Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»
Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Конструктивно-продукційне моделювання мультиатрибутивних просторових графових фракталів»
за освітньою програмою: **Інженерія програмного забезпечення**
зі спеціальності: **121 Інженерія програмного забезпечення**
Виконав: студент групи «ПЗ2322»

_____ /Олександр ЛЕТУЧИЙ/

Керівник: _____ /проф. Віктор ШИНКАРЕНКО/

Нормоконтролер: _____ /доц. Світлана ВОЛКОВА/

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

Дніпро – 2025 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note

to Master's Thesis

on the topic: «Constructive-synthesizing modeling of multi-attribute spatial graph fractals»
according to educational curriculum **software engineering**
in the Speciality: **121 Software engineering**

Done by the student of the group PZ2322: /Oleksandr LETUCHYI/

Scientific Supervisor: /Viktor SHYNKARENKO/

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2025

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Факультет «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: магістр
Освітня програма: Інженерія програмного забезпечення
Спеціальність: 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
(підпис)
Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу магістра
студенту Летучому Олександру Ігоровичу
1. Тема роботи: «Конструктивно-продукційне моделювання мультиатрибутивних просторових графових фракталів»
Керівник роботи: Шинкаренко Віктор Іванович, професор
затверджені наказом № 1443 ст від 10.10.2023
2. Строк подання студентом роботи: 18.01.2025 р.
3. Вихідні дані до роботи: _____
4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
4.1 Аналітична частина: Аналіз проблеми використання графових фракталів; обґрунтування напрямку дослідження мультиатрибутивних просторових графових фракталів
4.2 Основна частина: проектування та розробка інструментального забезпечення; дослідження ефективності просторових графових фракталів
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): презентація; відео роботи програми

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строки виконання етапів	Примітка
1	Огляд предметної галузі	29.01.24 – 16.03.24	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	17.03.24 – 27.05.24	
3	Огляд існуючих програмних рішень	25.03.24 – 27.05.24	
4	Проектування програми, визначення основних вимог	03.06.24 – 28.06.24	30%
5	Розробка програми	04.08.24 – 01.12.24	
6	Тестування та налагодження	02.12.24 – 15.12.24	60%
7	Проведення дослідів	16.12.24 – 08.01.2025	
8	Оформлення пояснювальної записки	01.11.2024 – 12.01.2025	100%
9	Подання кваліфікаційної роботи до кафедри	19.01.05	
10	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	23.01.25	

Студент _____ Олександр ЛЕТУЧИЙ
(підпис) (Ім’я ПРІЗВИЩЕ)

Керівник роботи _____ проф. Віктор ШИНКАРЕНКО
(підпис) (Ім’я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра.

Об’єкт розробки – десктопний застосунок для моделювання мультиатрибутивних просторових графових фракталів, що взаємодіє з ЛІРА-САПР.

Мета роботи – моделювання та дослідження мультиатрибутивних просторових графових фракталів та дослідження їх напружено-деформованого стану. Підтвердження гіпотези ефективності природньо створених структур кристалічних ґраток з точки зору макроструктури та напружено-деформованого стану у будівництві.

Методи дослідження – використання фрактальних властивостей при побудові просторових графових фракталів, використання ЛІРА-САПР для аналізу їх напружено-деформованого стану.

Результати та їх новизна: аналіз напружено-деформованого стану згенерованих стрижневих конструкцій з фрактальними властивостями кристалічної ґратки для їх подальшого використання у будівництві.

Вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 3 сторінок;

Перший розділ – аналіз проблеми та огляд інструментів. Складається з 11 сторінок;

Другий розділ – висвітлено обґрунтування напрямку дослідження. Складається з 12 сторінок;

Третій розділ – описано процес проектування і розробки інструментального програмного забезпечення для досліджень. Складається з 17 сторінок;

Четвертому розділ – дослідження напружено-деформованого стану згенерованих конструкцій з фрактальними властивостями кристалічних ґраток, аналіз експериментальної обчислюваної складності з їх генерації. Складається з 18 сторінок

Висновки – підсумок всієї роботи. Складається з 2 сторінок; список використаних джерел – включає в себе бібліографічний список використаної літератури. Складає 3 сторінки;

Додатки – технічне завдання, керівництво користувача, робочий код проекту, тези. Складають 127 сторінок.

Кількість таблиць: 10. Кількість рисунків: 50. Кількість бібліографічних посилань: 30.

Ключові слова: моделювання, САПР, атрибутика, жорсткість, навантаження, фрактал, ґратка Браве, кристалічна структура, граф, конструктивно-продукційні структури, Ліра.

ЗМІСТ

ВСТУП

Актуальність роботи. Мультиатрибутивні просторові графові фрактали є потужним інструментом для моделювання складних систем, що мають ієрархічну структуру, нерегулярну геометрію та властивості самоподібності. Вони знаходять широке застосування в таких галузях, як комп'ютерна графіка, аналіз природних явищ, біоінформатика, системи штучного інтелекту та інші. Особливу увагу заслуговує проектування інженерних конструкцій, зокрема ферма, яка може розглядатися як граф, що складається з ключових вузлів та ребер, що їх з’єднують. Конструкції, побудовані за допомогою мультиатрибутивних просторові графових фракталів, можуть застосовуватися для збереження витрачених ресурсів або підвищення стійкості.

Для генерації мультиатрибутивних просторових графових фракталів задіяно підхід конструктивно-продукційного моделювання, який складається з чотирьох складових: спеціалізація, інтерпретація, конкретизація, реалізація. На основі цього підходу було розроблено програмний додаток, який дозволяє не тільки генерувати мультиатрибутивні просторові графові фрактали, а також візуалізувати їх. Додана інтеграція зі сторонніми додатками для дослідження експериментальної обчислюваної складності даного підходу.

Об’єкт дослідження. Конструктивно-продукційне моделювання мультиатрибутивних просторових графових фракталів, інтеграційні можливості ЛІРА-САПР.

Предмет дослідження. Генерація стрижневих конструкцій з фрактальними властивостями кристалічних ґраток та розрахунки напружено-деформованого стану.

Мета і завдання дослідження. Метою є підтвердження гіпотези щодо ефективності в будівельних конструкціях структур подібних до кристалічних ґраток. Для цього потрібно було вирішити завдання інтеграції користувацьких програм з ЛІРА-САПР для виконання розрахунків напружено-деформованого стану

стрижневих конструкцій з фрактальними властивостями кристалічних ґраток, а також дослідити експериментальну обчислювальну складність з їх генерування.

Методи дослідження. Для генерації мультиатрибутивних графових фракталів використовувався конструктивно-продукційних підхід, в основі якого лежать формальні граматики. Для визначення експериментальної обчислюваної складності було використано алгоритм підрахунку ключових операцій. Аналіз напружено-деформованого стану виконувався за допомогою програмного додатку ЛІРА-САПР.

Наукова новизна. Удосконалено формування мультиатрибутивних просторових графових на основі конструктивно-продукційного підходу. Вперше виконано дослідження експериментальної обчислюваної складності алгоритмів конструктивно-продукційного моделювання. Аналіз напружено-деформованого стану стрижневих структур з фрактальними властивостями кристалічної ґратки.

Практичне значення. Виконані дослідження та дана програма може бути використана фахівцями, студентам, викладачам з галузі цивільного будівництва, інформатики, програмної інженерії, для дослідження та розрахунку напружено-деформованого стану металевих конструкцій з фрактальними властивостями кристалічних ґраток.

Апробація результатів дослідження. Процес та результати робити над даним дослідженням доповідалися на засіданнях наукових семінарів кафедри КІТ.

Також були надані доповіді на наступні конференції:

– IEEE 18th International Conference on Computer Science and Information Technologies (CSIT) (Львів, 2023 р.);

– XVII та XVIII міжнародні науково-практичні конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (Дніпро 2023 р., 2024 р.);

– Міжнародна науково-практична конференція «Інформаційні Технології в Металургії та Машинобудуванні ІТММ 2024» (Дніпро, 2024 р.);

Публікації за темою роботи. Публікація за темою «Constructive-synthesizing modeling of fractal crystal lattices» була представлена та включена до

міжнародного журналу «IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)», [1]

Також були опубліковані праці у збірниках до наступних конференцій:

– «Засоби формування графових 3D фракталів», тези XVII Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» [2];

– «Графові фрактали з варіативністю процесу формування», тези міжнародної науково-технічної конференції «Інформаційні технології в металургії та машинобудуванні ІТММ 2024» [3];

– «Використання ЛІРИ-САПР у клієнт-серверному додатку», тези XVIII Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» [4].

1 АНАЛІЗ ПРОБЛЕМИ ВИКОРИСТАННЯ ГРАФОВИХ ФРАКТАЛІВ

1.1 Огляд об’єктів дослідження

1.1.1 Особливості застосування фракталів

Фрактали [5] — це математичні структури, що мають властивість самоподібності, тобто частини фракталу повторюють його структуру на різних масштабах. Завдяки цій властивості фрактали широко застосовуються в різних галузях, зокрема в природничих науках, техніці, медицині та мистецтві. Їхнє використання дозволяє вирішувати складні проблеми, пов’язані з моделюванням, оптимізацією, аналізом даних та візуалізацією.

Природні об'єкти, такі як гори, дерева, хмари або річкові системи, мають складну геометрію, яку важко відтворити традиційними методами: традиційні геометричні об’єкти такі як лінія, коло, полігон тощо, не можуть відобразити їх складність. Фрактали дозволяють створювати реалістичні моделі цих явищ Рисунок 1.1.

Рисунок 1.1 – Згенерований ландшафт за допомогою фрактальних алгоритмів

Фрактали застосовуються для моделювання та аналізу складних біологічних систем, які мають нерегулярну або розгалужену структуру:

– кровоносні судини;
– пухлини;
– дихальна система Рисунок 1.2

Рисунок 1.2 – Модель бронхіального древа

У задачах комп’ютерного зору, обробки зображень і аналізу текстур часто необхідно працювати з великими обсягами даних або складними структурами. Традиційні методи стиснення не завжди ефективні для складних зображень із багатьма дрібними деталями. Фрактальні алгоритми стискання дозволяють зменшити розмір файлів без значної втрати якості, зберігаючи деталі завдяки математичній самоподібності.

1.1.2 Застосування ґраток Браве при формуванні кристалічної структури

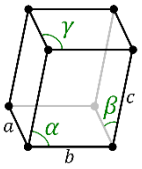
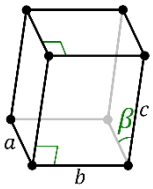
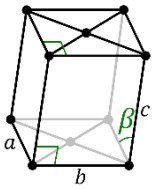
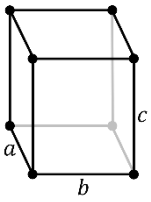
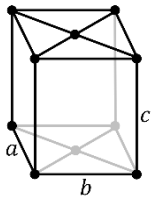
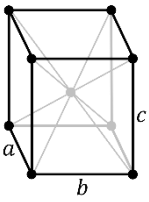
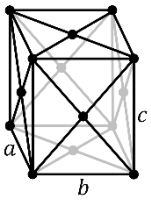
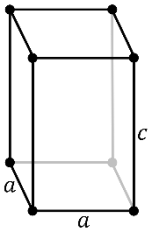
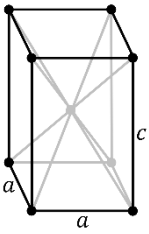
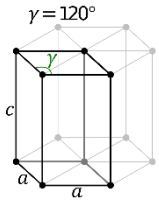
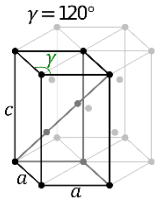
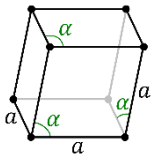
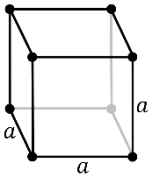
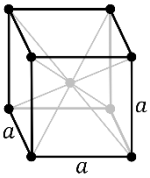
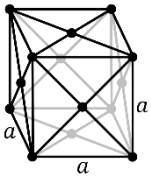
Ґратка Браве [6] — це математична модель, яка описує періодичну структуру кристалів у тривимірному просторі. Вона визначає, як розташовані точки (атоми, молекули або іони) у просторі, формуючи кристалічну ґратку. Ґратка Браве є базовою концепцією в кристалографії та твердотільній фізиці, оскільки вона описує симетрію кристалічних структур і є основою для класифікації всіх можливих кристалічних структур.

Під ґраткою або системою трансляцій Браве розуміється набір елементарних трансляцій (трансляційна група), які дозволяють утворити нескінченну періодичну структуру в просторі. Тип ґратки визначається співвідношеннями між довжинами трансляцій a, b, c та кутами між ними α, β, γ. Це дозволяє розділити ґратки на три категорії, шість сингоній та сім кристалічних систем залежно від геометричних параметрів і симетрій:

1. Нижча категорія (всі трансляції не рівні один одному)
   - триклінна: $a \neq b \neq c,\ \alpha \neq \beta \neq \gamma \neq 90°$
   - моноклінна: $a \neq b \neq c,\ \alpha = \gamma = 90°,\ \beta \neq 90°$
   - ромбічна: $a \neq b \neq c,\ \alpha = \beta = \gamma = 90°$
2. Середня категорія (дві трансляції із трьох рівні між собою)
   - тетрагональна: $a = b \neq c,\ \alpha = \beta = \gamma = 90°$
   - гексагональна: $a = b \neq c,\ \alpha = \beta = 90°,\ \gamma = 120°$
   - тригональна: $a = b = c,\ \alpha = \beta = \gamma < 120° \neq 90°$
3. Вища категорія (усі трансляції рівні між собою)
   - кубічна: $a = b = c, \alpha = \beta = \gamma = 90°$

Також є різні типи ґраток Браве залежно від центрувань Таблиця 1.1: примітивна, базоцентрована, об'ємноцентрована, гранецентрована, двічі об'ємноцентрована. Ґратки Браве є основою для вивчення кристалічної структури речовин, дозволяючи досліджувати їхні фізичні, хімічні й механічні властивості. Наприклад, гранецентровані кубічні ґратки, як у золота чи алюмінію, забезпечують вищу пластичність через щільніше пакування атомів порівняно з об’ємноцентрованими, як у заліза.

Таблиця 1.1 – Різновиди кристалічних ґраток

Кристалічна система	Тип центрування ґратки Браве				
	примітивна	базоцентрована	об'ємноцентрована	гранецентрована	двічі об'ємноцентрована
Триклінна					
Моноклінна					
Ромбічна					
Тетрагональна					
Гексагональна					
Тригональна					
Кубічна					

1.2 Аналіз сучасного стану дослідження за науковими літературними джерелами

Графом називається сукупність вузлів і ребер, що їх з'єднують. Графовий фрактал [7] — це фрактал, який має структуру графа, тобто його елементи організовані таким чином, що зберігають самоподібність при масштабуванні на різних рівнях, що у свою чергу властиво кристалічним структурам. Поняття графового фракталу є досить новим та невідомим в широкому сенсі. Використання графових фракталів присутнє в роботах [8, 9]

Дослідження [8] присвячене методам створення фрактальних полімерів Рисунок 1.3 і матеріалів шляхом хімічної самозбірки. Описано синтез наномасштабних фрактальних конструкцій, таких як шестикутний ґаскет, схожий на трикутник Серпінського, із заданими молекулярними властивостями. В роботі розглядаються способи контролю архітектури та властивостей цих конструкцій через використання різних методів зв'язку, таких як ковалентні, водневі чи йонні зв'язки. Такі фрактальні матеріали потенційно можуть застосовуватися у фотоелектричних пристроях, зберіганні енергії, каталізі й доставці ліків.

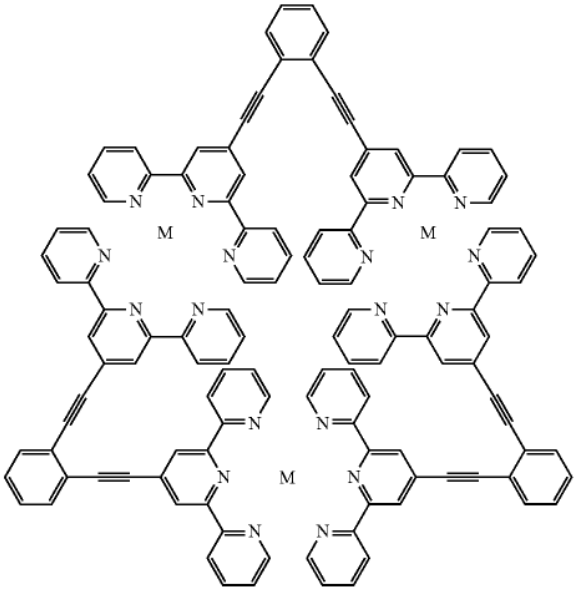


Рисунок 1.3 – Фрактальні конструкції другого покоління сформована з використанням моле-кули N-етілморфіну в поєднані з CH3-OH/CHCl3, де замість M може бути любий метал, такий як RU(II), Fe(II) або ZnII)

В дослідженні [9] акцентується увага на оптимізації фрактальних просторових рамних конструкцій Рисунок 1.4, виготовлених з порожнистих труб, для витримки легкого компресійного навантаження. У роботі розглядається вплив ієрархічної структури на механічну ефективність, стійкість до навантажень та допустимість помилок. Автори аналізують різні рівні ієрархії та фрактальні характеристики, що впливають на оптимальність конструкції, а також пропонують методи виготовлення таких структур із використанням сучасних технологій. Робота також наголошує на важливості врахування компромісів між механічною ефективністю, вартістю виробництва та стійкістю конструкції.

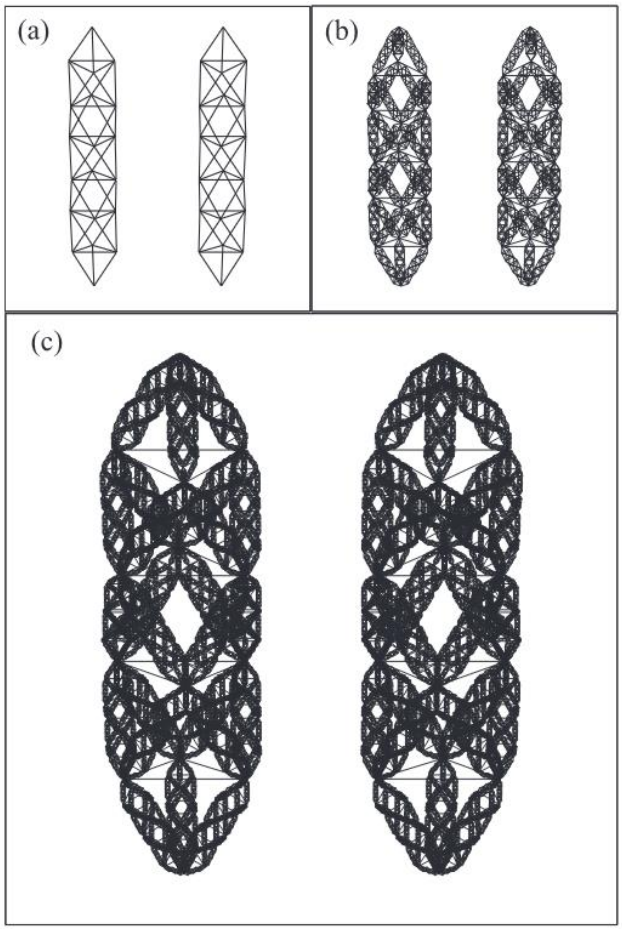


Рисунок 1.4 – Прогресія до вищих поколінь ієрархічної структури. (a) зображує простий просторовий фрейм, (b) показує просторовий фрейм із двома рівнями ієрархії, а (c) показує просторовий фрейм із трьома рівнями ієрархії

Проте можливість використання графових структур, які зустрічаються у природі, тобто структури з фрактальними подібностями кристалічних ґраток не було досліджено, зокрема в будівництві.

1.3 Аналіз існуючих програмних рішень

1.3.1 Аналіз кристалічних структур за допомогою Vesta

VESTA [10] — це багатофункціональна програма для візуалізації, моделювання та аналізу кристалічних структур Рисунок 1.5. Вона забезпечує широкий спектр інструментів для дослідження властивостей кристалів, таких як електронна щільність, розподіл заряду та структурні перетворення. Завдяки підтримці різних форматів даних, VESTA дозволяє завантажувати, редагувати та

відображати кристалічні структури, створювати тривимірні зображення та анімації, а також експортувати графічні матеріали високої якості.

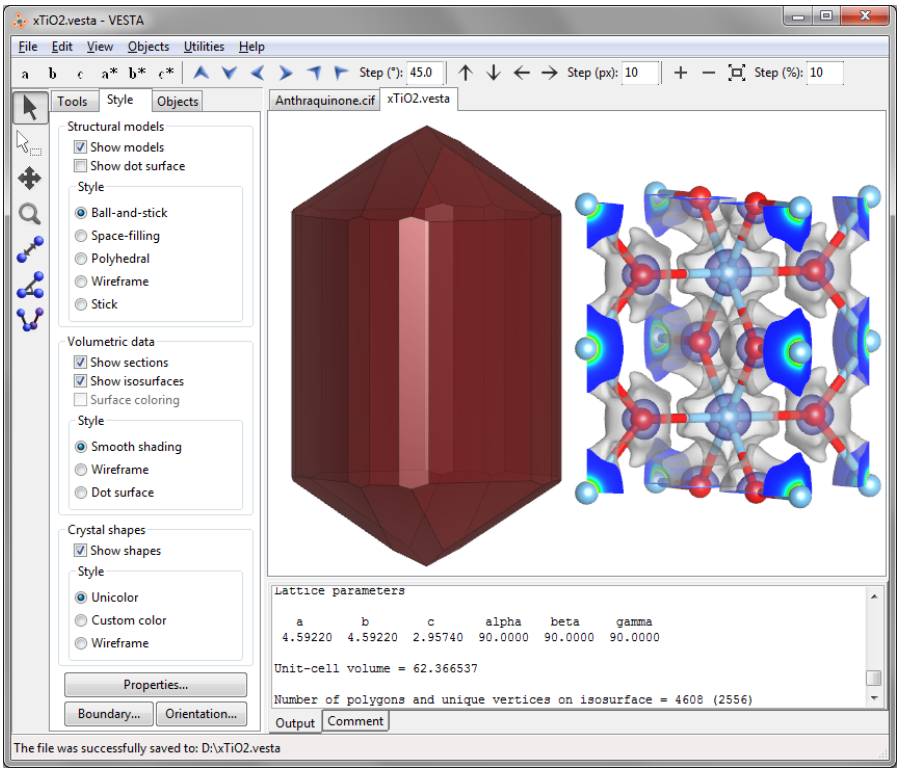


Рисунок 1.5 – Вікно програми VESTA, що демонструє геометричні властивості показаної кристалічної комірки кристала, а також її об’єм

Ця програма активно використовується в кристалографії, матеріалознавстві, хімії та фізиці, пропонуючи зручність у наукових дослідженнях і навчанні. Серед переваг VESTA — безкоштовність, кросплатформеність та підтримка багатьох форматів файлів. Недоліком можна вважати складний інтерфейс, який вимагає часу на освоєння. Програму можна завантажити для Windows, Mac OS X і Linux з офіційного сайту.

1.3.2 AutoCAD: Інструмент для точного проектування та моделювання

AutoCAD [11] — це програмне забезпечення для автоматизованого проектування, розроблене компанією Autodesk. Воно дозволяє створювати точні 2D-креслення та 3D-моделі, що робить його незамінним інструментом у таких галузях, як архітектура, будівництво, машинобудування, електротехніка та ландшафтний дизайн. Завдяки широкому набору функцій AutoCAD надає можливості для проектування складних об'єктів, моделювання тривимірних

структур Рисунок 1.6, рендерингу та аналізу фізичних характеристик, таких як площі, об'єми та довжини. Програма підтримує роботу з багатьма популярними форматами файлів, забезпечуючи інтеграцію з іншими інженерними та дизайнерськими додатками.

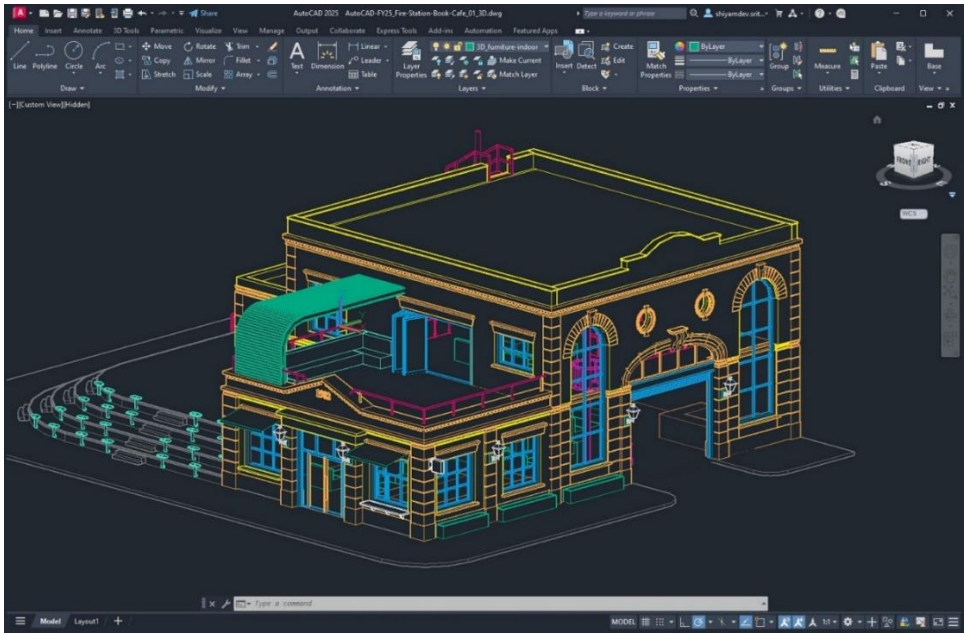


Рисунок 1.6 – Головне вікно програми AutoCAD, яке демонструє 3D проект будівлі

Однією з ключових переваг AutoCAD є його гнучкість, що дозволяє адаптувати інструменти під потреби конкретного користувача, а також точність, яка важлива для створення технічних проєктів. Проте, освоєння програми може вимагати значного часу через складний інтерфейс і великий обсяг функціоналу. Незважаючи на високу вартість ліцензії, AutoCAD залишається стандартом у багатьох професійних сферах завдяки своїй потужності, кросплатформенності та постійним оновленням.

1.3.3 Аналіз конструкцій за допомогою ЛІРА-САПР

ЛІРА-САПР [12] — це сучасне програмне забезпечення для розрахунку, аналізу та проектування будівельних конструкцій Рисунок 1.7. Воно орієнтоване на інженерів, архітекторів і конструкторів, які працюють над статичним і динамічним аналізом складних об'єктів. ЛІРА-САПР широко використовується

для розрахунків сталевих, залізобетонних, дерев'яних і кам'яних конструкцій у цивільному та промисловому будівництві.

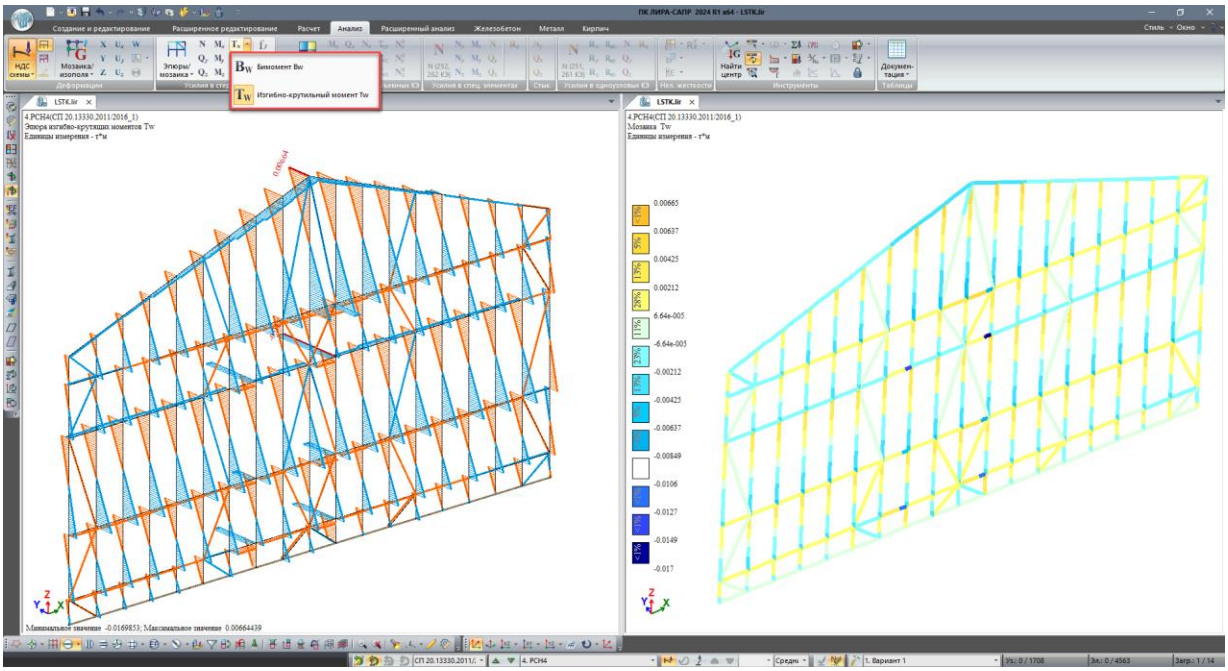


Рисунок 1.7 – Головне вікно програмного додатку ЛІРА-САПР, аналіз згинально-крутного моменту T_w для показаної конструкції

Однією з головних переваг ЛІРА-САПР є можливість виконання розрахунків відповідно до національних і міжнародних будівельних норм, таких як ДБН, Єврокоди чи СНіП. Програма дозволяє моделювати конструкції будь-якої складності, включаючи багатоповерхові будівлі, мости, інженерні споруди та інші об'єкти. Її функціонал охоплює широкий спектр задач, зокрема лінійний і нелінійний аналіз, розрахунок на сейсмічні навантаження, теплові й вітрові впливи, а також розрахунок ґрунтових основ.

Програма підтримує інтеграцію з іншими CAD-системами, такими як AutoCAD, Revit або Tekla Structures, що спрощує процес обміну даними та зменшує кількість ручної роботи. ЛІРА-САПР також оснащена зручним інтерфейсом для створення геометрії моделей, налаштування навантажень і аналізу результатів, що включають графіки, діаграми й візуалізації напружено-деформованого стану конструкцій.

ЛІРА-САПР вирізняється високою точністю розрахунків, гнучкістю у налаштуваннях і багатофункціональністю. Її кросплатформені можливості

дозволяють працювати як на локальних машинах, так і через хмарні обчислення. Серед недоліків програми можна відзначити необхідність часу для освоєння складного функціоналу, а також високі системні вимоги для роботи з великими моделями.

Висновки до першого розділу

У розділі було розглянуто об’єкти дослідження та які проблеми вирішуються за їх допомогою. Розглянуто та проаналізовано останні сучасні дослідження, що стосуються графових фракталів, їх практичне застосування, а також висвітлено недостатньо дослідженні теми, такі як ефективність стрижневих конструкцій з фрактальними властивостями кристалічних ґраток з точки зору макроструктури та напружено-деформованого стану. Розглянуто програмні додатки, їх переваги та недоліки. Дані програмні додатки можуть допомогти у вирішенні проблеми аналізу та розрахунку напружено-деформованого стану графових структур, які являють собою стрижневі конструкції з фрактальними властивостями кристалічної ґратки, проте вони фокусуються лише на окремих проблемах, так як аналіз геометричної складової або розрахунок напружено-деформованого стану, та не надають можливостей з генерації просторових мультиатрибутивних графових фракталів.

На основі аналізу літератури та наявних наукових підходів до розв’язання цієї проблеми було визначено такі ключові вимоги до програмних засобів:

- наявність конструктора, що базується на четвірці етапів конструктивно-продукційної ґраматики, що підтримує мультиатрибутику просторових графових фракталів;
- можливість задавати атрибути за допомогою користувацького інтерфейсу;
- підтримка інтеграції з ЛІРА-САПР для можливості аналізу напружено-деформованого стану;
- можливість отримувати результати розрахунків експериментальної обчислювальної складності алгоритмів з генерації просторових мультиатрибутивних графових фракталів.

2 ОБҐРУНТУВАННЯ НАПРЯМКУ ДОСЛІДЖЕННЯ МУЛЬТИАТРИБУТИВНИХ ПРОСТОРОВИХ ГРАФОВИХ ФРАКТАЛІВ

Просторові графові фрактали є мало дослідженим явищем, що відкриває нові можливості для моделювання складних структур, які повторюють закономірності природи. Унікальність таких фракталів полягає в їхній здатності відтворювати багаторівневі форми, що спостерігаються в природному середовищі, наприклад, у структурі дерев, системі кровоносних судин чи мінералів. Особливості їхньої природньої структури дозволяють використовувати фрактали для опису об’єктів, які мають властивості самоподібності, що забезпечує оптимальність із точки зору розподілу матеріалу і навантаження.

Наприклад, фрактали знаходять застосування у проектуванні аеродинамічних поверхонь і навіть у медицині для аналізу структур тканин та судин. Такі приклади демонструють широкі можливості використання принципів фрактальної геометрії в різних галузях науки та техніки.

