише готове зображення.

Також зараз використовується комбінована архітектура, де з кожним процесом рендерингу вже не тільки працює сервер, але і клієнт також буде запускати процес, який називається візуалізацією на стороні клієнта.

Клієнт виконує процес візуалізації, а обробкою інформації займається сервер, що дає менше часу на виконання програми, щоб задовольнити користувача.

2.1 VTK

Набір інструментів візуалізації (VTK) - це широко використовувана програмна система для обробки та візуалізації даних. Він використовується в наукових обчисленнях, аналізі медичних зображень, обчислювальній геометрії, візуалізації, обробці зображень та інформатика. У цьому розділі ми подаємо короткий огляд VTK, включаючи деякі основні закономірності проектування зробити його успішною системою.

Щоб по-справжньому зрозуміти програмну систему, важливо не тільки зрозуміти, яку проблему вона вирішує, а й конкретні обставини, в яких вона виникла. У випадку з VTK програмне забезпечення було нібито розроблено як 3D система візуалізації наукових даних. Але культурний контекст, в якому він виник, додає істотну історію і допомагає пояснити, чому програмне забезпечення було розроблено та розгорнуто таким, яким воно було. На момент задуму та написання VTK його початковими авторами (Уілл Шредер, Кен Мартін, Білл Лоренсен) були дослідники GE Corporate R&D.

Вони вклали значні кошти в систему попередників, відому як LYMB, яка була середовищем реалізованим мовою програмування C. Незважаючи на те, що це була чудова система для свого часу, як дослідників, вони розчаровувались двома основними перешкодами:

1) проблеми IP
2) нестандартне пропрієтарне програмне забезпечення.

Проблеми з IP були проблемою, оскільки спроба розповсюдити програмне забезпечення була майже неможливою, коли до цього долучились корпоративні юристи. По-друге, багато клієнтів не хотіли навчатись працювати в нестандартній пропрієтарній системі, оскільки навички роботи з

нею не переходили разом із працівником, коли вони покидали компанію, і вона не мала широкої підтримки стандартного набору інструментів.

Таким чином, зрештою, основною мотивацією для VTK була розробка відкритого стандарту, абоплатформа для співпраці завдяки чому ми могли б легко передати технологію нашим клієнтам. Таким чином, вибір ліцензії з відкритим кодом для VTK був, мабуть, найважливішим рішенням, яке прийняли розробники.

Остаточний вибір ліцензії (тобто BSD, а не GPL) з огляду на минуле був зразковим рішенням, прийнятим авторами, оскільки в кінцевому підсумку це дало можливість рости сервісному та консалтинговому бізнесу, яким став Kitware.

2.2 ОГЛЯД VTK

VTK спочатку задумувався як наукова система візуалізації даних. Багато людей поза сферою помилково вважають візуалізацію певним типом геометричної візуалізації: вивчаючи віртуальні об'єкти та взаємодіючи з ними. Хоча це справді частина візуалізації, загалом візуалізація даних включає весь процес перетворення даних у ввід, сприйнятний сенсорами людини, як правило, зображення, але також включає тактильну, слухову та інші форми. Форми даних складаються не тільки з геометричних та топологічних конструкцій, включаючи такі абстракції, як сітки або складні просторові розкладання, але атрибути основної структури, такі як скаляри (наприклад, температура або тиск), вектори (наприклад, швидкість), тензори (наприклад, напруження та деформація) плюс такі атрибути візуалізації, як нормалі поверхні та координати текстури.

Дані, що представляють просторово-часову інформацію, зазвичай вважаються частиною наукової візуалізації. Однак існують більш абстрактні форми даних, такі як демографічні показники маркетингу, веб-сторінки,

документи та інша інформація, яка може бути представлена лише за допомогою абстрактних (тобто непросторових часових) зв’язків, таких як неструктуровані документи, таблиці, графіки та дерева. Ці абстрактні дані зазвичай розглядаються методами візуалізації інформації.

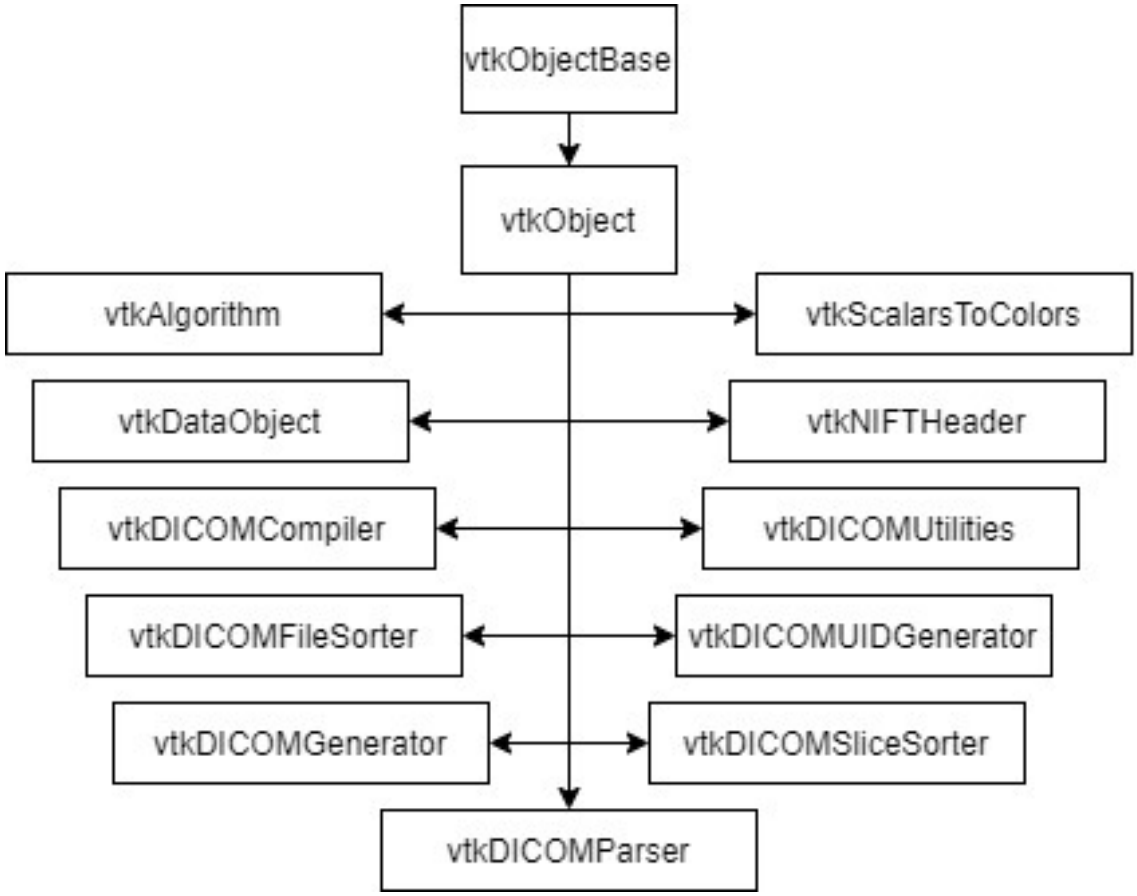
За допомогою спільноти VTK тепер здатний як до наукової, так і до візуалізації інформації. Перш ніж вдаватися занадто далеко до конкретних архітектурних особливостей VTK, є концепції високого рівня, які мають значний вплив на розробку та використання системи. Одним з них є гібридна обгортка VTK. Цей засіб автоматично генерує мовні прив’язки до Python, Java та Tcl із реалізації C ++ VTK (додаткові мови можуть бути і були додані). Більшість розробників працюють на C ++. Користувач та розробники додатків можуть використовувати C ++, але часто інтерпретовані мови, згадані вище, є кращими. Це гібридне компільоване / інтерпретоване середовище поєднує в собі найкраще з обох сторін: високопродуктивні обчислювальні алгоритми та гнучкість під час створення прототипів або розробки додатків. Такий підхід до багатомовних обчислень знайшов прихильність у багатьох науково-обчислювальних колах, і вони часто використовують VTK як шаблон для розробки власного програмного забезпечення. Що стосується програмного процесу, VTK прийняв CMake для управління збіркою; CDash / CTest для тестування; і CPack для розгортання кроссплатформних платформ. VTK може бути скомпільований майже на будь-якому комп'ютері, включаючи суперкомп'ютери, які часто є загальновідомими середовищами розробки. Крім того, веб-сторінки, wiki, списки розсилки, засоби створення документації (тобто Doxygen) та інструмент відстеження помилок (Mantis) доповнюють набір засобів розробки.

Однією з сильних сторін VTK є відносно спрощений спосіб подання та управління даними. Зазвичай різні масиви даних певних типів (наприклад,vtkFloatArray) використовуються для представлення суміжних

частин інформації. Наприклад, список із трьох точок XYZ буде представлений за допомогою vtkFloatArray з дев’яти полів (x, y, z, x, y, z тощо) У цих масивах існує поняття кортежу, тому 3D-точка є 3-кратною, тоді як симетрична тензорна матриця 3 × 3 представлена 6 кортежами. Ця конструкція була прийнята спеціально, оскільки в наукових обчисленнях загальноприйнято взаємодіяти із системами, що маніпулюють масивами (наприклад, Fortran), і набагато ефективніше розподіляти та вивільняти пам'ять великими суміжними шматками. Крім того, зв'язок, серіалізація та виконання вводувиводу, як правило, набагато ефективніші із суміжними даними. Ці основні масиви даних (різних типів) представляють значну частину даних у VTK і мають безліч зручних методів для вставки та доступу до інформації, включаючи способи швидкого доступу та методи, які автоматично розподіляють пам'ять за необхідності при додаванні більшої кількості даних.

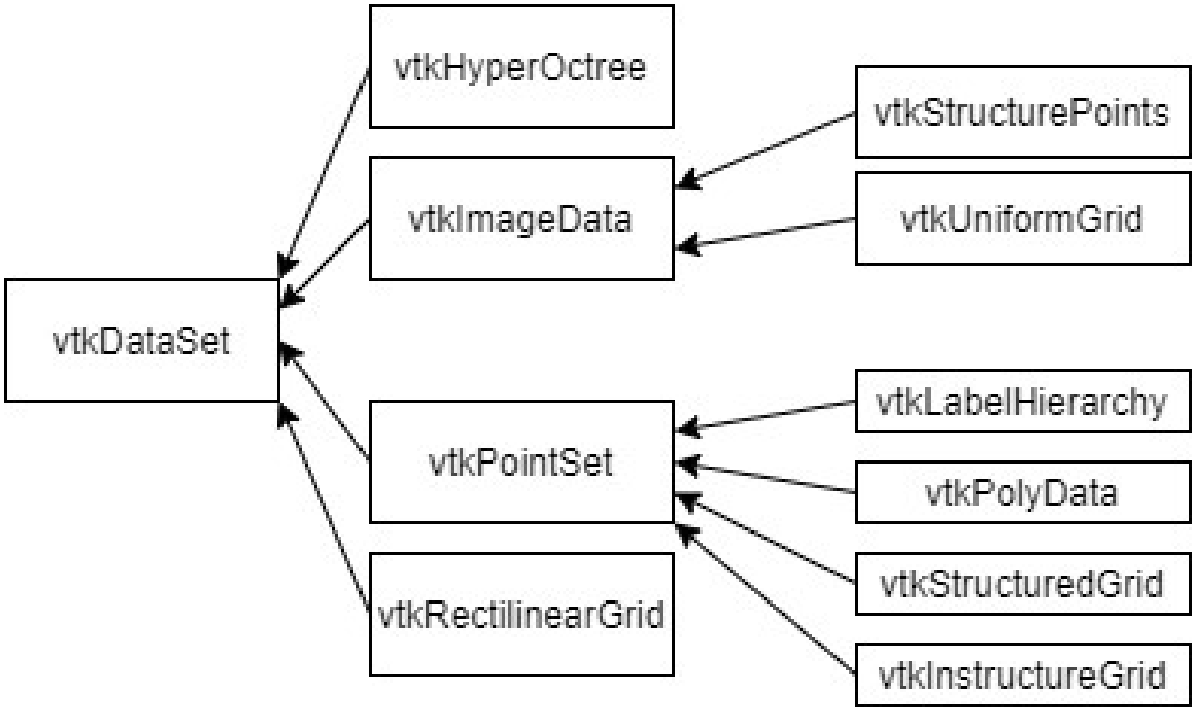
2.3 ПРЕДСТАВЛЕННЯ ДАНИХ

Однією з сильних сторін VTK є його здатність представляти складні форми даних. Ці форми даних варіюються від простих таблиці до складних структур, таких як сітки кінцевих елементів. Всі ці форми даних є підкласами vtkDataObject як показано на рисунку.



Малюнок 1 - Часткова діаграма успадкування класів об’єктів даних

Одна з найважливіших характеристик vtkDataObject полягає в тому, що його можна обробити в конвеєрі візуалізації. З багатьох показаних класів є лише кілька, які зазвичай використовуються в більшості реальних програм: vtkDataSet та похідні класи використовуються для наукової візуалізації. З розвитком VTK постійно створюються нові класи, що базуються на vtkObject, що робить VTK більш гнучким та дозволяє працювати з великою кількістю різноманітних типів даних.