У рамках даного дослідження особливу увагу буде приділено аналізу стрижневих структур з фрактальними властивостями кристалічної ґратки. Природні кристалічні структури часто характеризуються високою ефективністю з точки зору макроструктури та напружено-деформованого стану. Гіпотеза, яка перевіряється в роботі, полягає в тому, що такі структури можуть бути ефективно використані у будівництві для створення стійких конструкцій.

Також буде виконано аналіз експериментальної обчислюваної складності з генерації таких структур, які генеруються за допомогою конструктивно-продукційного підходу.

2.1 Обґрунтування використання конструктивно-продукційного моделювання

Для моделювання фракталів існує кілька підходів, серед яких виділяються алгоритмічний і функціонально-алгоритмічний методи. Зокрема, часто використовують систему ітерованих функцій, що базується на сукупності стискаючих відображень, L-системи, а також стискаючі афінні автомати.

Окремої уваги заслуговує підхід, що базується на конструктивно-продукційній структурі (КПС) [13, 14], яка розроблена на основі формальних граматик.

КПС відкриває широкі можливості для моделювання різноманітних конструкцій і процесів у багатьох сферах, включаючи інформаційні технології, машинобудування, робототехніку, біологію, хімію та їх відгалуження. Цей підхід дозволяє ефективно створювати й описувати складні об'єкти й процеси завдяки своїй універсальності та адаптивності до різних завдань.

КПС доцільно застосовувати для моделювання та поетапної деталізації стрижневої структури з фрактальними властивостями кристалічної ґратки. Така структура базується на елементарних комірках, які багаторазово повторюються, утворюючи суперкомірки. Особливістю підходу є акцент на атрибутах елементів, їхній формі та параметрах, що притаманні кристалічним ґраткам Браве, зокрема довжині ребер, кутам між ними та додатковим характеристикам, які визначають геометрію й симетрію ґратки.

Проте, на відміну від суто геометричних властивостей, у моделюванні також враховуються фізичні характеристики, такі як жорсткість елементів, прикладена сила, напруження та інші механічні параметри, що впливають на поведінку структури в реальних умовах. Поєднання геометричних і фізичних властивостей дозволяє більш точно описувати особливості кристалічних ґраток і їхніх фрактальних аналогів.

Присутні і допоміжні віртуальні атрибути, такі як колір вузла, які допомагають у деталізації генерованих структур.

2.1.1 Визначення конструктивно-продукційної структури

Конструктивно-продукційною структурою називається послідовність із трьох складових:

$$C = \langle M, \Sigma, \Lambda \rangle,$$

де M – неоднорідний носій структури, Σ – сигнатура, що складається з безлічі операцій зв'язування, підстановки і висновку, операцій над атрибутами і відносин підстановки, Λ – інформаційне забезпечення, яке включає: онтологію, ціль, правила, обмеження, умови початку та завершення конструювання.

Призначення конструктивно-продукційної структури (КПС) [] полягає у формуванні множин конструкцій шляхом застосування таких операцій, як зв'язування, підстановка, виведення тощо, які визначаються правилами аксіоматики.

В M можна виділити підмножини: T – терміналів, N – нетерміналів (допоміжних, абстрактних елементів), із властивостями $T \cap N = \emptyset$, $\varepsilon \in T$, $\varepsilon \notin N$, де ε – порожній елемент.

Сигнатура Σ складається з множини операцій: Ξ - зв'язування, Θ - підстановки і виводу, Φ - операцій над атрибутами. Сигнатура містить також відношення підстановки «→». Таким чином, формально сигнатурою є $\Sigma = \langle \Xi, \Theta, \Phi, \{\rightarrow\} \rangle$ з властивостями: $\Xi \cap \Theta = \emptyset$; $\Xi \cap \Phi = \emptyset$; $\Theta \cap \Phi = \emptyset$, $\varepsilon \in \Phi$. Сигнатура складається з імен операцій $\{\otimes_j\}$, що містять набір атрибутів w_i, та представляється як ${}_w\otimes \in \Sigma$.

Під формою ${}_{w_l}l$ з набором атрибутів w_l розуміють форму ${}_lw$ з атрибутом w називається набір терміналів і нетерміналів, що об'єднуються операціями зв'язування. Конструкцією називається форма, яка містить тільки термінали:

- ${}_{w_l}l = {}_{w_0}\otimes ({}_{w_1}m_1, {}_{w_2}m_2, \dots, {}_{w_k}m_k)$ для $\forall {}_{w_i}m_i \in M$;
- ${}_{w_l}l = {}_{w_j}m_j$, якщо $l = {}_{w_0}\otimes (\varepsilon, \dots, \varepsilon, {}_{w_j}m_j, \varepsilon, \dots, \varepsilon)$;
- ${}_{w_l}f = {}_{w_0}\otimes ({}_{w_1}f_1, {}_{w_2}f_2, \dots, {}_{w_k}f_k)$, якщо ${}_{w_1}f_1, {}_{w_2}f_2, \dots, {}_{w_k}f_k$ – форми.

Правила підстановки мають вигляд $\psi_r: \langle s_r, g_r \rangle \in \Psi$, де s_r – відношення підстановки, g_r – набір операцій над атрибутами. Відношення підстановки – двомісне відношення з атрибутами ${}_{w_i}l_i \rightarrow {}_{w_j}l_j$. . Для форми ${}_{w_l}f_l = {}_{w_0}\bigoplus ({}_{w_1}f_1, {}_{w_2}f_2, \dots, {}_{w_h}f_h, \dots, {}_{w_k}f_k)$ і доступного відношення підстановки${}_{w_h}l_h \rightarrow {}_{w_q}l_q$ такого ${}_{w_h}l_h \prec {}_{w_l}f_l$, що (${}_{w_h}l_h$ є частиною ${}_{w_l}f_l$), результатом тримісної операції підстановки $W_p \Rightarrow ({}_{w_h}l_h, {}_{w_q}l_q, {}_{w_l}f_l)$ буде форма ${}_{w_l}f = {}_{w_0}\otimes ({}_{w_1}f_1, {}_{w_2}f_2, \dots, {}_{w_k}f_k)$, де $\bigoplus$– будь-яка операція зв'язування з Σ.

Операція часткового виведення ${}_{w_l^*}f^* = {}_{v_p} \mid \Rightarrow (\Psi, {}_{w_l}f)$ полягає у:

– виборі одного з доступних правил підстановки $p_r: \langle s_r, g_r \rangle$ з відношенням підстановки s_r;
– виконанні на його основі операцій підстановки;
– виконання операцій над атрибутами у відповідній послідовності.

Для формування конструкцій необхідно виконувати ряд уточнюючих перетворень узагальненої КПС [14]:

– **спеціалізація** визначає предметну область, включаючи семантичну природу носія, кінцеву множину операцій та їх семантику, атрибутику операцій, порядок їх виконання, а також обмеження на правила підстановки $C_S \mapsto {}_SC$;
– **інтерпретація** передбачає зв'язування операцій сигнатури з алгоритмами виконання певної алгоритмічної структури. У процесі інтерпретації відбувається узгодження інформаційної моделі способу побудови конструкцій із моделлю виконавця ${}_SC, C_{A_I} \mapsto < {}_{S,I}C, C_A >$;
– **конкретизація** КПС полягає у розширенні аксіоматики множиною правил продукцій, визначенні конкретних множин нетермінальних і термінальних символів разом із їх атрибутами, а за потреби – й значеннями цих атрибутів ${}_{S,I}C_K \mapsto {}_{S,I,K}C$.
– **реалізація** КПС полягає у формуванні конструкції з елементів носія КПС через виконання алгоритмів, які відповідають операціям сигнатури. Вона можлива лише для КПС, яка була попередньо спеціалізована, інтерпретована та конкретизована ${}_{S,I,K}C_R \mapsto \Omega$.

2.1.2 Спеціалізація конструктора

Спеціалізація конструктора – формування геометричних фігур (конструкцій) у просторі R^3, у тому числі просторових графових фракталів:

$$C = \langle M,\ \Sigma,\ \Lambda \rangle_S \mapsto C_G = \langle M_G, \Sigma_G, \Lambda_G \rangle,$$

де $\Lambda_G = \Lambda \cup \Lambda_1 \cup \Lambda_2$, $\Lambda_1 = \{M_G \supset \boldsymbol{R}^3 \cup T_G \cup F_G \cup K_G$, $\Sigma_G = \{\Xi_G, \Theta_G, \Phi_G, \{\rightarrow, \hookleftarrow\}\}$, $\Phi_G = \{\gg, \circ, +, -, *, /, \star\}$, $\Theta_G = \{\Rightarrow, | \Rightarrow, || \Rightarrow\}\}$. Тут множина: T_G – умовно неподільних графічних елементів конструювання, F_G, K_G – проміжних форм та

конструкцій відповідно, Ξ_G – відносини взаємного розташування геометричних фігур (зв'язування елементів), які задаються графічно; операції: Θ_G – підстановки та виведення, Φ_G – операції над атрибутами, а також відношення підстановки (→) и атрибутивності (↲).

Λ_2 містить наступні визначення, доповнення та обмеження, які уточнюють алфавіт, атрибути носія, відносини підстановки, задають особливості виконання операцій підстановки та виведення.

Фігурою або об'єктом будемо називати зв'язану компактну множну в просторі R^3.

Наявність атрибута w у елемента носія m будем визначати як ${}_w m$ (ідентифікатор m з атрибутом w), а те, що w є атрибутом ідентифікатора m – $w↲m$.

Термінальний алфавіт T_G складається з множини геометричних об'єктів з атрибутами.

У конструкторі, що буде розглядатися далі, передбачені наступні операції зв'язування і над атрибутами:

- дублювання множини $f_i \gg f_j$, множина f_i копируется в f_j (відповідне відношення – може бути скопійовано);
- копіювання сентенційної форми f_k в $\bar{f}_k$, з послідовною заміною одних фігур на інші, у порядку віддалення від останньої $\circ(\bar{f}_k, f_k)$;
- додаванню дійсних чисел $+(c, a, b)$, реалізована як $c := a + b$;
- відніманню дійсних чисел $-(c, a, b)$, реалізована як $c := a - b$;
- множення дійсних чисел $*(a, b)$, реалізована як $a := a * b$;
- ділення дійсних чисел $/(a, b)$, реалізована як $a := a/b$;
- $\star$ заміна кольору фігури на інший (перефарбування) $\star(a, b)$, реалізована як $a := a \star b$.

Умовою закінчення висновку є виконання N - ї операції часткового виведення (гіпотетично N може дорівнювати ∞).

2.1.3 Інтерпретація конструктора

Для інтерпретації операцій конструювання необхідно встановити базову алгоритмічну структуру [15] (БАС).

Нехай є наступна БАС:

$$C_A = \langle M_A,\ V_A,\ \Sigma_A,\ \Lambda_A \rangle,$$

де M_A – неоднорідний носій, Σ_A – сигнатура та Λ_A – аксіоматика, $V_{\text{A}} \subset M_A$ – множина базових алгоритмів внутрішнього виконавця конструювання, $V_{\text{A}} \supset \{A_1|^{f_j}_{l_h,l_q,f_i}, A_2|^{f_j}_{f_i,\Psi}, A_3|^{f_j}_{f_i,\Psi}, A_4|^{a}_{a,b} A_5|^{a}_{a,b}, A_6|^{c}_{a,b}, A_7|^{c}_{a,b}, A_8|^{a}_{a,b} A_9|^{\bar{f}_i}_{f_i}, A_{10}|^{f_i,f_j}_{f_i}\}$.

Зазначені вище алгоритми реалізують наступні операції:

– $A_1|^{f_j}_{l_h,l_q,f_i}$ – підстановки;

– $A_2|^{f_j}_{f_i,\Psi}, A_3|^{\bar{\Omega}}_{\sigma,\Psi}$ – часткового та повного виводу, f_i, f_j – форми, σ – початковий термінал, $\bar{\Omega}$ – множина сформованих конструкцій;

– $A_4|^{a}_{a,b} A_5|^{a}_{a,b}, A_6|^{c}_{a,b}, A_7|^{c}_{a,b}$ – множення, ділення, додавання і віднімання дійсних чисел;

– $A_8|^{a}_{a,b}$ – перефарбування;

– $A_9|^{\bar{f}_i}_{f_i}$ – реалізація операції $\circ(\bar{f}_i, f_i)$

– $A_{10}|^{f_i,f_j}_{f_i}$ – реалізація операції $f_i \gg f_j$

Інтерпретація операцій конструювання (формування конструктивної системи):

$$\langle C_G = \langle M_G, \Sigma_G, \Lambda_G \rangle, C_A = \langle M_A,\ V_A,\ \Sigma_A,\ \Lambda_A \rangle \rangle_I \mapsto C_{GI} = \langle M_{GI}, \Sigma_{GI}, \Lambda_{GI} \rangle,$$

де $I \mapsto$ – операція інтерпретації; $\Lambda_{GI} = \Lambda_G \cup \Lambda_A \cup \Lambda_3$, $\Lambda_3 = \{(A_1|^{f_j}_{l_h,l_q,f_i} \hookleftarrow \Rightarrow),$

$(A_2|^{f_j}_{f_i,\Psi} \hookleftarrow | \Rightarrow), A_3|^{\bar{\Omega}}_{\sigma,\Psi} \hookleftarrow || \Rightarrow), (A_4|^{a}_{a,b} \hookleftarrow *), (A_5|^{a}_{a,b} \hookleftarrow /), (A_6|^{c}_{a,b} \hookleftarrow +), (A_7|^{c}_{a,b} \hookleftarrow -),$

$(A_8|^{a}_{a,b} \hookleftarrow *), \left(A_9|^{\bar{f}_i}_{f_i} \hookleftarrow \star\right), (A_{10}|^{f_i,f_j}_{f_i} \hookleftarrow \gg)\}$

8.1.1. Конкретизація конструктора

Виконаємо конкретизацію конструктора C_{MSGF}, назва конструктора містить перші літери англійської назви «мультиатрибутивні просторові графові фрактали»:

$$C_{GI} = \langle M_{GI}, \Sigma_{GI}, \Lambda_{GI} \rangle_K \mapsto C_{MSGF} = \langle M_{GI}, \Sigma_{GI}, \Lambda_{MSGF} \rangle,$$

де $\Lambda_{MSGF} = \Lambda_{GI} \cup \Lambda_4$, частина Λ_4 визначає:

- фізичні атрибути жорсткості, такі як: модуль пружності E Т/м^2, питома вага матеріалу R_o Т/м^3, коефіцієнт Пуассона v; тип навантаження на вузли, значення навантаження на вузли;
- атрибути закріплення, які обмежують рух вузлів при аналізі в заданих траєкторіях: X, Y, Z, UX, UY, UZ, W;
- вузол (термінал) – n, з атрибутами розміру $size = 1$, та координатами $pos = (x, y, z)$, атрибутом кольору $col = yellow$, атрибутами зв'язків по осям, атрибутами навантаження, атрибутами закріплення;
- ребро (термінал) – e, з атрибутами початку $\mathrm{start} = (x, y, z)$, та кінця $\mathrm{e}nd = (x, y, z)$ та фізичними атрибутами жорсткості;
- граф (нетермінал) – graph, складається з вузлів n та ребер e;
- кристалічна ґратка (нетермінал) – lattice, складається з графу та має власні атрибути: довжини ребер у метрах a, b, c, кути між ребрами α, β, γ;
- початкові умови – граф, що має один вузол n_0, з атрибутами ${}_{size}n_0 = 1$, ${}_{pos}v_0 = (0, 0, 0)$, ${}_{col}v_0 = yellow$, інші фізичні атрибути не застосовано;
- правило підстановки $\psi =< s, g >$, в якому операція складається з єдиного відношення підстановки, яке представлене Рисунок 2.1 та операцій над атрибутами $g =< \star\,(left, right) >$, права частина підстановки повинна містити хоча б один вузол з кольором як у лівому правилі;
- фізичні атрибути задаються та редагуються окремими операціями.

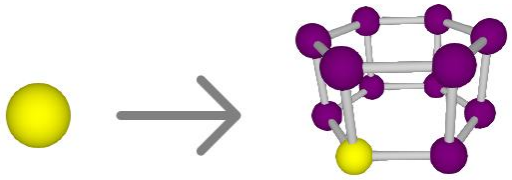


Рисунок 2.1 – Відношення підстановки конструктора C_{MSGF}

2.1.4 Реалізація конструктора

Реалізацію представимо у вигляді послідовності сентенційних форм f_1, f_2, f_3, f_4 ... Рисунок 2.2 (зображення збільшено по відношенню до s для більшої розбірливості).

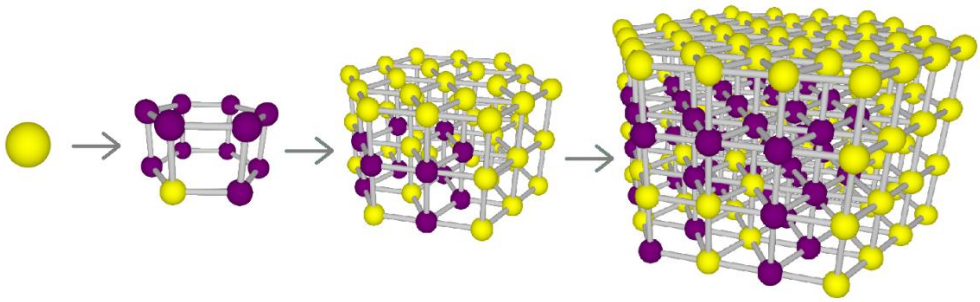


Рисунок 2.2 – Послідовність конструювання графового фракталу

Реалізація виконується в такий спосіб.

– Операція часткового висновку здійснює перетворення форми f_i в f_{i+1};
– початкову форму f_0 (graph), згідно з підстановкою (див. рис), одноразово застосовуючи операцію підстановки, перетворює в f_1:
– виконуємо операцію над атрибутам кольору: взаємне перефарбування вузлів з кольором left на right і навпаки.

Форма f_3 Рисунок 2.3 отримана операцією часткового виведення з триразовим застосуванням операції підстановки з подальшою операцією над атрибутами кольорів

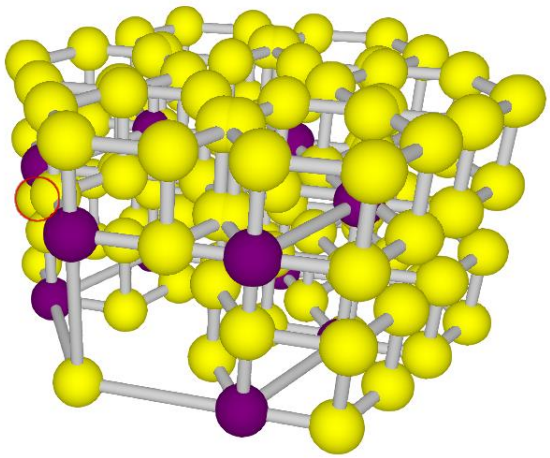


Рисунок 2.3 – Графовий фрактал із застосуванням операції над атрибуами розмірності під час формування структури

Змінюючи правило підстановки в процесі побудови суперкомірки можна отримати різні бажані результати

До модифікації правила підстановки належать не лише типи кристалічних ґраток, а й їх параметри. Як приклад, візьмемо гексагональну ґратку та змінимо на кожному кроці довжину ребер (див рис)

2.2 Використання ЛІРА-САПР для аналізу згенерованих конструкцій

За допомогою ЛІРА-САПР відбувається аналіз переданих згенерованих конструкцій, що включає визначення напружено-деформованого стану Рисунок 2.4, розрахунок внутрішніх зусиль у стержнях і перевірку їх відповідності нормативним вимогам. Програма дозволяє моделювати різні умови навантаження, враховувати геометричну нелінійність, змінювати характеристики матеріалів і виконувати детальний аналіз поведінки конструкцій під впливом зовнішніх чинників. Отримані результати використовуються для оцінки ефективності кожного варіанту конструкції, порівняння їх характеристик та вибору оптимального рішення для подальшого впровадження в проектування.

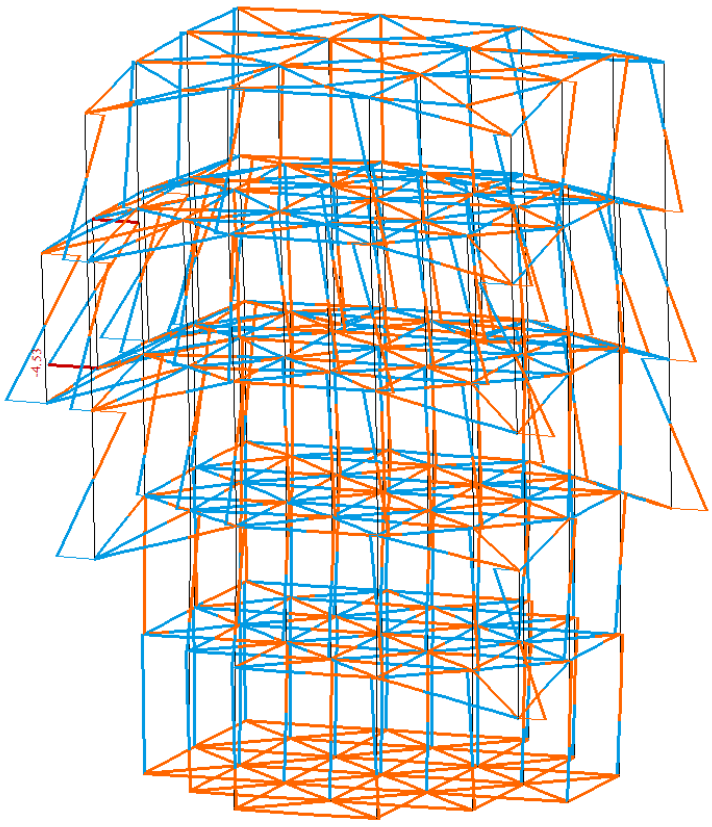


Рисунок 2.4 Вигляд проаналізованого згенерованого графа з фрактальними властивостями кристалічної ґратки; показано аналіз епірюв по осі Z

2.3 Методи розрахунків експериментальної обчислюваної складності

Експериментальна обчислювальна складність [16] — це галузь дослідження в теорії алгоритмів і обчислювальної математики, яка фокусується на емпіричному вивченні складності алгоритмів та програм шляхом проведення експериментів. На відміну від теоретичного аналізу складності, що базується на математичних оцінках часу чи простору в найгіршому, середньому або найкращому випадках, експериментальна обчислювальна складність вивчає фактичну поведінку алгоритмів на практиці.

Основні операції, які задіяні для підрахунку показників обчислювальної складності формування мультиатрибутивних просторових графових фракталів:

– арифметичні;
– присвоєння;
– порівнювання;
– переходу.

Для цього було переглянута дизасимбльована версія коду на мові C# Рисунок 2.5, та написана версія алгоритму з додатковими лічильниками операцій.

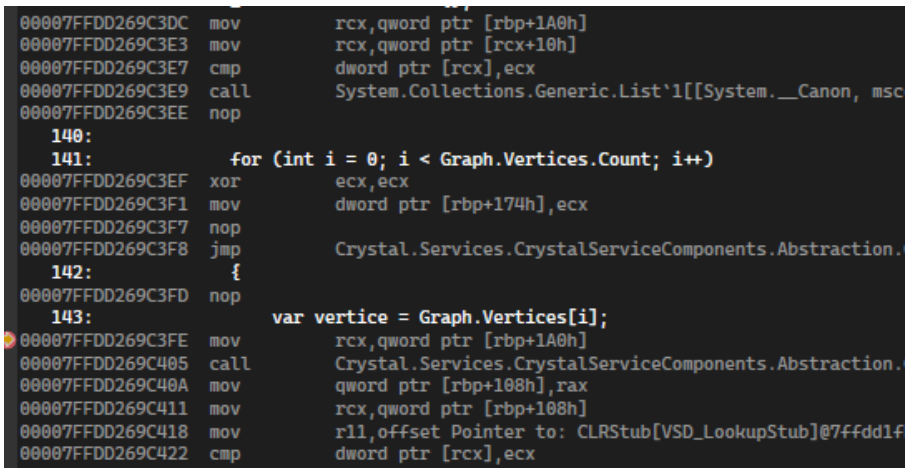


Рисунок 2.5 – Приклад дизасимбльованого коду на мові C# в режимі відладки в середовищі Visual Studio

Порівнюватися будуть просторові графові фрактали, які побудовані за допомогою різних типів кристалічних ґраток. Для генерації кожного графового фракталу буде виконуватись десять ітерацій росту фрактала. На основі

отриманих даних будуть складені графіки залежності кількості операцій під час виконання кожної ітерації. Також для отриманих графіків виконається аналіз для визначення типу аналітичної залежності, та виду формули із параметрами. Для визначення параметрів формули для аналітичної залежності використовується метод найменших квадратів, попередньо провівши заміну змінних за потреби:

$$b = \frac{n*\sum_{i=1}^{n} x_i * y_i - \sum_{i=1}^{n} x_i * \sum_{i=1}^{n} y_i}{n*\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2},\ a = \frac{\sum_{i=1}^{n} y_i - b*\sum_{i=1}^{n} x_i}{n}$$

Висновки до другого розділу

У розділі обґрунтовано доцільність досліджень, описано підхід конструктивно-продукційного моделювання, що базується на формальних граматиках. Описано та визначено усі четвірки конструктивно-продукційної структури для моделювання мультиатрибутивного просторового графового фракталу, а саме: спеціалізація, інтерпретація, конкретизація та реалізація. Під час конкретизації описано інформаційну частину, яка необхідна для моделювання графових структур з фрактальними властивостями кристалічної ґратки, а також описано усі атрибути які умовно можна поділити на геометричні, фізичні та суто віртуальні, такі як колір, що допомагають при їх генерації.

Дано невеликий опис можливостей ЛІРА-САПР, яка може використовуватись для аналізу напружено-деформованого стану згенерованих графових структур.

Описано як і яким чином розраховується експериментальна обчислювальна складність графових фракталів.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНСТРУМЕНТАЛЬНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Формалізація задачі

Формалізація задачі може бути представлена у вигляді діаграми прецедентів Рисунок 3.1. Дана діаграма містить усі основні та відмінні варіанти використання програми, що можуть включати інші, більш деталізовані варіанти використання, або навпаки розширювати їх, тобто надавати додаткову функціональність. Актором є сутність яка взаємодіє з програмою, яка представлена людиною, що має базові навички взаємодії з ПК.

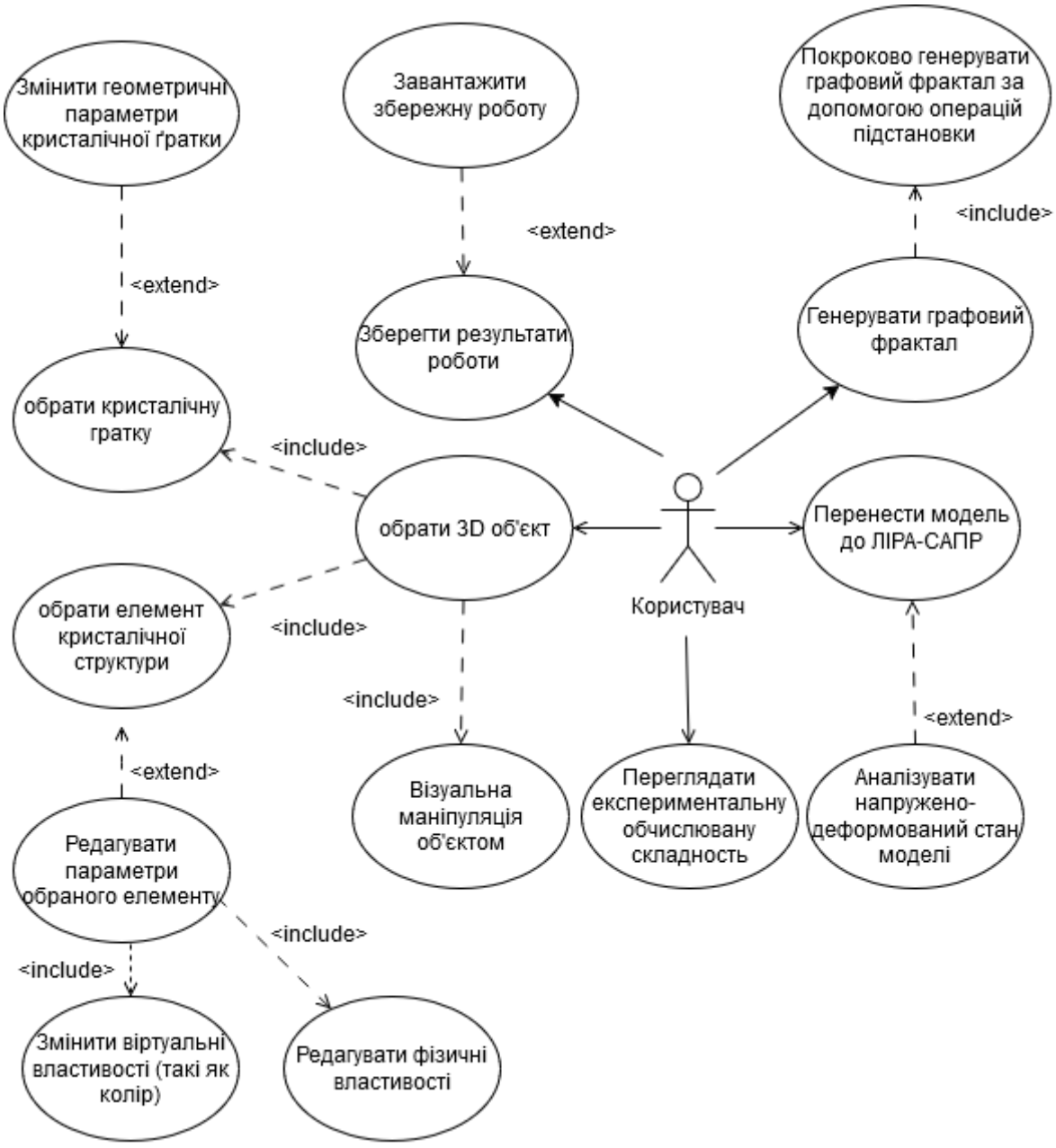


Рисунок 3.1 – Діаграма прецедентів

З діаграми прецедентів може здатися що деякі варіанти використання мають йти напряму від користувача, такі як редагування та зміна параметрів об’єктів, але це не так. Не можна змінити параметри об’єкту, попередньо не обравши його, і в той же час це не є включення, бо об’єкт можна просто обрати, щоб переглянути його, не змінюючи параметрів, тому варіант розширення є коректним з точки зору діаграми прецедентів.

На діаграмі представлені усі ключові та відмінні варіанти використання. Деякі варіанти використання було виключено, через можливість завданню навмисного або ненавмисного збою програмі. Мова йде про зберігання файлу в формат XML або JSON, зокрема зміну числових параметрів на текстові.

3.2 Базова архітектура системи

3.2.1 Загальні складові програми

Архітектура системи поділена на чотири окремі частини: представницьку, сервісів, абстрактних сутностей, та окремий додаток для перегляду експериментальної обчислюваної складності. Загалом ці частини можна представити за допомогою загальної діаграми компонентів Рисунок 3.2.

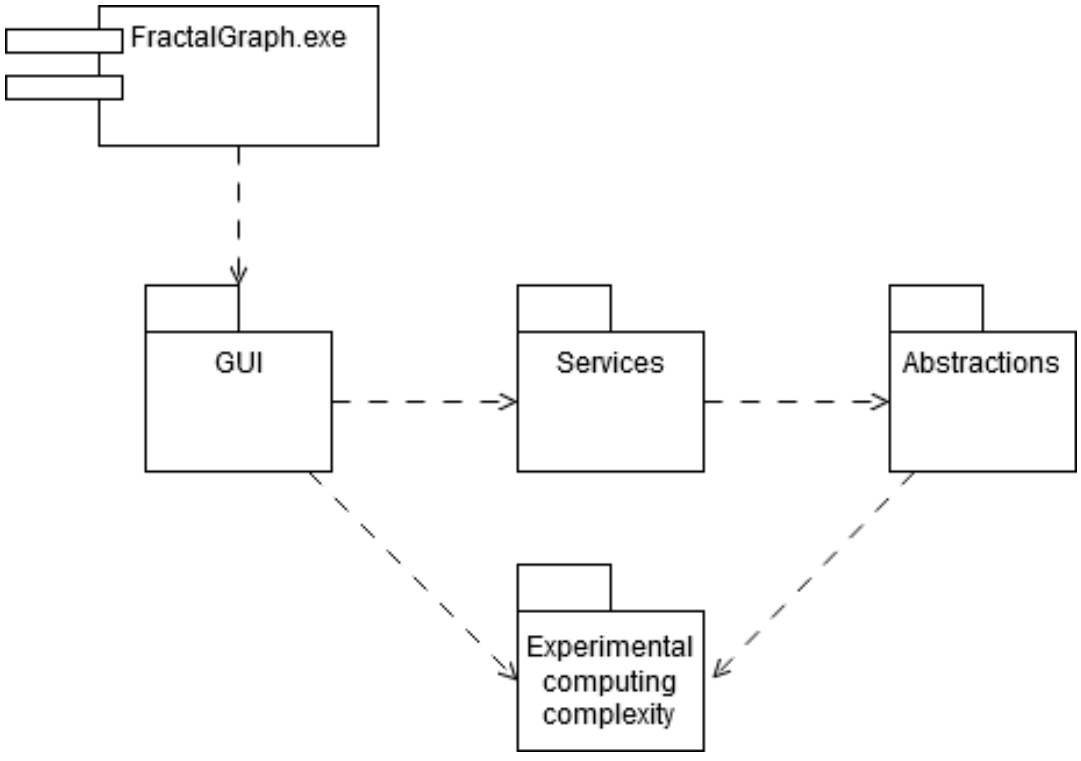


Рисунок 3.2 – Загальна діаграма компонентів з небажаним зв’язком

В ідеальному світі зв’язок між «Abstractions» та «Experimental computing complexity» не мав би бути таким, а замість нього повинен бути зв’язок від «Services» до «Experimental computing complexity». Проблема експериментальної обчислювальної складності полягає в потребі до надто близького зв’язку до абстрактних сутностей, щоб врахувати всі атомарні операції, які ними виконуються, тому виникає небажана ситуація, коли абстракція використовує сервіс, а не навпаки.