Малюнок 2 - Класу наборів даних

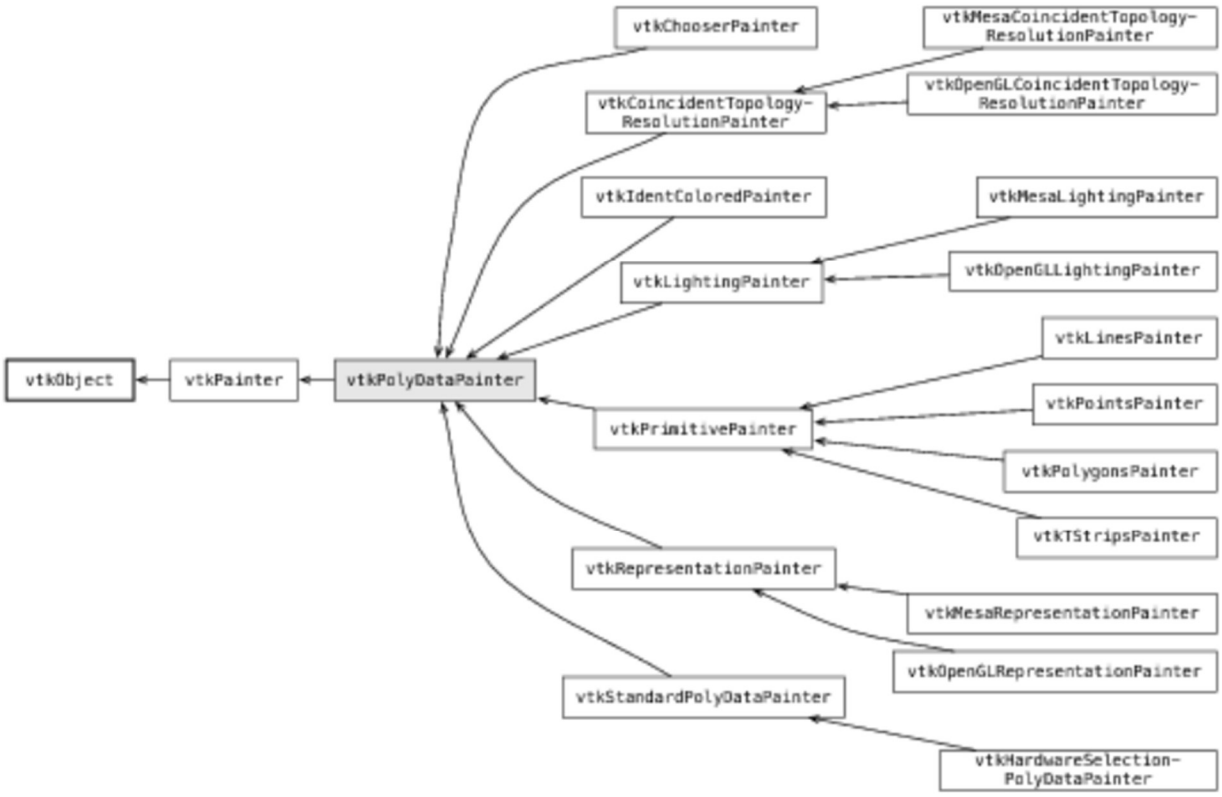
Наприклад,vtkPolyData використовується для представлення полігональних сіток, vtkUnstructuredGrid представляє сітки, та vtkImageData представляє 2D та 3D піксельні та воксельні дані.

2.4 ПІДСИСТЕМА ВІЗУАЛІЗАЦІЇ

На перший погляд VTK має просту об’єктно-орієнтовану модель візуалізації з класами, що відповідають компонентам, що складають 3D-сцену. Наприклад,vtkActor s - це об'єкти, які відображаються a vtkRenderer у поєднанні з a vtkCamera , з можливо кількома vtkRenderer s, що існують у a vtkRenderWindow . Сцена висвітлюється одним або кількома vtkLight s. Позиція кожного vtkActor контролюється vtkTransform , а зовнішність актора визначається через vtkProperty . Нарешті, геометричне зображення актора визначається vtkMapper . Карти відіграють важливу роль у VTK, вони служать для завершення конвеєру обробки даних, а також для інтерфейсу до системи візуалізації.

З часом ця об'єктна модель стала більш досконалою. Значна частина складності пов’язана з розробкою похідних класів, які спеціалізуються на певному аспекті процесу візуалізації.vtkActor Зараз вони спеціалізуються на vtkProp (як реквізит, знайдений на сцені), і існує ціла низка цих реквізитів для візуалізації 2D графіки та тексту, накладених 3D-об'єктів і навіть для підтримки передових методів візуалізації, таких як об'ємна візуалізація або реалізація графічного процесора.

Подібно до того, як зростала модель даних, що підтримується VTK, зростали і різні формати представлення даних, які взаємодіють із системою візуалізації. Інша область значного розширення - це ієрархія трансформації. Що спочатку було простою матрицею лінійних перетворень 4 × 4, стало потужною ієрархією, яка підтримує нелінійні перетворення, включаючи тонкопластинкове сплайн-перетворення. Наприклад, спочатку vtkPolyDataMapper мали спеціальні підкласи для класів (наприклад, vtkOpenGLPolyDataMapper)



Малюнок 3 - Класи відображення

2.5 SERVER-SIDE РЕНДЕРИНГ

Завдяки обгортці VTK дуже легко створювати додатки для віддаленого рендерингу зображення. У документації є достатньо багато матеріалів з прикладами на різних мовах програмування. Розглянему створення такого серверу для рендерингу за допомогою Python.

Для цього можна встановити модифіковану версію Python під назвою vtkPython, яка вже містить всі необхідні модулі та бібліотеки для роботи з VTK.

Приймати підключення до сервера та обробляти запити можна за допомогою простого блоку коду, який наведено у документації.

```
global renderer, renderWindow, renderWindowInteractor, cone, mapper,
actor
```

```
        self.registerVtkWebProtocol(protocols.vtkWebMouseHandler())
        self.registerVtkWebProtocol(protocols.vtkWebViewPort())

self.registerVtkWebProtocol(protocols.vtkWebPublishImageDelivery(decode=
False))

self.registerVtkWebProtocol(protocols.vtkWebViewPortGeometryDelivery())

        self.updateSecret(_WebCone.authKey)
        self.getApplication().SetImageEncoding(0)

        if not _WebCone.view:
            renderer = vtk.vtkRenderer()
            renderWindow = vtk.vtkRenderWindow()
            renderWindow.AddRenderer(renderer)

            renderWindowInteractor = vtk.vtkRenderWindowInteractor()
            renderWindowInteractor.SetRenderWindow(renderWindow)

renderWindowInteractor.GetInteractorStyle().SetCurrentStyleToTrackballCa
mera()

            cone = vtk.vtkConeSource()
            mapper = vtk.vtkPolyDataMapper()
            actor = vtk.vtkActor()

            mapper.SetInputConnection(cone.GetOutputPort())
            actor.SetMapper(mapper)

            renderer.AddActor(actor)
            renderer.ResetCamera()
            renderWindow.Render()

            _WebCone.view = renderWindow
            self.getApplication().GetObjectIdMap().SetActiveObject("VIEW",
renderWindow)
```

Для ефективної та зручної роботи сервера потрібно додати можливість задавати різні параметри для підключення до сервера, наприклад інтерфейс сервера, порт, чи секретні ключі для авторизації.

```
parser = argparse.ArgumentParser(
      description="Server-side rendering")
  server.add_arguments(parser)
  args = parser.parse_args()
  _WebCone.authKey = args.authKey
```

Повну версію коду наведено у додатку А

Запустити такий сервер можна за допомогою vtkpython, наприклад

```
$: vtkpython vtk_server.py --port 1234
```

Для взаємодії між сервером та клієнтом використовується wslink.
Wslink дозволяє двосторонню взаємодію між сервером на Python та клієнтами на JavaScript чи C++. Клієнт може робити виклики процедур (RPC), а сервер публікує відповідні повідомлення, до запитів клієнта. У цих повідомленнях сервер може передавати бінарні додатки, що передаються як двійкове повідомлення вебсокету, що зменшує накладні витрати на кодування та декодування інформації.

Wslink також досить добро документований, тому не складно запустити клієнт з прикладів від Kitware.
Приклад підключення до сервера за допомогою JavaScript.

```
const view = vtkRemoteView.newInstance({
  rpcWheelEvent: 'viewport.mouse.zoom.wheel',
});
view.setContainer(divRenderer);
view.setInteractiveRatio(0.5);
view.setInteractiveQuality(90); // jpeg quality
window.addEventListener('resize', view.resize);
```

```
const clientToConnect = vtkWSLinkClient.newInstance();
clientToConnect.onConnectionError((httpReq) => {
  const message =
    (httpReq && httpReq.response && httpReq.response.error) ||
    `Connection error`;
  console.error(message);
  console.log(httpReq);
});
const config = {
  application: 'cone',
  sessionURL: 'ws://localhost:8888/ws',
};

clientConnect
  .connect(config)
  .then((validClient) => {
    connectImageStream(validClient.getConnection().getSession());

    const session = validClient.getConnection().getSession();
    view.setSession(session);
    view.setViewId(-1);
    view.render();
  })
  .catch((error) => {
    console.error(error);
  });
```

Для підключення потрібно вказати sessionURL. Якщо сервер знаходиться в одній мережі з клієнтом, достатньо вказати локальну адресу. У випадку якщо сервер знаходиться у іншій мережі, потрібно вказати адресу маршрутизатора та налаштувати NAT на цьому маршрутизаторі. При цих налаштуваннях може змінитися порт.

Також у клієнта є можливість налаштувати якість зображення, що може бути дуже корисним якщо клієнт має повільне підключення до мережі Інтернет.

2.6 CLIENT-SIDE РЕНДЕРИНГ

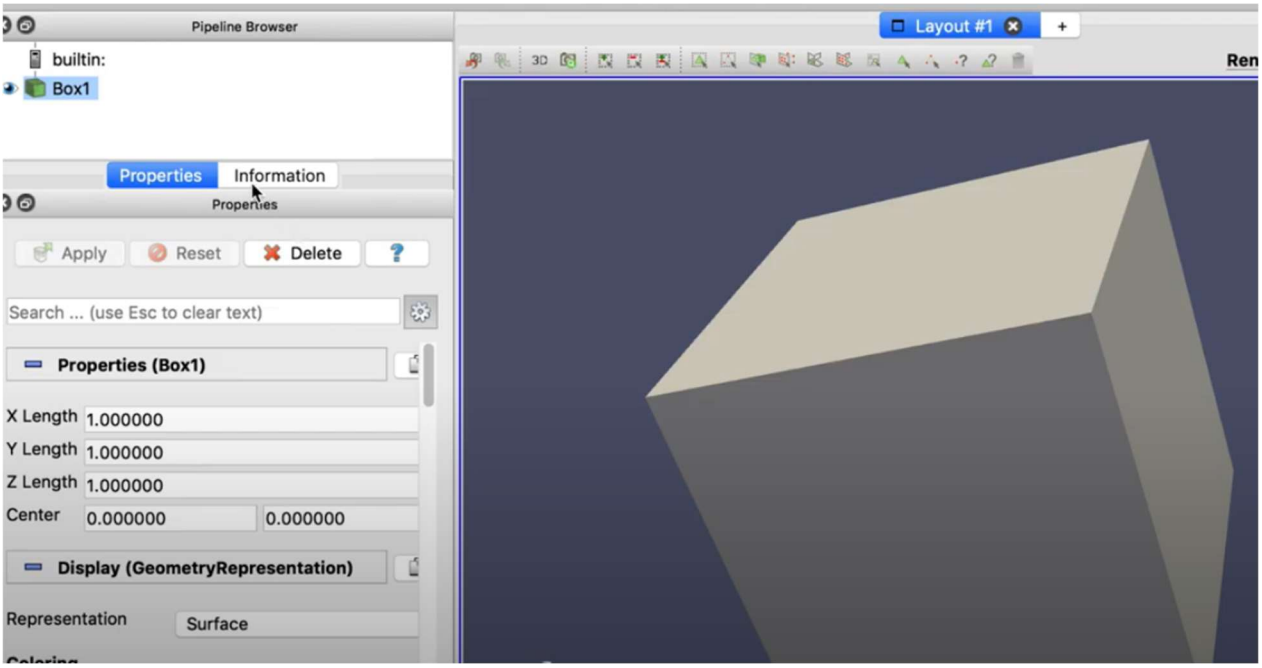
За десятки років існування VTK було створено чимало програм для рендерингу зображення, серед них

VisIt

VtkReader

Paraview

Більшість з цих програм мають інтуїтивно зрозумілий користувальницький інтерфейс, наприклад ось так виглядає рендеринг куба у VTK Reader.



Малюнок 4 - Приклад рендерингу у VTK Reader

Включити/Виключити виділення – ввімкнення та вимкнення виділення виду.

Показати/Сховати вісі.

Вмістити все – масштабує зображення таким чином, щоб воно повністю помістилося у поле перегляду.

Вмістити виділене – масштабує виділену частину зображення таким чином, щоб воно повністю помістилося у поле перегляду.