3.2.2 Використані принципи проектування

Програма написана в об’єктно-орієнтованому стилі з дотриманням наступних принципів проектування, відомих як SOLID [17]:

- відкритості/закритості – програмні сутності відкриті для розширення, але закриті для змін;
- єдиного обов’язку – кожен клас має відповідати лише за одну задачу;
- підстановки Лісков – об'єкти підкласів можуть замінятися об'єктами батьківського класу без порушення працездатності програми;
- розділення інтерфейсів – інтерфейси повинні бути вузькоспеціалізованими, щоб клієнти використовували лише необхідний їм функціонал;
- інверсії залежностей [18] Рисунок 3.3 – високорівневі модулі не залежать від низькорівневих, обидва повинні залежати від абстракцій.

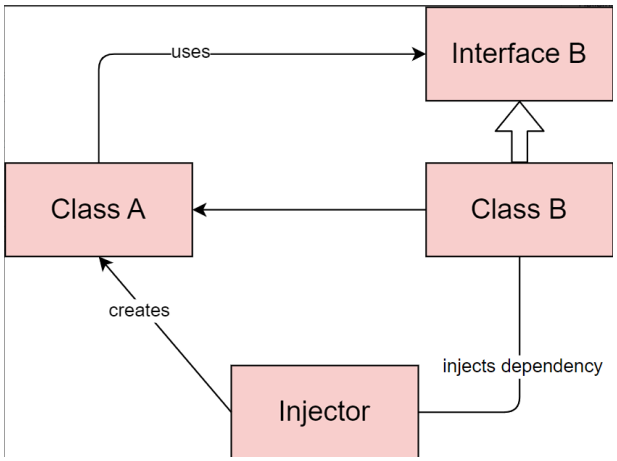


Рисунок 3.3 – Приклад «ін’єкції залежності», що є складовою принципа інверсії контролю

Інверсія залежностей проглядається за допомогою ін’єкції залежностей, де в якості Injector зазвичай виступає Service Provider, який надає необхідні сервіси рис.

3.2.3 Гнучкість архітектурного патерну MVVM

У програмі реалізовано архітектурний патерн MVVM (Model-View-ViewModel) [19] Рисунок 3.4, який забезпечує чітке розділення між логікою програми та її зовнішнім виглядом. Основна ідея цього підходу полягає у поділі відповідальностей: **Model** відповідає за управління даними, **View** – за відображення інтерфейсу користувача, а **ViewModel** виступає посередником, що пов’язує логіку програми з інтерфейсом рис.

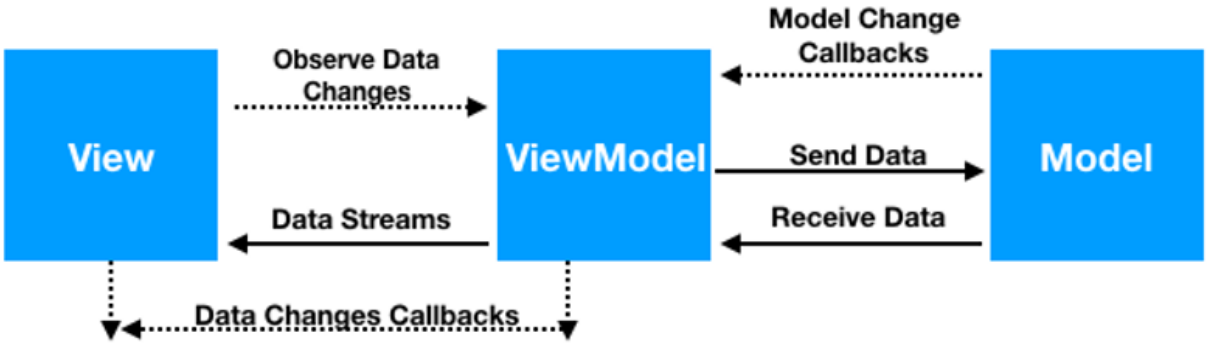


Рисунок 3.4 – Принципи взємодії скадових патерну MVVM

Завдяки такій структурі, патерн сприяє гнучкості й повторному використанню коду. Додавання шару **ViewModel** дозволяє модифікувати поведінку інтерфейсу без необхідності змінювати його безпосередню реалізацію або структуру даних. Це значно полегшує тестування, обслуговування та адаптацію програми під різні вимоги.

Однією з ключових особливостей MVVM є **біндинг** [20] – механізм, який автоматично синхронізує дані між інтерфейсом користувача (View) і логікою програми (ViewModel). Біндинг дозволяє досягти двосторонньої взаємодії, завдяки чому зміни, внесені користувачем, автоматично відображаються у моделі, а зміни в даних негайно відображаються у користувацькому інтерфейсі. За потреби можна контролювати як елементи варто оновлювати, а які ні.

3.3 Внутрішнє проектування

3.3.1 Вибір мови програмування

Для написання програми обрана мова програмування C#, розроблена компанією Microsoft як аналог мові Java. Вибір мови зумовлений наявністю готових рішень, бібліотек, технологій Рисунок 3.5, що використовують мову C#. Технології, на яких вона працює, тобто фреймворк .NET Framework, має збірник сміття (англ. – garbage collector), який дозволяє швидше писати програму, не відволікаючись про контроль пам’яті, хоча від цього не слід зовсім не перейматися яким чином виділяється пам’ять, бо можуть виникнути вузькі місця, які дуже навантажують програму.

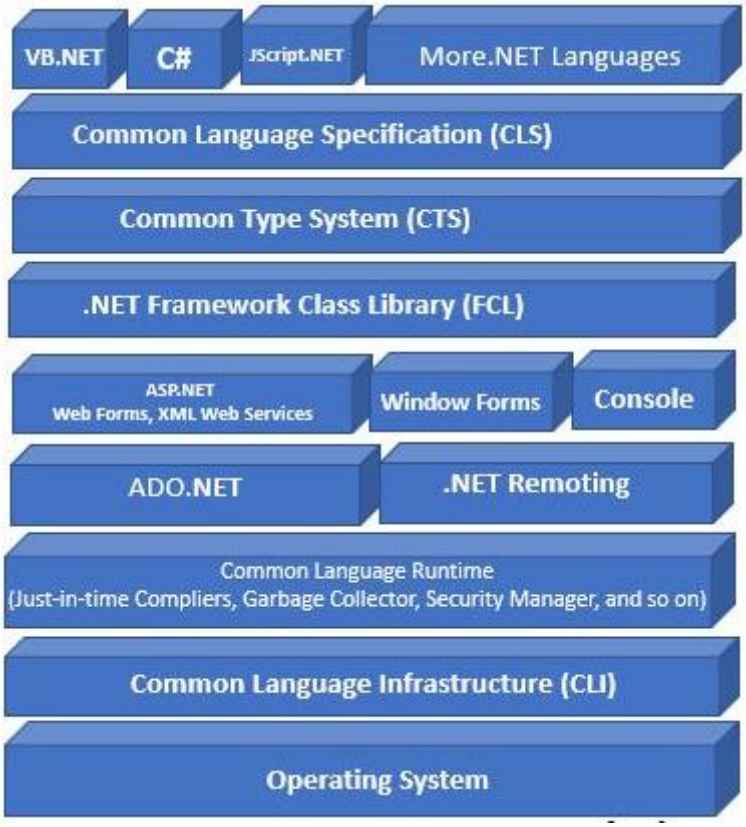


Рисунок 3.5 –Зв'язок між мовою C# та технологіями, що стоять за нею

3.3.2 Використані технологій

Перш за все головною технологію, рушій на якому все відбувається, це є .NET Framework 4.8.1, саме версія Framework 4.8.1, яка хоча трохи застаріла, все одно залишається кращим варіантом для написання десктопних додатків під ОС Windows.

В якості відображення графічного інтерфейсу використовується Windows Presentation Framework (WPF) [21], який підтримує усі можливості патерну MVVM. За допомогою неї можна створювати інтерфейс у декларативному стилі на мові XAML, що дозволяє чітко розмежувати дизайн та логіку роботи програми. Такий підхід забезпечує легкість у підтримці та оновленні додатку, оскільки візуальна частина та програмна логіка взаємодіють через механізми прив’язки даних та подій.

Однією з ключових переваг WPF є потужна підтримка стилів і шаблонів, що дозволяє створювати сучасні, адаптивні та естетично привабливі інтерфейси користувача. Крім того, ця технологія забезпечує можливість роботи з векторною графікою, анімацією та мультимедіа, що дає змогу створювати додатки з високим рівнем інтерактивності та динамічності.

Для використання 3D графіки використовується Helix Toolkit [22] Рисунок 3.6. Helix Toolkit надає зручні та прості засоби з побудови 3D примітивів, що не вимагають глибоких знань з 3D графіки та легко використовуються разом з WPF

Рисунок 3.6 Графічні можливості Helix Toolkit

Іншим варіантом застосування 3D є або написання власного рушію за допомогою DirectX [23] або Vulkan [24], проте це вимагає більшого часу, а різниця чимало виправдовує очікувань.

Слід зауважити, що існує ще один відомий варіант – 3D рушії, які підтримують мову C#, такі як Unity [25], Godot [26], Stride [27]. Написання користувацького інтерфейсу вимагає більше часу і більшості випадків не таке гнучке як в WPF, хоча їх можна інтегрувати в WPF, це може виявитися не простою задачею, а від появи підводних каменів, які можуть трапитися у такий спосіб, ніхто не застрахований.

Для аналізу напружено-деформованого стану використовується Component Object Model взаємодія з додатком ЛІРА-САПР. ЛІРА-САПР — це сучасний програмний комплекс для проведення розрахунків та аналізу будівельних конструкцій і споруд. Він використовується для моделювання, оцінки міцності, стійкості та надійності конструкцій з урахуванням різних навантажень. Проте остання версія API не дозволяє повністю уникнути взаємодію з ЛІРА-САПР, лише автоматизувати більшість операцій. Головною причиною тому є неможливість введення навантажень через COM API. Проте є надія що це лише проблема часу, так як в початку 2024 року вийшло окреме API для отримання результатів, що у свою чергу спростило інтеграцію ЛІРА-САПР з іншими додатками.

3.4 Моделювання сутностей системи

Під час створення архітектури програми було проаналізовано предметну область та створення декількох груп сутностей.

Перша група абстрактні сутності предметної області: вузол, ребро, граф, конструктор, правило заміни, генератор кристалічної ґратки, вбудовані атрибути кристалічної ґратки, фізичні атрибути жорсткості, атрибути закріплення вузлів.

Обов’язки першої групи сутностей:

– вузол – абстрактна атомарна частина просторового графу, що має позицію, розмір, колір, атрибути закріплення;

– ребро – абстрактна атомарна частина просторового графу що поєднує два вузли між собою, має розмір, а також фізичні атрибути жорсткості;

– граф – сукупність вузлів та ребер, що уявляють собою мультиатрибутивний просторовий графовий фрактал;

– конструктор – абстрактний об’єкт що будує просторовий графовий фрактал за допомогою операцій заміни;

– правило заміни – містить інформацію про те як повинен поширюватися граф;

– генератор кристалічної ґратки – генерує просторовий графову структур з фрактальними властивостями кристалічної ґратки;

– вбудовані атрибути кристалічної ґратки – геометричні атрибути ґратки Браве, такі як довжини ребер a, b, c, кути між ними;

– фізичні атрибути жорсткості – атрибути жорсткості притаманні ребру, такі як тип перерізу, його розмірні характеристики, модуль пружності, коефіцієнт Пуассона , питома вага матеріалу;

– атрибути закріплення вузлів – напрямки X, Y, Z, UX, UY, UZ, W, по яким треба заборонити переміщення вузлів;

Атрибути та операції, необхідні для виконання обов'язків кожної сутності-класу першої групи наведемо у наступній таблиці:

Таблиця 3.1 – Сутності, атрибути та методи абстрактні сутностей

Сутність	Атрибути	Методи
вузол (node)	Size – розмір вузла, який буде використовуватись для графічного примітиву на 3D сцені – додатне дійсне число; Color – колір вузла, використовується в алгоритмах та для кольору графічного примітиву – структура, яка містить числову інформацію про відтінки кольорів.	створити вузол; клонувати.

Продовження Таблиці 3.1

	Position – позиція у світових координатах на 3D сцені – структура Point, яка містить числову інформацію координат x, y, z. Restrictions – напрямки X, Y, Z, UX, UY, UZ, W, по яким треба заборонити переміщення;	
ребро (edge)	Size – розмір вузла, який буде використовуватись для графічного примітиву на 3D сцені – додатне дійсне число; Start – вузол, з якої починається ребро, являє собою сутність vertice; End – вузол, в якій закінчується ребро, являє собою сутність vertice; Stiffness – фізичні атрибути жорсткості	створити ребро; клонувати.
жорсткість (stiffness)	E – модуль пружності; v – коефіцієнт Пуассона; Ro – питома вага матеріалу; Розмірні характеристики – залежать від типу перерізу, наприклад брус або двотавр	клонувати; перевести до ЛІРА-САПР формату.
закріплення (restrains)	X, Y, Z, UX, UY, UZ, W – змінні типу бул, якщо правда, то переміщення вузла по напрямку заборонено	клонувати; застосувати всі закріплення.

Продовження Таблиці 3.1

граф (graph)	Nodes – вузли графу – список сутностей типу Node; Edges – ребра графу – список сутностей типу Edge;	додати вузол; додати ребро; клонувати.
правило заміни (rule)	LeftGraph – ліва частина правила, елементи якої замінюються на елементи правої частини правила – сутність Graph; RightGraph – права частина правила, в яку повинно перетворитися ліва частина – сутність Graph; LeftColor – виконує функцію фільтрації елементів, які будуть використовуватись для заміни з лівого правила – сутність Color. RightColor – виконує функцію фільтрації елементів у правому правилі, які будуть замінені – сутність Color.	встановити ліву частину правила; встановити праву частину правила; перемалювати ліву частину; перемалювати праву частину; встановити головний колір лівого правила; встановити головний колір правого правила.
конструктор	Graph – граф, що являє собою суперкомірку, яка поширюється – сутність Graph: CurrentRule – правило, яке використовується для виконання замін та побудови суперкомірки – сутність Rule: AddedNodes – список вузлів, які були додані під час заміни, елементи сутності Node;	виконати заміну; очистити суперкомірку.

Продовження Таблиці 3.1

	AddedEdges – список ребер, які були додані під час заміни, елементи сутності Edge;	
генератор кристалічної ґратки	являє собою сімейство класів, кожна з яких створює свою графову структуру з фрактальними властивостями кристалічної ґратки; містить атрибути кристалічної ґратки, головний лівій та правий кольори, а також атрибути, що позначають можливість редагувати параметри кристалічної ґратки	створити просторовий граф з фрактальними властивостями кристалічної ґратки.

Друга група складається з абстрактних сутностей, які не мають альтернативу у реальному світі, але необхідні для зручної архітектури та потоку управління програми. Серед них є сервіси, контролери, менеджери стану та команд:

- IServiceProvider – контейнер, який надає доступ до інших класів за запитом;
- StateManager – містить спільний стан програми, спрощує доступ до даних з різних її частин;
- CommandManager – дає змогу реагувати на події користувача в різних місцях програми;
- ViewportController – інкапсулює стан 3D сцени та надає лише необхідні методи для роботи з нею;
- RuleController – інкапсулює функціонал з маніпуляцією та модифікуванням правила;
- UIService – надає можливості створення та вибору компонентів користувацького інтерфейсу
- MainController – конфігрує початкові налаштування програми.

Так як для побудови всієї діаграми класів та не вистачить місця, побудуємо діаграму для окремих її частин:

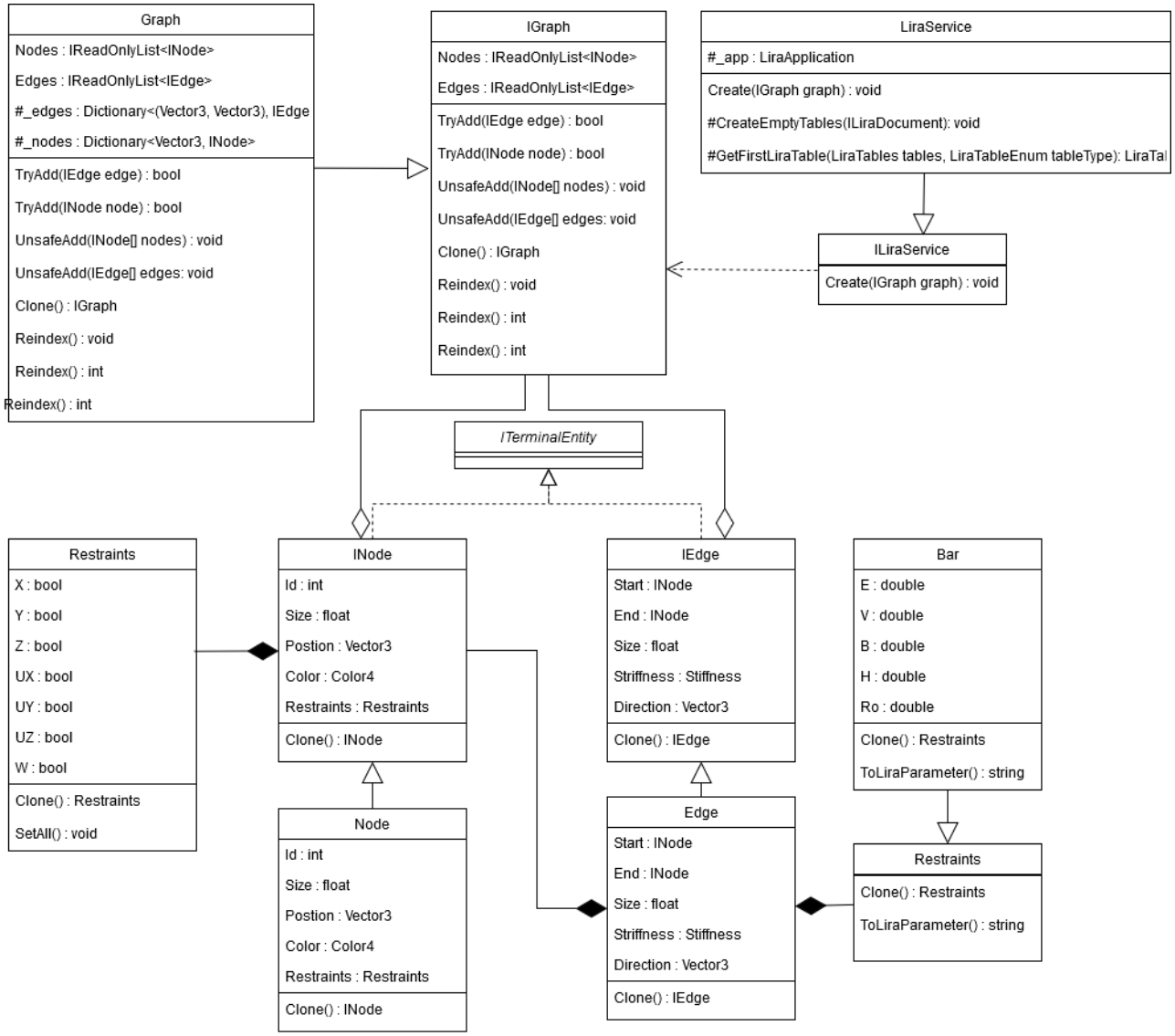


Рисунок 3.7 – Діаграма класів абстрактних сутностей, що становлять логіку програми

Третя група складається з сутностей графічного інтерфейсу та його поділ у вигляді компонентів як пари View, ViewModel як частину патерну MVVM. Компонент інтерфейсу – цілісна частина інтерфейсу, яка займає певну площину користувацького інтерфейсу та об’єднує спільні за призначенням елементи управління. Кожен компонент складається з пари двох класів: View – клас інтерфейсу, ViewModel – прошарок між моделюю даних та інтерфейсом, визначає як саме

дані повинні використовуватись інтерфейсом. Серед таких пар-компонентів можна визначити:

- вікно команд – розташовано вгорі програми, містить усі кнопки інтерфейсу;
- ліве меню - меню атрибутів обраного генератора просторових графів з фрактальними властивостями кристалічної ґратки;
- список генераторів – візуально відображає генератори та тип кристалічної ґратки, який буде використовуватися під час росту фракталу;
- представлення правила – відображає два правила, які будуть виконуватись під час ітерації: заміна візуальних об’єктів та взаємне перефарбування;
- головна сцена програми – відображає 3D сцену.

3.5 Розробка інтерфейсу користувача

При розробці інтерфейсу користувача ставилась задача зробити максимально простий та інтуїтивно зрозумілий інтерфейс:

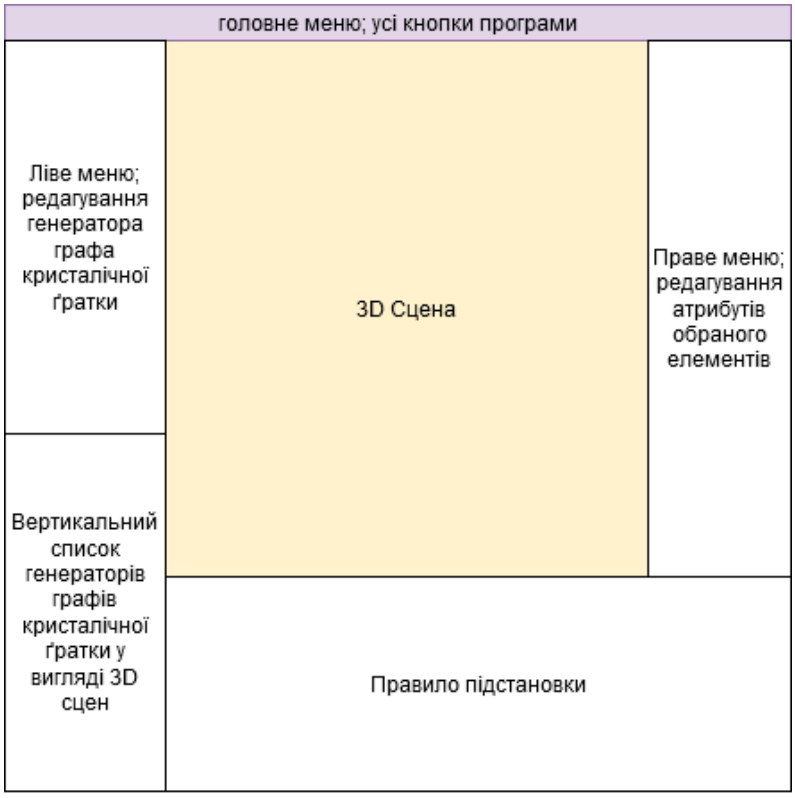


Рисунок 3.8 – Перший макет майбутньої програми

Здійснити такий візуальний вигляд не вдалося через особливість 3D сцени: вона завжди перекриває будь-які елементи не зважаючи не на що:

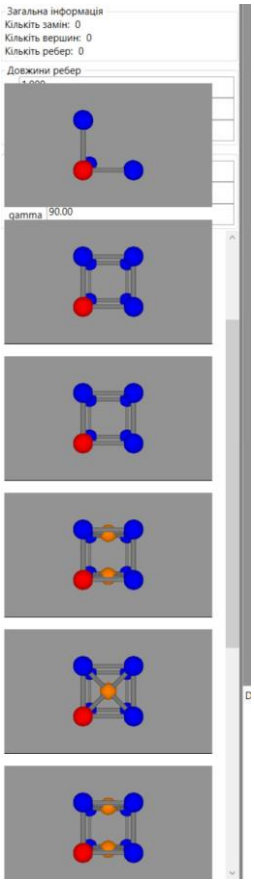


Рисунок 3.9 – Перекриття списком сцен полів для редагування обраного генератора

Тому було прийнято рішення зробити цей список горизонтальним, де він не може перетинатися з іншими елементами:

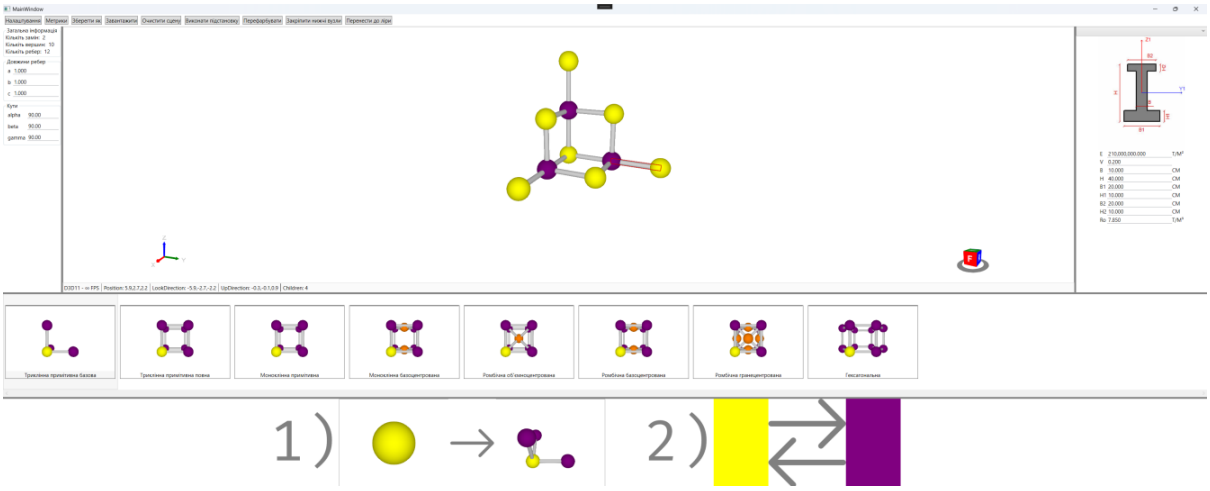


Рисунок 3.10 – Кінцевий варіант інтерфейсу

3.6 Тестування та налагодження чорною скринькою

3.6.1 Тестування чорною скринькою

Тестування програми проводилось методами чорної скриньки [28]. Цей процес мав особливе значення під час написання програми, так як він підтверджує відповідність програми щодо її специфікації.

Метод чорної скриньки дозволяє оцінити роботу програми, ґрунтуючись лише на вхідних даних та очікуваних вихідних результатах. Для його ефективності критично важлива чітка документація: необхідно розуміти, які дані потрібно вводити, які результати очікуються та як сторонні виклики можуть впливати на функції, що тестуються. Хоча цей метод дозволяє перевірити, чи відповідає поведінка програми очікуванням, він не розкриває причин можливих помилок чи несправностей.

За допомогою методів чорної скриньки, а саме методів припущення про помилку, тестувалося API ЛІРА-САПР. Документація ЛІРА-САПР майже нічого не пояснює: не вказує типи параметрів які саме їх необхідно. В мові C# вони позначенні як тип object, що ні про що не говорить. Особливо проблемним було заповнення таблиць жорсткості. В офіційні довідці програми вказується, що можна перевірити як правильно надсилати дані, перевіривши необхідну таблицю у режимі розширеного редагування. По-перше, такі таблиці автоматично не створюються, що вже ускладнює перегляд даних які необхідно, при створенні вони ж пусті. Тобто спочатку треба їх створити, а потім вже робити дії в програмі, щоб побачити, що вони чимось заповнюються. Проте головна та критична відмінність полягає у розбіжності параметрів, що вимагаються:

- дані які показує комірка: «Ro:2.5 E:300000 B:60 H:60 Mu:0.2 STD_END»
- дані які вимагає апі в цій комірці: «Ro:2.5 E:3000000 B:60 H:60 NGrndPar:0 GF:0 nFlags:0 Length:0 B_:0 H_:0 NEL:0 UseRbNel:0 BAR_END Mu:0.2 EF:0 EIy:0 EIz:0 GIk:0 GFz:0 GFy:0 IsRigParamChanges:0 IsSavedAsKoef:0 IsNormSect:1 IsIter:0 Uli:0 STD_END";»

Лише за допомогою режиму відладки та тестуванню чорною скринькою вдалося віднайти правильні параметри, що вимагаються.

3.7 Налагодження

Загальна кількість суттєвих помилок траплялась при взаємодії з API ЛІРА-САПР, деякі з них, якщо не всі, були зовсім не очевидні. Для їх усунення було використано режим відладки в Visual Studio 2022 [29].

Результат налагодження програми представлені в наступній таблиці:

Таблиця 3.2 Протокол налагодження програми

Опис помилки	Опис ситуації	Дії, що були застосовані для усунення
Не можливо створити зразок типу LiraApplication	При створенні об'єкту LiraApplication вибивало помилку з назвою адреси комірки пам'яті, в які це відбулося, що ні про що не говорить	Спробовано створити цей примірник в різних середовищах, та за допомогою ActiveX в JavaScript. Помилка була вирішена завдяки запуску у режимі відладки x64 замість будь-якого іншого.
Таблиця не переносилась в ЛІРА-САПР	Після встановлення заповнення таблиці ЛІРА-САПР не бачила зміни	Після операції SetContent додатково треба викликати ApplyChanges.
UI вузол не може мати більше одного нащадка	При спробі створити окрему 3D сцену для обраних елементів виникала помилка	Замість спроби відобразити один і той же елемент в двох 3D сценах було прийнято рішення надати підсвітку обраному елементу
Невідповідність встановленого параметру	При створені таблиці жорсткості, неправильно вказувалась текстовий параметр.	За допомогою відладки було переглянуто приклад параметру, згенерованого програмою.

Висновки до третього розділу

Виконано проектування системи, створено діаграму прецедентів, де вказуються усі відмінні варіанти використання програми, а також обґрунтовано типи зв’язків що на ній представлено.

Загальна діаграма компонентів демонструє основні модулі програми, які використовуються, а також зауваження щодо їх зв’язків.

Під час проектування програми дотримувалися принципи SOLID, а принцип інверсії залежності розглянуто більш детально, за його допомогою зменшується зв’язність компонентів.

На вибір технологій вплинув архітектурний патерн MVVM, за допомогою якого легко та гнучко писати графічний інтерфейс користувача. Фреймворк WPF підтримує усі особливості патерну MVVM, включаючи прив’язку даних.

Для відображання 3D графіки використовується рушій Helix, в якому легко розібратися та має інтеграцію з WPF.

Розглянуто основні сутності програми та продемонстровано діаграму класів для їх частини.

Для розробки користувацького інтерфейсу було розроблено макет, який не вдалося втілити в життя в певній мірі через особливості використання 3D графіки, через що були внесені певні зміни в розташування користувацьких елементів.

За допомогою методів чорної скриньки та режиму відладки було встановлено та перевірено на відповідність API ЛІРА-САПР, з який було пов’язана більша кількість помилок в програмі.

4 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ПРОСТОРОВИХ ГРАФОВИХ ФРАКТАЛІВ

4.1 Аналіз експериментальної обчислюваної складності з генерування просторових графових фракталів

4.1.1 Порядок випробування

Для отримання показників експериментальної обчислюваної складності з генерування просторових графових фракталів використовуються спеціальні версії алгоритмів з лічильниками. Результати вимірів виводяться в окремому вікні, з якого можна експортувати результати до таблиці, файлу *.xlsx.

Випробування складається з наступних пунктів:

- перед кожним випробуванням програма запускається заново;
- визначаються два основні кольори, які приймають участь у генерації графових структур: червоний та синій;
- кількість кроків для росту фракталу становить десять;
- після генерації фракталів, дані про десять кроків експортуються з іншого вікна програми до файлу *.xlsx у вигляді таблиці;
- на основі отриманих даних будується графік залежності кількості операцій від номеру кроку;
- аналізується отриманий графік, визначається тип аналітичної залежності;
- рахується коефіцієнт кореляції r для вихідного набору даних;
- за потреби вводиться заміна змінних;
- за допомогою методів найменших квадратів розраховуються параметри для формули.