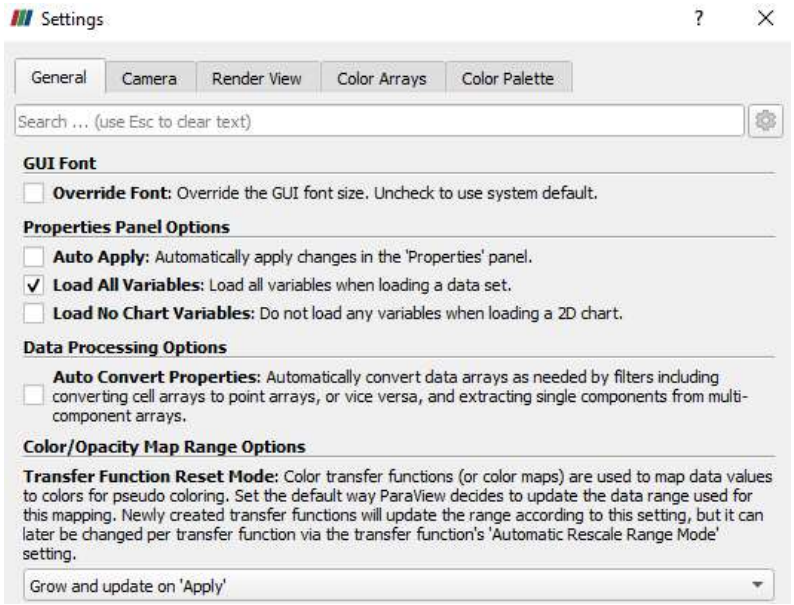
Приближення – дозволяє приближати та віддаляти зображення

Панорамирование – Сдвинути вид зображення

У клієнта є безліч налаштувань які зберігаються у $HOME/.config/ParaView, що дозволяє легко переносити свої налаштування на інші клієнти.

У основних налаштуваннях можна налаштувати поведінку програми при закритті або при втраті зв'язку з сервером. Також можна зробити щоб усі зміни властивостей моделі відбувалися одразу без нобхідності натискати на кнопку «apply» .

Усі налаштування можна зберігати у профіль та швидко перемикати між профілями. Таким чином можна мати різні профілі для роботи з різними типами моделей.



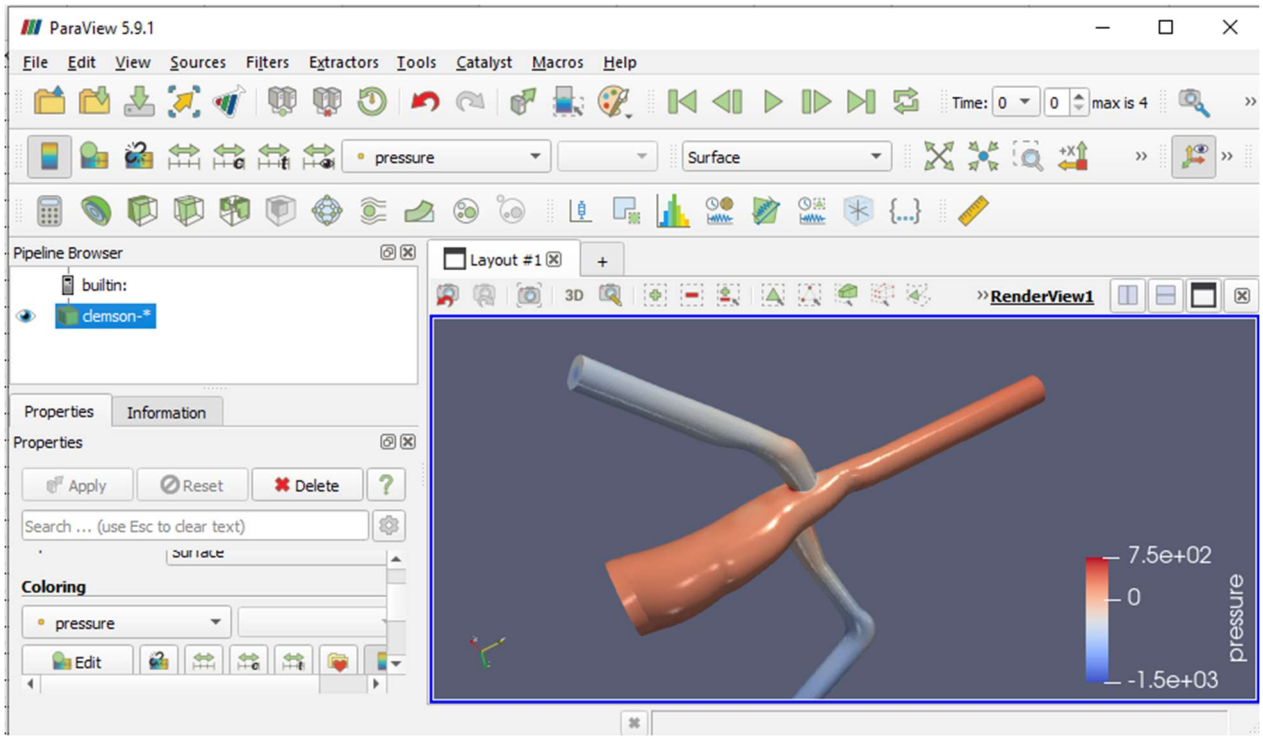
Малюнок 5 - Приклад налаштувань

3.1 ParaView

ParaView - це багатоплатформова програма з відкритим кодом для інтерактивної наукової візуалізації. Він має архітектуру клієнт-сервер для полегшення віддаленої візуалізації наборів даних та генерує моделі рівня деталізації (LOD) для підтримки інтерактивних частот кадрів для великих наборів даних. Це програма, побудована поверх бібліотек Visualization Toolkit (VTK). ParaView - це програма, призначена для паралелізму даних на мультикомп’ютерах і кластерах із загальною та розподіленою пам'яттю.
Його також можна запустити як однокомп'ютерну програму.

ParaView розроблений як багатошарова архітектура, що базується на інструментарії візуалізації VTK.
Він забезпечує основу ParaView: представлення даних, алгоритми та а
механізм для з'єднання цих уявлень та алгоритмів разом для формування робочої програми.
Другий шар - це паралельні розширення набору інструментів візуалізації.
Шар розпаралелювання VTK,
розширений VTK для підтримки потокової передачі всіх типів даних та паралельне виконання на спільних та
розподілених серверах.
Третій шар - це сам ParaView. ParaView надає графічний
користувальницький інтерфейс і прозоро підтримує візуалізацію та рендеринг великих наборів даних через
апаратне прискорення, паралелізм та прийоми деталізації. Кожен шар відповідає підмножині вимог і додає додаткову функціональність шару нижче.

ParaView надає графічний користувацький інтерфейс для інтерактивного дослідження великих наборів даних. Він створює таку функціональність на паралельному та розподіленому VTK. Оглянемо інструмент з перспективи користувача.



Малюнок 6 - Інтерфейс Paraview

Зразок сеансу ParaView показаний на малюнку 6. Інтерфейс користувача має кілька панелей, що включають панель меню зверху програми, панель інструментів трохи нижче панелі меню, панель ліворуч, а область відображення справа. Кожна з цих областей описана
докладніше нижче:

- Меню (зверху) - У верхній панелі меню є кнопки меню для завантаження та збереження даних, створення джерел та фільтрів, перегляд інших вікон, відображення довідки та інших стандартних можливостей
- Панель інструментів - містить кнопки для роботи з камерою, перемикання між 2D і 3D режимами взаємодії, а також зміна центру

обертання. Крім того, панель інструментів містить ярлики для створення екземплярів деяких часто використовуваних фільтрів.

- Ліва панель - Верхня частина цієї панелі містить вікно вибору або навігації. Вікно вибору містить список екземплярів джерел та фільтрів. Вікно навігації надає графічне представлення дій користувача над потоком даних. Під вікном вибору / навігації розташовані властивості джерел та фільтрів, а також список властивостей. Властивості містять параметри модулів, такі як поточні значення ізоповерхні, обчислені модулем ізоповерхні.
- Область відображення - це місце, де відображається 3D-зображення сцени. У цій області передбачена взаємодія з мишею та клавіатурою.

Щоб додати фільтри, користувач обирає джерело або фільтр із меню Source або Filter в верхньому меню. До джерел належать різні типи зчитувачів або комп’ютерні джерела.

Можливе використання наступних фільтрів:

- Контури та ізоповерхні можуть бути вилучені з усіх типів даних за допомогою скалярів або векторних компонентів. Результати можуть бути забарвлені будь-якою іншою змінною або оброблені далі. Структуровані контури даних / ізоповерхні витягуються за допомогою швидких та ефективних алгоритмів, що використовують структуровану модель даних
- Поля векторів можна перевірити, застосовуючи гліфи (на даний момент стрілки, конуси та сфери) до точок набору даних. Гліфи можна масштабувати за допомогою скалярів, векторних компонент або векторних величин.
- Підрегіон набору даних можна отримати, відсікаючи довільною площиною, або шляхом зазначення порогових критеріїв для виключення

комірок та / або вказувати обсяг, що цікавить (лише для структурованих типів даних). Результати можуть відображатися у вигляді точок, ліній, трубочок і стрічок і можуть бути оброблені великою кількістю фільтрів.

- Точки в наборі даних можуть бути викривлені / зміщені скалярами або векторами.

3.2 Використання бібліотеки ParaView при створенні додатків

Дуже часто при виконанні задач рендерингу з’являється необхідність створення власних додатків, які б розширювали функціональні можливості існуючої системи. ParaView має гарну підтримку основних популярних мов програмування, що дозволяє розробляти такі додатки.

Всі функції бібліотек для відповідних мов програмування гарно задокументовані, знайти їх можна на офіційному сайті Kitware

3.3 Архітектура ParaView

ParaView має трирівневу клієнт-серверну архітектуру. Три логічні одиниці ParaView такі:

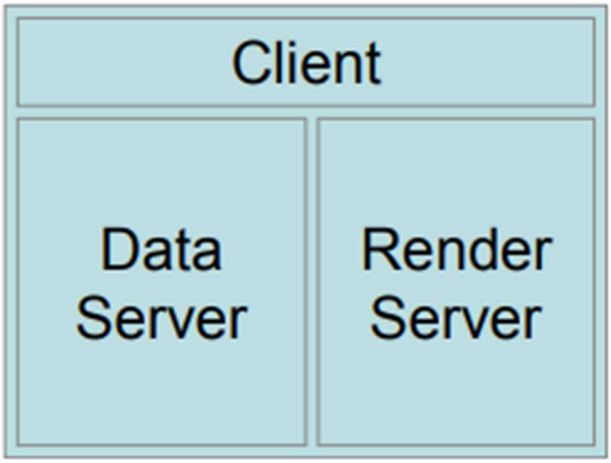
1. Клієнт - відповідає за встановлення візуалізації. клієнт контролює створення, виконання та знищення на серверах, але не отримуж файлів даних (таким чином дозволяючи серверам масштабуватися не обмежуючись параметрами клієнта). У випадку використання графічного інтерфейсу, він також у клієнта. Клієнт завжди є послідовним додатком.
2. Сервер рендерингу - відповідає за візуалізацію. Сервер рендерингу може також бути паралельним

3. Сервер даних - відповідає за зчитування, фільтрацію та запис даних. Усі об'єкти, які бачимо в браузері, містяться на сервері даних.

Ці логічні одиниці не обов’язково повинні бути фізично розділені. Логічні одиниці часто вбудовані в одному додатку, усуваючи необхідність створення засобів будь-якого спілкування між ними.

Далі розглянемо 3 режими у яких можна запустити ParaView

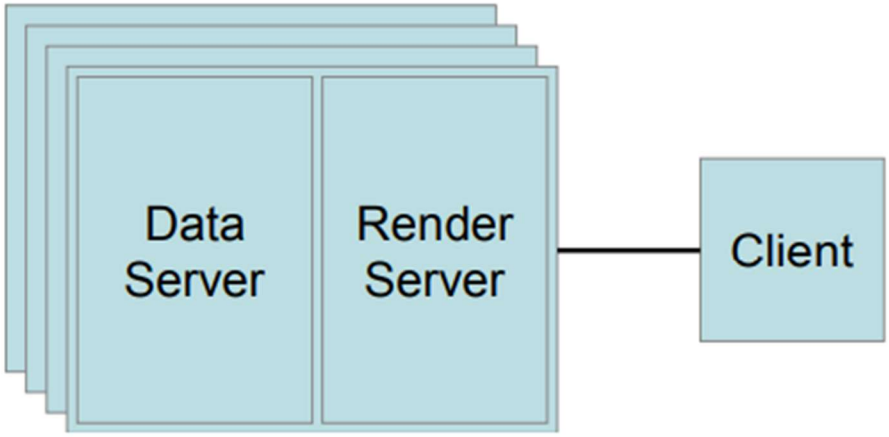
- **Автономний клієнт**



Малюнок 7 - Автономний клієнт

У автономному режимі клієнт, сервер даних та сервер візуалізації об'єднуються в одну послідовну систему. Під час запуску програми paraview ви автоматично підключаєтесь на вбудований сервер, щоб ви були готові використовувати всі можливості ParaView. Такий спосіь добре підійде для роботи з невеликими обсягами даних, чи для запуску на потужній робочій станції.

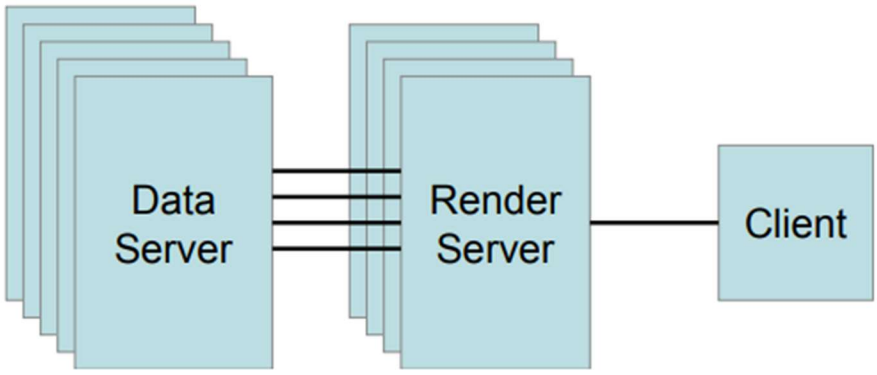
- **Клієнт - сервер**



Малюнок 8 - Клієнт-сервер

Другий режим - це режим клієнт-сервер. У режимі клієнт-сервер pvserver виконується на паралельній машині, а клієнт підключається до нього за допомогою клієнта Paraview. Програма pvserver має в собі як сервер даних, так і сервер візуалізації, тому там обробляються дані та здійснюється візуалізація. Клієнт і Сервер з'єднаний через сокет, який вважається відносно повільним режимом зв'язку, тому передача даних через цей сокет зведена до мінімуму.