4.1.2 Проведення випробування

Після проведення досліджень отримали наступні результати для різних типів графових структур:

Таблиця 4.1 – Показники графової структури «Триклинна примітивна базова»

№	Arithmetics	Assignments	Comparisons	Jumps	Total
1	81	159	68	34	342
2	243	461	194	98	996

Продовження таблиці 4.1

3	567	1053	440	222	2282
4	1053	1941	809	408	4211
5	1782	3264	1358	684	7088
6	2754	5028	2090	1052	10924
7	4050	7372	3062	1540	16024
8	5670	10302	4277	2150	22399
9	7695	13957	5792	2910	30354
10	10125	18343	7610	3822	39900

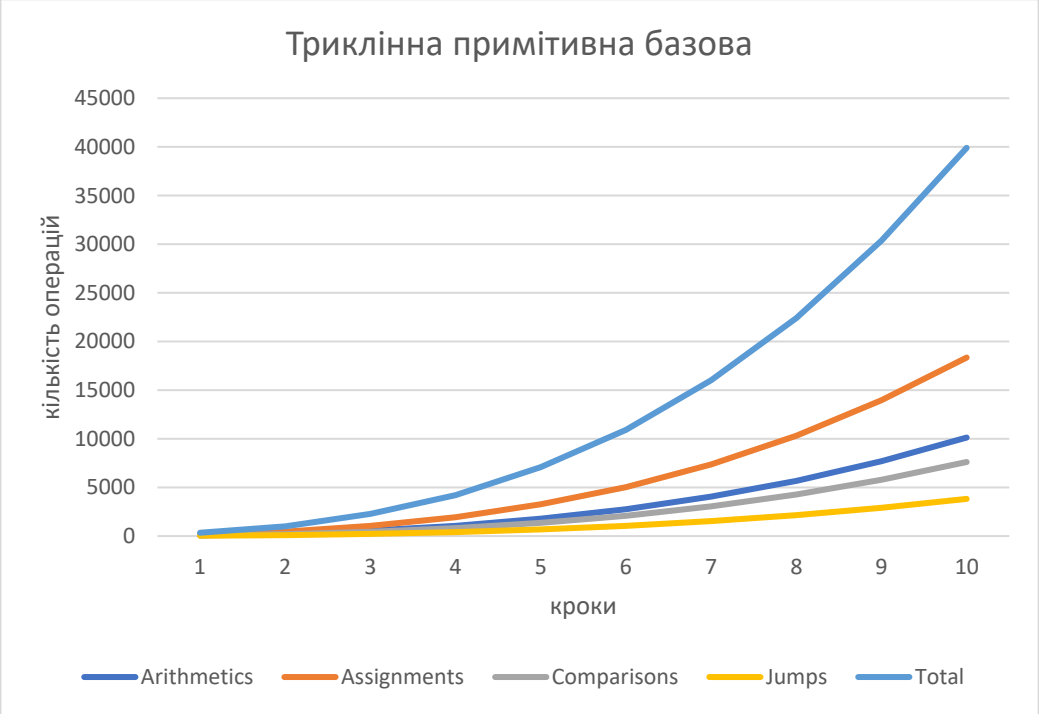


Рисунок 4.1 – Залежність операцій від ітерації при використанні триклінної примітивної базової ґратки

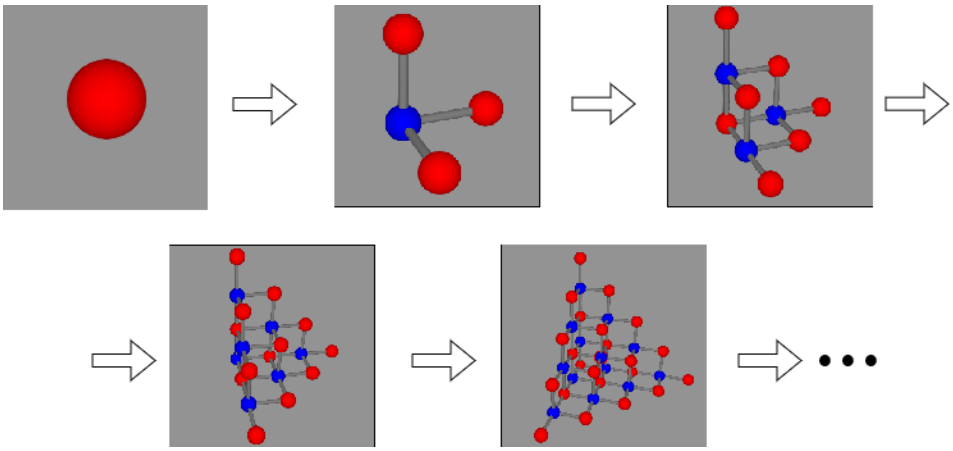


Рисунок 4.2 – Зростання фракталу з використанням структури триклинної примітивної базової кристалічної ґратки

Таблиця 4.2 – Показники графової структури «Триклінна примітивна повна»

№	Arithmetics	Assignments	Comparisons	Jumps	Total
1	317	633	311	164	1425
2	2219	4245	2078	1090	9632
3	6340	12046	5894	3088	27368
4	13948	26404	12917	6762	60031
5	25677	48521	23735	12420	110353
6	42795	80765	39506	20666	183732
7	65936	124338	60818	31808	282900
8	96368	181608	88829	46450	413255
9	134725	253777	124127	64900	577529
10	182275	343213	167870	87762	781120

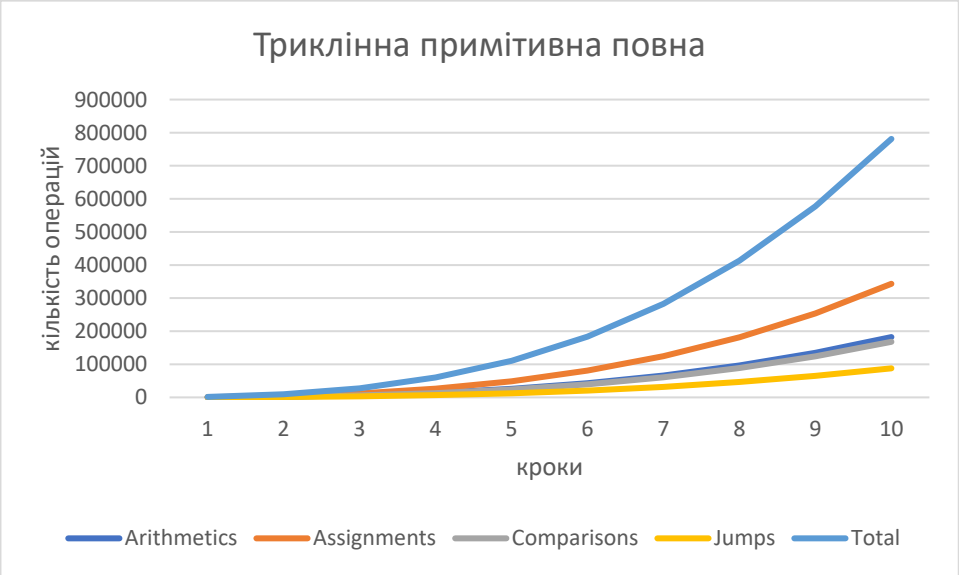


Рисунок 4.3 – Залежність операцій від ітерації при використанні триклінної примітивної повної ґратки

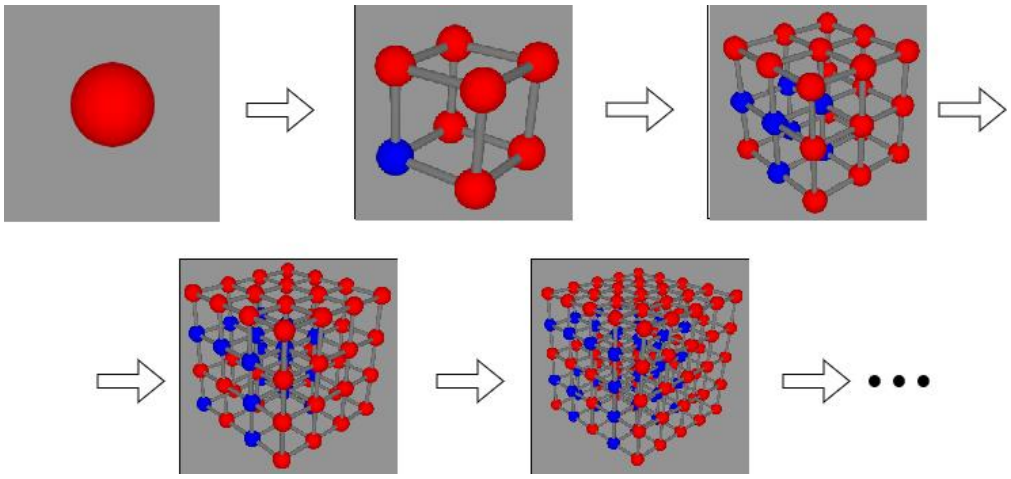


Рисунок 4.4 – Зростання фракталу з використанням структури триклинної примітивної повної кристалічної ґратки

Таблиця 4.3 – Показники графової структури «Моноклінна базоцентрована»

№	Arithmetics	Assignments	Comparisons	Jumps	Total
1	541	1109	571	300	2521
2	3787	7567	3892	2038	17284
3	10820	21546	11082	5800	49248
4	23804	47308	24333	12730	108175
5	43821	87017	44759	23412	199009
6	73035	144935	74552	38990	331512
7	112528	223226	114826	60048	510628
8	164464	326152	167773	87730	746119
9	229925	455877	234507	122620	1042929
10	311075	616663	317220	165862	1410820

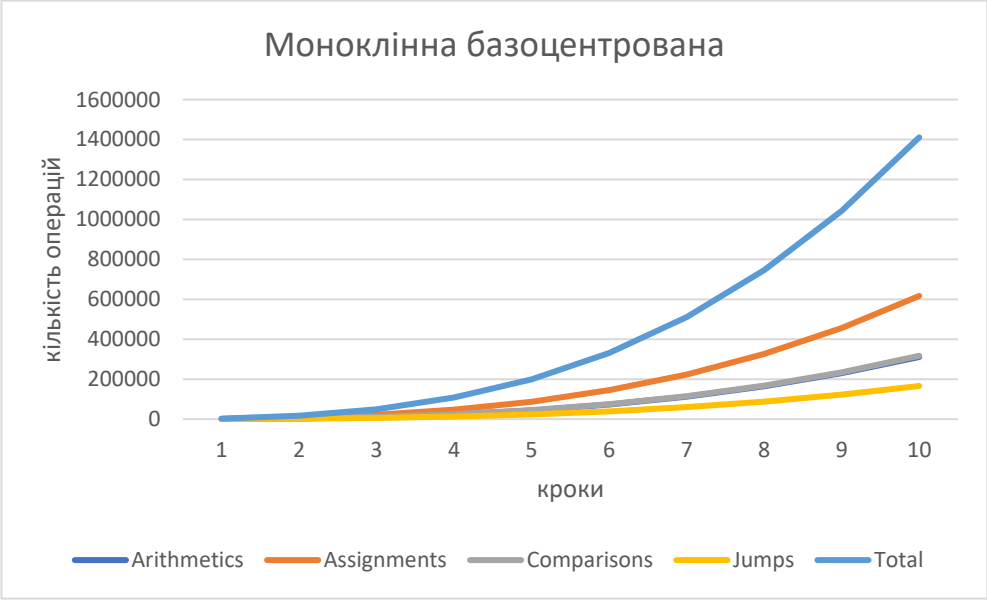


Рисунок 4.5 – Залежність операцій від ітерації при використанні моноклінної базоцентрованої кристалічної ґратки

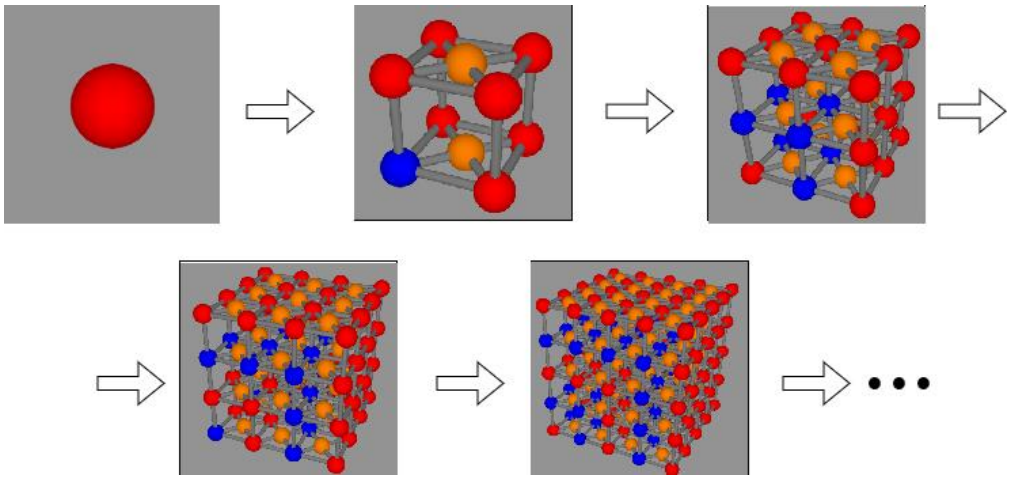


Рисунок 4.6 – Зростання фракталу з використанням структури моноклінної базоцентрованої кристалічної ґратки

Таблиця 4.4 – Показники графової структури «Ромбічна об’ємноцентрована»

№	Arithmetics	Assignments	Comparisons	Jumps	Total
1	529	1083	557	292	2461
2	3703	7400	3803	1988	16894
3	10580	21081	10835	5662	48158
4	23276	46304	23801	12434	105815
5	42849	85191	43793	22876	194709
6	71415	141920	72959	38108	324402
7	110032	218613	112391	58702	499738
8	160816	319448	164237	85778	730279
9	224825	446547	229589	119908	1020869
10	304175	604088	310595	162212	1381070

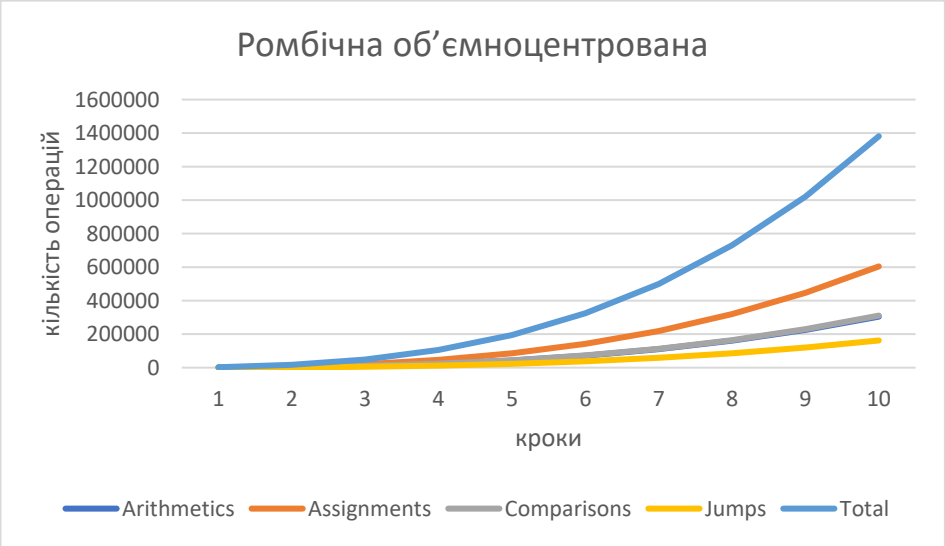


Рисунок 4.7 – Залежність операцій від ітерації при використанні ромбічної об’ємноцентрованої кристалічної ґратки

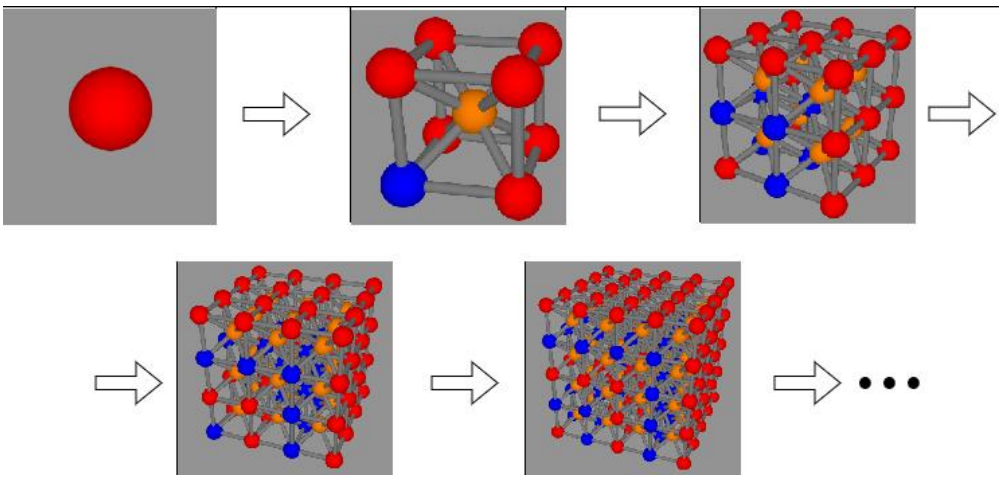


Рисунок 4.8 – Зростання фракталу з використанням структури ромбічної об’ємноцентрованої кристалічної ґратки

Таблиця 4.5 – Показники графової структури «Ромбічна гранецентрована»

№	Arithmetics	Assignments	Comparisons	Jumps	Total
1	1037	2205	1187	620	5049
2	7259	15219	8192	4270	34940
3	20740	43426	23378	12184	99728
4	45628	95452	51389	26778	219247
5	83997	175673	94583	49284	403537
6	139995	292715	157604	82118	672432
7	215696	450954	242810	126512	1035972
8	315248	659016	354845	184882	1513991
9	440725	921277	496067	258460	2116529
10	596275	1246363	671120	349662	2863420

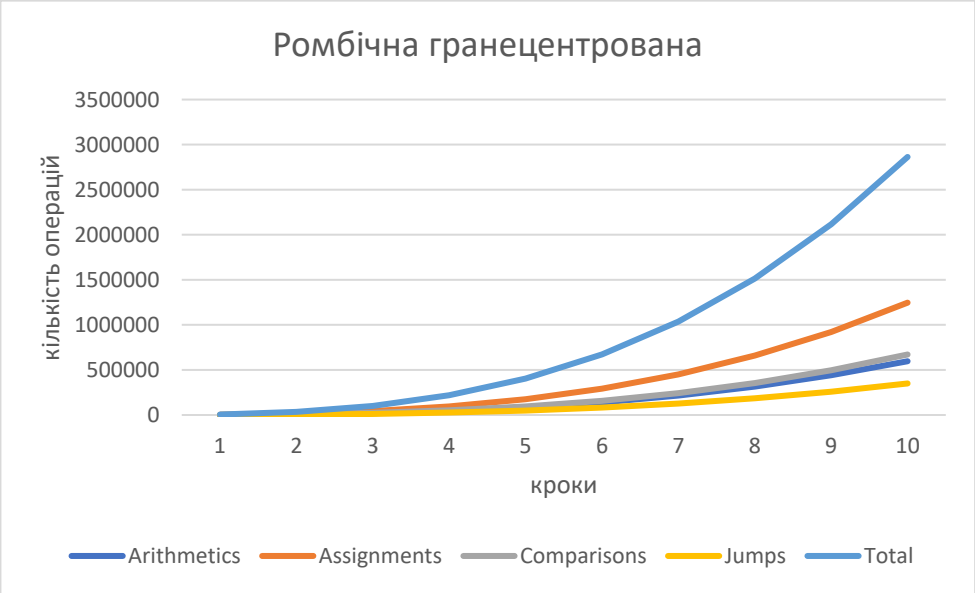


Рисунок 4.9 – Залежність операцій від ітерації при використанні ромбічної гранецентрованої кристалічної ґратки

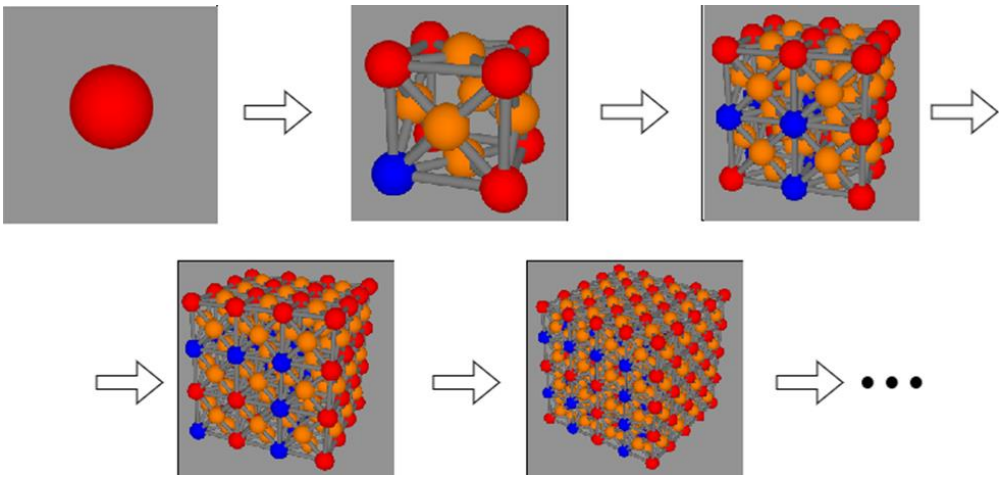


Рисунок 4.10 – Зростання фракталу з використанням структури ромбічної гранецентрованої кристалічної ґратки

Таблиця 4.6 – Показники графової структури «Гексагональна»

№	Arithmetics	Assignments	Comparisons	Jumps	Total
1	547	1155	605	318	2625
2	6017	12235	6410	3348	28010
3	25162	50616	26528	13820	116126
4	55794	112094	58751	30598	257237
5	111041	222721	116741	60774	511277
6	187621	376119	197150	102620	863510
7	298662	598340	313640	163228	1373870
8	440882	883006	462863	240870	2027621
9	627409	1256169	658481	342638	2884697
10	854961	1711451	897146	466804	3930362

Рисунок 4.11 – Залежність операцій від ітерації при використанні гексагональної кристалічної ґратки

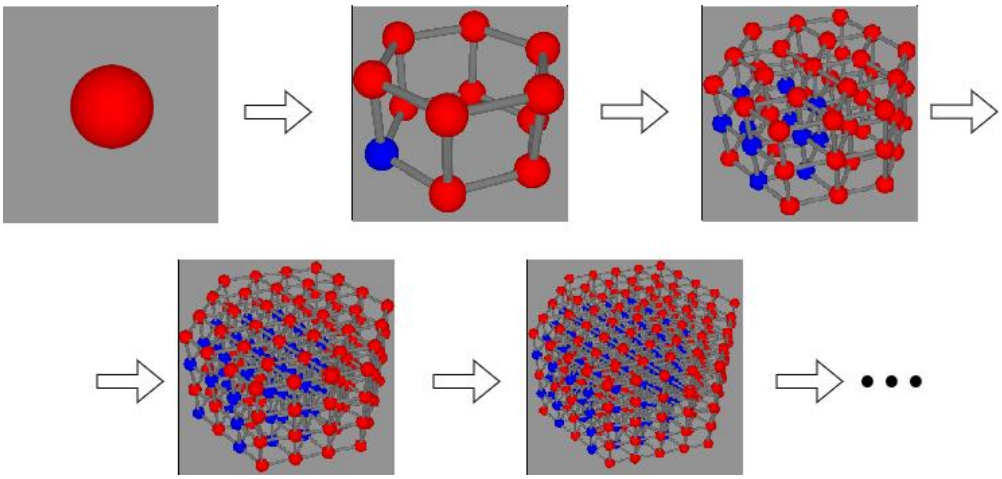


Рисунок 4.12 – Зростання фракталу з використанням структури гексагональної кристалічної ґратки

4.1.3 Аналіз результатів експериментальної обчислюваної складності

Було проведено дослідження на шести графових структурах з різною кількістю ребер та вузлів рис.

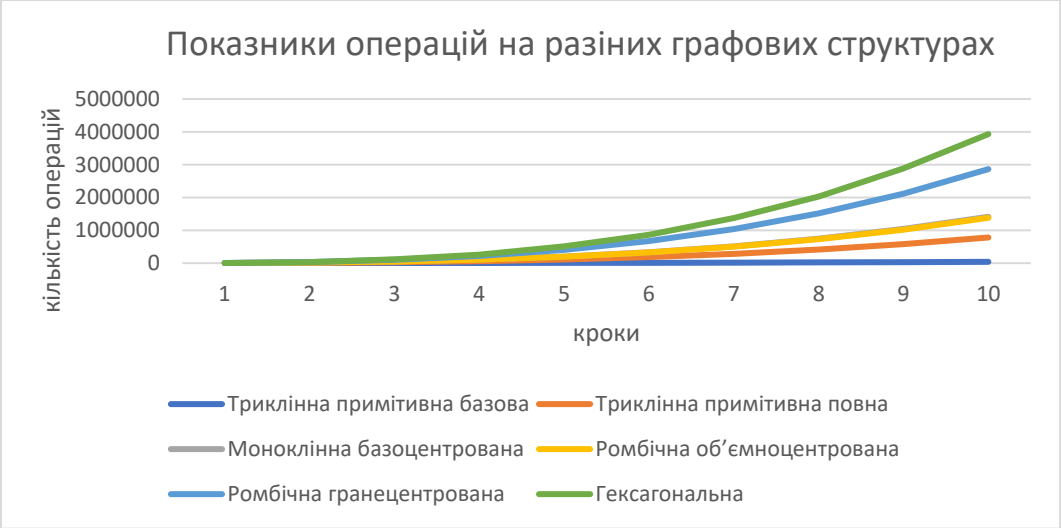


Рисунок 4.13 – Порівняння експериментальної обчислюваної складності з генерації графових структур на основі різних кристалічних ґраток

Було проведено дослідження на шести графових структурах з різною кількістю ребер та вузлів (див Рисунок 4.13).

Як видно із графіків найбільша кількість операцій належить до гексагональної графової структури, в той час як найменша кількість операцій належить до триклінної примітивної базової.

Знайдемо тип емпіричної залежності Рисунок 4.14 для значень триклінної примітивної базової графової структури:

Таблиця 4.7 – Показники графової структури «Гексагональна»

№	X	Y
1	1	342
2	2	996
3	3	2282
4	4	4211
5	5	7088
6	6	10924
7	7	16024
8	8	22399
9	9	30354
10	10	39900

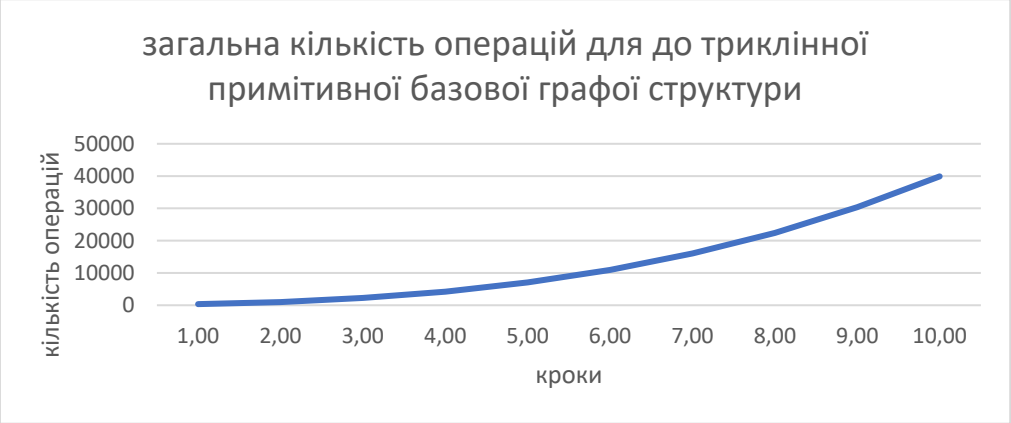


Рисунок 4.14 – Залежність загальної кількості операцій від ітерації при використанні триклінної примітивної базової ґратки

Проведемо допоміжні обчислення та знайдемо для крайніх значень незалежної змінної:

x1=1

x10=10

x_{arif} = 5.5

x_{geom} = 3.16

x_{garm} = 1.82

З графіка знайдемо значення функції, що відповідають розрахунковим значенням:

y_1* = f(5.5) = 9006;

y_2* = f(3.16) = 2595.03

y_3* = f(1.82) = 877.09

Виконаємо додаткові розрахунки для залежної змінної. Знайдемо для крайніх значень:

y_{arif} = 20121

y_{geom} = 3694.02

y_{garm} = 678.19

Порівняємо знайдені графічно значення залежної змінної з yarif, ygeom та ygarm:

e1 = |y1* - yarif| = 11115

e2 = |y1* - ygeom| = 5311.98

e3 = |y1* - ygarm| = 9327.81

e4 = |y2* - yarif| = 17525.97

e5 = |y2* - ygeom| = 1098.99

e6 = |y3* - yarif| = 19243.91

e7 = |y3* - ygarm| = 198.9

Оскільки найменша з абсолютних помилок є e7, то як аналітичну залежність слід вибрати дрібно-раціональну залежність виду y = x/(a + bx).

Вирахуємо коефіцієнт кореляції для вихідного набору даних за наступною формулою:

$$r = \frac{n*\sum_{i=1}^{n} x_i * y_i - \sum_{i=1}^{n} x_i * \sum_{i=1}^{n} y_i}{\sqrt{\left[n*\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]} * \sqrt{n*\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}} = 0.9488$$

Якщо ввести нові змінні z=1/y та q=1/x, то в площині qOz можна отримати лінійну залежність:

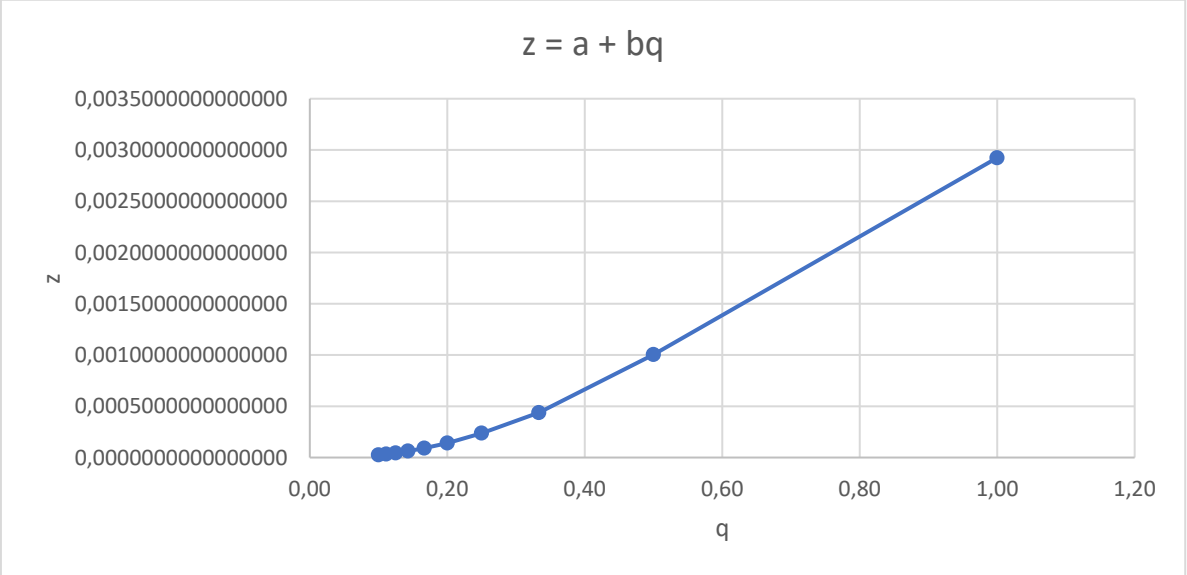


Рисунок 4.15 – Вид аналітичної залежності при заміні змінних

За допомогою методу найменших квадратів порахуємо параметри за даними q та z із попереднього пункту, форму для розрахунків:

$$b = \frac{n*\sum_{i=1}^{n} x_i * y_i - \sum_{i=1}^{n} x_i * \sum_{i=1}^{n} y_i}{n*\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \text{ та } a = \frac{\sum_{i=1}^{n} y_i - b*\sum_{i=1}^{n} x_i}{n}$$

Отримуємо a = -0.00044, b = 0.00322.

Кінцева функція має вигляд:

$$y = \frac{x}{-0.00044 + 0.00322 * \mathrm{x}}$$

Виконаємо тип аналітичної залежності та вид формули для інших графових структур

Триклінна примітивна повна – степенева функція, r = 0.99, формула:

$$y = 2.73 * 3.15^x$$

Моноклінна базоцентрована – степенева функція, r = 0.99, формула:

$$y = 2.74 * 3.4^x$$

Ромбічна об’ємноцентрована – степенева функція, r = 0.99, формула:

$$y = 2.74 * 3.39^x$$

Ромбічна гранецентрована – степенева функція , r = 0.99, формула:

$$y = 2.74 * 3.7^x$$

Гексагональна – дрібно-раціональна функція, r = 0.93, формула:

$$y = \frac{x}{0.00039 + 0.00007 * \mathrm{x}}$$

4.2 Аналіз просторових графових фракталів за допомогою ЛІРА-САПР

Для аналізу напружено-деформованого буде сформовано дві графові структури з фрактальними властивостями кристалічної ґратки. Для генерації кожної графової структури буде застосовано по чотири операції підстановки. Тип перерізу, що буде використовуватися – двотавр. Ребрам графових структур будуть застосовані фізичні властивості використовуваних елементів, так само як і для геометричних параметрів кристалічної ґратки буде використовуватися відповідні параметри. Усі нижні вузли по осі Z будуть закріплені по усім напрямкам. Варто зазначити що показані прикладі є ознайомчими та вимагають аналізу досвідченими фахівцями у сфері будівельної інженерії та у поєднанні з реально існуючими проектами інженерних конструкцій.

4.2.1 Аналіз графової структури з властивостями заліза

Для першого прикладу використовується залізо (Fe). Для генерації графової структури використовуються наступні значення атрибутів:

– тип кристалічної ґратки – кубічна об'ємноцентрована
– атрибути кристалічної ґратки: a = 2.856, b = 2.856, c = 2.856, $\alpha = 90°$, $\beta = 90°$, $\gamma = 90°$
– модуль пружності E = 210000000 Т/м^2;
– питома вага матеріалу R_o = 7.85 Т/м^3;
– коефіцієнт Пуассона v = 0.29;
– розміри двотаврову представлені на Рисунок 4.16.