- **клієнт - сервер візуалізації - сервер даних**



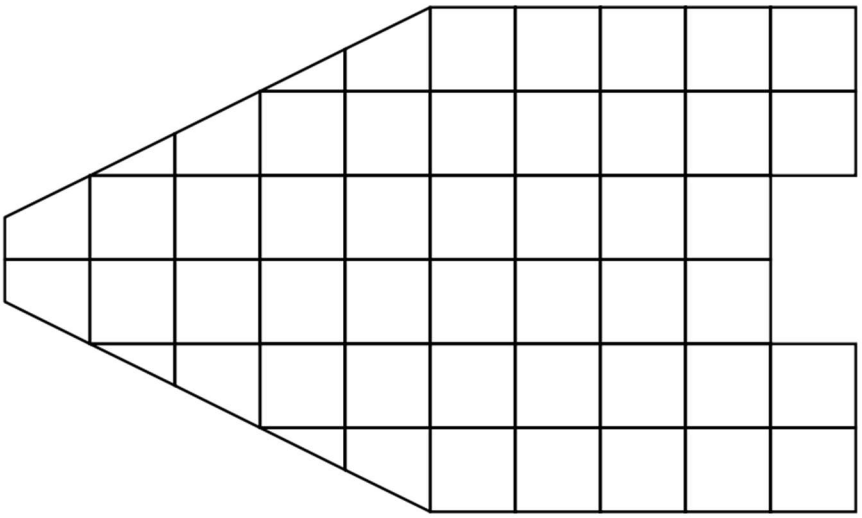
Малюнок 9 - клієнт - сервер візуалізації - сервер даних

Третій режим - це режим клієнт - сервер візуалізації - сервер даних. У цьому режимі всі три логічні частини працюють в окремих програмах. Як і раніше, клієнт підключений до сервера ренденингу через одне сокетне підключення. Сервер візуалізації та сервер даних з'єднані між собою
багатома підключеннями сокетів, по одному для кожного процесу на сервері візуалізації. Передача даних по сокетах зведена до мінімуму через їх повільність.

Хоча режим клієнт - сервер візуалізації - сервер даних підтримується, він майже ніколи не рекомендується для використанння. Початковий намір цього режиму полягає у використанні незвичайних середовищ, де можуть бути великі, потужні обчислювальні машини та друга менша паралельна машина з графічним обладнанням. Однак на практиці ми виявляємо, що будь-яка вигода майже завжди незначна порівняно з часом, необхідним для руху геометрії від сервера даних до сервера візуалізації. Якщо обчислювальна платформа набагато потужніша за графічний кластер, тоді слід використовувати програмне забезпечення для рендерингу на обчислювальній платформі. Якщо дві платформи мають приблизно однакову потужність, слід виконувати всі обчислення на графічному кластері.

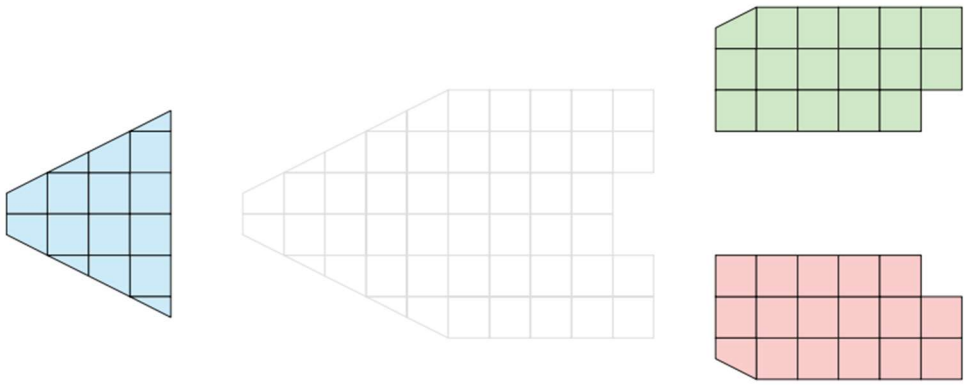
3.4 АЛГОРИТМИ ПАРАЛЕЛЬНОЇ ВІЗУАЛІЗАЦІЇ

У ParaView вже є вбудований паралельний фреймворк, тому паралельне виконання задач візуалізації є простою задачею. Дані, з якими ми маємо працюваьт, містяться в сітці, що означає, що дані вже були розбиті на дрібні шматочки. Ми можемо зробити візуалізацію на розподіленій паралельній машині, попередньо розділивши комірки міжпроцесами. Для демонстративних цілей розглянемо цю спрощену сітку.



Малюнок 10 - Приклад сітки

Тепер припустимо, що ми хочемо виконати візуалізацію цієї сітки за допомогою трьох процессорів. Ми можемо розділити клітини сітки, як показано нижче, на блакитні, зелені та рожеві області.



Малюнок 11 - Приклад розділення сітки на області

Після розділення деякі алгоритми візуалізації працюватимуть, просто дозволяючи кожному процесу самостійно запускати алгоритм на його власному набору комірок.

Наприклад, візьмемо операцію відсікання. Скажімо, що ми визначаємо площину відсікання і даємо ту саму площину кожному з

процесів.

Кожен процес може самостійно відсікає свої клітини цією площиною. Кінцевий результат співпадає наче ми виконували відсікання послідовно. Якщо ми зберемо клітини разом ми побачимо, що операція відсікання була виконана правильно.

3.5 РЕНДЕРИНГ

Рендеринг, або візуалізація - це процес синтезу зображень, які ви бачите на основі даних. Здатність ефективно взаємодіяти з вашими даними сильно залежить від швидкості візуалізації.

Завдяки досягненню апаратного прискорення 3D, ми маємо можливість швидкого відтворення 3D навіть на недорогих системах. Але, звичайно,

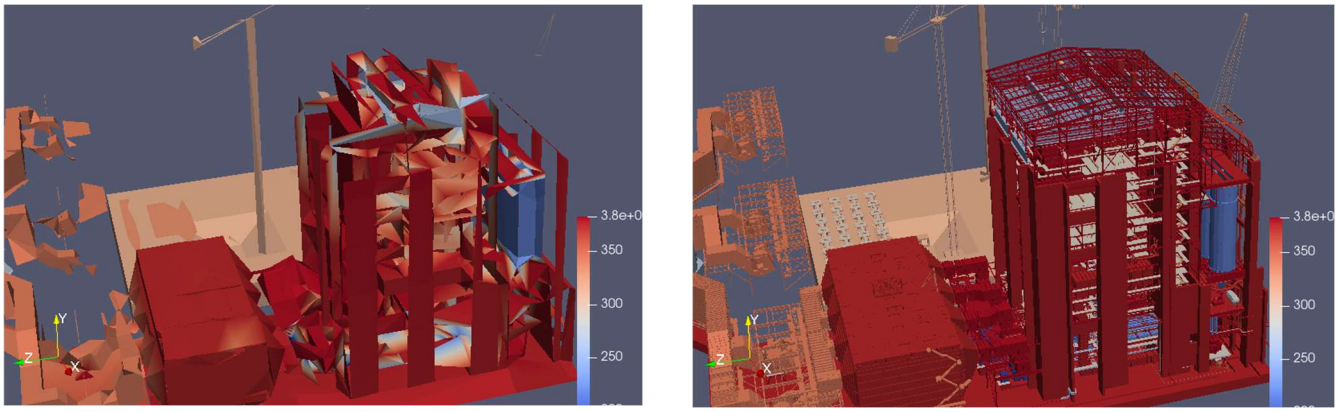
швидкість візуалізації пропорційна обсягу даних, що відображаються. Зі збільшенням обсягів даних процес візуалізації стає повільнішим.

Щоб ваш сеанс візуалізації був інтерактивним, ParaView підтримує два режими візуалізації, які автоматично змінюються при необхідності. У першому режимі візуалізації, дані відображаються з найвищим рівнем деталізації. Цей режим візуалізації забезпечує точне представлення даних. У другому режимі інтерактивний візуалізації, швидкість має перевагу над точністю. Цей режим візуалізації намагається надати швидку візуалізацію незалежно від розміру даних.

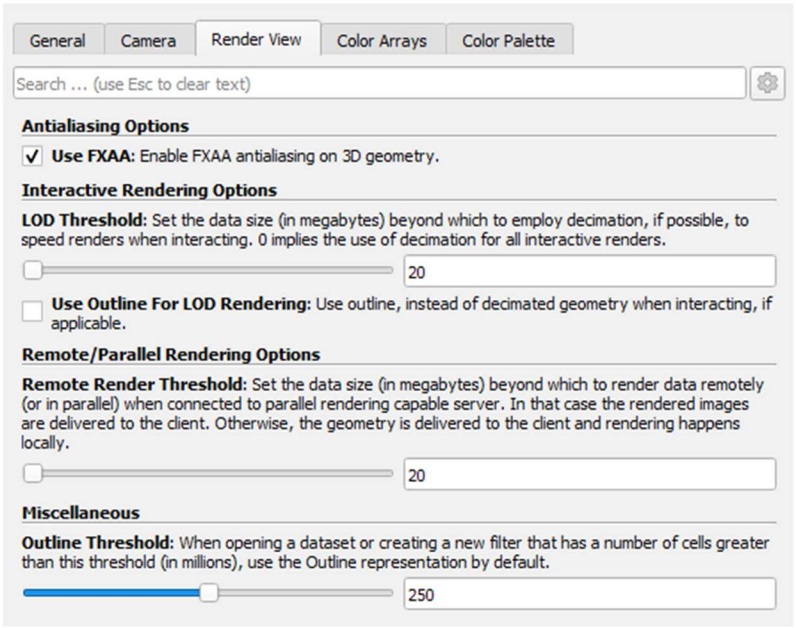
Під час взаємодії з тривимірним видом, наприклад при обертанні, панорамуванні або масштабуванні
за допомогою миші ParaView використовує інтерактивну візуалізацію. Це тому, що при взаємодії необхідна висока частота кадрів, щоб зробити ці функції придатними для використання і тому, що кожен
кадр негайно замінюється новим рендером, поки взаємодія відбувається так що в цьому режимі дрібні деталі менш важливі. У будь-який час, коли взаємодія з 3D-переглядом не відбувається, ParaView використовує нерухому точну візуалізацію, щоб отримати повну деталізацію даних для аналізу. Під час руху миші в тривимірному вигляді, щоб перемістити дані, буде відображена приблизна візуалізація, а повна візуалізація
буде представлена лише коли ви відпустите кнопку миші.

Інтерактивний рендер - це компроміс між швидкістю та точністю. Таким чином, багато з цих параметри візуалізації стосуються того, коли і як використовуються нижчі рівні деталізації.

Одними з найважливіших параметрів візуалізації є параметри деталізація — LOD (level of detail). Протягом інтерактивної візуалізації, геометрія може бути замінена на нижчий рівень деталізації (LOD), це приблизна геометрія з меншою кількістю примітивів.



Малюнок 12 - Порівняння LOD при інтерактивному та точному рендерингу



Малюнок 13 - Налаштування виду рендеру у Paraview

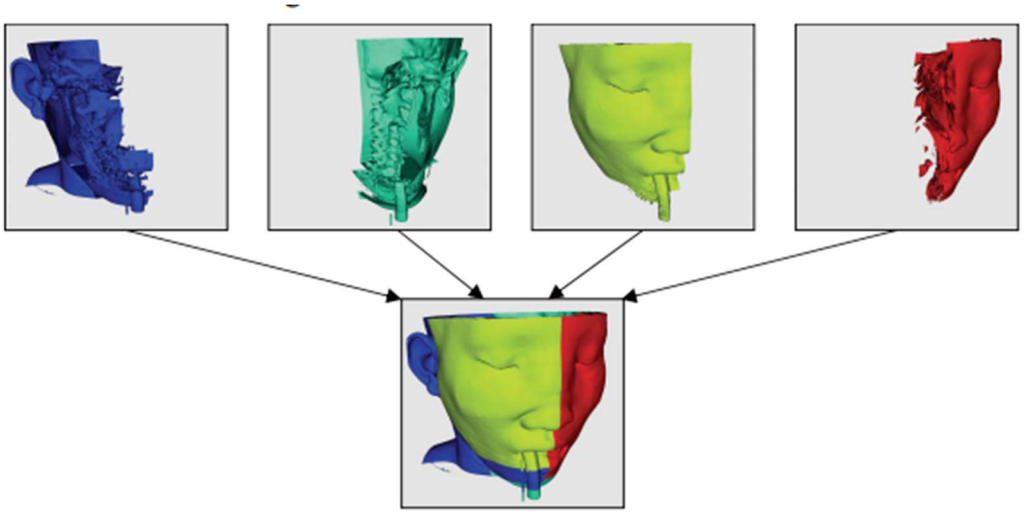
Можна змінювати роздільну здатність геометричного наближення. Алгоритм Децимації прагне розмістити багатокутники у сітці. На Зображенні ліве

зображення - повна роздільна здатність; середнє зображення - це децимація на 50^3 сітка та праворуч зображення - децимація на 10^3.

Параметри 3D візуалізації розміщені вікні налаштувань Edit → Settings → Render

Виконуючи паралельну візуалізацію, нам потрібно бути впевненими, що дані залишаються розділеними між усіма процесорами на всіх етапах аж до процесів візуалізації. ParaView використовує бібліотеку паралельної візуалізації під назвою IceT.