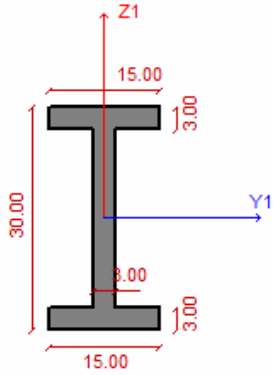


Рисунок 4.16 – Розміри використовувані у двотаврові

Після виконання чотирьох операцій підстановки отримуємо наступну графову структуру:

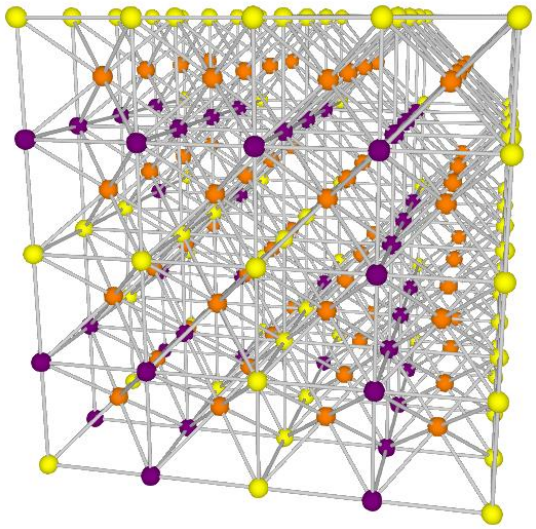


Рисунок 4.17 – Графова структура з фрактальними властивостями кристалічної ґратки заліза після 4 операцій заміни

В лірі отримаємо наступне зображення:

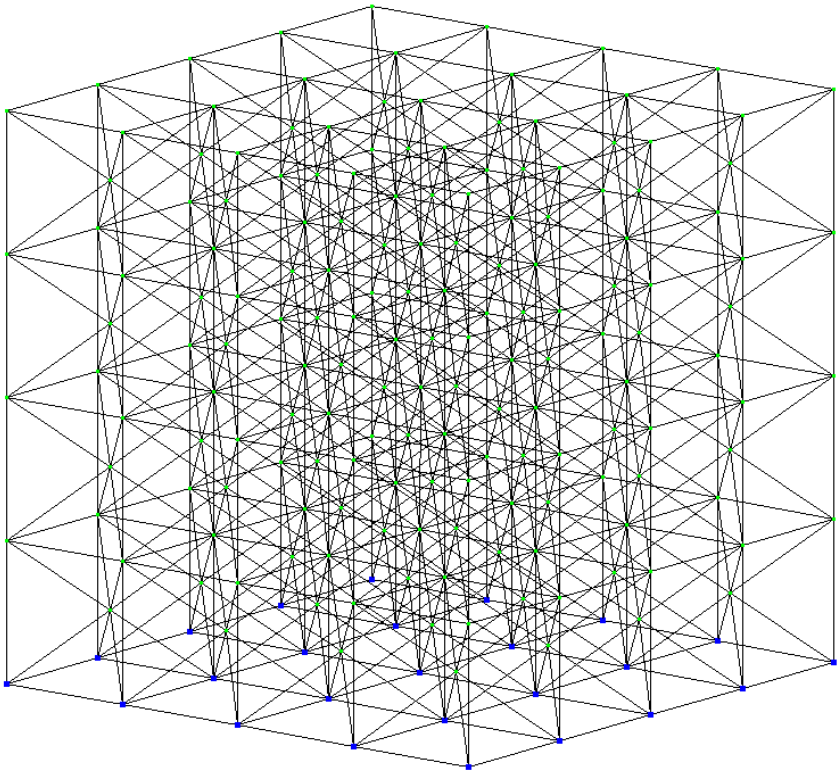


Рисунок 4.18 Графова структура з фрактальними властивостями кристалічної ґратки заліза після 4 операцій заміни у ЛІРА-САПР

Для аналізу застуємо навантаження на усі вузли в 50 Т по осі Z:

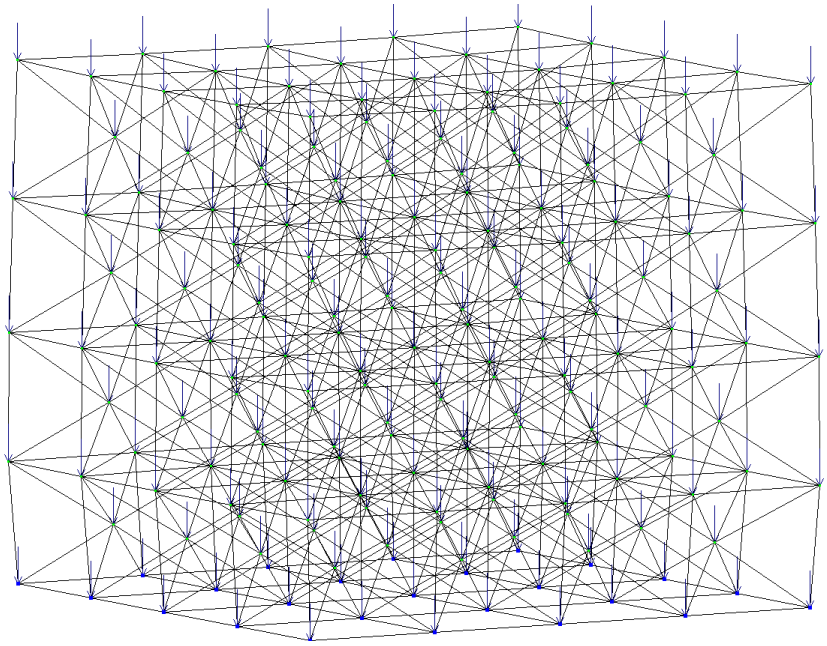


Рисунок 4.19 Графова структура з фрактальними властивостями кристалічної ґратки заліза під навантаженням 50 т.

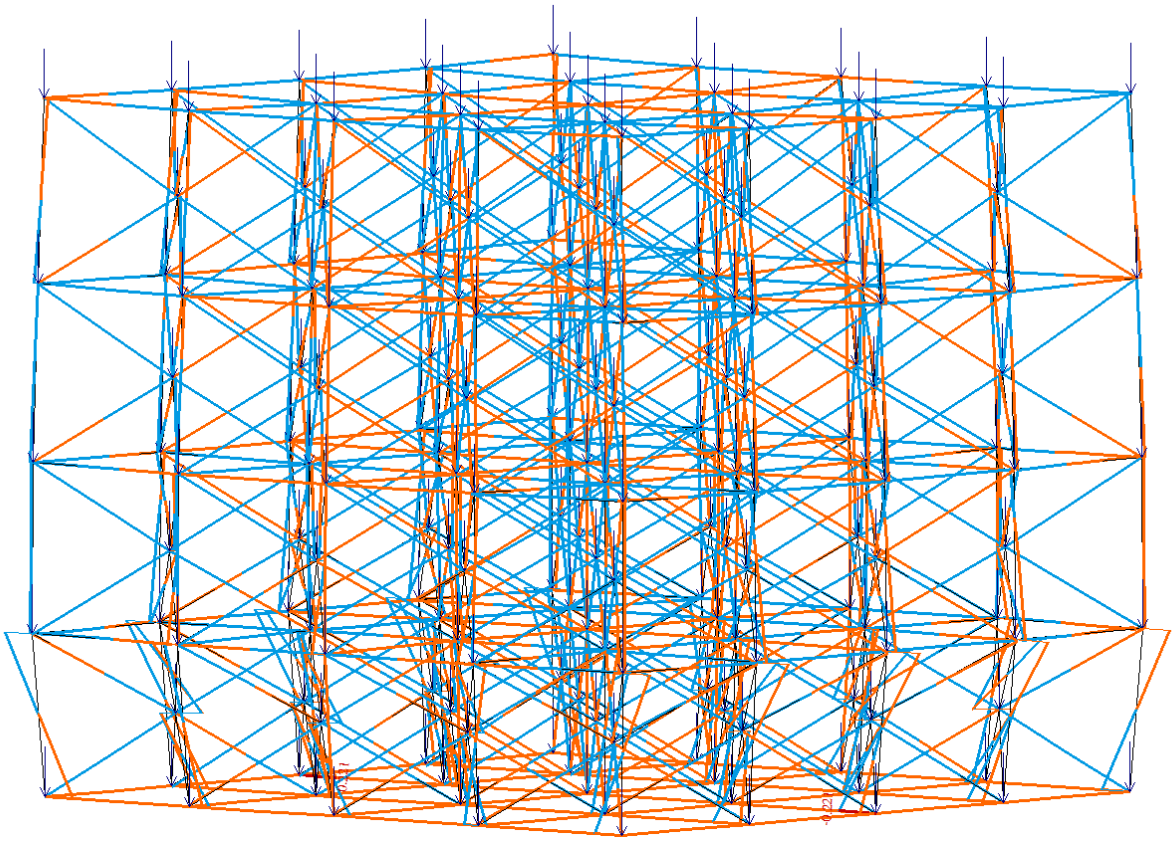


Рисунок 4.20 Епюри навантажень по осі Z для залізної графової структури

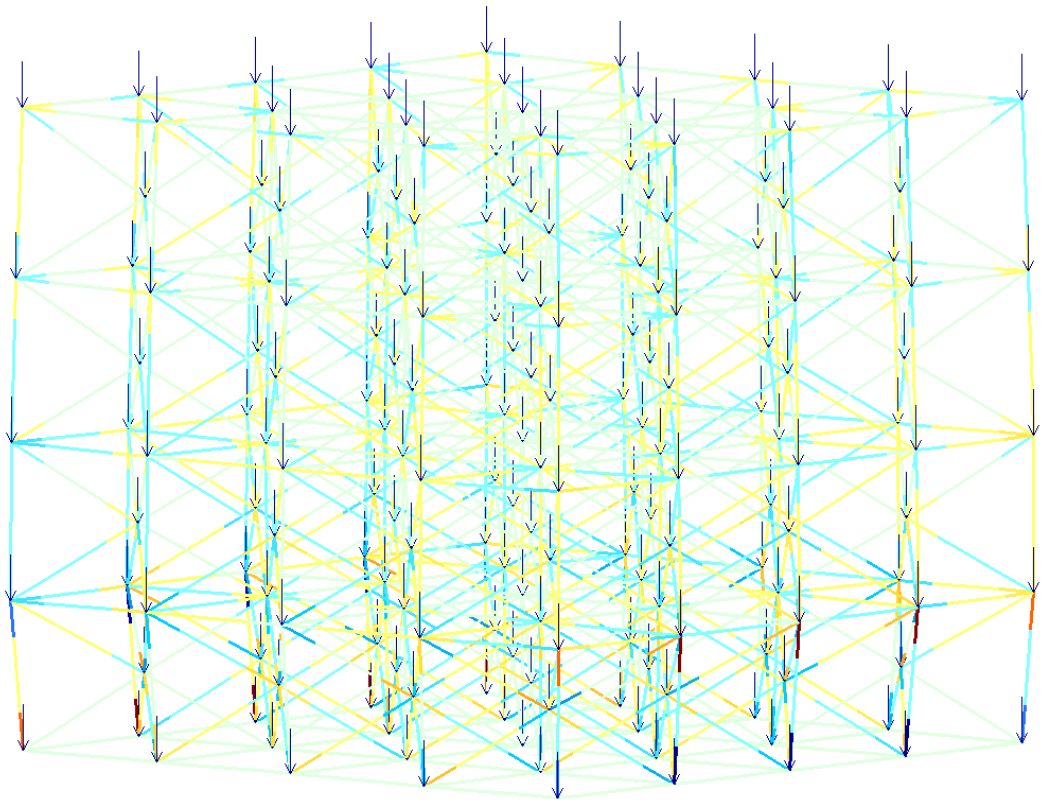


Рисунок 4.21 – Мозаїка навантажень по осі Z для залізної графової структури

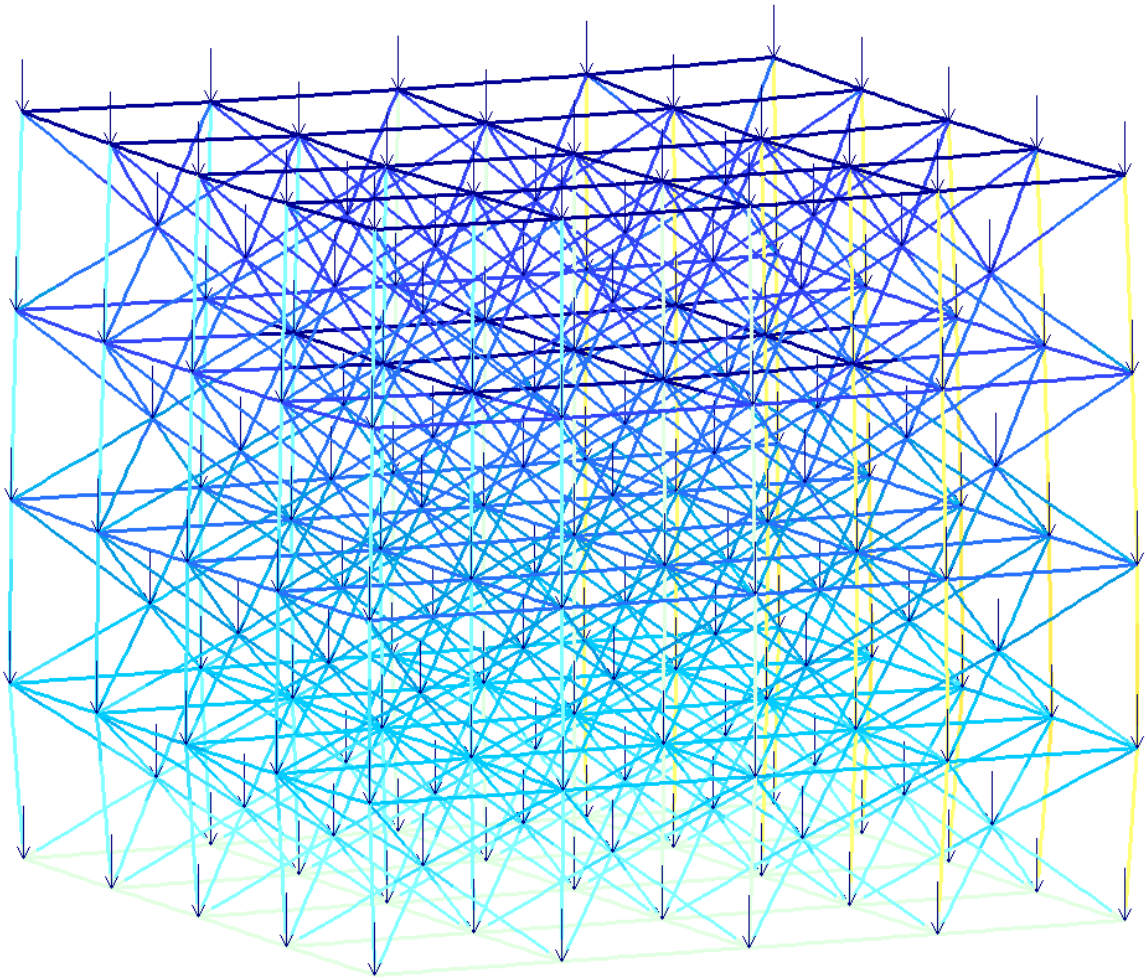


Рисунок 4.22 Мозаїка навантажень по всім осям для залізної графової структури

4.2.2 Аналіз графової структури з властивостями ніобата літію

Для другого прикладу використовується ніобат літію ($LiNbO_3$) [30] – хімічна сполука, змішаний оксид ніобію, літію, безбарвний кристал із ромбоедричною структурою. Для генерації графової структури використовуються наступні значення атрибутів:

- тип кристалічної ґратки – кубічна об'ємноцентрована
- атрибути кристалічної ґратки: a = 0.515, b = 0.515, c = 0. 549, $\alpha = 62°$, $\beta = 62°$, $\gamma = 60°$
- модуль пружності E = 180000000 Т/м^2;
- питома вага матеріалу R_o = 4.3 Т/м^3;
- коефіцієнт Пуассона ν = 0.29;
- розміри двотаврову використовувані як для заліза (див Рисунок 4.16).

Після виконання чотирьох операцій підстановки отримуємо наступну графову структуру:

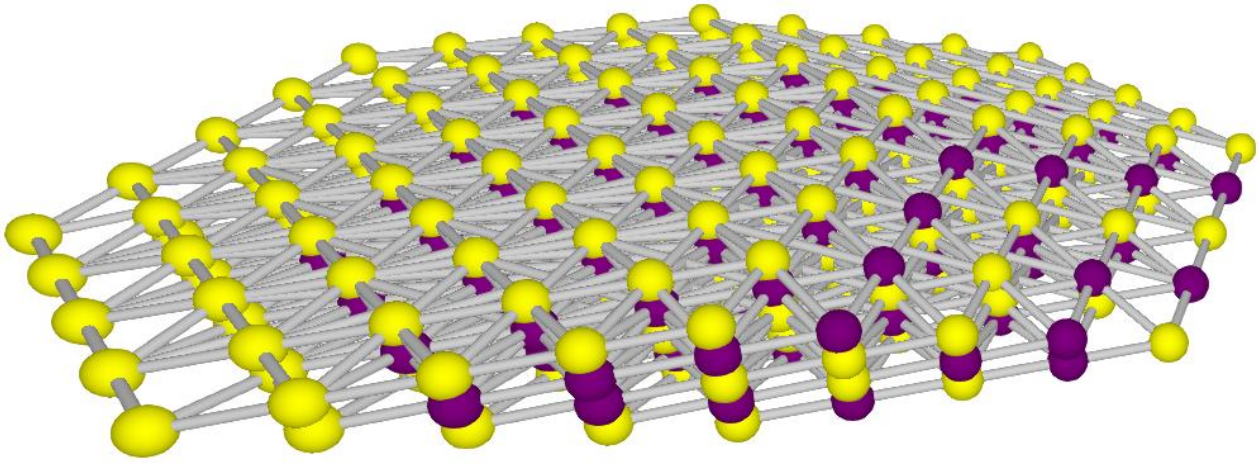


Рисунок 4.23 Графова структура з фрактальними властивостями кристалічної ґратки ніобату літію після 4 операцій заміни

В лірі отримаємо наступне зображення:

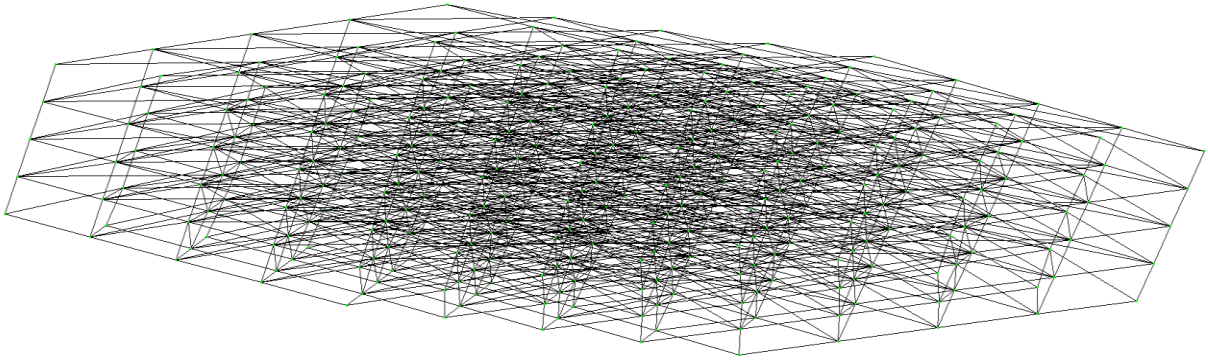


Рисунок 4.24 – Графова структура з фрактальними властивостями кристалічної ґратки ніобату літію після 4 операцій заміни у ЛІРА-САПР

Для аналізу застуємо навантаження на усі вузли в 50 Т по осі Z:

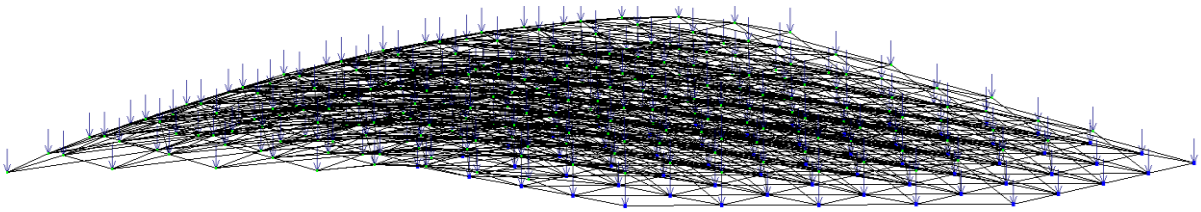


Рисунок 4.25 – Графова структура з фрактальними властивостями кристалічної ґратки ніобату літію під навантаженням 50 т.

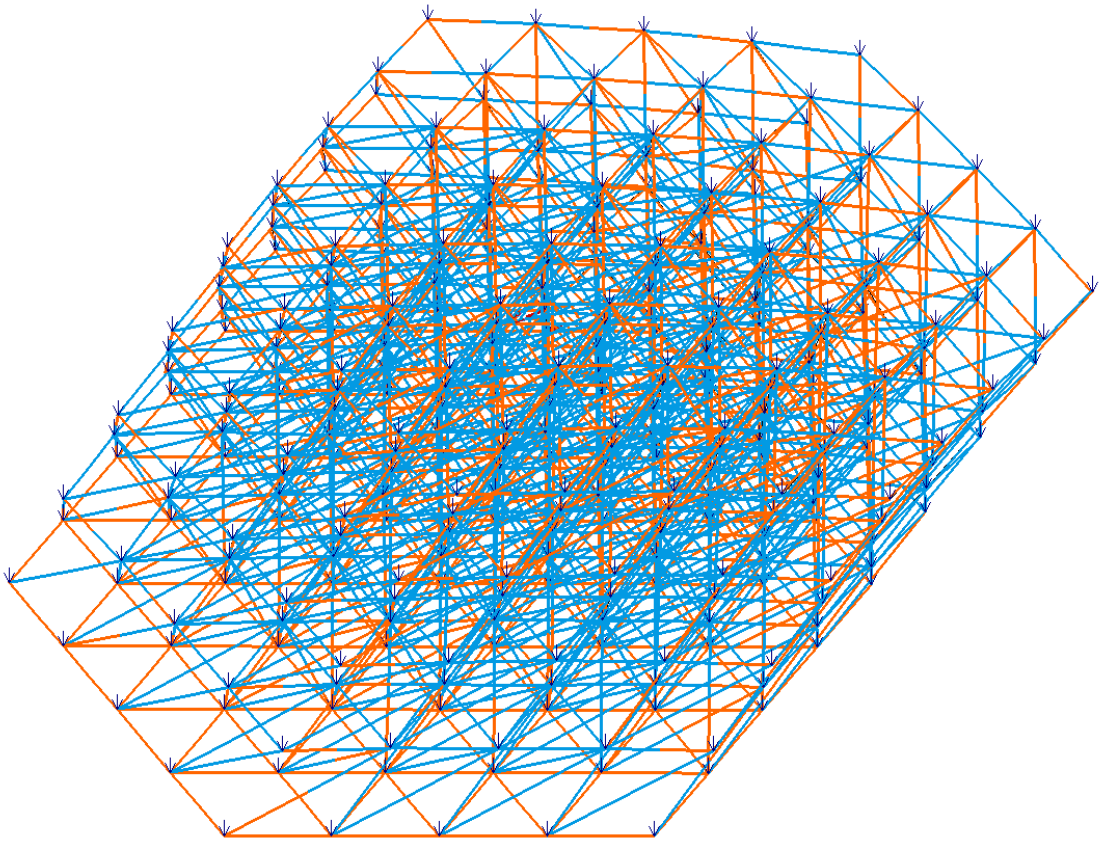


Рисунок 4.26 Епюри навантажень по осі Z для графової структури ніобату літію, вид згори

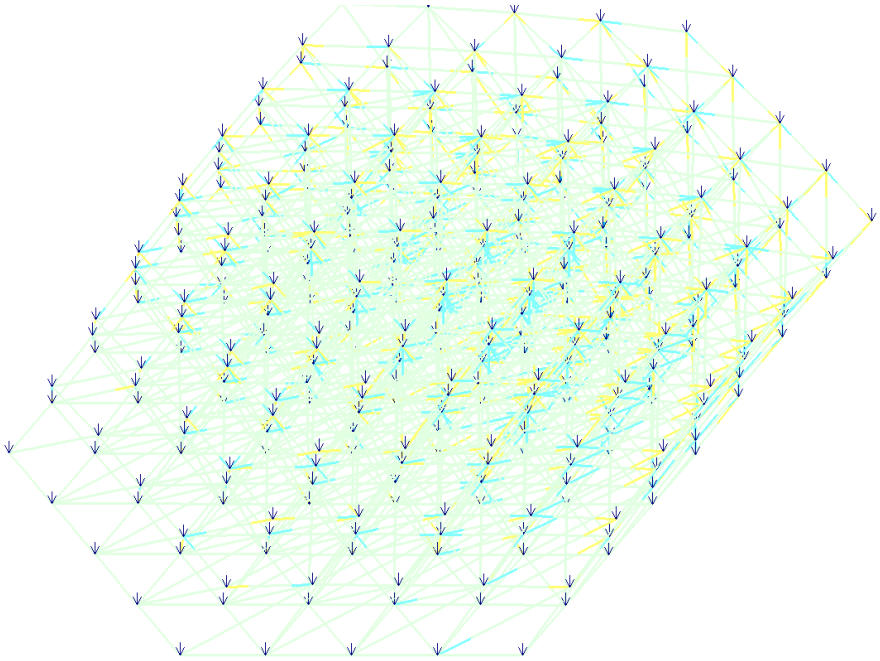


Рисунок 4.27 – Мозаїка навантажень по осі Z для графової структури ніобату літію, вид згори

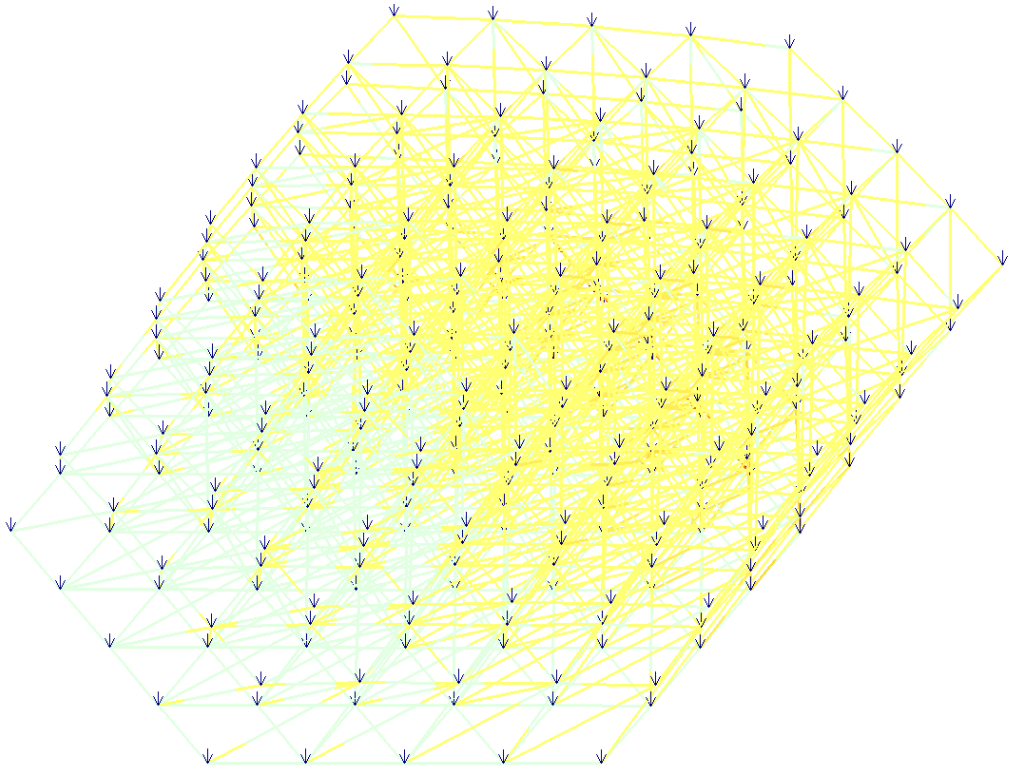


Рисунок 4.28 – Мозаїка навантажень по всім осям для графової структури ніобату літію, вид згори

Висновки до четвертого розділу

У розділі описано процес досліду та результати аналізу по двом напрямкам досліджень: експериментальна обчислювальна складність з генерації мультиатрибутивних просторових графових структур з фрактальними властивостями кристалічної ґратки та аналіз їх напружено-деформованого стану за допомогою ЛІРА-САПР.

При аналізі експериментальної обчислювальної складності було виявлено, що емпірична залежність кількості операцій від номеру підстановки загалом має вигляд степеневої функції. З графіків емпіричної залежності також видно наступне: чим більше елементів має права частина правила, тим більше буде виконано операцій.

Показані приклади напружено-деформованого стану графових структур. Проте для аналізу їх ефективності треба розглядати їх в купі з реально існуючими проектами будівельних конструкцій та враховувати набагато більше факторів, ніж наданими атрибутами конструктора.

ВИСНОВКИ

У результаті роботи над дипломним проектом було проведено дослідження з експериментальної обчислювальної складності генерації мультиатрибутивних просторових графових структур з фрактальними властивостями кристалічної ґратки, а також надано зразки для подальшого дослідження напружено-деформованого стану таких структур.

Дослідження ефективності напружено-деформованого стану графових структур з фрактальними властивостями кристалічної ґратки, що може генерувати програма, вимагає експертності в галузі будівельної інженерії та великої кількості аналізів на прикладах реально існуючих проектів. Визначати їх ефективність поза межами прикладів реальних проектів не є можливим через недостатню кількість факторів та змінних, що можуть вплинути на результат аналізів.

Для дослідження ефективності напружено-деформованого була зроблена інтеграція з програмним комплексом ЛІРА-САПР, за допомогою останньої версії API, яка з’явилась під час виконання дипломної роботи. Надана документація API не є до кінця зрозумілою, в ній існують розбіжності в визначенні параметрів для передачі даних до API, один із таких прикладів є передача параметрів жорсткості, яка по суті є чорним ящиком, так як дані передаються у текстовому форматі без чіткого визначення, що стосується й усього API, де передані параметри мають загальний тип object в мові c#, що буквально є чорною скринькою. Для вирішення проблем з інтеграцією застосовано режим відладки та тестування методами чорної скриньки. Варто також зазначити, що хоча версія API є найостаннішою, вона не надає можливості для повної передачі усіх параметрів через API, до таких параметрів відноситься, як приклад, навантаження та багато іншого. Проте не дивлячись на усі проблеми з API, ЛІРА-САПР є потужним інструментом в проектуванні і розрахунку машинобудівних та будівельних конструкцій різного призначення, та вирішує поставлені перед нею задачі.

Для дослідження з експериментальної обчислювальної складності виконувався підрахунок кількості виконаних атомарних операцій на рівні дизасимбльованого коду програми. Визначено тип аналітичної залежності між кількістю виконаних операцій на номером підстановки, що генерує мультиатрибутивний графовий фрактал. У більшості випадків аналітична залежність мала вид степеневої функції, а коефіцієнт кореляції становив R = 0.99. При більшій кількості задіяних елементів в правій частині правила підстановки збільшувалась і кількість операцій, проте це цілком не впливало на вид аналітичної залежності, тому можна вважати, що експериментальна обчислювана складність з генерації просторових графових структур з фрактальними властивостями кристалічної ґратки приймає вигляд степеневої функції, що у свою чергу дозволяє визначати час та кількість використаних ресурсів для їх генерації.

Для генерації мультиатрибутивних просторових графових фракталів задіяно конструктивно-продукційну структуру, для якої задіяну низку перетворень, що в результаті дає потужний інструмент у вигляді конструктора. Особливу увагу заслуговує зв’язок між мультиатрибутикою та, задіяними в конструкторі, алгоритмічними системами. Мультиатрибутика вказує, що атрибути є різного характеру та по-різному впливають на алгоритмічні структури: геометричні атрибути впливають на зовнішній вигляд 3D моделі графової структури, атрибут кольору використовується для операції підстановки одних елементів графового структури на інші, а фізичні атрибути, такі як жорсткість, впливають на напружено-деформований стан.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Shynkarenko, V., Letuchyi, O., Chyhir, R., Constructive-synthesizing modeling of fractal crystal lattices [Virtual Resource] /Viktor Shynkarenko, Oleksandr Letuchyi, Robert Chyhir // IEEE 18th International Conference on Computer Science and Information Technologies (CSIT). – 2023. – pp. 1-4 – Access Mode: DOI: 10.1109/CSIT61576.2023.10324251.
2. Летучий О., Шинкаренко В., Засоби формування графових 3D фракталів [Текст]о / О. І. Летучий, В. І. Шинкаренко // Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті: тез. XVII міжнар. Наук.-практ. Конф. (13.12.– 14.12.2023) / Мін-во інфраструктури України, Укр. держ. ун-т науки та техн. – Дніпро – 2023 – С. 76.
3. Letuchyi, O., Shynkarenko, V., Graph fractals with the variability of the formation process [Virtual Resource] / Oleksandr Letuchyi, Viktor Shynkarenko, // Information Technologies in Metallurgy and Machine building – ITMM. – 2024. – pp. 271-274 – Access Mode: DOI: 10.34185/1991-7848.itmm.2024.01.049.
4. Летучий О., Шинкаренко В., Використання ЛІРИ-САПР у клієнт-серверному додатку [Текст]о / О. І. Летучий, В. І. Шинкаренко // Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті: тез. XVIII міжнар. Наук.-практ. Конф. (12.12.– 13.12.2024) / Мін-во інфраструктури України, Укр. держ. ун-т науки та техн. – Дніпро – 2024 – С. 95.
5. Feder, Jens. *Fractals*. [Text] Springer Science & Business Media, 2013 – 1 p.
6. Bravias Lattice [Virtual Resource] Access Mode: URL: https://byjus.com/chemistry/bravais-lattice/#14-Types-of-Bravais-Lattices. Date of Access: 14 Jan 2025
7. Ille, P., Woodrow, R., Fractal graphs [Virtual Resource] / Pierre Ille, Robert Woodrow, // Journal of Graph Theory. – 2019 – pp. 53-72 – Access Mode: DOI: 10.1002/jgt.22420.
8. Pub. No.: US 2013/0066078 A1. Methods of nanoassembly ofa fractal polymer and materals formed thereby [Text] / George R. Newkome (US), Charles N. Moorefield (US); patent application publication: Newkome et al. – № 13/548,664 – filled 13.07.2012 – published 14.03.2013.
9. Rayneau-Kirkhope, D., Mao, Y., Farr, R., Optimization of fractal space frames under gentle compressive load [Virtual Resource] / Daniel Rayneau-Kirkhope, Yong Mao, Robert Farr // Physical Review E. – 2013 – pp. 063204-1 - 063204-7 – Access Mode: DOI: 10.1103/PhysRevE.87.063204.

10. VESTA: Visualization for Electronic and STructual Analysis [Virtual Resource] Access Mode: URL https://jp-minerals.org/vesta/en/. Date of Access: 14 January 2025
11. Reimagine what’s possible in design and manufacturing [Virtual Resource] Access Mode: URL https://www.autodesk.com/industry/manufacturing Date of Access: 14 January 2025
12. Оптимальна конфігурація комп'ютера для LIRA-FEM [Електронний ресурс] Режим доступу: https://help.liraland.com/uk-ua/general/system-requirements.html Дата доступу: 14 січня 2025
13. Shynkarenko, V. I., Ilman, V. M. Constructive-Synthesizing Structures and Their Grammatical Interpretations. I. Generalized Formal Constructive-Synthesizing Structure [Text] /V. I. Shynkarenko, V. M. Ilman// Cybernetics and Systems Analysis, Volume 50, Issue 5, pp 655-662
14. Shynkarenko, V. I., Ilman, V. M. Constructive-Synthesizing Structures and Their Grammatical Interpretations. II. Refining Transformations [Text] / V. I. Shynkarenko, V. M. Ilman// Cybernetics and Systems Analysis, Volume 50, Issue 6, pp 829-841.
15. Шинкаренко, В. І., Ільман, В. М., Скальозуб, В. В. Структурні моделі алгоритмів у задачах прикладного програмування. Частина І. [Текст] / В. І. Шинкаренко, В. М. Ільман, В. В. Скальозуб //Формальні алгоритмічні структури. Кібернетика і системний аналіз. 2009. №3. С. 3–14.
16. Goldsmith, S. F., Aiken, A. S., Wilkerson, D. S. Measuring empirical computational complexity. [Text] : / S. F. Goldsmith, A. S. Aiken, D. S. Wilkerson // In Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, 2007 September. – pp. 395-404 .
17. A Solid Guide to SOLID Principles. [Virtual Resource] Access Mode: URL: https://www.baeldung.com/solid-principles. Date of Access: 14 January 2025.
18. What is Inversion of Control. [Virtual Resource] Access Mode: URL: https://www.educative.io/answers/what-is-inversion-of-control. Date of Access: 14 January 2025.
19. Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming [TEXT] WPF Notifications, Validations, Commands, and MVVM / Phillip Japikse, Andrew Troelsen, 2021. – pp. 1143 – 1177.
20. Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming [TEXT] WPF Notifications, Validations, Commands, and MVVM / Phillip Japikse, Andrew Troelsen, 2021. – pp. 1182 – 1186.

21. What is WPF. [Virtual Resource] Режим доступу: URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-7.0 Date of Access: 14 January 2025 p.
22. Helix Toolkit. [Virtual Resource] Access Mode: URL: https://github.com/helix-toolkit/helix-toolkit. Date of Access: 14 June 2025 p.
23. Direct3D 11 Reference. [Virtual Resource] Access Mode: URL: https://learn.microsoft.com/en-us/windows/win32/direct3d11/d3d11-graphics-reference. Date of Access: 14 January 2025
24. Vulkan API. [Virtual Resource] Access Mode: URL: https://www.vulkan.org/. Date of Access: 14 January 2025
25. Made with Unity: 2023 in review. [Virtual Resource] Access Mode: URL: https://unity.com/blog/games/made-with-unity-2023-review . Date of Access: 14 January 2025
26. Vanhove, S. Learning GDScript by Developing a Game with Godot 4: A Fun Introduction to Programming in GDScript 2.0 and Game Development Using the Godot Engine. Germany: Packt Publishing, 2024. – pp 4-23.
27. Welcome to the Stride Game Engine [Virtual Resource] Access Mode: URL: https://github.com/stride3d/stride Date of Access: 14 January 2025
28. Nidhra, S., & Dondeti, J. Black box and white box testing techniques-a literature review. International Journal of Embedded Systems and Applications (IJESA), 2012 – pp 29-50.
29. First look at the Visual Studio Debugger [Virtual Resource] Access Mode: URL: https://learn.microsoft.com/en-us/visualstudio/debugger/debugger-feature-tour?view=vs-2022 . Date of Access: 14 January 2025
30. Weis, R. S., and T. K. Gaylord. Lithium niobate: Summary of physical properties and crystal structure. [Text]: Applied Physics, 1985. – pp 191-203.

ДОДАТОК А

Технічне завдання

ЗАТВЕРДЖУЮ

Проректор Українського державного

університету науки і технологій

Анатолій РАДКЕВИЧ

«КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ МУЛЬТИАТРИБУТИВНИХ ПРОСТОРОВИХ ГРАФОВИХ ФРАКТАЛІВ»

Технічне завдання

44165850.1443-01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Віктор ШИНКАРЕНКО
Виконавець
_____Олександр ЛЕТУЧИЙ
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.1443-01

«КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ МУЛЬТИАТРИБУТИВНИХ ПРОСТОРОВИХ ГРАФОВИХ ФРАКТАЛІВ»

Технічне завдання

44165850.1443-01

Листів 12

2025

44165850.1443-01

ЗМІСТ

ВСТУП

Фрактали знаходять широке застосування в науці, техніці та мистецтві завдяки своїм властивостям самоподібності та здатності моделювати складні системи. Вони використовуються для створення реалістичних природних об’єктів у комп’ютерній графіці, аналізу біологічних структур у медицині, розробки компактних антен у телекомунікаціях, моделювання природних процесів в екології та геології, а також для вивчення фізичних і хімічних явищ.

Просторові графові фрактали є мало дослідженим явищем, що відкриває нові можливості для моделювання складних структур, які повторюють закономірності природи.

Мета цього дослідження — підтвердження гіпотези щодо ефективності в будівельних конструкціях структур подібних до кристалічних ґраток, а також отримання показників з експериментальної обчислювальної складності з їх генерування.

У рамках роботи проводиться:

- аналіз експерементальної обчислювальної складності з генерування графових фракталів на основі різних типів кристалічних ґраток;
- інтеграція з ЛІРА-САПР;
- аналіз напружено-деформованого стану згенерованих конструкцій в ЛІРА-САПР.

44165850.1443-01

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Основою для розробки є наказ ректора Українського державного університету науки і технології Радкевич А.В. «Про затвердження тем та призначення керівників дипломних проектів» №1186 ст від 29.12. 2023 року.

Тема проекту: «Конструктивно-продукційне моделювання мультиатрибутивних просторових графових фракталів».

Керівник дипломного проекту: Шинкаренко В. І.

9. ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – генерація стрижневих конструкцій з фрактальними властивостями кристалічних ґраток, розрахунки напружено-деформованого стану, підрахунок експериментальної обчислювальної складності з їх генерації.

Експлуатаційне призначення – забезпечення можливості моделювання та аналізу стрижневих конструкцій з фрактальними властивостями для дослідження їх механічних характеристик, оптимізації матеріалів та конструктивних рішень.

10. ВИМОГИ ДО ПРОГРАМИ

10.1. Вимоги до функціональних характеристик

Програма надає можливості для генерації просторових графових фракталів та їх перегляду використовуючи 3D графіку. За допомогою інтеграції з ЛІРА-САПР є можливість передати візуально створену модель до ЛІРА-САПР та продовжити її подальший аналіз. В самій програмі є можливість викликати вікно для перегляду показників експериментальної обчислювальної складності з генерації просторових графових фракталів

Вимоги до складу виконуваних функцій:

- візуальне представлення просторових графових фракталів за допомогою 3D графіки
- можливість передати згенеровану модель до ЛІРА-САПР;
- задання атрибутики для генерації графових фракталів;
- збереження згенерованої моделі в окремому файлі;
- завантаження збереженої раніше моделі до програми;
- оновлення програми.

Вхідні дані є параметри атрибутики для формування графових фракталів.

Вихідні дані:

- візуальна 3D модель
- таблиця з метриками експериментальної обчислювальної складності

10.2. Вимоги до надійності

Програма має відповідати таким вимогам:

- при збої обладнання робота програми може бути продовжена шляхом повторного запуску програми;
- програма не повинен допускати невимушену втрату даних;
- кількість помилок не повинна перевищувати однієї на 10000 операцій;
- програма стійка до неправильного вводу даних;
- наявність архівної копії тексту програми на зовнішньому носії;

– наявність резервної копії бази даних на зовнішньому носії.

10.3. Умови експлуатації

ПЗ буде функціонувати у книжкових магазинах. Температурний режим в приміщенні, де встановлений ПК, має бути 21-25°С, а вологість 40-60%. Користувачами є завідуючі або менеджери магазину, які ознайомлені з керівництвом користувача.

10.4. Вимоги до складу і параметрів технічних засобів

Програмний продукт розрахований на IBM-сумісні ПК, що мають такі мінімальні характеристики:

– Чотириядерний процесор Intel Core з тактовою частотою 3.0 ГГц;
– Зовнішній або вбудований 3D відеоадаптер Nvidia, Intel, AMD
– об’єм оперативної пам’яті 8 ГБ;
– пам’ять на жорсткому диску 800 Мб.
– USB-порт для перенесення програми на комп’ютер;
– маніпулятор «миша»;
– клавіатура;
– монітор з роздільною здатністю 1920*1080;
– підключення до локальної мережі;

10.5. Вимоги до інформаційної і програмної сумісності;

Для роботи з програмним продуктом на ПК має бути встановлене таке програмне забезпечення:

– ОС: Windows 10/11;
– Microsoft .NET Framework 4.8 (веб-інсталятор);
– мова програмування C#, .NET Framework;
– ЛІРА-САПР 2024.

11. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації повинні входити:

- текст програми;
- керівництвом користувача.

Програмна документація повинна відповідати вимогам ДСТУ [1].

12. ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Показники та методи їх розрахунку не були детально описані в документації, оскільки розробка програмного продукту має навчальний, а не комерційний характер.

13. СТАДІЇ І ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки програмного продукту приведені в Таблиця *4.8*

Таблиця 4.8. Стадії та етапи розробки програмного продукту

№	Етапи розробки	Терміни виконання
1	Аналіз предметної області, визначення структури даних, складання функціональних і експлуатаційних вимог, складання вимог до програмного забезпечення і технічної реалізації, прототипування і дизайн. Узгодження та затвердження Технічного завдання.	03.27.24 – 28.06.24
2	Програмування і відладка програми	04.08.24 – 01.12.24
3	Тестування програми	02.12.24 – 15.12.24
4	Розробка, узгодження та затвердження програмної документації.	16.12.24 – 30.12.24

14. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль здійснює керівник розробки Шинкаренко В.І.

Прийом здійснює комісія у складі:

- Волкова С.А.
- Гарячкін В. М.

15. БІБІЛОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

ДОДАТОК Б

Керівництво користувача

ЗАТВЕРДЖУЮ

Проректор Українського державного

університету науки і технологій

Анатолій РАДКЕВИЧ

«КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ МУЛЬТИАТРИБУТИВНИХ ПРОСТОРОВИХ ГРАФОВИХ ФРАКТАЛІВ»

Керівництво користувача

44165850.01443 – 01 І3 01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Віктор ШИНКАРЕНКО
Виконавець
_____Олександр ЛЕТУЧИЙ
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.01443-01 І3 01

«КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ МУЛЬТИАТРИБУТИВНИХ ПРОСТОРОВИХ ГРАФОВИХ ФРАКТАЛІВ»

Керівництво користувача

44165850.01443 – 01 І3 01

Листів 9

2025

44165850.01443-01 І3 01
АНОТАЦІЯ

Документ 1116130.01443 – 01 І3 01 «Конструктивно-продукційне моделювання мультиатрибутивних просторових графових фракталів. Керівництво користувача».

Програма написані на мові C# з використанням фреймворках .Net Framework 4.8.1, з використанням бібліотек Helix та COM бібліотекою LiraSapr у програмному середовищі Visual Studio.

ЗМІСТ

1 ВВЕДЕННЯ

Програмний додаток «Конструктивно-продукційне моделювання мультиатрибутивних просторових графових фракталів» призначене для вивчення процесу побудові графових фракталів, аналізу показників експериментальної обчислюваної складності з їх генерації. Підтримується інтеграція з ЛІРА-САПР. За допомогою ЛІРА-САПР можливий подальший аналіз напружено-деформованого стану згенерованих конструкцій з фрактальними властивостями кристалічної ґратки.

Програма надає наступні можливості:

- генерація графових фракталів;
- інтеграція з ЛІРА-САПР;
- можливість збереження процесу роботи, тобто згенерованої моделі;
- завантаження збереженої раніше роботи;
- перегляд показників обчислювальної складності з генерації графових фракталів.

Передбачається, що користувач впевнено володіє комп’ютером, має просунуті навички для роботи з ЛІРА-САПР.

Додаткова експлуатаційна документація непотрібна.

2 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Дана програма призначена для введення досліджень просторових графових фракталів, а також для полегшення користування ЛІРА-САПР.

ПЗ буде функціонувати у НІІ. Температурний режим в приміщенні, де встановлений ПК, має бути 21-25°С, а вологість 40-60%. Користувачами є студенти або фахівці з комп’ютерної інженерії, будівельної інженерії або матеріалознавства, які ознайомлені з керівництвом користувача

Програмний продукт розрахований на ІВМ-сумісні ПК, що мають такі мінімальні характеристики:

- Чотириядерний процесор Intel Core з тактовою частотою 3.0 ГГц;
- Зовнішній або вбудований 3D відеоадаптер Nvidia, Intel, AMD
- об’єм оперативної пам’яті 8 ГБ;
- пам’ять на жорсткому диску 800 Мб.
- USB-порт для перенесення програми на комп’ютер;
- маніпулятор «миша»;
- клавіатура;
- монітор з роздільною здатністю 1920*1080;
- підключення до локальної мережі;

Для роботи з програмним продуктом на ПК має бути встановлене таке програмне забезпечення:

- ОС: Windows 10/11;
- Microsoft .NET Framework 4.8 (веб-інсталятор);
- мова програмування C#, .NET Framework;
- ЛІРА-САПР 2024.

3 ПІДГОТОВКА ДО РОБОТИ

ПЗ можна транспортувати на будь-якому запам’ятовуючому носії, що сумісний з usb портом та можуть зберігати дані. Носій повинен мати як мінімум 4 Gb пам’яті.

Місце та строк зберігання обумовлені термінами зберігання відповідним носієм, але для додаткової безпеки вони не повинні потрапляти під прямі сонячні промені та повинні перебувати у сухому місці.

Встановлення програми відбувається за допомогою дистрибутивного носія даних, оновлення програми здійснюється за допомогою мережі Інтернет.

Перше, що необхідно зробити для перевірки працездатності – перевірити апаратну та програмну сумісність. Після цього необхідно перевірити достатність місця для встановленої програми. Наступним кроком є впевнитися в доступі та стабільній роботі мережі Інтернет, а також перевірити актуальність версії програми, тобто щоб версія була остання.

4 ОПИС ОПЕРАЦІЙ

Після встановлення та запуску програмного додатку " Конструктивно-продукційне моделювання мультиатрибутивних просторових графових фракталів", користувач бачить наступний інтерфейс. На рис.1 представлено головне вікно програми.

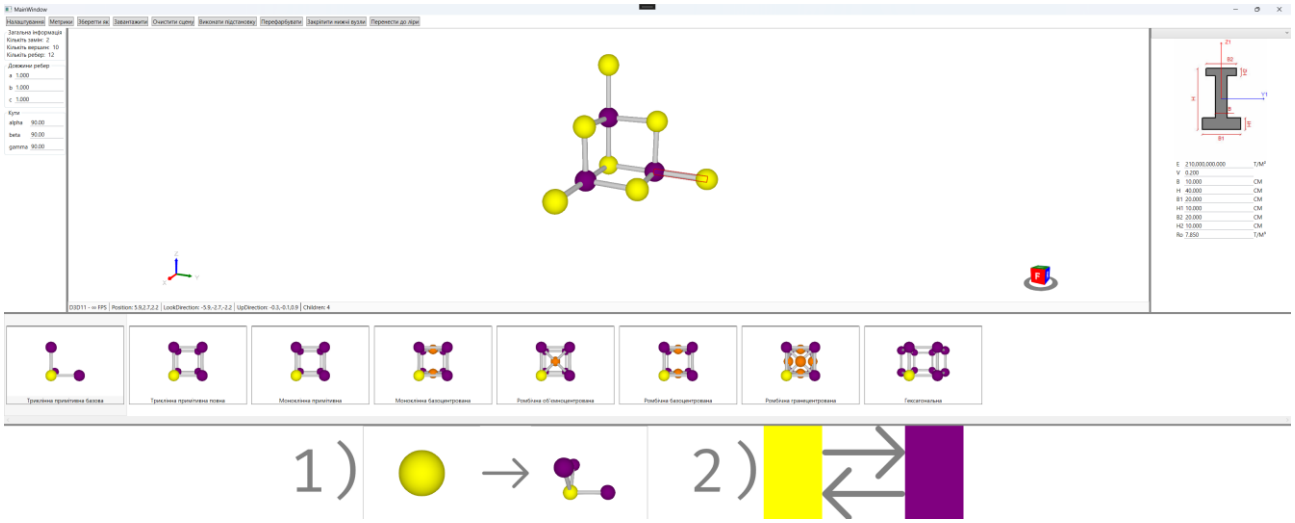


Рисунок 1 – Інтерфейс додатку

5 АВАРІЙНІ СИТУАЦІЇ

Втрата живлення вимагає повторення операцій усіх операцій. Якщо процес роботи не було збережено заздалегідь, то його втрачено назавжди.

6 РЕКОМЕНДАЦІЇ ЩОДО ЗАСТОСУВАННЯ

При закритті програми дані не збережуться автоматично, їх треба зберігати власноруч.

Для правильності передачі даних до ЛІРА-САПР, ЛІРА-САПР має бути оновлений до останньої версії.

ДОДАТОК В

Текст програми

ЗАТВЕРДЖУЮ

Проректор Українського державного

університету науки і технологій

Анатолій РАДКЕВИЧ

«КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ МУЛЬТИАТРИБУ-ТИВНИХ ПРОСТОРОВИХ ГРАФОВИХ ФРАКТАЛІВ»

Текст програми

44165850.01443–01 12–01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Віктор ШИНКАРЕНКО
Виконавець
_____Олександр ЛЕТУЧИЙ
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.01443–01 12–01

КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ МУЛЬТИАТРИБУТИВНИХ ПРОСТОРОВИХ ГРАФОВИХ ФРАКТАЛІВ

Текст програми

44165850.01443-01 12 01

Листів 82

2025

АНОТАЦІЯ

Документ 44165850.01443–01 12 01 «Конструктивно-продукційне моделювання мультиатрибутивних просторових графових фракталів» входить до складу програмної документації на програму, що описує результати аналізу продуктивності та ефективності веб-додатків.

У даному документі представлений текст додатку. Додаток написаний на мові JavaScript з використанням фреймворків React та React-native у програмному середовищі Visual Studio Code.

ЗМІСТ

1 ТЕКСТ ПРОГРАМИ

1.1. MetricsWindow.xaml

```
<Window x:Class="MetricsUI.MetricsWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:MetricsUI"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800">
   <DockPanel>
      <StackPanel DockPanel.Dock="Top" Orientation="Horizontal">
         <Button Content="Зберегти" Command="{Binding Path=SaveMetricsCommand}" Width="60" Margin="0 4 0 4"/>
      </StackPanel>
      <ContentControl Content="{Binding MetricTreeControl}"/>
   </DockPanel>
</Window>
```

1.2. MetricsWindow.xaml.cs

```
using System;
using System.Windows;
using FractalGraph.Abstractions;
using MetricsService;

namespace MetricsUI
{
   /// <summary>
   /// Interaction logic for MetricsWindow.xaml
   /// </summary>
   public partial class MetricsWindow : Window
   {
      IMetricsOwner _owner;
      public MetricsWindow(IMetricsService metricsService, IMetricsOwner metricsOwner)
      {
         _owner = metricsOwner;
         _owner.Closed += Owner_Closed;

         MetricsNode root = null;
         IMetricsServiceInternals internals = metricsService as IMetricsServiceInternals;
         root = internals.GetRoot();

         var metricsTree = new MetricsTree(new MetricsTreeModel(root), internals);
         var viewModel = new MetricsViewModel(metricsService);
         viewModel.MetricTreeControl = metricsTree;

         DataContext = viewModel;

         InitializeComponent();
      }

      private void Owner_Closed(object sender, EventArgs e)
      {
         this.Close();
      }
   }
}
```

1.3. MetricsTree.xaml

```
<UserControl x:Class="MetricsUI.MetricsTree"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:tree="clr-namespace:MetricsUI.Tree">
   <Grid>
      <tree:TreeList x:Name="_treeList">
         <tree:TreeList.View>
            <GridView>
```

```
            <GridView.Columns>
              <GridViewColumn
Header="Name">

<GridViewColumn.CellTemplate>
                  <DataTemplate>
                    <StackPanel
Orientation="Horizontal">
                      <tree:RowExpander/>
                      <TextBlock
Text="{Binding Name}"></TextBlock>
                    </StackPanel>
                  </DataTemplate>

</GridViewColumn.CellTemplate>
              </GridViewColumn>

              <GridViewColumn
Header="Arithmetics"
DisplayMemberBinding="{Binding
Arithmetics}"/>
              <GridViewColumn
Header="Assignments"
DisplayMemberBinding="{Binding
Assignments}"/>
              <GridViewColumn
Header="Comparisons"
DisplayMemberBinding="{Binding
Comparisons}"/>
              <GridViewColumn
Header="Jumps"
DisplayMemberBinding="{Binding Jumps}"/>

            </GridView.Columns>
          </GridView>
        </tree:TreeList.View>
      </tree:TreeList>
    </Grid>
</UserControl>
```

1.4. MetricsTree.xaml.cs

```
using System;
using System.Windows.Controls;
using MetricsService;

namespace MetricsUI
{
    /// <summary>
    /// Interaction logic for MetricsTree.xaml
    /// </summary>
    public partial class MetricsTree : UserControl
    {
        IMetricsServiceInternals _internals;
        public MetricsTree()
        {
            InitializeComponent();
        }

        internal MetricsTree(MetricsTreeModel
metricsTreeModel, IMetricsServiceInternals
internals) : base()
        {
            InitializeComponent();

            if (metricsTreeModel == null)
                throw new ArgumentException();

            _internals = internals;
            _internals.MethodStuckBecameEmtpy
+= _internals_MethodStuckBecameEmtpy;

            _treeList.Model = metricsTreeModel;
        }

        private void
_internals_MethodStuckBecameEmtpy(object
sender, EventArgs e)
        {
            _treeList.Invalidate();
        }

        public void Invalidate()
        {
            _treeList.Invalidate();
        }
    }
}
```

1.5. RelayCommand.cs

```
using System;
using System.Windows.Input;

namespace MetricsUI
{
    public class RelayCommand<T> : ICommand
    {
```

```
        readonly Action<T> _execute;
        readonly Func<T, bool> _canExecute;

        public event EventHandler
CanExecuteChanged;

        public RelayCommand(Action<T> execute,
Func<T, bool> canExecute = null)
        {
            if (execute == null)
                throw new
ArgumentNullException("execute");

            _execute = execute;
            _canExecute = canExecute;
        }

        public bool CanExecute(object parameter)
        {
            return _canExecute == null ||
_canExecute((T)parameter);
        }

        public void Execute(object parameter)
        {
            _execute((T)parameter);
        }
    }
}
```

1.6. Generic.xaml

```
<ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/200
6/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2
006/xaml"
                    xmlns:tree="clr-
namespace:MetricsUI.Tree"
                    xmlns:local="clr-
namespace:MetricsUI">
    <Style x:Key="ExpandCollapseToggleStyle"
TargetType="{x:Type ToggleButton}">
        <Setter Property="Focusable"
Value="False"/>
        <Setter Property="Width" Value="19"/>
        <Setter Property="Height" Value="13"/>
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate
TargetType="ToggleButton">
                    <Border Background="#00FFFFFF"
Width="19" Height="13">
                        <Border
BorderThickness="1,1,1,1"
CornerRadius="1,1,1,1"
BorderBrush="#FF7898B5" Width="9"
Height="9" SnapsToDevicePixels="True">
                            <Border.Background>
                                <LinearGradientBrush
StartPoint="0,0" EndPoint="1,1">

<LinearGradientBrush.GradientStops>
                                    <GradientStop
Color="#FFFFFFFF" Offset="0.2" />
                                    <GradientStop
Color="#FFC0B7A6" Offset="1" />

</LinearGradientBrush.GradientStops>
                                </LinearGradientBrush>
                            </Border.Background>
                            <Path Data="M0,2L0,3 2,3 2,5
3,5 3,3 5,3 5,2 3,2 3,0 2,0 2,2z"
Fill="#FF000000" Name="ExpandPath"
Margin="1,1,1,1" />
                        </Border>
                    </Border>
                    <ControlTemplate.Triggers>
                        <Trigger
Property="ToggleButton.IsChecked"
Value="True">
                            <Setter Property="Path.Data"
TargetName="ExpandPath">
                                <Setter.Value>

<StreamGeometry>M0,2L0,3 5,3
5,2z</StreamGeometry>
                                </Setter.Value>
                            </Setter>
                        </Trigger>
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
```

```
   <tree:LevelToIndentConverter
x:Key="LevelToIndentConverter"/>
   <tree:CanExpandConverter
x:Key="CanExpandConverter"/>

   <Style TargetType="{x:Type
tree:RowExpander}">
      <Setter Property="Focusable"
Value="False"/>
      <Setter Property="Template">
         <Setter.Value>
            <ControlTemplate
TargetType="{x:Type tree:RowExpander}">
               <ToggleButton x:Name="Expander"

                  Style="{StaticResource
ExpandCollapseToggleStyle}"

                  Margin="{Binding Node.Level,
Converter={StaticResource
LevelToIndentConverter},RelativeSource={Rela
tiveSource AncestorType={x:Type
tree:TreeListItem}}}"

                  IsChecked="{Binding
Node.IsExpanded, Mode=TwoWay,
RelativeSource={RelativeSource
AncestorType={x:Type tree:TreeListItem}}}"

                  Visibility="{Binding
Node.IsExpandable, Converter={StaticResource
CanExpandConverter},RelativeSource={Relativ
eSource AncestorType={x:Type
tree:TreeListItem}}}"

                  ClickMode="Press"/>
            </ControlTemplate>
         </Setter.Value>
      </Setter>
   </Style>
</ResourceDictionary>
```

1.7. MetricsTreeModel.cs

```
using System;
using System.Collections;
using FractalGraph.Abstractions;
using MetricsUI.Tree;

namespace MetricsUI
{
   internal class MetricsTreeModel : ITreeModel
   {
      MetricsNode _metricsRoot;

      public MetricsTreeModel(MetricsNode
metricsRoot)
      {
         if (metricsRoot == null)
            throw new ArgumentException();

         _metricsRoot = metricsRoot;
      }

      public IEnumerable GetChildren(object
parent)
      {
         if (parent == null)
            parent = _metricsRoot;

         return (parent as MetricsNode).Children;
      }

      public bool HasChildren(object parent)
      {
         return (parent as
MetricsNode).Children.Count > 0;
      }
   }
}
```

1.8. MetricsViewModel.cs

```
using System;
using System.Collections;
using FractalGraph.Abstractions;
using MetricsUI.Tree;

namespace MetricsUI
{
   internal class MetricsTreeModel : ITreeModel
   {
      MetricsNode _metricsRoot;

      public MetricsTreeModel(MetricsNode
metricsRoot)
      {
         if (metricsRoot == null)
```

```
            throw new ArgumentException();

        _metricsRoot = metricsRoot;
      }

      public IEnumerable GetChildren(object
parent)
      {
         if (parent == null)
            parent = _metricsRoot;

         return (parent as MetricsNode).Children;
      }

      public bool HasChildren(object parent)
      {
         return (parent as
MetricsNode).Children.Count > 0;
      }
   }
}
```

1.9. Converters.cs

```
using System;
using System.Globalization;
using System.Windows.Data;
using System.Windows;

namespace MetricsUI.Tree
{
   internal class LevelToIndentConverter :
IValueConverter
   {
      private const double IndentSize = 19.0;

      public object Convert(object o, Type type,
object parameter, CultureInfo culture)
      {
         return new Thickness((int)o * IndentSize,
0, 0, 0);
      }

      public object ConvertBack(object o, Type
type, object parameter, CultureInfo culture)
      {
         throw new NotSupportedException();
      }
   }

   internal class CanExpandConverter :
IValueConverter
   {
      public object Convert(object o, Type type,
object parameter, CultureInfo culture)
      {
         if ((bool)o)
            return Visibility.Visible;
         else
            return Visibility.Hidden;
      }

      public object ConvertBack(object o, Type
type, object parameter, CultureInfo culture)
      {
         throw new NotSupportedException();
      }
   }
}
```

1.10. ITreeModel.cs

```
using System.Collections;

namespace MetricsUI.Tree
{
   public interface ITreeModel
   {
      IEnumerable GetChildren(object parent);

      bool HasChildren(object parent);
   }
}
```

1.11. ObservableCollectionAdv.cs

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.ComponentModel;

namespace MetricsUI.Tree
{
   internal class ObservableCollectionAdv<T> :
ObservableCollection<T>
   {
      public void RemoveRange(int index, int
count)
      {
```

```
            this.CheckReentrancy();
            var items = this.Items as List<T>;
            items.RemoveRange(index, count);
            OnReset();
        }

        public void InsertRange(int index,
IEnumerable<T> collection)
        {
            this.CheckReentrancy();
            var items = this.Items as List<T>;
            items.InsertRange(index, collection);
            OnReset();
        }

        private void OnReset()
        {
            OnPropertyChanged("Count");
            OnPropertyChanged("Item[]");
            OnCollectionChanged(new
NotifyCollectionChangedEventArgs(

NotifyCollectionChangedAction.Reset));
        }

        private void OnPropertyChanged(string
propertyName)
        {
            OnPropertyChanged(new
PropertyChangedEventArgs(propertyName));
        }
    }
}
```

1.12. RowExpander.cs

```
using System.Windows.Controls;
using System.Windows;

namespace MetricsUI.Tree
{
    public class RowExpander : Control
    {
        static RowExpander()
        {