IceT використовує алгоритм останнього сортування для паралельної візуалізації. Цей паралельний алгоритм візуалізації дозволяє кожному процесу самостійно зробити свій розділ геометрії, а потім скласти часткові зображення разом, щоб сформувати остаточне зображення.



Малюнок 14 - Розділення рендерингу по потокам в IceT

Велика перевага паралельної візуалізації останнього сортування полягає в тому, що його ефективність повністю нечутлива до обсягу даних, що відображаються. Це робить його дуже масштабованим алгоритмом

і добре підходить для великих даних. Однак паралельні накладні витрати зростають лінійно з кількістю пікселів на зображенні. Отже, частина параметрів рендерингу залежать від зображення.

3.6 НАЛАШТУВАННЯ PARAVIEW СЕРВЕРА

Налаштування автономного ParaView зазвичай досить просте. Можна завантажити заздалегідь скомпільований бінарний файл, встановити його на свій комп'ютер і почати працювати.

Налаштування сервера ParaView є досить складнішим.

По-перше, доведеться скомпілювати сервер самостійно, бо там багато версій MPI, бібліотеки, яка робить можливим паралельне програмування, і кожна версія MPI може бути змінена відповідно до апаратних можливостей паралельного комп'ютера. Саму тому неможливо надійно забезпечити бінарні релізи сервера, щоб вони відповідали всім можливим комбінаціям конфігурацій серверів.

Для компіляції ParaView на паралельній машині вам знадобиться наступне.

- CMake Засоби збірки проекту (www.cmake.org)
- MPI
- OpenGL (або можна використовувати Mesa 3D, якщо не має графічного процессора, що підтримує OpenGL)
- Qt 4.2.3 (Додатково)
- Python (Додатково)

Компіляція без однієї з додаткових бібліотек означає, що функція буде недоступна. Компіляція без Qt означає, що у вас не буде графічного інтерфейсу

програми, без Python означає, що у вас не будуть доступні скрипти. Щоб скомпілювати ParaView, спочатку потрібно запустити CMake, що дозволить налаштувати компіляцію параметрів і вказати необхідні бібліотеки у вашій системі. Це створить файли make, які потім можна використовувати для збірки ParaView.

Паралельний запуск ParaView також є суттєво складнішим, ніж запуск автономного клієнту. Як правило, це включає ряд кроків, які будуть різні залежно від обладнання, на якому ви працюєте:

- вхід на віддалені комп'ютери
- виділення паралельних вузлів,
- запуск паралельно ї програми
- встановлення підключення та прокладання тунелів через брандмауери.

Клієнт-серверні зв'язки встановлюються через клієнтську програму paraview. Підключення та завершення з'єднання до серверів виконується за допомогою графічного інтерфейсу кнопками  та  . При запуску ParaView він автоматично підключається до спеціального вбудованого сервера. Він також підключається до вбудованого сервера у випадку, коли він втрачає підключення до сервера.

3.7 СТВОРЕННЯ ПРОГРАМИ ДЛЯ SERVER-SIDE РЕНДЕРИНГУ

Завдяки гарній падтримці бібліотек розробниками, э можливість створення власних додатків з використанням Paraview. Створимо додаток для рендерингу на стороні сервера за документацією з сайту розробників.

Методи наведені в документації передбачають рендеринг на екран, у цьому методі є декілька недоліків:

- неможливо зробити рендер з більшою роздільною здатністю ніж роздільна здатній дисплея
- необхідна відеокарта

Саме тому було вирішено зкомпілювати Paraview з використанням Mesa3D – бібліотеки, що реалізує графічні API OpenGL та Vulkan.

Це дасть можливість рендерити на сервері, на якому немає відеокарти, але рендеринг на відеокартах теж буде підтримуватися.

Програми OpenGL вимагають певного механізму для створення контексту візуалізації та управління ним. У системах Linux та Unix, що використовують X11 та графічні процесори, це зазвичай робиться за допомогою бібліотеки GLX (libGL), наданої графічним драйвером постачальника. Для систем без графічних карт або потужних графічних процесорів, пакет Mesa 3D забезпечує бібліотеку OSMesa (поза екраном Mesa) та кілька візуаторів на базі процесора, що забезпечують драйвери OpenGL. Далі будемо розглядати 2 візуалізаттра OSMesa: llvmpipe і swr. Візуатори llvmpipe і swr від Mesa є одними із найбільш корисних останніх нововведень у Mesa. Вони являють собою багатопоточні програмні драйвери OpenGL, які використовують LLVM і clang для компіляції JIT шейдерів GLSL. Драйвер llvmpipe для OSMesa був доданий у випуску 9.2.0 2013 року, а драйвер swr у випуску 12.0.0 2016 року. Драйвер llvmpipe доступний на більшості підтримуваних платформ Mesa, тоді як драйвер swr - набагато більш оптимізований і багатопоточний драйвер, доступний лише для новіших процесорів x86_64 з можливостями AVX або AVX2. На даний момент це найкращий варіант програмного забезпечення на базі OpenGL для ParaView та VTK як з точки зору функцій OpenGL, так і продуктивності (swr на

процесорах Intel та llvmpipe на всіх інших). Класична версія OSMesa більше не підтримує необхідні функції OpenGL, які вимагають новіші версії VTK та ParaView. Софтпайп Mesa, однопотокове програмне забезпечення, все ще може бути корисним у випадку помилки в драйверах llvmpipe або swr і забезпечує стабільну резервну можливість.

Для компіляції Paraview з підтримкою OSMesa потрібно спочатку встановити LLVM та Clang

Встановлення Clang досить тривіальне, а от встановлення LLVM відбувається за допомогою Cmake та потребує додаткових флагів при компіляції.

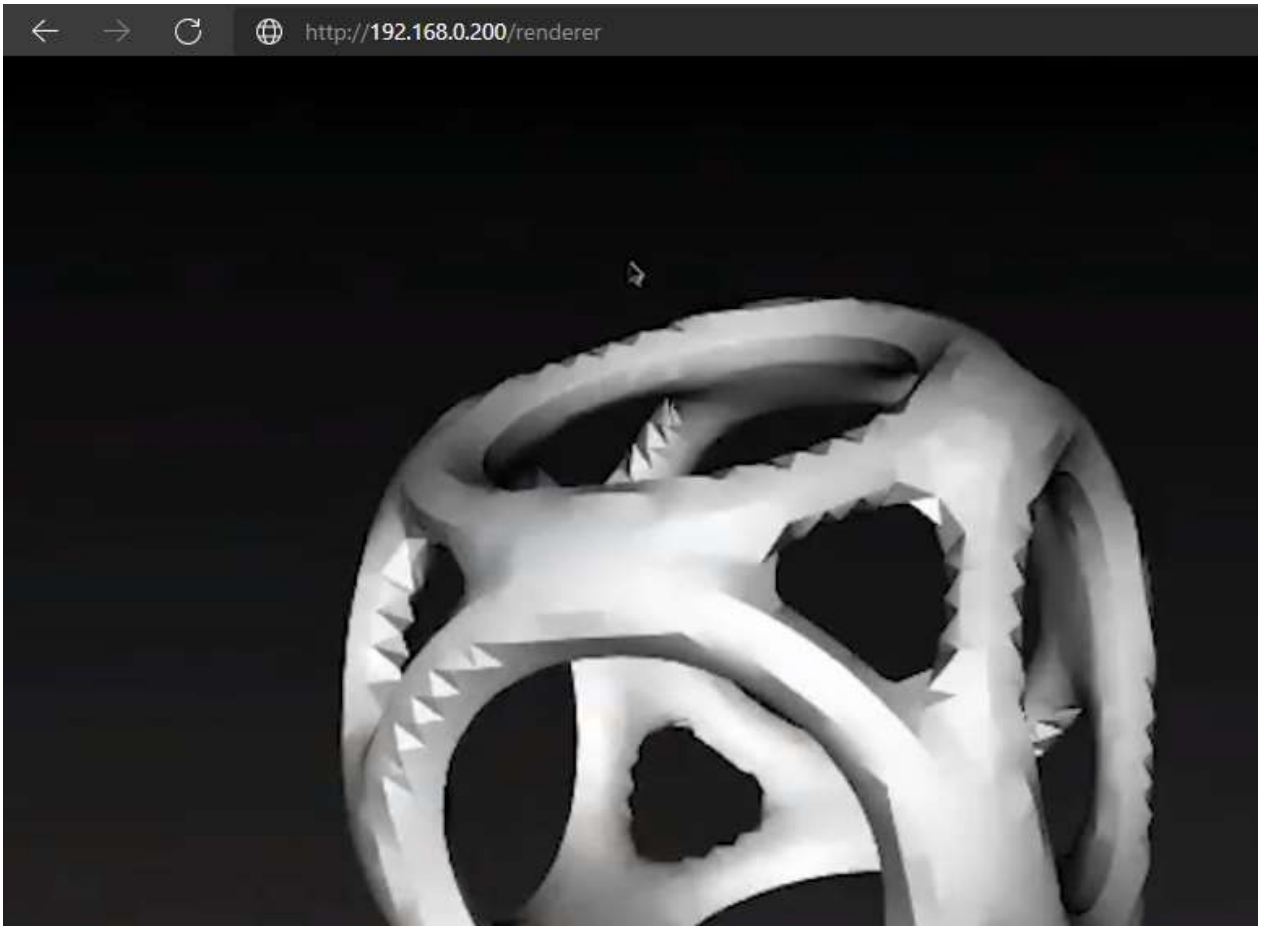
```
cmake                                           \
  -DCMAKE_BUILD_TYPE=Release                    \
  -DCMAKE_INSTALL_PREFIX=/path/to/install/llvm  \
  -DLLVM_BUILD_LLVM_DYLIB=ON                    \
  -DLLVM_ENABLE_RTTI=ON                         \
  -DLLVM_INSTALL_UTILS=ON                       \
  -DLLVM_TARGETS_TO_BUILD:STRING=X86            \
  ../llvm-12.0.0.src
```

Після цього можна встановити LLVM за допомогою make build install

Після цього потрібно встановити один з драйверів llvmpipe або swr. Згідно з даними зі сторінки розробників, llvmpipe працює трохи швидше, тому змінемо драйвер саме на нього.

```
GALLIUM_DRIVER=llvmpipe
glxinfo |grep "^OpenGL\(renderer\|core profile version\|vendor\)"
OpenGL vendor string: Asus, Inc.
OpenGL renderer string: Gallium 0.5 on llvmpipe (LLVM 12.0, 256 bits)
OpenGL core profile version string: 3.3 (Core Profile) Mesa 13.0.1 (git-
f4f382e)
```

Налаштування системи завершено, далі можна спробувати використати нашу програму, наведену в додатку



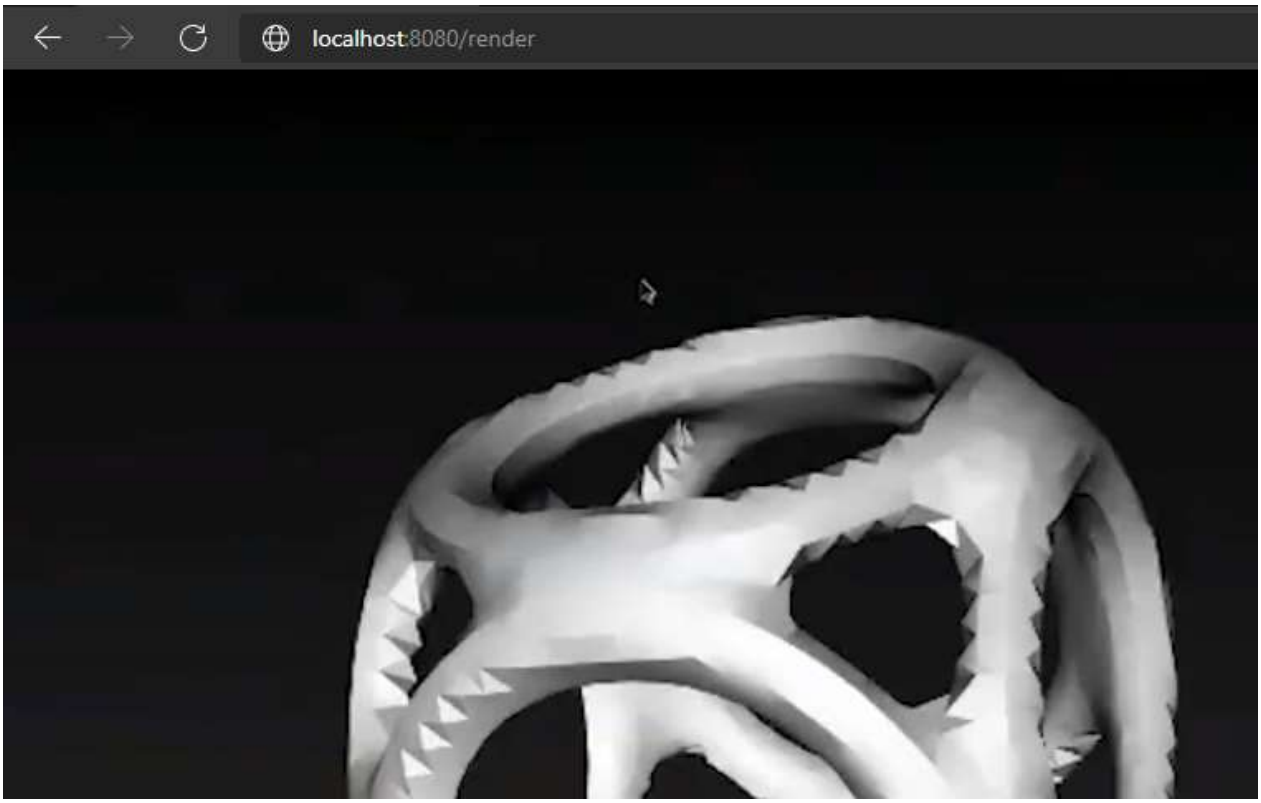
Малюнок 15 - Веб інтерфейс клієнта

Програма дозволяє оглядати зображення отримані з сервера та маніпулювати моделлю: повертати, збільшувати, зменшувати.

Порівняно з офіційним клієнтом Paraview наша веб програма виявилась досить повільною, тому у подальшому порівнянні будемо використовувати саме офіційний клієнт та сервер Paraview.

3.8 РЕНДЕРИНГ НА КЛІЄНТІ

Іноді сервера для серндерингу може не бути, а бо він може бути недоступний чи зайнятий. У такому випадку можна локально запустити сервер на свойому комп’ютері. Після запуску серверу можна отримати до нього доступ за адресою 127.0.0.1:8080/render або localhost:8080/render



Малюнок 16 - Рендеринг у веб сервісі, запущеному локально

Таким чином було виконано локальний рендеринг на клієнту в веб браузері. Таким рендеринг був значно повільнішим ніж використання офіційного клієнту Paraview. Офіційний клієнт також є більш гнучким до розмірів даних та LOD, тому рекомендуэться використовувати саме його.

4.1 ПОРІВНЯННЯ SERVER-SIDE ТА CLIENT-SIDE РЕНДЕРИНГУ

Для порівняння засобів Server-side та Client-side рендерингу будемо використовувати офіційний клієнт та сервер Paraview, оскільки він ефективніше використовує обчислювальні ресурси комп'ютера, та у результаті цього має менший час рендерингу.

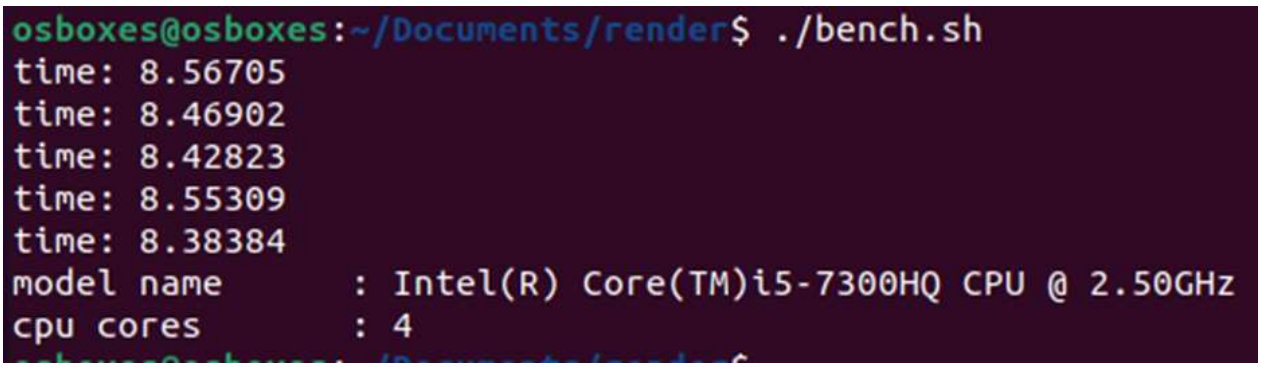
При порівнянні дуже важливо враховувати де знаходяться дані, які будуть візуалізовуватися. Для візуалізації на сервері, дані потрібно на нього завантажити, що теж викликає суттєву затримку, залежно від швидності, та затримки мережі.

Було розроблено програму, яка рендерить модель у високій роздільній здатності та виводить час, витрачений на рендеринг. Для зменшення похибок для кожної системи програму було запушено 5 разів наступним чином:

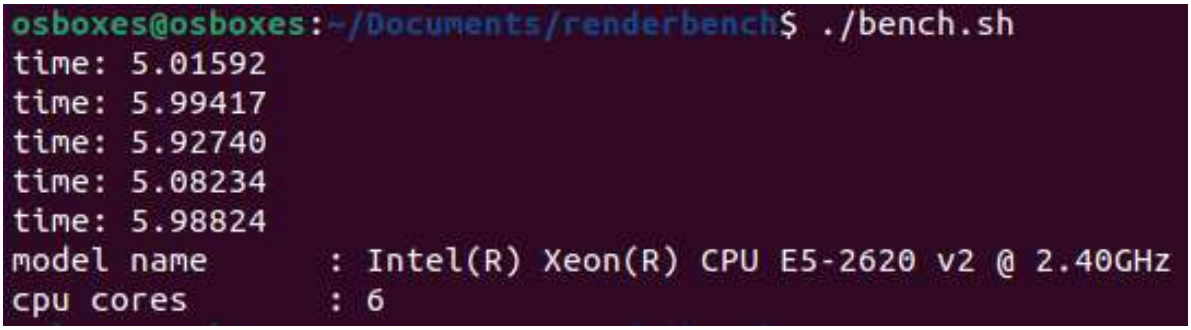
```
for i in {1..5};
do
        ./pvpython render.py;
done
cat /proc/cpuinfo  | grep "model\|cpu\ cores"  | tail -n2
```

код програми Render.py наведений у додатку Б

Для початку порівняємо швидкість рендерингу для середнього сервера та середнього ноутбука.



Малюнок 17 - Швидкість рендерингу ноутбука



Малюнок 18 - Швидкість рендерингу сервера

Як ми бачимо на малюнках 17 та 18 вище рендеринг на сервері виявився у 1,5 рази більш швидким. Якщо сервер буде мати більш сучасний процесор, або графічний процесор, він може мати перевагу у швидкості рендерингу до 5 разів при великих об’ємах даних. Під сервером будемо мати на увазі потужний віддалений комп’ютер з мережевими можливостями який здатен проводити рендеринг зображення.

4.2 ДАНІ ЗНАХОДЯТЬСЯ НА СТОРОНІ КЛІЄНТА

Розглянемо ситуацію коли дані у нас знаходяться на клієнті. Для рендерингу на сервері нам потрібно передати дані, зробити візуалізацію та передати дані назад до клієнта. У цьому випадку дуже важливо розглядати швидкість мережі між клієнтом та сервером. Повільна передача даних по мережі робить такий процес не дуже ефективним, хоча все ще може використовуватись для рендерингу великих файлів. Ефективність залежить від потужності сервера, а саме від того, у скільки разів швидше дані візуалізуються на сервері.

Після проведення експерименту, було виявлено час рендерингу на клієнті та сервері з урахуванням часу на передачу даних по мережі Fast Ethernet (100 Мбіт/с)

Розмір даних	5мб	10мб	50мб	200мб
Шкидкість рендерингу на сервері	5 сек	6 сек	14 сек	52 сек
Швидкість рендерингу на клієнті	3 сек	5,5 сек	19 сек	67 сек

Таблиця 1. Порівняння швидкості рендерингу при даних на клієнті

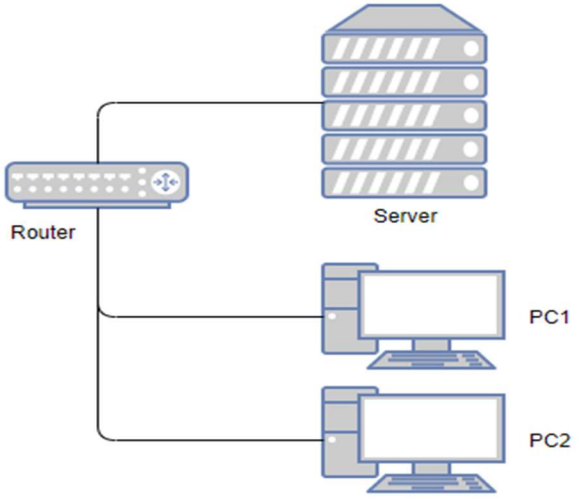
З наведеної таблиці можна зробити висновок, що при знаходженні даних на клієнті рендеринг на стороні сервера має значення лише у випадку роботи з середніми та великими обсягами даних.

Якщо сервер не достатньо потужний, знаходиться в іншій мережі або швидкість передачі даних менша, слід використовувати рендеринг на клієнті.

4.3 ДАНІ ЗНАХОДЯТЬСЯ НА СТОРОНІ СЕРВЕРА

У більшості задач реального світу великі обсяги даних зберігаються на серверах.

Для початку розглянемо приклад локальної мережі великої швидкості.



Малюнок 19 - Приклад: локальна мережа

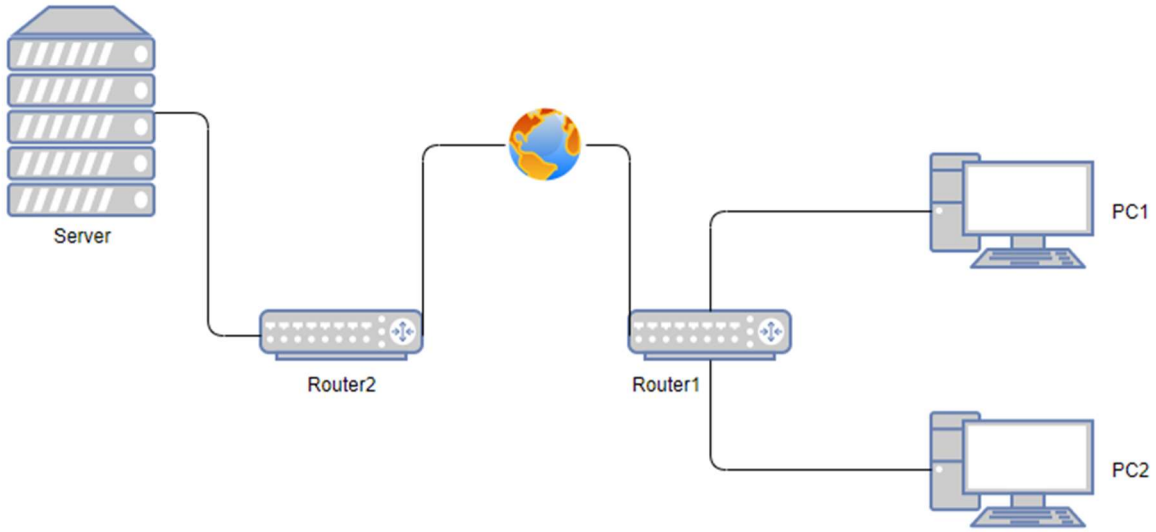
Для такої швидкої локальної мережі були отримані наступні результати

Розмір даних	5мб	10мб	50мб	200мб
Шкидкість рендерингу на сервері	3 сек	6 сек	21 сек	52 сек
Швидкість рендерингу на клієнті	5 сек	12 сек	42 сек	74 сек

Таблиця 2. Порівняння швидкості рендерингу, локальний сервер.

Такі результати є досить непоганими, навіть при не дуже потужному сервері. Це рекомендований спосіб використання рендерингу на сервері, оскільки затримки передачі даних по мережі майже не помітні.

У випадку коли сервер знаходиться у іншій мережі або на великій відстані від клієнта все стає значно складніше, оскільки, швидкість може бути значно меншою і залежить від навантеженості проміжних вузлів.



Маклюнок 20 - Приклад: сервер у іншій мережі

Розмір даних	5мб	10мб	50мб	200мб
Шкидкість рендерингу на сервері	5 сек	8 сек	25 сек	58 сек
Швидкість рендерингу на клієнті	6 сек	14 сек	53 сек	82 сек

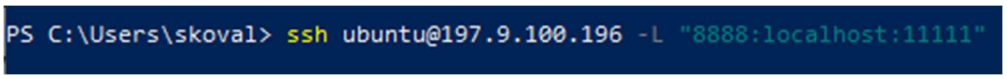
Таблиця 3. Порівняння швидкості рендерингу, віддалений сервер.

При такій мережі швидкість рендерингу буде такою самою як у локальній мережі, але буде більша затримка при інтерактивному маніпулюванні моделлю. Затримка буде залежати від відстані до мережі.

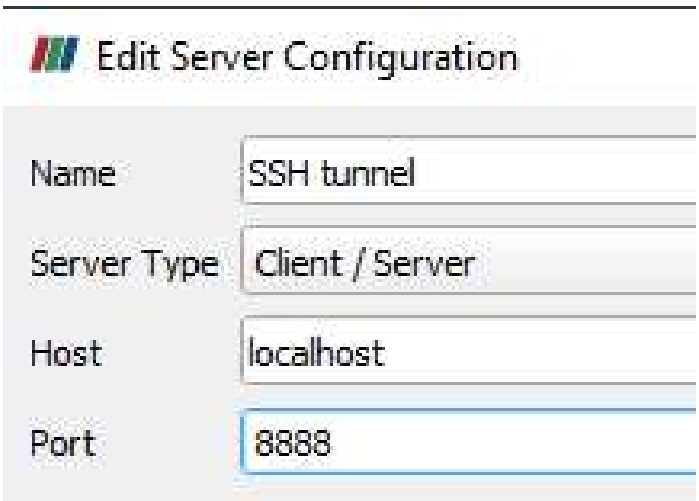
Для доступу до такого серверу рекомендується використання VPN або створення захищений тунелей, наприклад за допомогою SSH. Наприклад

ssh ubuntu@197.9.100.196 -L "8888:localhost:11111"

Після такого підключення, створюється зашифрований тунель, і після цього ми зможемо отримати доступ до сервера за адресою localhost:8888.