DefaultStyleKeyProperty.OverrideMetadata(typ
eof(RowExpander), new
FrameworkPropertyMetadata(typeof(RowExpan
der)));
        }
    }
}
```

1.13. TreeList.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.Linq;
using System.Windows.Controls.Primitives;
using System.Windows.Controls;
using System.Windows;

namespace MetricsUI.Tree
{
    public class TreeList : ListView
    {
        #region Properties

        /// <summary>
        /// Internal collection of rows representing
visible nodes, actually displayed in the ListView
        /// </summary>
        internal
ObservableCollectionAdv<TreeNode> Rows
        {
            get;
            private set;
        }


        private ITreeModel _model;
        public ITreeModel Model
        {
            get { return _model; }
            set
            {
                if (_model != value)
                {
                    _model = value;
                    _root.Children.Clear();
                    Rows.Clear();
                    CreateChildrenNodes(_root);
                }
```

```
        }
    }

    public void Invalidate()
    {
        _root.Children.Clear();
        Rows.Clear();
        CreateChildrenNodes(_root);
    }

    private TreeNode _root;
    internal TreeNode Root
    {
        get { return _root; }
    }

    public ReadOnlyCollection<TreeNode>
Nodes
    {
        get { return Root.Nodes; }
    }

    internal TreeNode PendingFocusNode
    {
        get;
        set;
    }

    public ICollection<TreeNode>
SelectedNodes
    {
        get
        {
            return
SelectedItems.Cast<TreeNode>().ToArray();
        }
    }

    public TreeNode SelectedNode
    {
        get
        {
            if (SelectedItems.Count > 0)
                return SelectedItems[0] as
TreeNode;
            else
                return null;
        }
    }

    #endregion

    public TreeList()
    {
        Rows = new
ObservableCollectionAdv<TreeNode>();
        _root = new TreeNode(this, null);
        _root.IsExpanded = true;
        ItemsSource = Rows;
        ItemContainerGenerator.StatusChanged
+= ItemContainerGeneratorStatusChanged;
    }

    void
ItemContainerGeneratorStatusChanged(object
sender, EventArgs e)
    {
        if (ItemContainerGenerator.Status ==
GeneratorStatus.ContainersGenerated &&
PendingFocusNode != null)
        {
            var item =
ItemContainerGenerator.ContainerFromItem(Pe
ndingFocusNode) as TreeListItem;
            if (item != null)
                item.Focus();
            PendingFocusNode = null;
        }
    }

    protected override DependencyObject
GetContainerForItemOverride()
    {
        return new TreeListItem();
    }

    protected override bool
IsItemItsOwnContainerOverride(object item)
    {
        return item is TreeListItem;
    }

    protected override void
PrepareContainerForItemOverride(Dependency
Object element, object item)
    {
        var ti = element as TreeListItem;
        var node = item as TreeNode;
        if (ti != null && node != null)
```

```
        {
            ti.Node = item as TreeNode;

base.PrepareContainerForItemOverride(element,
node.Tag);
        }
    }

    internal void SetIsExpanded(TreeNode
node, bool value)
    {
        if (value)
        {
            if (!node.IsExpandedOnce)
            {
                node.IsExpandedOnce = true;
                node.AssignIsExpanded(value);
                CreateChildrenNodes(node);
            }
            else
            {
                node.AssignIsExpanded(value);
                CreateChildrenRows(node);
            }
        }
        else
        {
            DropChildrenRows(node, false);
            node.AssignIsExpanded(value);
        }
    }

    internal void
CreateChildrenNodes(TreeNode node)
    {
        var children = GetChildren(node);
        if (children != null)
        {
            int rowIndex = Rows.IndexOf(node);
            node.ChildrenSource = children as
INotifyCollectionChanged;
            foreach (object obj in children)
            {
                TreeNode child = new
TreeNode(this, obj);
                child.HasChildren =
HasChildren(child);
                node.Children.Add(child);
            }
            Rows.InsertRange(rowIndex + 1,
node.Children.ToArray());
        }
    }

    private void CreateChildrenRows(TreeNode
node)
    {
        int index = Rows.IndexOf(node);
        if (index >= 0 || node == _root) // ignore
invisible nodes
        {
            var nodes =
node.AllVisibleChildren.ToArray();
            Rows.InsertRange(index + 1, nodes);
        }
    }

    internal void DropChildrenRows(TreeNode
node, bool removeParent)
    {
        int start = Rows.IndexOf(node);
        if (start >= 0 || node == _root) // ignore
invisible nodes
        {
            int count = node.VisibleChildrenCount;
            if (removeParent)
                count++;
            else
                start++;
            Rows.RemoveRange(start, count);
        }
    }

    private IEnumerable GetChildren(TreeNode
parent)
    {
        if (Model != null)
            return Model.GetChildren(parent.Tag);
        else
            return null;
    }

    private bool HasChildren(TreeNode parent)
    {
        if (parent == Root)
            return true;
        else if (Model != null)
            return Model.HasChildren(parent.Tag);
```

```
            else
               return false;
         }

         internal void InsertNewNode(TreeNode
parent, object tag, int rowIndex, int index)
         {
            TreeNode node = new TreeNode(this,
tag);
            if (index >= 0 && index <
parent.Children.Count)
               parent.Children.Insert(index, node);
            else
            {
               index = parent.Children.Count;
               parent.Children.Add(node);
            }
            Rows.Insert(rowIndex + index + 1, node);
         }
      }
}
```

1.14. TreeListItem.cs

```
using System.ComponentModel;
using System.Windows.Controls;
using System.Windows.Input;

namespace MetricsUI.Tree
{
   public class TreeListItem : ListViewItem,
INotifyPropertyChanged
   {
      #region Properties

      private TreeNode _node;
      public TreeNode Node
      {
         get { return _node; }
         internal set
         {
            _node = value;
            OnPropertyChanged("Node");
         }
      }

      #endregion

      public TreeListItem()
      {
      }

      protected override void
OnKeyDown(KeyEventArgs e)
      {
         if (Node != null)
         {
            switch (e.Key)
            {
               case Key.Right:
                  e.Handled = true;
                  if (!Node.IsExpanded)
                  {
                     Node.IsExpanded = true;
                     ChangeFocus(Node);
                  }
                  else if (Node.Children.Count > 0)

ChangeFocus(Node.Children[0]);
                  break;

               case Key.Left:

                  e.Handled = true;
                  if (Node.IsExpanded &&
Node.IsExpandable)
                  {
                     Node.IsExpanded = false;
                     ChangeFocus(Node);
                  }
                  else
                     ChangeFocus(Node.Parent);
                  break;

               case Key.Subtract:
                  e.Handled = true;
                  Node.IsExpanded = false;
                  ChangeFocus(Node);
                  break;

               case Key.Add:
                  e.Handled = true;
                  Node.IsExpanded = true;
                  ChangeFocus(Node);
                  break;
            }
         }
```

```
            if (!e.Handled)
                base.OnKeyDown(e);
        }

        private void ChangeFocus(TreeNode node)
        {
            var tree = node.Tree;
            if (tree != null)
            {
                var item =
tree.ItemContainerGenerator.ContainerFromItem
(node) as TreeListItem;
                if (item != null)
                    item.Focus();
                else
                    tree.PendingFocusNode = node;
            }
        }

        #region INotifyPropertyChanged Members

        public event PropertyChangedEventHandler
PropertyChanged;

        private void OnPropertyChanged(string
name)
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new
PropertyChangedEventArgs(name));
        }

        #endregion
    }
}
```

1.15. TreeNode.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Collections.Specialized;
using System.ComponentModel;
using System.Linq;

namespace MetricsUI.Tree
{
    public sealed class TreeNode :
INotifyPropertyChanged
    {
        #region NodeCollection
        private class NodeCollection :
Collection<TreeNode>
        {
            private TreeNode _owner;

            public NodeCollection(TreeNode owner)
            {
                _owner = owner;
            }

            protected override void ClearItems()
            {
                while (this.Count != 0)
                    this.RemoveAt(this.Count - 1);
            }

            protected override void InsertItem(int
index, TreeNode item)
            {
                if (item == null)
                    throw new
ArgumentNullException("item");

                if (item.Parent != _owner)
                {
                    if (item.Parent != null)

item.Parent.Children.Remove(item);
                    item._parent = _owner;
                    item._index = index;
                    for (int i = index; i < Count; i++)
                        this[i]._index++;
                    base.InsertItem(index, item);
                }
            }

            protected override void RemoveItem(int
index)
            {
                TreeNode item = this[index];
                item._parent = null;
                item._index = -1;
                for (int i = index + 1; i < Count; i++)
                    this[i]._index--;
                base.RemoveItem(index);
            }
```

```
        protected override void SetItem(int index,
TreeNode item)
        {
            if (item == null)
                throw new
ArgumentNullException("item");
            RemoveAt(index);
            InsertItem(index, item);
        }
    }
    #endregion

    #region INotifyPropertyChanged Members

    public event PropertyChangedEventHandler
PropertyChanged;
    private void OnPropertyChanged(string
name)
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new
PropertyChangedEventArgs(name));
    }

    #endregion

    #region Properties

    private TreeList _tree;
    internal TreeList Tree
    {
        get { return _tree; }
    }

    private INotifyCollectionChanged
_childrenSource;
    internal INotifyCollectionChanged
ChildrenSource
    {
        get { return _childrenSource; }
        set
        {
            if (_childrenSource != null)
                _childrenSource.CollectionChanged
-= ChildrenChanged;

            _childrenSource = value;

            if (_childrenSource != null)
                _childrenSource.CollectionChanged
+= ChildrenChanged;
        }
    }

    private int _index = -1;
    public int Index
    {
        get
        {
            return _index;
        }
    }

    /// <summary>
    /// Returns true if all parent nodes of this
node are expanded.
    /// </summary>
    internal bool IsVisible
    {
        get
        {
            TreeNode node = _parent;
            while (node != null)
            {
                if (!node.IsExpanded)
                    return false;
                node = node.Parent;
            }
            return true;
        }
    }

    public bool IsExpandedOnce
    {
        get;
        internal set;
    }

    public bool HasChildren
    {
        get;
        internal set;
    }

    private bool _isExpanded;
    public bool IsExpanded
    {
        get { return _isExpanded; }
```

```
        set
        {
            if (value != IsExpanded)
            {
                Tree.SetIsExpanded(this, value);
                OnPropertyChanged("IsExpanded");

OnPropertyChanged("IsExpandable");
            }
        }
    }

    internal void AssignIsExpanded(bool value)
    {
        _isExpanded = value;
    }

    public bool IsExpandable
    {
        get
        {
            return (HasChildren &&
!IsExpandedOnce) || Nodes.Count > 0;
        }
    }

    private bool _isSelected;
    public bool IsSelected
    {
        get { return _isSelected; }
        set
        {
            if (value != _isSelected)
            {
                _isSelected = value;
                OnPropertyChanged("IsSelected");
            }
        }
    }


    private TreeNode _parent;
    public TreeNode Parent
    {
        get { return _parent; }
    }

    public int Level
    {
        get
        {
            if (_parent == null)
                return -1;
            else
                return _parent.Level + 1;
        }
    }

    public TreeNode PreviousNode
    {
        get
        {
            if (_parent != null)
            {
                int index = Index;
                if (index > 0)
                    return _parent.Nodes[index - 1];
            }
            return null;
        }
    }

    public TreeNode NextNode
    {
        get
        {
            if (_parent != null)
            {
                int index = Index;
                if (index < _parent.Nodes.Count - 1)
                    return _parent.Nodes[index + 1];
            }
            return null;
        }
    }

    internal TreeNode BottomNode
    {
        get
        {
            TreeNode parent = this.Parent;
            if (parent != null)
            {
                if (parent.NextNode != null)
                    return parent.NextNode;
                else
                    return parent.BottomNode;
            }
```

```
            return null;
        }
    }

    internal TreeNode NextVisibleNode
    {
        get
        {
            if (IsExpanded && Nodes.Count > 0)
                return Nodes[0];
            else
            {
                TreeNode nn = NextNode;
                if (nn != null)
                    return nn;
                else
                    return BottomNode;
            }
        }
    }

    public int VisibleChildrenCount
    {
        get
        {
            return AllVisibleChildren.Count();
        }
    }

    public IEnumerable<TreeNode>
AllVisibleChildren
    {
        get
        {
            int level = this.Level;
            TreeNode node = this;
            while (true)
            {
                node = node.NextVisibleNode;
                if (node != null && node.Level >
level)
                    yield return node;
                else
                    break;
            }
        }
    }

    private object _tag;

    public object Tag
    {
        get { return _tag; }
    }

    private Collection<TreeNode> _children;
    internal Collection<TreeNode> Children
    {
        get { return _children; }
    }

    private ReadOnlyCollection<TreeNode>
_nodes;
    public ReadOnlyCollection<TreeNode>
Nodes
    {
        get { return _nodes; }
    }

    #endregion

    internal TreeNode(TreeList tree, object tag)
    {
        if (tree == null)
            throw new
ArgumentNullException("tree");

        _tree = tree;
        _children = new NodeCollection(this);
        _nodes = new
ReadOnlyCollection<TreeNode>(_children);
        _tag = tag;
    }

    public override string ToString()
    {
        if (Tag != null)
            return Tag.ToString();
        else
            return base.ToString();
    }

    void ChildrenChanged(object sender,
NotifyCollectionChangedEventArgs e)
    {
        switch (e.Action)
        {
            case
NotifyCollectionChangedAction.Add:
```

```
            if (e.NewItems != null)
            {
               int index = e.NewStartingIndex;
               int rowIndex =
Tree.Rows.IndexOf(this);
               foreach (object obj in e.NewItems)
               {
                  Tree.InsertNewNode(this, obj,
rowIndex, index);
                  index++;
               }
            }
            break;

         case
NotifyCollectionChangedAction.Remove:
            if (Children.Count >
e.OldStartingIndex)

RemoveChildAt(e.OldStartingIndex);
            break;

         case
NotifyCollectionChangedAction.Move:
         case
NotifyCollectionChangedAction.Replace:
         case
NotifyCollectionChangedAction.Reset:
            while (Children.Count > 0)
               RemoveChildAt(0);
            Tree.CreateChildrenNodes(this);
            break;
      }
      HasChildren = Children.Count > 0;
      OnPropertyChanged("IsExpandable");
   }

   private void RemoveChildAt(int index)
   {
      var child = Children[index];
      Tree.DropChildrenRows(child, true);
      ClearChildrenSource(child);
      Children.RemoveAt(index);
   }

   private void ClearChildrenSource(TreeNode
node)
   {
      node.ChildrenSource = null;
      foreach (var n in node.Children)
         ClearChildrenSource(n);
   }
  }
}
```

1.16. Bar.cs

```
namespace FractalGraph.Abstractions
{
  public class Bar : Stiffness
  {
    public double E { get;set; }
    public double V { get;set; }
    public double B { get;set; }
    public double H { get;set; }
    public double Ro { get;set; }

    public override int LiraType => 0;

    public override Stiffness Clone()
    {
      return (Stiffness)MemberwiseClone();
    }

    public override string ToLiraParameter()
    {
      return $"Ro:{Ro} E:{E} B:{B} H:{H}
NGrndPar:0 GF:0 nFlags:0 Length:0 B_:0 H_:0
NEL:0 UseRbNel:0 BAR_END Mu:{V} EF:0
EIy:0 EIz:0 GIk:0 GFz:0 GFy:0
IsRigParamChanges:0 IsSavedAsKoef:0
IsNormSect:1 IsIter:0 Uli:0 STD_END";
    }
  }
}
```

1.17. Constructor.cs

```
using System;
using System.Linq;
using System.Collections.Generic;
using FractalGraph.Abstractions.Extensions;
using MetricsService;

namespace FractalGraph.Abstractions
{
  public interface IConstructor
  {
    IGraph Graph { get; set; }
```

```
    IGraph TestGraph { get; set; }
    IRule CurrentRule { get; set; }

    IReadOnlyCollection<IEdge> AddedEdges
{ get; }
    IReadOnlyCollection<INode> AddedNodes
{ get; }


    void Invalidate();
    void PerformSubstitution();
    void Clear();
    void Repaint();
    void ApplyStiffness(Stiffness stiffness);
    void FixBottomNodes();
    void SetMetrics(IMetricsService
metricsService);

    event
EventHandler<SubstitutionEventArgs>
Substituted;
    event
EventHandler<IReadOnlyCollection<INode>>
Repainted;
  }

  public class SubstitutionEventArgs :
EventArgs
  {
    public IReadOnlyCollection<IEdge>
AddedEdges { get; }
    public IReadOnlyCollection<INode>
AddedNodes { get; }

    public
SubstitutionEventArgs(IReadOnlyCollection<IE
dge> addedEdges,
IReadOnlyCollection<INode> addedNodes)
    {
      AddedEdges = addedEdges;
      AddedNodes = addedNodes;
    }
  }

  public class Constructor : IConstructor
  {
    List<IEdge> _addedEdges = new
List<IEdge>();
    List<INode> _addedNodes = new
List<INode>();
    List<INode> _repaintedVetrices = new
List<INode>();
    IGraph _graph;
    IMetricsService _metrics;

    public Constructor(IGraph graph)
    {
      _graph = graph;
    }

    public void SetMetrics(IMetricsService
metricsService)
    {
      _metrics = metricsService;
    }

    public IGraph Graph { get => _graph; set
=> _graph = value; }
    public IGraph TestGraph { get; set; }

    public IRule CurrentRule { get; set; }

    public IReadOnlyCollection<IEdge>
AddedEdges => _addedEdges.AsReadOnly();
    public IReadOnlyCollection<INode>
AddedNodes => _addedNodes.AsReadOnly();

    public event
EventHandler<SubstitutionEventArgs>
Substituted;
    public event
EventHandler<IReadOnlyCollection<INode>>
Repainted;

    public void Invalidate()
    {
      _graph = CurrentRule.Left.Clone();
      _addedEdges.Clear();
      _addedNodes.Clear();
    }

    public void Clear()
    {
      _graph = new Graph();
      _addedEdges.Clear();
      _addedNodes.Clear();
    }
```

```
    public void Repaint()
    {
        Repaint(Graph.Nodes);
    }

    private void
Repaint(IReadOnlyList<INode> nodes)
    {
        _metrics.MethodStart(nameof(Repaint));

        _metrics.Assignment();
        _metrics.Comparison();
        if (!(0 < nodes.Count))
        {
            _metrics.Jump();
        }

        for (int i = 0; i < nodes.Count; i++)
        {
            _metrics.Assignment();
            var node = nodes[i];

            _metrics.Assignment();
            bool condition = node.Color ==
CurrentRule.MainLeftColor;

            _metrics.Comparison();
            if (condition)
            {
                _metrics.Assignment();
                node.Color =
CurrentRule.MainRightColor;
                _repaintedVetrices.Add(node);
            }
            else
            {
                _metrics.Jump();

                _metrics.Assignment();
                condition = node.Color ==
CurrentRule.MainRightColor;
                _metrics.Comparison();
                if (condition)
                {
                    _metrics.Assignment();
                    node.Color =
CurrentRule.MainLeftColor;
                    _repaintedVetrices.Add(node);
                }
            }

        }

        Repainted(this, _repaintedVetrices);
        _metrics.MethodEnd(nameof(Repaint));
    }

    public void ApplyStiffness(Stiffness
stiffness)
    {
        foreach(var edge in _addedEdges)
            if(edge.Stiffness == null)
                edge.Stiffness = stiffness.Clone();
    }

    public void FixBottomNodes()
    {
        double lowest = double.PositiveInfinity;
        foreach(var v in _graph.Nodes)
        {
            if(v.Position.Z < lowest)
                lowest = v.Position.Z;
        }

        var epsilon = 0.000001;
        lowest += epsilon;

        foreach (var v in _graph.Nodes)
        {
            if (v.Position.Z < lowest)
                v.Restraints.SetAll(true);
        }
    }

    public void PerformSubstitution()
    {

_metrics.MethodStart(nameof(PerformSubstituti
on));

        _addedEdges.Clear();
        _addedNodes.Clear();

        _metrics.Assignment();
        _metrics.Comparison();
        if (!(0 < Graph.Nodes.Count))
        {
```

```
            _metrics.Jump();
        }

        for (int i = 0; i < Graph.Nodes.Count;
i++)
        {
            _metrics.Assignment();
            var node = Graph.Nodes[i];

            _metrics.Assignment();
            bool condition = node.Color ==
CurrentRule.MainLeftColor;

            _metrics.Comparison();
            if (!condition)
            {
                _metrics.Jump();
                continue;
            }

            SingleSubstitution(node, out var edges,
out var nodes);

            _metrics.Assignment();
            condition = !edges.IsNullOrEmpty();
            _metrics.Comparison();
            if (condition)
            {
                _metrics.Assignment();
                _metrics.Comparison();
                if (!(0 < edges.Count))
                {
                    _metrics.Jump();
                }

                for (int j = 0; j < edges.Count; j++)
                {
                    _metrics.Assignment();

                    _metrics.Assignment();
                    condition =
Graph.TryAdd(edges[j]);

                    _metrics.Comparison();
                    if (condition)
                    {
                        _addedEdges.Add(edges[j]);
                    }

                    //j++
                    _metrics.Arithmetic();

                    _metrics.Assignment();
                    condition = j < edges.Count;

                    _metrics.Comparison();
                    if (condition)
                    {
                        _metrics.Jump();
                    }
                }
            }

            _metrics.Assignment();
            condition = !nodes.IsNullOrEmpty();
            _metrics.Comparison();
            if (condition)
            {
                _metrics.Assignment();
                _metrics.Comparison();
                if (!(0 < nodes.Count))
                {
                    _metrics.Jump();
                }

                for (int j = 0; j < nodes.Count; j++)
                {
                    _metrics.Assignment();

                    _metrics.Assignment();
                    condition =
Graph.TryAdd(nodes[j]);

                    _metrics.Comparison();
                    if (condition)
                    {
                        _addedNodes.Add(nodes[j]);
                    }

                    //j++
                    _metrics.Arithmetic();

                    _metrics.Assignment();
                    condition = j < nodes.Count;

                    _metrics.Comparison();
                    if (condition)
                    {
```

```
                    _metrics.Jump();
                }
            }
        }


        //i++
        _metrics.Arithmetic();

        _metrics.Assignment();
        condition = i < Graph.Nodes.Count;

        _metrics.Comparison();
        if (condition)
        {
            _metrics.Jump();
        }
    }

    Repaint();
    Substituted?.Invoke(this, new
SubstitutionEventArgs(_addedEdges,
_addedNodes));


_metrics.MethodEnd(nameof(PerformSubstitutio
n));
}


private void SingleSubstitution(INode node,
out List<IEdge> edges, out List<INode> nodes)
{
    INode mainRightNode = null;
    var pos = node.Position;
    var rightNodes =
CurrentRule.Right.Nodes.ToList();
    var rightEdges =
CurrentRule.Right.Edges.ToList();

    for (int i = 0; i < rightNodes.Count; i++)
    {
        var n = rightNodes[i];
        if (n.Color ==
CurrentRule.MainLeftColor)
        {
            mainRightNode = n;
            break;
        }
    }

    if (mainRightNode is null)
    {
        edges = null;
        nodes = null;
        return;
    }

    var newEdges = new List<IEdge>();
    var newNode = new List<INode>();

    for (int i = 0; i < rightNodes.Count; i++)
    {
        var n = rightNodes[i];
        var distance = n.Position -
mainRightNode.Position;

        var v = new Node();

        v.Position = (pos + distance).Round(3);
        v.Color = n.Color;
        v.Size = n.Size;

        newNode.Add(v);
    }

    for (int i = 0; i < rightEdges.Count; i++)
    {
        var edge = rightEdges[i];

        var distanceStart = edge.Start.Position -
mainRightNode.Position;
        var distanceEnd = edge.End.Position -
mainRightNode.Position;

        var newStart = (pos +
distanceStart).Round(3);
        var newEnd = (pos +
distanceEnd).Round(3);

        //looking for start pos
        INode start = null;

        for (int j = 0; j < newNode.Count; j++)
        {
            var n = newNode[j];

            if (newStart == n.Position)
```

```
            {
               start = n;
               break;
            }
         }


         //looking for end pos
         INode end = null;
         for (int j = 0; j < newNode.Count; j++)
         {
            var n = newNode[j];

            if (newEnd == n.Position)
            {
               end = n;
               break;
            }

         }

         newEdges.Add(new Edge(start, end)
         {
            Size = edge.Size
         });
      }

      edges = newEdges;
      nodes = newNode;
   }

  }
}
```

1.18. Edge.cs

```
using SharpDX;

namespace FractalGraph.Abstractions
{
  public interface IEdge : ITerminalEntity
  {
    INode Start { get; set; }
    INode End { get; set; }
    float Size { get; set; }
    Stiffness Stiffness { get; set; }
    Vector3 Direction { get; }
  }

  public class Edge : IEdge
  {
    public Edge(INode start, INode end)
    {
      Start = start;
      End = end;
    }

    public INode Start { get; set; }
    public INode End { get; set; }
    public float Size { get; set; } =
Defaults.DefaultEdgeSize;
    public Stiffness Stiffness { get; set; } = null;
    public Vector3 Direction => Start.Position -
End.Position;

  }
}
```

1.19. Graph.cs

```
using System.Linq;
using System.Collections.Generic;
using System.Xml.Serialization;
using SharpDX;
using FractalGraph.Abstractions.Extensions;

namespace FractalGraph.Abstractions
{
  public interface IGraph
  {
    IReadOnlyList<INode> Nodes { get; }
    IReadOnlyList<IEdge> Edges { get; }
    bool TryAdd(IEdge edge);
    bool TryAdd(INode node);
    void UnsafeAdd(params INode[] nodes);
    void UnsafeAdd(params IEdge[] edge);
    void UnsafeAdd(IEnumerable<INode>
nodes);
    void UnsafeAdd(IEnumerable<IEdge>
edge);
    IGraph Clone();
    void Reindex();
    int EdgesCount();
    int NodesCount();
  }

  public class Graph : IGraph
  {
```

```
    Dictionary<Vector3, INode> _nodes = new
Dictionary<Vector3, INode>();

    Dictionary<(Vector3, Vector3), IEdge>
_edges = new Dictionary<(Vector3, Vector3),
IEdge>();

    [XmlAttribute]
    public IReadOnlyList<INode> Nodes =>
_nodes.Values.ToList();
    [XmlAttribute]
    public IReadOnlyList<IEdge> Edges =>
_edges.Values.ToList();

    public bool TryAdd(IEdge edge)
    {
      if
(_edges.ContainsKey((edge.Start.Position,
edge.End.Position)) ||

_edges.ContainsKey((edge.End.Position,
edge.Start.Position)))
        return false;

      _edges.Add((edge.Start.Position,
edge.End.Position), edge);
      return true;
    }

    public bool TryAdd(INode node)
    {
      if (_nodes.ContainsKey(node.Position))
        return false;

      _nodes.Add(node.Position, node);
      return true;
    }

    public void UnsafeAdd(params INode[]
nodes)
    {
      foreach (var node in nodes)
        _nodes.Add(node.Position, node);
    }

    public void UnsafeAdd(params IEdge[]
edge)
    {
      foreach (var e in edge)
        _edges.Add((e.Start.Position,
e.End.Position), e);
    }

    public void
UnsafeAdd(IEnumerable<INode> nodes)
    {
      foreach (var node in nodes)
        _nodes.Add(node.Position, node);
    }

    public void
UnsafeAdd(IEnumerable<IEdge> edge)
    {
      foreach (var e in edge)
        _edges.Add((e.Start.Position,
e.End.Position), e);
    }


    public static IGraph
SingleNodeGraph(Color4 color, float size =
Defaults.DefaultNodeSize)
    {
      var graph = new Graph();
      graph.TryAdd(new Node() { Color =
color, Size = size });
      return graph;
    }

    public IGraph Clone()
    {
      var originRefCloned = new
HashSet<INode>();
      var clonedNodes = new
HashSet<INode>();
      var clonedEdges = new List<Edge>();
      foreach (var edge in Edges)
      {
        var startClone = edge.Start.Clone();
        var endClone = edge.End.Clone();
        originRefCloned.TryAdd(edge.Start);
        originRefCloned.TryAdd(edge.End);

        var edgeClone = new Edge(startClone,
endClone);

        clonedEdges.Add(edgeClone);
        clonedNodes.TryAdd(startClone);
```

```
            clonedNodes.TryAdd(endClone);
        }

        var nodesToClone =
Nodes.Except(originRefCloned);
        var nodesUnclonedClones =
nodesToClone.Select(node => node.Clone());

        var graph = new Graph();
        graph.UnsafeAdd(clonedNodes);

graph.UnsafeAdd(nodesUnclonedClones);
        graph.UnsafeAdd(clonedEdges);
        return graph;
    }

    //set right Ids
    public void Reindex()
    {
        var indexed = new Dictionary<Vector3,
int>(_nodes.Count);
        int i = 1;
        foreach (var node in _nodes)
        {
            node.Value.Id = i;
            indexed[node.Key] = i;
            i++;
        }

        foreach (var edge in _edges.Values)
        {
            edge.Start.Id =
indexed[edge.Start.Position];
            edge.End.Id =
indexed[edge.End.Position];
        }
    }

    public int EdgesCount() => _edges.Count;

    public int NodesCount() => _nodes.Count;
  }
}
```

1.20. ITerminalEntity.cs

```
namespace FractalGraph.Abstractions
{
  public interface ITerminalEntity
  {
  }
}
```

1.21. MetricsNode.cs

```
using System.Collections.ObjectModel;

namespace FractalGraph.Abstractions
{
  public class MetricsNode
  {
    private readonly
ObservableCollection<MetricsNode> _children
= new ObservableCollection<MetricsNode>();

    public
ObservableCollection<MetricsNode> Children
    {
      get { return _children; }
    }

    public string Name { get; set; }
    public int Arithmetics { get; set; }
    public int Assignments { get; set; }
    public int Comparisons { get; set; }
    public int Jumps { get; set; }
  }
}
```

1.22. Node.cs

```
using SharpDX;

namespace FractalGraph.Abstractions
{
  public interface INode : ITerminalEntity
  {
    int Id { get; set; }
    float Size { get; set; }
    Vector3 Position { get; set; }
    Color4 Color { get; set; }
    INode Clone();
    Restraints Restraints { get; }
  }

  public class Node : INode
  {
    public int Id { get; set; }
```

```
        public float Size { get; set; } =
Defaults.DefaultNodeSize;
        public Vector3 Position { get; set; }
        public Color4 Color { get; set; }

        public INode Clone()
        {
            return new Node()
            {
                Id = this.Id,
                Color = this.Color,
                Size = this.Size,
                Position = this.Position,
                Restraints = this.Restraints.Clone()
            };
        }

        public Restraints Restraints { get; private
set; } = new Restraints();
    }
}
```

1.23. Restraints.cs

```
namespace FractalGraph.Abstractions
{
    public class Restraints
    {
        public bool X { get; set; }
        public bool Y { get; set; }
        public bool Z { get; set; }
        public bool UX { get; set; }
        public bool UY { get; set; }
        public bool UZ { get; set; }
        public bool W { get; set; }

        public void SetAll(bool value)
        {
            X = value;
            Y = value;
            Z = value;
            UX = value;
            UY = value;
            UZ = value;
            W = value;
        }

        public Restraints Clone()
        {
            return (Restraints)MemberwiseClone();
        }
    }
}
```

1.24. Rule.cs

```
using System;
using System.Linq;
using SharpDX;
using FractalGraph.Abstractions.Extensions;

namespace FractalGraph.Abstractions
{
    public interface IRule
    {
        IGraph Left { get; }
        IGraph Right { get; }

        Color4 MainLeftColor { get; }
        Color4 MainRightColor { get; }

        void SetLeftPart(IGraph left, Color4
mainLeftColor);
        void SetRightPart(IGraph right, Color4
mainRigthColor);

        void RepaintLeftMain(Color4 color);
        void RepaintRightMain(Color4 color);

        void SetLeftColor(Color4 color);
        void SetRightColor(Color4 color);

        event EventHandler<RuleEventArgs>
LeftPartSet;
        event EventHandler<RuleEventArgs>
RightPartSet;
    }

    public class RuleEventArgs : EventArgs
    {
        Color4 Color { get; }
        IGraph Graph { get; set; }
        public RuleEventArgs(IGraph graph,
Color4 color)
        {
            Graph = graph;
            Color = color;
        }
```

```
    }

    public class Rule : IRule
    {
        public IGraph Left { get; private set; }
        public IGraph Right { get; private set; }

        public Color4 MainLeftColor { get; private
set; }
        public Color4 MainRightColor { get;
private set; }

        public event
EventHandler<RuleEventArgs> RightPartSet;
        public event
EventHandler<RuleEventArgs> LeftPartSet;
        public event
EventHandler<RuleEventArgs>
LeftPartRepainted;
        public event
EventHandler<RuleEventArgs>
RightPartRepainted;

        public static RuleErrorCode
TryCreateRule(IGraph left, IGraph right, Color4
mainLeftColor, Color4 mainRigthColor, out
Rule rule)
        {
            var colorOccurrence =
left.Nodes.FirstOrDefault(node => node.Color
== mainLeftColor);
            if (colorOccurrence is null)
            {
                rule = null;
                return
RuleErrorCode.WrongLeftColor;
            }

            colorOccurrence =
right.Nodes.FirstOrDefault(node => node.Color
== mainRigthColor);
            if (colorOccurrence is null)
            {
                rule = null;
                return
RuleErrorCode.WrongRightColor;
            }

            rule = new Rule(left, right,
mainLeftColor, mainRigthColor);
            return RuleErrorCode.Succeed;
        }

        public void SetLeftColor(Color4 color) =>
MainLeftColor = color;
        public void SetRightColor(Color4 color) =>
MainRightColor = color;

        //unsafe
        public void SetRightPart(IGraph right,
Color4 mainRigthColor)
        {
            Right = right;
            MainRightColor = mainRigthColor;
            RightPartSet?.Invoke(this, new
RuleEventArgs(right, mainRigthColor));
        }

        //unsafe
        public void SetLeftPart(IGraph left, Color4
mainLeftColor)
        {
            Left = left;
            MainLeftColor = mainLeftColor;
            LeftPartSet?.Invoke(this, new
RuleEventArgs(left, mainLeftColor));
        }

        public void RepaintLeftMain(Color4 color)
        {
            foreach (var node in Left.Nodes)
                if (node.Color == MainLeftColor)
                    node.Color = color;

            foreach (var node in Right.Nodes)
                if (node.Color == MainLeftColor)
                    node.Color = color;

            MainLeftColor = color;
        }

        public void RepaintRightMain(Color4
color)
        {
            foreach (var node in Right.Nodes)
                if (node.Color == MainRightColor)
                    node.Color = color;
```

```
            foreach (var node in Left.Nodes)
                if (node.Color == MainRightColor)
                    node.Color = color;