Малюнок 21 - Створення шифрованого SSH тунелю



Малюнок 22 - Підключення до створеного тунелю

Після підключення до серверу можна продовжити працювати так само, як і у випадку з рендерингом на клієнті, але рендерингом буде займатися сервер. Таким чином було налаштовано доступ до віддаленого сервера рендеринну. SSH використовує криптографічні ключі RSA, що дозволяють забезпечити велику надійність.

4.4 ВИСНОВКИ ДО РОЗДІЛУ 4

У розділі 4 було протестовано декілька конфігурацій server-side та client-side рендерингу за умов зберігання даних на сервері та на клієнті. Було проаналізовано отпримані результати та виявлено недоліки кожної з конфігурацій. У випадку коли дані знаходяться на сервері, доцільніше виконувати рендеринг на сервері. Однак у випадку коли модель не велика та у клієнта вистачає потужностей для візуалізації, найкращим методом буде рендеринг на клієнті, оскільки це забезпечить меншу затриму при маніпулюванні моделлю.

У випадку коли дані знаходяться на клієнті, рендеринг слід виконувати також на клієнті. Лише у ситуації, коли обсяги даних дуже великі має зміст використання серверного рендерингу.

5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, для вирішення задач 3D рендерингу за допомогою Paraview.

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

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

5.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи рендерингу 3D моделей за допомогою Paraview. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її

задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для рендерингу 3D моделей.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

5.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який вирішує задачу рендерингу за допомогою Paraview та будує його модель. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір мови програмування;

F_2 – вибір апаратного забезпечення

F_3 – вибір типу рендерингу client-side\server-side

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Python

б) C++

Функція F_2:

а) Рендеринг на процесорі CPU;

б) Рендеринг на графічному процесорі GPU

Функція F_3:

а) server-side

б) client-side

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 5.1).

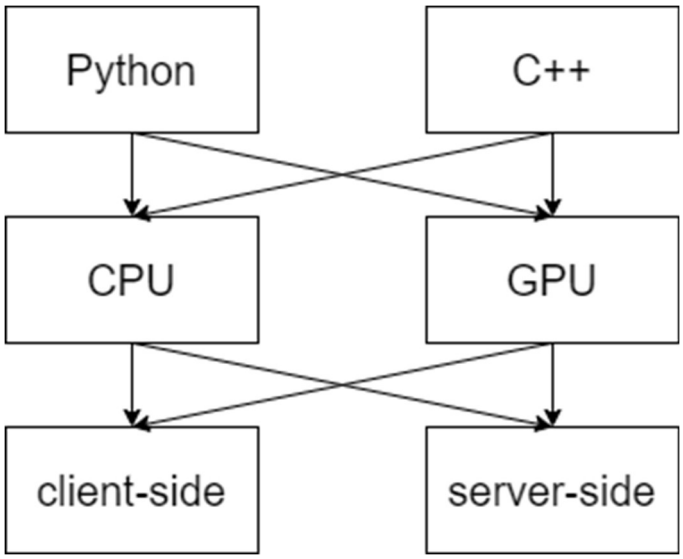


Рисунок 5.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіанти основних функцій.

Таблиця 5.2 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	*А*	Швидка розробка програми, доступність бібліотек, кросплатформеність	Низька швидкість роботи, особливо, якщо потрібно обробляти велику кількість даних
	Б	Код швидко виконується	Буде витрачено багато часу на розробку програми

F_2	*А*	Доступна ціна	Повільна швидкість рендерингу
	Б	Велика швидкість рендерингу	Висока ціна, трохи складніше налаштування системи
F_3	*А*	Простота реалізації, надійність системи.	Невелика швидкість рендерингу, неможливість роботи з великими проектами
	Б	Велика швидкість рендерингу, можливість рендерити складні проекти	Складність системи

На основі цієї карти будуємо позитивно-негативну матрицю варіантів основних функцій (Таб.5.2). Робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1*:*

Перевагу віддаємо швидкості вивчення, простоті використання та наявності стандартних бібліотек для обчислення. Для спрощення роботи по написанню коду варіант Б має бути відкинутий.

Функція F_2*:*

Обидва варіанти можна використовувати в розробці.

Функція F_3*:*

Віддаємо перевагу варіанту Б для забезпечення високої швидкості роботи системи та роботи з великими проектами.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а – F_2а – F_3б

F_1а – F_2б – F_3б

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

5.3 Обґрунтування системи параметрів ПП

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

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

- *X1* – швидкодія мови програмування;
- *X2* – швидкість обробки даних
- *X3* – швидкість розробки та налаштування системи;
- *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів обираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у таблиці 5.4.

Таблиця 5.4 - Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі

Швидкодія мови програмування;	X1	оп/мс	10000	14000	19000
Швидкість обробки даних	X2	Мб/с	20	100	400
Швидкість розробки та налаштування системи;	X3	годин	600	300	50
Потенційний об’єм програмного коду	X4	кількість рядків коду	5000	2500	1000

За даними таблиці 5.3 будуються графічні характеристики параметрів – рис. 5.2 – рис. 5.5.

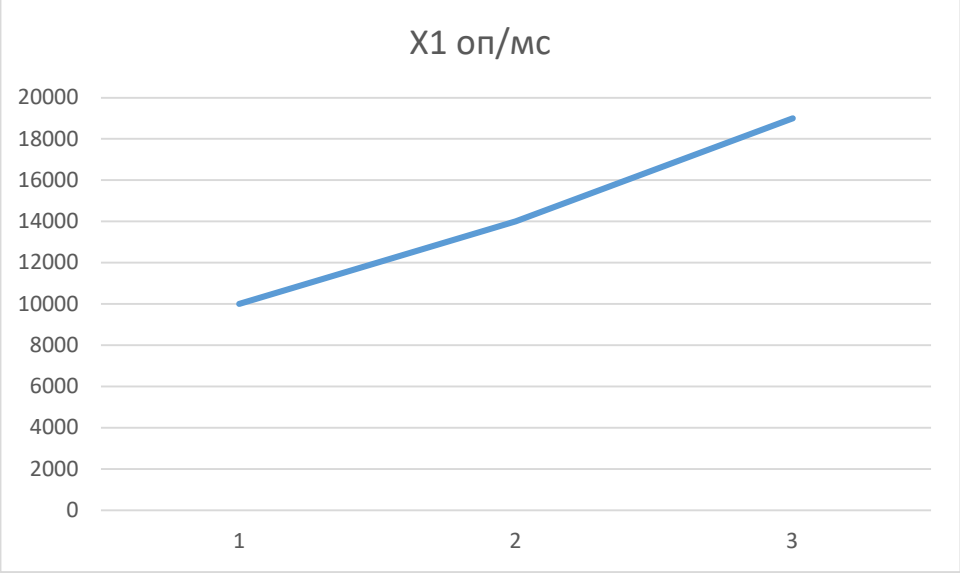


Рисунок 5.2 – X1, швидкодія мови програмування

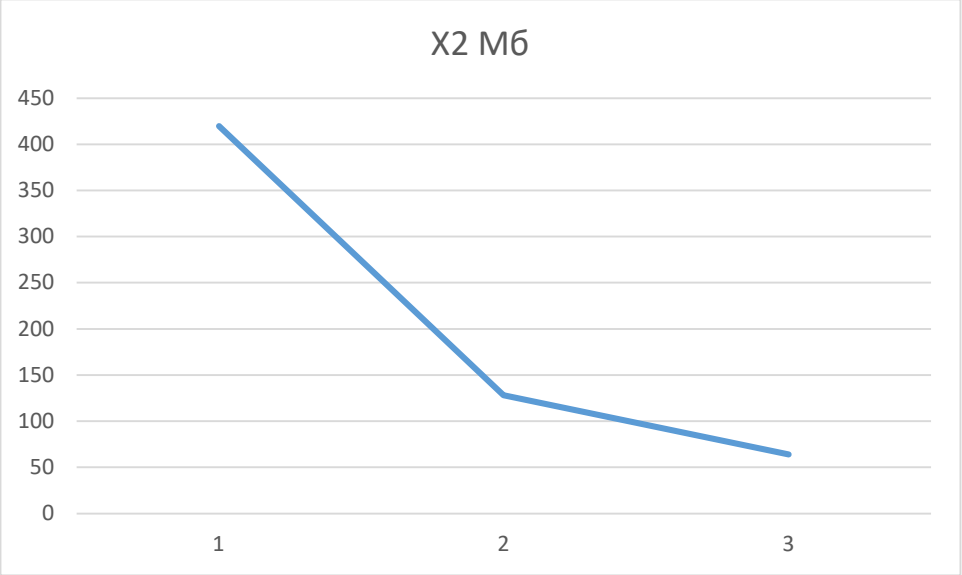


Рисунок 5.3 – Швидкість обробки даних

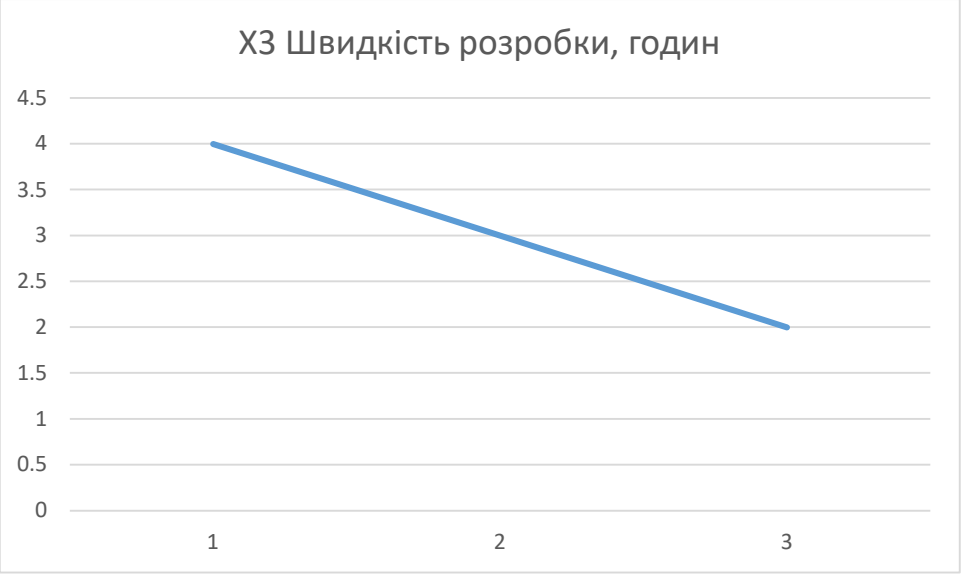


Рисунок 5.4 – X3, Швидкість розробки та налаштування системи

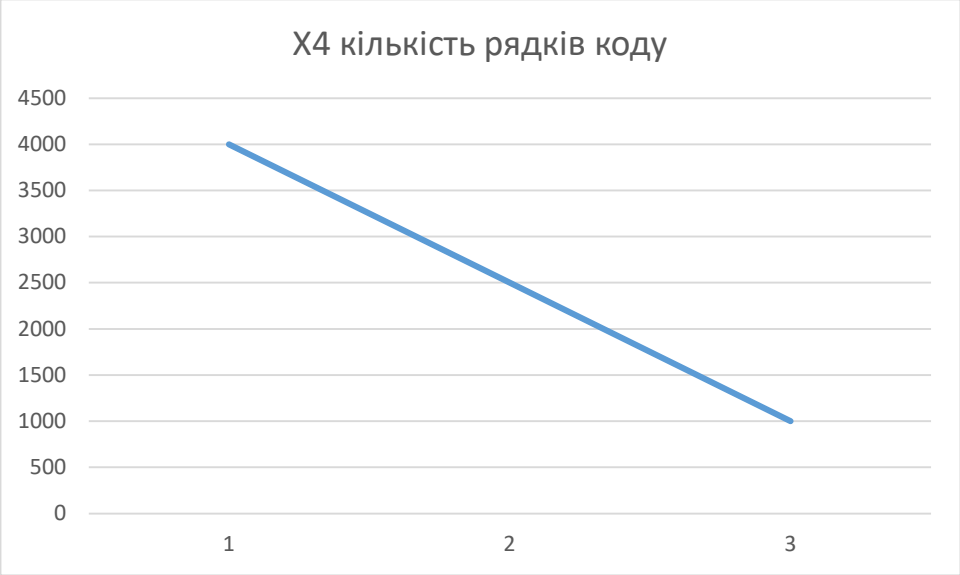


Рисунок 5.5 – X4, потенційний об’єм програмного коду

5.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

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

– визначення рівня значимості параметра шляхом присвоєння різних рангів;
– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 5.4

Таблиця 5.4 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	R_i
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	1	3	2	2	3	2	3	16	-8,5	72.25
X2	Швидкість обробки даних	Мб/с	5	5	5	6	6	5	4	36	11.5	132.25
X3	Швидкість розробки та налаштування системи;	мс	5	4	6	5	4	4	5	33	8.5	72.25
X4	Потенційний об’єм програмного коду	Кількість рядків коду	3	2	1	1	1	3	2	13	-11.5	132.25
	Разом		14	14	14	14	14	14	14	98	0	409

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

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

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 98, \quad (5.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 24{,}5 \quad (5.2)$$

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

$$\Delta_i = R_i - T. \quad (5.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 409 \quad (5.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 409}{7^2(4^3 - 4)} = 1{,}66 > W_k = 0{,}67. \quad (5.5)$$

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

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 5.5.

Таблиця 5.5 - Попарне порівняння параметрів.

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

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

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (5.6)$$

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

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

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (5.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \tag{5.8}$$

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

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{5.9}$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \tag{5.10}$$

.Як видно з таблиці 5.6, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,22	12,25	0,18	42,88	0,13
X2	1,5	1,0	1,5	1,5	5,5	0,34	30,25	0,44	166,38	0,53
X3	1,5	0,5	1,0	1,5	4,5	0,28	20,25	0,29	91,13	0,29
X4	0,5	0,5	0,5	1,0	2,5	0,16	6.25	0,09	15,63	0,05
Всього:					16	1	69	1	316	1

5.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Швидкість обробки даних), *X3* (Швидкість розробки та налаштування системи;) та *X4* (Потенційний об'єм програмного коду відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (Швидкодія мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 5.7):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (5.11)$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 5.7 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Пара метри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	A	X1	14000	4	0,13	0,85
F2	A	X2	100	5	0,53	1.05

	Б	X2	300	3	0,29	0.63
F3	А	X3	2500	7	0,05	1.48

За даними з таблиці 5.7 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \quad (5.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 0{,}85 + 1{,}05 + 1{,}48 = 3{,}38 ,$$

$$K_{K2} = 0{,}85 + 0{,}63 + 1{,}48 = 2{,}96 .$$

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

5.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Налаштування програмного продукту

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

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

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

Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.13)$$

де T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

$K_М$ – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

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

$$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

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

$$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино-днів.}$$

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

$$T_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328{,}64 \text{ людино-годин.}$$

$$T_{II} = (122.4 + 19.44 + 6.91 + 19.44) \cdot 8 = 1345.52 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два python програмісти з окладом 20000 грн., один backend розробник в області даних з окладом 15000. Визначимо середню зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.,} \qquad (5.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$С_{ч} = \frac{20000 + 20000 + 15000}{3 \cdot 21 \cdot 8} = 109{,}12 \text{ грн.} \quad (5.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_{ч} \cdot Т_{i} \cdot К_{Д}, \quad (5.16)$$

де $С_{ч}$– величина погодинної оплати праці програміста;

$Т_{i}$ – трудомісткість відповідного завдання;

$К_{Д}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 64.48 \cdot 1328.64 \cdot 1.2 = 173977{,}43$ грн.

II. $С_{ЗП} = 64.48 \cdot 1345.52 \cdot 1.2 = 146823{,}14$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 173977{,}43 \cdot 0.22 = 38274.94$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 146823{,}14 \cdot 0.22 = 32301{,}06$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_{М}$)

Так як одна ЕОМ обслуговує одного програміста з окладом 20000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_{Г} = 12 \cdot М \cdot К_{З} = 12 \cdot 20000 \cdot 0{,}2 = 48000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 48000 \cdot (1 + 0.2) = 57600 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 57600 \cdot 0{,}22 = 12{,}672 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 20000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 20000 = 5750 \text{ грн.},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 20000 \cdot 0.05 = 1150 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.9 =$$

= 1677.6 годин,

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1677.6 \cdot 0,2 \cdot 0,5 \cdot 3,51 = 588,83 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 20000 \cdot 0,67 = 13400 \text{ грн}.$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \qquad (5.17)$$

$С_{ЕКС}$ = 57600 + 12672 + 5750 + 1150 + 588,83 + 13400 = 91160,83 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{М\text{-}Г} = C_{ЕКС} / T_{ЕФ} = 91160{,}83 / 1677.6 = 54.34 \text{ грн/год.}$$

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

$$C_М = C_{М-Г} \cdot T, \quad (5.18)$$

I. $C_М = 54.34 \cdot 1328{,}64 = 72198{,}29$ грн.

II. $C_М = 54.34 \cdot 1345.52 = 73115{,}55$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0{,}67, \quad (5.19)$$

грн.

I. $C_Н = 173977{,}43 \cdot 0{,}67 = 116564{,}59$ грн.

II. $C_Н = 146823 \cdot 0{,}67 = 98371{,}41$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н, \quad (5.20)$$

I. $С_{ПП}$ = 173977,43 + 38274.94 + 72198,29 + 116564,59 = 401015.25 грн.

II. $С_{ПП}$ = 146823,14 + 32301,06 + 73115,55 + 98371,41 = 350611,16 грн.

грн.

5.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj} / С_{Фj}, \quad (5.21)$$

$$К_{ТЕР1} = 4,25/ 240195,29 = 1,77 \cdot 10^{-5},$$

$$К_{ТЕР2} = 3,74/ 243246,91 = 1,53 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}$= 1,77 $\cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР}$ = 1,77 $\cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Python;

– Велика швидкість рендерингу

– Можливість роботи з великими проектами

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

5.8 Висновки до розділу 5

Проведено повний функціонально-вартісний аналіз програмного продукту. Визначено та проведено оцінку основних функцій програмного продукту. Визначено параметри, які характеризують програмний продукт. Проведено експертне оцінювання параметрів та аналіз якості варіантів реалізації функцій.

Проведено економічний аналіз варіантів розробки – трудомісткість, витрати на заробітну плату та інші витрати.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

У результаті виконання дипломної роботи було розглянуто основні проблеми використання віддаленого рендеринга за допомогою Paraview та VTK, а також актуальність подібних методів для створення систем рендерингу.

Основними недоліками client-side рендерингу була неможливість рендерингу середніх та великих проектів у реальному часі, а також великі витрати при масштабуванні.

Основними причинами розгляду server-side рендерингу була значно більша швидкість рендерингу порівняно з client-side рендерингом.

Однією з головних причин також були безпека, та контроль доступу до даних, у системах з великою кількістю клієнтів.

Було розглянуто засоби VTK та Paraview для вирішення задач рендерингу та створено власний додаток для клієнт серверної взаємодії.

Також було проведено аналіз особливостей VTK та Paraview, виявлені переваги та недоліки кожної з систем.

Таким чином, технічним результатом дипломної роботи є створення програми для віддаленого рендерингу на сервері, та порівняння client-side та server-side рендерингу. Виявлені та вирішені проблеми у налаштуванні таких систем.

ПЕРЕЛІК ПОСИЛАНЬ

1. Paraview Documentation. [Електронний ресурс]. https://docs.paraview.org/
2. VTK Documentation. [Електронний ресурс]. https://vtk.org/documentation/
3. Візуалізація та анімація геофізичних моделей. [Електронний ресурс]. https://habr.com/ru/post/492362/
4. Paraview and Offscreen rendering. [Електронний ресурс]. https://kitware.github.io/paraview-docs/latest/cxx/Offscreen.html/
5. Mesa: Agent-based modeling in Python 3+ [Електронний ресурс]. https://mesa.readthedocs.io/en/stable/
6. Кристофер Р. Джонсон, Чарльз Д. Хансен The Visualization Handbook
7. Уилл Шредер The visualization toolkit

Додаток А

```
import os
import sys


# Try handle virtual env if provided
if "--virtual-env" in sys.argv:
    virtualEnvPath = sys.argv[sys.argv.index("--virtual-env") + 1]
    virtualEnv = virtualEnvPath + "/bin/activate_this.py"
    if sys.version_info.major < 3:
        execfile(virtualEnv, dict(__file__=virtualEnv))
    else:
        with open(virtualEnv) as venv:
            exec(venv.read(), dict(__file__=virtualEnv))


# import paraview modules.
from paraview.web import pv_wslink
from paraview.web import protocols as pv_protocols


from wslink import register as exportRpc


from paraview import simple
```

```
from wslink import server


import pv_protocol as local_protocol


import argparse
class _Server(pv_wslink.PVServerProtocol):

    authKey = "wslink-secret"
    viewportScale = 1.0
    viewportMaxWidth = 2560
    viewportMaxHeight = 1440
    settingsLODThreshold = 102400

    @staticmethod
    def add_arguments(parser):
        parser.add_argument(
            "--virtual-env", default=None, help="Path to virtual environment to use"
        )
        parser.add_argument(
            "--viewport-scale",
            default=1.0,
```

```
    type=float,
    help="Viewport scaling factor",
    dest="viewportScale",
)
parser.add_argument(
    "--viewport-max-width",
    default=2560,
    type=int,
    help="Viewport maximum size in width",
    dest="viewportMaxWidth",
)
parser.add_argument(
    "--viewport-max-height",
    default=1440,
    type=int,
    help="Viewport maximum size in height",
    dest="viewportMaxHeight",
)
parser.add_argument(
    "--settings-lod-threshold",
    default=102400,
```

```
        type=int,
        help="LOD Threshold in Megabytes",
        dest="settingsLODThreshold",
    )

@staticmethod
def configure(args):
    _Server.authKey = args.authKey
    _Server.viewportScale = args.viewportScale
    _Server.viewportMaxWidth = args.viewportMaxWidth
    _Server.viewportMaxHeight = args.viewportMaxHeight
    _Server.settingsLODThreshold = args.settingsLODThreshold

def initialize(self):
    self.registerVtkWebProtocol(pv_protocols.ParaViewWebMouseHandler())
    self.registerVtkWebProtocol(
        pv_protocols.ParaViewWebViewPort(
            _Server.viewportScale,
            _Server.viewportMaxWidth,
            _Server.viewportMaxHeight,
        )
```

```
        )
        self.registerVtkWebProtocol(
            pv_protocols.ParaViewWebPublishImageDelivery(decode=False)
        )
        self.registerVtkWebProtocol(local_protocol.ParaViewCone())
        self.updateSecret(_Server.authKey)
        self.getApplication().SetImageEncoding(0)
        view = simple.GetRenderView()
        view.EnableRenderOnInteraction = 0
        view.OrientationAxesVisibility = 0
        view.Background = [0.5, 0.5, 0.5]
        pxm = simple.servermanager.ProxyManager()
        interactionProxy = pxm.GetProxy("settings",
"RenderViewInteractionSettings")
        interactionProxy.Camera3DManipulators = [
            "Rotate", "Pan",    "Zoom",
            "Pan",    "Roll",   "Pan",
            "Zoom",   "Rotate", "Zoom",
        ]
        renderingSettings = pxm.GetProxy("settings", "RenderViewSettings")
        renderingSettings.LODThreshold = _Server.settingsLODThreshold
```

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="ParaView Cone Sample application")
    server.add_arguments(parser)
    _Server.add_arguments(parser)
    args = parser.parse_args()
    _Server.configure(args)
    server.start_webserver(options=args, protocol=_Server)
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />
    <link rel="icon" href="<%= BASE_URL %>favicon.ico" />
    <title>vue-vtkjs-pvw-template</title>
  </head>
  <body>
```

```
    <noscript>
      <strong
        >We're sorry but vue-vtkjs-pvw-template doesn't work properly without
        JavaScript enabled. Please enable it to continue.</strong
      >
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
import os, time

from wslink import register as exportRpc

from paraview import simple, servermanager
from paraview.web import protocols as pv_protocols
cone = simple.Cone()

class ParaViewCone(pv_protocols.ParaViewWebProtocol):

    @exportRpc("vtk.initialize")
```

```
def createVisualization(self):
    simple.Show(cone)
    return self.resetCamera()


@exportRpc("vtk.camera.reset")
def resetCamera(self):
    view = self.getView('-1')
    simple.Render(view)
    simple.ResetCamera(view)
    try:
        view.CenterOfRotation = view.CameraFocalPoint
    except:
        pass

    self.getApplication().InvalidateCache(view.SMProxy)
    self.getApplication().InvokeEvent('UpdateEvent')

    return view.GetGlobalIDAsString()
```

```
@exportRpc("vtk.cone.resolution.update")
def updateResolution(self, resolution):
    cone.Resolution = resolution
    self.getApplication().InvokeEvent('UpdateEvent')



@exportRpc("viewport.mouse.zoom.wheel")
def updateZoomFromWheel(self, event):
  if 'Start' in event["type"]:
    self.getApplication().InvokeEvent('StartInteractionEvent')

  viewProxy = self.getView(event['view'])
  if viewProxy and 'spinY' in event:
    rootId = viewProxy.GetGlobalIDAsString()
    zoomFactor = 1.0 - event['spinY'] / 10.0

    fp = viewProxy.CameraFocalPoint
    pos = viewProxy.CameraPosition
    delta = [fp[i] - pos[i] for i in range(3)]
    viewProxy.GetActiveCamera().Zoom(zoomFactor)
    viewProxy.UpdatePropertyInformation()
```

```
        pos2 = viewProxy.CameraPosition
        viewProxy.CameraFocalPoint = [pos2[i] + delta[i] for i in range(3)]

      if 'End' in event["type"]:
        self.getApplication().InvokeEvent('EndInteractionEvent')
```

ДОДАТОК Б

```
for i in {1..5};
do
                        ./pvpython render.py;
done
cat /proc/cpuinfo  | grep "model\|cpu\ cores"  | tail -n2

from paraview.simple import *
import time

#read a vtp
reader = XMLPolyDataReader(FileName="unc-power-plant.vtp")

#position camera
view = GetActiveView()
if not view:
```

```
    # When using the ParaView UI, the View will be present, not otherwise.
    view = CreateRenderView()
view.CameraViewUp = [0, 1, 0]
view.CameraFocalPoint = [0, 0, 0]
view.CameraViewAngle = 45
view.CameraPosition = [0.2,0.2,0.2]

#draw the object
Show()

#set the background color
view.Background = [1,1,1]  #white

#set image size
view.ViewSize = [2000, 2000] #[width, height]

dp = GetDisplayProperties()

#set point color
dp.AmbientColor = [1, 0, 0] #red
```

```
#set surface color

dp.DiffuseColor = [0, 1, 0] #blue


#set point size

dp.PointSize = 2


#set representation

dp.Representation = "Surface"

t0 = time.perf_counter()

Render()

t1 = time.perf_counter() - t0

print("Time elapsed: ", t1) # CPU seconds elapsed (floating point)

#save screenshot

WriteImage("test.png")
```