            MainRightColor = color;
        }

        private Rule(IGraph left, IGraph right,
Color4 mainColorLeft, Color4 rightColorLeft)
        {
            Left = left;
            Right = right;
            MainLeftColor = mainColorLeft;
            MainRightColor = rightColorLeft;
        }
    }
}
```

1.25. SectionI.cs

```
namespace FractalGraph.Abstractions
{
    public class SectionI : Stiffness
    {
        public double E { get; set; }
        public double V { get; set; }
        public double B { get; set; }
        public double H { get; set; }
        public double B1 { get; set; }
        public double H1 { get; set; }
        public double B2 { get; set; }
        public double H2 { get; set; }
        public double Ro { get; set; }

        public override int LiraType => 3;

        public override Stiffness Clone()
        {
            return (Stiffness)MemberwiseClone();
        }

        public override string ToLiraParameter()
        {
            return $"Ro:{Ro} E:{E} B:{B} H:{H}
NGrndPar:0 GF:0 nFlags:0 Length:0 B1:{B1}
H1:{H1} H2:{H2} B2:{B2} B_:0 H_:0 B1_:0
H1_:0 B2_:0 H2_:0 NEL:0 CDT_END Mu:{V}
EF:0 EIy:0 EIz:0 GIk:0 GFz:0 GFy:0
IsRigParamChanges:0 IsSavedAsKoef:0
IsNormSect:1 IsIter:0 Uli:0 STD_END";
        }
    }
}
```

1.26. Stifness.cs

```
namespace FractalGraph.Abstractions
{
    public abstract class Stiffness
    {
        public abstract Stiffness Clone();
        public abstract string ToLiraParameter();
        public abstract int LiraType { get; }
    }
}
```

1.27. UIEdge.cs

```
namespace FractalGraph.Abstractions
{
    public class UIEdge : UITerminalEntity
    {
        public UIEdge(IEdge edge)
            : base(edge)
        {
        }

        public IEdge Edge => (IEdge)_terminal;
    }
}
```

1.28. UINode.cs

```
using SharpDX;
using HelixToolkit.Wpf.SharpDX;

namespace FractalGraph.Abstractions
{
    public class UINode : UITerminalEntity
    {
        public UINode(INode node)
            : base(node)
        {
        }

        public INode Node => (INode)_terminal;

        public void SetColor(Color4 color)
```

```
        {
            Node.Color = color;
            Material = new DiffuseMaterial() {
DiffuseColor = color };
        }
    }
}
```

1.29. UITerminalEntity.cs

```
using HelixToolkit.Wpf.SharpDX.Model.Scene;

namespace FractalGraph.Abstractions
{
    public class UITerminalEntity : MeshNode
    {
        protected ITerminalEntity _terminal;

        public UITerminalEntity(ITerminalEntity
terminal)
        {
            _terminal = terminal;
        }

        public ITerminalEntity Terminal =>
_terminal;
    }
}
```

1.30. CameraType.cs

```
namespace
FractalGraph.Abstractions.Extensions
{
    public enum CameraType
    {
        Orthographic,
        Perspective
    }
}
```

1.31. CollectionExtensions.cs

```
using System.Linq;
using System.Collections.Generic;

namespace
FractalGraph.Abstractions.Extensions
{
    public static class CollectionExtensions
    {
        public static bool IsNullOrEmpty<T>(this
IEnumerable<T> enumerable)
        {
            if (enumerable == null)
            {
                return true;
            }

            var collection = enumerable as
ICollection<T>;
            if (collection != null)
            {
                return collection.Count < 1;
            }

            return !enumerable.Any();
        }

        public static bool TryAdd<T>(this ISet<T>
set, T value)
        {
            if (set == null || value == null)
            {
                return false;
            }

            if (!set.Contains(value))
            {
                set.Add(value);
                return true;
            }

            return false;
        }
    }
}
```

1.32. ErrorCode.cs

```
namespace
FractalGraph.Abstractions.Extensions
{
    public enum RuleErrorCode
    {
        Succeed = 0,
        WrongLeftColor = 1,
        WrongRightColor = 2,
    }
```

```
}
```

1.33. MathExtenstion.cs

```
using System;
using SharpDX;
using HelixToolkit.Wpf.SharpDX;
using Media =
System.Windows.Media.Media3D;

namespace
FractalGraph.Abstractions.Extensions
{
   public static class MathExtenstion
   {
      public static Vector3 Round(this Vector3
vec, int digits)
      {
         return new Vector3(
            (float)Math.Round(vec.X, digits),
            (float)Math.Round(vec.Y, digits),
            (float)Math.Round(vec.Z, digits));
      }

      public static Vector3 Rotate(this ref Vector3
point, Vector3 axis, float angleInDegree, Vector3
center)
      {
         var radians = (float)(Math.PI / 180 *
angleInDegree);
         var q = CreateFromAxisAngle(axis,
radians);
         var matr = Media.Matrix3D.Identity;
         matr.OffsetX = point.X;
         matr.OffsetY = point.Y;
         matr.OffsetZ = point.Z;
         matr.RotateAt(q, center.ToPoint3D());
         return new Vector3((float)matr.OffsetX,
(float)matr.OffsetY, (float)matr.OffsetZ);
      }

      public static Media.Quaternion
CreateFromAxisAngle(Vector3 axis, float
radians)
      {
         var quaternion = new
Media.Quaternion();
         float halfAngle = radians * 0.5f;
         float halfAngleSin =
(float)Math.Sin(halfAngle);
         quaternion.X = axis.X * halfAngleSin;
         quaternion.Y = axis.Y * halfAngleSin;
         quaternion.Z = axis.Z * halfAngleSin;
         quaternion.W =
(float)Math.Cos(halfAngle);
         return quaternion;
      }


      public static float ToRadians(float degrees)
      {
         return (float)(Math.PI / 180 * degrees);
      }
   }
}
```

1.34. ServiceProviderExtensions.cs

```
using System;
using System.ComponentModel.Design;

namespace
FractalGraph.Abstractions.Extensions
{
   public static class ServiceProviderExtensions
   {
      public static T GetService<T>(this
IServiceProvider provider)
      {
         return (T)provider.GetService(typeof(T));
      }

      public static void AddService<T>(this
IServiceContainer container, T serviceInstance)
      {
         container.AddService(typeof(T),
serviceInstance);
      }
   }
}
```

1.35. IMetricsOwner.cs

```
using System;

namespace MetricsService
{
   public interface IMetricsOwner
```

```
   {
      event EventHandler Closed;
   }
}
```

1.36. IMetricsService.cs

```
namespace MetricsService
{
   public interface IMetricsService
   {
      void MethodStart(string methodName);
      void MethodEnd(string methodName);

      void Assignment();
      void Arithmetic(int i = 1);
      void Comparison();
      void Jump();
   }
}
```

1.37. IMetricsServiceInternals.cs

```
using System;
using FractalGraph.Abstractions;

namespace MetricsService
{
   //should be internal
   public interface IMetricsServiceInternals :
IMetricsService
   {
      MetricsNode GetRoot();
      event EventHandler
MethodStuckBecameEmtpy;
   }
}
```

1.38. MetricsService.cs

```
using System;
using System.Collections.Generic;
using FractalGraph.Abstractions;

namespace MetricsService
{
   public class MetricsService :
IMetricsServiceInternals
   {
      MetricsNode _root = new MetricsNode();
      MetricsNode _current;
      Stack<MetricsNode> _methodsInProgress
= new Stack<MetricsNode>();

      public event EventHandler
MethodStuckBecameEmtpy;

      public void MethodStart(string
methodName)
      {
         var node = new MetricsNode();
         node.Name = methodName;

         if (_methodsInProgress.Count == 0)
         {
            _root.Children.Add(node);
            _current = node;
            _methodsInProgress.Push(node);
         }
         else
         {
            _methodsInProgress.Push(_current);
            _current.Children.Add(node);
            _current = node;
         }
      }

      public void MethodEnd(string
methodName)
      {
         if (_methodsInProgress.Count == 0)
            new InvalidOperationException();

         var node = _methodsInProgress.Pop();

         if (node.Name != _current.Name)
            new InvalidOperationException();

         _current = node;

         if (_methodsInProgress.Count == 0)
         {
            _current = null;

MethodStuckBecameEmtpy?.Invoke(this,
EventArgs.Empty);
         }
      }
```

```
        public MetricsNode GetRoot()
        {
            return _root;
        }

        public void Arithmetic(int i = 1)
        {
            _current.Arithmetics += i;
        }

        public void Assignment()
        {
            _current.Assignments++;
        }

        public void Comparison()
        {
            _current.Comparisons++;
        }

        public void Jump()
        {
            _current.Jumps++;
        }
    }
}
```

1.39. Defaults.cs

```
using SharpDX;
using HelixToolkit.Wpf.SharpDX;
using Media = System.Windows.Media;

namespace FractalGraph.Abstractions
{
    public static class Defaults
    {
        public const float DefaultNodeSize = 0.2f;
        public const float DefaultEdgeSize = 0.1f;

        public static readonly Color4 LeftColor =
Color.Yellow;
        public static readonly Color4 RightColor =
Color.Purple;
        public static readonly Color4 EdgeColor =
Color.LightGray;
        public static readonly Color4 NeutralColor
= Media.Color.FromArgb(255, 255, 128,
0).ToColor4(); // orange
    }
}
```

1.40. BaseCenteredLatticeGenerator.cs

```
using System;
using SharpDX;
using FractalGraph.Abstractions;

namespace
FractalGraph.Services.LatticeGenerators
{
    public class BaseCenteredLatticeGenerator :
LatticeGenerator
    {
        public override void CreateCell()
        {
            float a = A;
            float b = B;
            float c = C;
            float alpha = Alpha;
            float beta = Beta;
            float gamma = Gamma;

            var b1 = Vector3.Zero;
            float x = b1.X;
            float y = b1.Y;
            float z = b1.Z;

            var b2 = new Vector3(x + b, y, z);
            var b3 = new Vector3(x + a, y, z);
            Matrix.RotationZ(gamma, out var rotZ);
            Vector3.Transform(ref b3, ref rotZ, out
b3);
            var b4 = b2 + b3;

            float c_x = (float)(c * Math.Cos(alpha));
            float c_y = (float)((c * (Math.Cos(beta) -
Math.Cos(alpha) * Math.Cos(gamma))) /
Math.Sin(gamma));
            float c_z = (float)(Math.Sqrt(c * c - c_x *
c_x - c_y * c_y));
            var t1 = new Vector3(c_x, c_y, c_z);
            var t2 = b2 + t1;
            var t3 = b3 + t1;
            var t4 = b4 + t1;

            var graph = new Graph();
```

```
        var v1 = new Node() { Position = b1, Size
= NodeSize, Color = LeftColor };
        var v2 = new Node() { Position = b2, Size
= NodeSize, Color = RightColor };
        var v3 = new Node() { Position = b3, Size
= NodeSize, Color = RightColor };
        var v4 = new Node() { Position = b4, Size
= NodeSize, Color = RightColor };
        var v5 = new Node() { Position = t1, Size
= NodeSize, Color = RightColor };
        var v6 = new Node() { Position = t2, Size
= NodeSize, Color = RightColor };
        var v7 = new Node() { Position = t3, Size
= NodeSize, Color = RightColor };
        var v8 = new Node() { Position = t4, Size
= NodeSize, Color = RightColor };
        //bot
        var e1 = new Edge(v1, v2);
        var e2 = new Edge(v1, v3);
        var e3 = new Edge(v2, v4);
        var e4 = new Edge(v3, v4);
        //side
        var e5 = new Edge(v1, v5);
        var e6 = new Edge(v2, v6);
        var e7 = new Edge(v3, v7);
        var e8 = new Edge(v4, v8);
        //top
        var e9 = new Edge(v5, v6);
        var e10 = new Edge(v5, v7);
        var e11 = new Edge(v6, v8);
        var e12 = new Edge(v7, v8);

        graph.UnsafeAdd(v1, v2, v3, v4, v5, v6,
v7, v8);
        graph.UnsafeAdd(e1, e2, e3, e4, e5, e6,
e7, e8, e9, e10, e11, e12);

        //bot
        var botCent = (b1 + b4) / 2;
        var botCenter = new Node() { Position =
botCent, Size = NodeSize, Color =
Defaults.NeutralColor };
        var ebc1 = new Edge(v1, botCenter);
        var ebc2 = new Edge(v2, botCenter);
        var ebc3 = new Edge(v3, botCenter);
        var ebc4 = new Edge(v4, botCenter);
        graph.UnsafeAdd(botCenter);
        graph.UnsafeAdd(ebc1, ebc2, ebc3,
ebc4);

        //top
        var topCent = (t1 + t4) / 2;
        var topCenter = new Node() { Position =
topCent, Size = NodeSize, Color =
Defaults.NeutralColor };
        var etc1 = new Edge(v5, topCenter);
        var etc2 = new Edge(v6, topCenter);
        var etc3 = new Edge(v7, topCenter);
        var etc4 = new Edge(v8, topCenter);
        graph.UnsafeAdd(topCenter);
        graph.UnsafeAdd(etc1, etc2, etc3, etc4);

        Graph = graph;
      }
   }
}
```

1.41. BodyCenteredLatticeGenerator.cs

```
using System;
using SharpDX;
using FractalGraph.Abstractions;

namespace
FractalGraph.Services.LatticeGenerators
{
   public class BodyCenteredLatticeGenerator :
LatticeGenerator
   {
      public override void CreateCell()
      {
         float a = A;
         float b = B;
         float c = C;
         float alpha = Alpha;
         float beta = Beta;
         float gamma = Gamma;

         var b1 = Vector3.Zero;
         float x = b1.X;
         float y = b1.Y;
         float z = b1.Z;

         var b2 = new Vector3(x + b, y, z);
         var b3 = new Vector3(x + a, y, z);
         Matrix.RotationZ(gamma, out var rotZ);
         Vector3.Transform(ref b3, ref rotZ, out
b3);
```

```
        var b4 = b2 + b3;

        float c_x = (float)(c * Math.Cos(alpha));
        float c_y = (float)((c * (Math.Cos(beta) -
Math.Cos(alpha) * Math.Cos(gamma))) /
Math.Sin(gamma));
        float c_z = (float)(Math.Sqrt(c * c - c_x *
c_x - c_y * c_y));
        var t1 = new Vector3(c_x, c_y, c_z);
        var t2 = b2 + t1;
        var t3 = b3 + t1;
        var t4 = b4 + t1;

        var graph = new Graph();
        var v1 = new Node() { Position = b1, Size
= NodeSize, Color = LeftColor };
        var v2 = new Node() { Position = b2, Size
= NodeSize, Color = RightColor };
        var v3 = new Node() { Position = b3, Size
= NodeSize, Color = RightColor };
        var v4 = new Node() { Position = b4, Size
= NodeSize, Color = RightColor };
        var v5 = new Node() { Position = t1, Size
= NodeSize, Color = RightColor };
        var v6 = new Node() { Position = t2, Size
= NodeSize, Color = RightColor };
        var v7 = new Node() { Position = t3, Size
= NodeSize, Color = RightColor };
        var v8 = new Node() { Position = t4, Size
= NodeSize, Color = RightColor };
        //bot
        var e1 = new Edge(v1, v2);
        var e2 = new Edge(v1, v3);
        var e3 = new Edge(v2, v4);
        var e4 = new Edge(v3, v4);
        //side
        var e5 = new Edge(v1, v5);
        var e6 = new Edge(v2, v6);
        var e7 = new Edge(v3, v7);
        var e8 = new Edge(v4, v8);
        //top
        var e9 = new Edge(v5, v6);
        var e10 = new Edge(v5, v7);
        var e11 = new Edge(v6, v8);
        var e12 = new Edge(v7, v8);

        var cent = (b1 + t4) / 2;

        var center = new Node() { Position =
cent, Size = NodeSize, Color =
Defaults.NeutralColor };

        var v1c = new Edge(v1, center);
        var v2c = new Edge(v2, center);
        var v3c = new Edge(v3, center);
        var v4c = new Edge(v4, center);
        var v5c = new Edge(v5, center);
        var v6c = new Edge(v6, center);
        var v7c = new Edge(v7, center);
        var v8c = new Edge(v8, center);

        graph.UnsafeAdd(v1, v2, v3, v4, v5, v6,
v7, v8, center);
        graph.UnsafeAdd(e1, e2, e3, e4, e5, e6,
e7, e8, e9, e10, e11, e12,
            v1c, v2c, v3c, v4c, v5c, v6c, v7c, v8c);
        Graph = graph;
      }
   }
}
```

1.42. FaceCenteredLatticeGenerator.cs

```
using System;
using SharpDX;
using FractalGraph.Abstractions;

namespace
FractalGraph.Services.LatticeGenerators
{
   public class FaceCenteredLatticeGenerator :
LatticeGenerator
   {
      public override void CreateCell()
      {
         float a = A;
         float b = B;
         float c = C;
         float alpha = Alpha;
         float beta = Beta;
         float gamma = Gamma;

         var b1 = Vector3.Zero;
         float x = b1.X;
         float y = b1.Y;
         float z = b1.Z;
```

```
        var b2 = new Vector3(x + b, y, z);
        var b3 = new Vector3(x + a, y, z);
        Matrix.RotationZ(gamma, out var rotZ);
        Vector3.Transform(ref b3, ref rotZ, out
b3);
        var b4 = b2 + b3;

        float c_x = (float)(c * Math.Cos(alpha));
        float c_y = (float)((c * (Math.Cos(beta) -
Math.Cos(alpha) * Math.Cos(gamma))) /
Math.Sin(gamma));
        float c_z = (float)(Math.Sqrt(c * c - c_x *
c_x - c_y * c_y));
        var t1 = new Vector3(c_x, c_y, c_z);
        var t2 = b2 + t1;
        var t3 = b3 + t1;
        var t4 = b4 + t1;

        var graph = new Graph();
        var v1 = new Node() { Position = b1, Size
= NodeSize, Color = LeftColor };
        var v2 = new Node() { Position = b2, Size
= NodeSize, Color = RightColor };
        var v3 = new Node() { Position = b3, Size
= NodeSize, Color = RightColor };
        var v4 = new Node() { Position = b4, Size
= NodeSize, Color = RightColor };
        var v5 = new Node() { Position = t1, Size
= NodeSize, Color = RightColor };
        var v6 = new Node() { Position = t2, Size
= NodeSize, Color = RightColor };
        var v7 = new Node() { Position = t3, Size
= NodeSize, Color = RightColor };
        var v8 = new Node() { Position = t4, Size
= NodeSize, Color = RightColor };
        //bot
        var e1 = new Edge(v1, v2);
        var e2 = new Edge(v1, v3);
        var e3 = new Edge(v2, v4);
        var e4 = new Edge(v3, v4);
        //side
        var e5 = new Edge(v1, v5);
        var e6 = new Edge(v2, v6);
        var e7 = new Edge(v3, v7);
        var e8 = new Edge(v4, v8);
        //top
        var e9 = new Edge(v5, v6);
        var e10 = new Edge(v5, v7);
        var e11 = new Edge(v6, v8);
        var e12 = new Edge(v7, v8);

        graph.UnsafeAdd(v1, v2, v3, v4, v5, v6,
v7, v8);
        graph.UnsafeAdd(e1, e2, e3, e4, e5, e6,
e7, e8, e9, e10, e11, e12);

        //bot
        var botCent = (b1 + b4) / 2;
        var botCenter = new Node() { Position =
botCent, Size = NodeSize, Color =
Defaults.NeutralColor };
        var ebc1 = new Edge(v1, botCenter);
        var ebc2 = new Edge(v2, botCenter);
        var ebc3 = new Edge(v3, botCenter);
        var ebc4 = new Edge(v4, botCenter);
        graph.UnsafeAdd(botCenter);
        graph.UnsafeAdd(ebc1, ebc2, ebc3,
ebc4);

        //top
        var topCent = (t1 + t4) / 2;
        var topCenter = new Node() { Position =
topCent, Size = NodeSize, Color =
Defaults.NeutralColor };
        var etc1 = new Edge(v5, topCenter);
        var etc2 = new Edge(v6, topCenter);
        var etc3 = new Edge(v7, topCenter);
        var etc4 = new Edge(v8, topCenter);
        graph.UnsafeAdd(topCenter);
        graph.UnsafeAdd(etc1, etc2, etc3, etc4);

        //side 1
        var sideCent1 = (b1 + t2) / 2;
        var sideCenter1 = new Node() { Position
= sideCent1, Size = NodeSize, Color =
Defaults.NeutralColor };
        var es1c1 = new Edge(v1, sideCenter1);
        var es1c2 = new Edge(v2, sideCenter1);
        var es1c3 = new Edge(v5, sideCenter1);
        var es1c4 = new Edge(v6, sideCenter1);
        graph.UnsafeAdd(sideCenter1);
        graph.UnsafeAdd(es1c1, es1c2, es1c3,
es1c4);

        //side 2
        var sideCent2 = (b1 + t3) / 2;
```

```
        var sideCenter2 = new Node() { Position
= sideCent2, Size = NodeSize, Color =
Defaults.NeutralColor };
        var es2c1 = new Edge(v1, sideCenter2);
        var es2c2 = new Edge(v3, sideCenter2);
        var es2c3 = new Edge(v5, sideCenter2);
        var es2c4 = new Edge(v7, sideCenter2);
        graph.UnsafeAdd(sideCenter2);
        graph.UnsafeAdd(es2c1, es2c2, es2c3,
es2c4);


        //side 3
        var sideCent3 = (b2 + t4) / 2;
        var sideCenter3 = new Node() { Position
= sideCent3, Size = NodeSize, Color =
Defaults.NeutralColor };
        var es3c1 = new Edge(v2, sideCenter3);
        var es3c2 = new Edge(v4, sideCenter3);
        var es3c3 = new Edge(v6, sideCenter3);
        var es3c4 = new Edge(v8, sideCenter3);
        graph.UnsafeAdd(sideCenter3);
        graph.UnsafeAdd(es3c1, es3c2, es3c3,
es3c4);


        //side 4
        var sideCent4 = (b3 + t4) / 2;
        var sideCenter4 = new Node() { Position
= sideCent4, Size = NodeSize, Color =
Defaults.NeutralColor };
        var es4c1 = new Edge(v3, sideCenter4);
        var es4c2 = new Edge(v4, sideCenter4);
        var es4c3 = new Edge(v7, sideCenter4);
        var es4c4 = new Edge(v8, sideCenter4);
        graph.UnsafeAdd(sideCenter4);
        graph.UnsafeAdd(es4c1, es4c2, es4c3,
es4c4);

        Graph = graph;
      }
   }
}
```

1.43. HexagonalGenerator.cs

```
using System;
using SharpDX;
using FractalGraph.Abstractions;
using FractalGraph.Abstractions.Extensions;

namespace
FractalGraph.Services.LatticeGenerators
{
   public class HexagonalGenerator :
LatticeGenerator
   {
      public HexagonalGenerator()
      {
         Gamma = (float)(Math.PI / 3 * 2);
      }

      public override void CreateCell()
      {
         float a = A;
         float b = B;
         float c = C;
         float alpha = Alpha;
         float beta = Beta;
         float gamma = Gamma;

         var b1 = Vector3.Zero;
         float x = b1.X;
         float y = b1.Y;
         float z = b1.Z;

         var center = new Vector3(x + a, y, z);
         Matrix.RotationZ((float)Math.PI / 3, out
var rotZ);
         Vector3.Transform(ref center, ref rotZ,
out center);

         var b2 = b1.Rotate(Vector3.UnitZ, 60,
center);
         var b3 = b1.Rotate(Vector3.UnitZ, 120,
center);
         var b4 = b1.Rotate(Vector3.UnitZ, 180,
center);
         var b5 = b1.Rotate(Vector3.UnitZ, 240,
center);
         var b6 = b1.Rotate(Vector3.UnitZ, 300,
center);

         var graph = new Graph();
         var v1b = new Node() { Position = b1,
Size = NodeSize, Color = LeftColor };
         var v2b = new Node() { Position = b2,
Size = NodeSize, Color = RightColor };
```

```
        var v3b = new Node() { Position = b3,
Size = NodeSize, Color = RightColor };
        var v4b = new Node() { Position = b4,
Size = NodeSize, Color = RightColor };
        var v5b = new Node() { Position = b5,
Size = NodeSize, Color = RightColor };
        var v6b = new Node() { Position = b6,
Size = NodeSize, Color = RightColor };

        float c_x = (float)(c * Math.Cos(alpha));
        float c_y = (float)((c * (Math.Cos(beta) -
Math.Cos(alpha) * Math.Cos(gamma))) /
Math.Sin(gamma));
        float c_z = (float)(Math.Sqrt(c * c - c_x *
c_x - c_y * c_y));
        var t1 = new Vector3(c_x, c_y, c_z);

        var topCenter = new Vector3(c_x + a,
c_y, c_z);
        Matrix.RotationZ((float)Math.PI / 3, out
rotZ);
        Vector3.Transform(ref topCenter, ref
rotZ, out topCenter);

        var t2 = t1.Rotate(Vector3.UnitZ, 60,
topCenter);
        var t3 = t1.Rotate(Vector3.UnitZ, 120,
topCenter);
        var t4 = t1.Rotate(Vector3.UnitZ, 180,
topCenter);
        var t5 = t1.Rotate(Vector3.UnitZ, 240,
topCenter);
        var t6 = t1.Rotate(Vector3.UnitZ, 300,
topCenter);

        var v1t = new Node() { Position = t1, Size
= NodeSize, Color = RightColor };
        var v2t = new Node() { Position = t2, Size
= NodeSize, Color = RightColor };
        var v3t = new Node() { Position = t3, Size
= NodeSize, Color = RightColor };
        var v4t = new Node() { Position = t4, Size
= NodeSize, Color = RightColor };
        var v5t = new Node() { Position = t5, Size
= NodeSize, Color = RightColor };
        var v6t = new Node() { Position = t6, Size
= NodeSize, Color = RightColor };

        graph.UnsafeAdd(v1b, v2b, v3b, v4b,
v5b, v6b,
            v1t, v2t, v3t, v4t, v5t, v6t);

        //side edges
        var eb1t1 = new Edge(v1b, v1t);
        var eb2t2 = new Edge(v2b, v2t);
        var eb3t3 = new Edge(v3b, v3t);
        var eb4t4 = new Edge(v4b, v4t);
        var eb5t5 = new Edge(v5b, v5t);
        var eb6t6 = new Edge(v6b, v6t);

        //bot eges
        var eb1b1 = new Edge(v1b, v2b);
        var eb2b2 = new Edge(v2b, v3b);
        var eb3b3 = new Edge(v3b, v4b);
        var eb4b4 = new Edge(v4b, v5b);
        var eb5b5 = new Edge(v5b, v6b);
        var eb6b6 = new Edge(v6b, v1b);

        //top edges
        var et1t1 = new Edge(v1t, v2t);
        var et2t2 = new Edge(v2t, v3t);
        var et3t3 = new Edge(v3t, v4t);
        var et4t4 = new Edge(v4t, v5t);
        var et5t5 = new Edge(v5t, v6t);
        var et6t6 = new Edge(v6t, v1t);


        graph.UnsafeAdd(eb1t1, eb2t2, eb3t3,
eb4t4, eb5t5, eb6t6,
            eb1b1, eb2b2, eb3b3, eb4b4, eb5b5,
eb6b6,
            et1t1, et2t2, et3t3, et4t4, et5t5, et6t6);
        Graph = graph;
      }
    }
}
```

1.44. LatticeContructor.cs

```
using System;
using SharpDX;
using FractalGraph.Abstractions;
using FractalGraph.Abstractions.Extensions;

namespace
FractalGraph.Services.LatticeGenerators
{
```

```
  public abstract class LatticeGenerator
  {
     public LatticeGenerator()
     {
     }

     public void UpdateLatice()
     {
        CreateCell();
        LaticeUpdated?.Invoke(this, Graph);
     }

     public abstract void CreateCell();

     public event EventHandler<IGraph>
LaticeUpdated;

     public IGraph Graph { get; protected set; }

     public LatticeGenerator(float a, float b, float
c,
        float alpha, float beta, float gamma, float
nodeSize = 1)
     {
        A = a;
        B = b;
        C = c;
        Alpha = alpha;
        Beta = beta;
        Gamma = gamma;
        NodeSize = nodeSize;
     }

     public Color4 LeftColor { get; set; } =
Defaults.LeftColor;
     public Color4 RightColor { get; set; } =
Defaults.RightColor;
     public float A { get; set; } = 1.0f;
     public float B { get; set; } = 1.0f;
     public float C { get; set; } = 1.0f;

     public float Alpha { get; set; } =
MathExtenstion.ToRadians(90);
     public float Beta { get; set; } =
MathExtenstion.ToRadians(90);
     public float Gamma { get; set; } =
MathExtenstion.ToRadians(90);
     public float NodeSize { get; set; } =
Defaults.DefaultNodeSize;

     //IE stands for interactively editable
     public bool IEA { get; set; } = true;
     public bool IEB { get; set; } = true;
     public bool IEC { get; set; } = true;
     public bool IEAlpha { get; set; } = true;
     public bool IEBeta { get; set; } = true;
     public bool IEGamma { get; set; } = true;
  }
}
```

1.45. TriclinicLatticeFullGenerator.cs

```
using System;
using SharpDX;
using FractalGraph.Abstractions;

namespace
FractalGraph.Services.LatticeGenerators
{
  public class TriclinicLatticeFullGenerator :
LatticeGenerator
  {
     public TriclinicLatticeFullGenerator()
     {
     }

     public TriclinicLatticeFullGenerator(float a,
float b, float c,
        float alpha, float beta, float gamma, float
nodeSize = 1)
        : base(a, b, c, alpha, beta, gamma,
nodeSize)
     {
     }

     public override void CreateCell()
     {
        float a = A;
        float b = B;
        float c = C;
        float alpha = Alpha;
        float beta = Beta;
        float gamma = Gamma;

        var b1 = Vector3.Zero;
        float x = b1.X;
```

```
            float y = b1.Y;
            float z = b1.Z;

            var b2 = new Vector3(x + b, y, z);
            var b3 = new Vector3(x + a, y, z);
            SharpDX.Matrix.RotationZ(gamma, out
var rotZ);
            Vector3.Transform(ref b3, ref rotZ, out
b3);
            var b4 = b2 + b3;

            float c_x = (float)(c * Math.Cos(alpha));
            float c_y = (float)((c * (Math.Cos(beta) -
Math.Cos(alpha) * Math.Cos(gamma))) /
Math.Sin(gamma));
            float c_z = (float)(Math.Sqrt(c * c - c_x *
c_x - c_y * c_y));
            var t1 = new Vector3(c_x, c_y, c_z);
            var t2 = b2 + t1;
            var t3 = b3 + t1;
            var t4 = b4 + t1;

            var graph = new Graph();
            var v1 = new Node() { Position = b1, Size
= NodeSize, Color = LeftColor };
            var v2 = new Node() { Position = b2, Size
= NodeSize, Color = RightColor };
            var v3 = new Node() { Position = b3, Size
= NodeSize, Color = RightColor };
            var v4 = new Node() { Position = b4, Size
= NodeSize, Color = RightColor };
            var v5 = new Node() { Position = t1, Size
= NodeSize, Color = RightColor };
            var v6 = new Node() { Position = t2, Size
= NodeSize, Color = RightColor };
            var v7 = new Node() { Position = t3, Size
= NodeSize, Color = RightColor };
            var v8 = new Node() { Position = t4, Size
= NodeSize, Color = RightColor };
            //bot
            var e1 = new Edge(v1, v2);
            var e2 = new Edge(v1, v3);
            var e3 = new Edge(v2, v4);
            var e4 = new Edge(v3, v4);
            //side
            var e5 = new Edge(v1, v5);
            var e6 = new Edge(v2, v6);
            var e7 = new Edge(v3, v7);
            var e8 = new Edge(v4, v8);
            //top
            var e9 = new Edge(v5, v6);
            var e10 = new Edge(v5, v7);
            var e11 = new Edge(v6, v8);
            var e12 = new Edge(v7, v8);
            graph.UnsafeAdd(v1, v2, v3, v4, v5, v6,
v7, v8);
            graph.UnsafeAdd(e1, e2, e3, e4, e5, e6,
e7, e8, e9, e10, e11, e12);
            Graph = graph;
        }
    }
}
```

1.46. TriclinicLatticeSimpleGenerator.cs

```
using System;
using SharpDX;
using FractalGraph.Abstractions;

namespace
FractalGraph.Services.LatticeGenerators
{
    public class TriclinicLatticeSimpleGenerator :
LatticeGenerator
    {
        public TriclinicLatticeSimpleGenerator()
        {
        }

        public
TriclinicLatticeSimpleGenerator(float a, float b,
float c,
            float alpha, float beta, float gamma, float
nodeSize = 1)
            : base(a, b, c, alpha, beta, gamma,
nodeSize)
        {
        }

        public override void CreateCell()
        {
            float a = A;
            float b = B;
            float c = C;
            float alpha = Alpha;
            float beta = Beta;
```

```
            float gamma = Gamma;

            var b1 = Vector3.Zero;
            float x = b1.X;
            float y = b1.Y;
            float z = b1.Z;

            var b2 = new Vector3(x + b, y, z);
            var b3 = new Vector3(x + a, y, z);
            SharpDX.Matrix.RotationZ(gamma, out
var rotZ);
            Vector3.Transform(ref b3, ref rotZ, out
b3);

            float c_x = (float)(c * Math.Cos(alpha));
            float c_y = (float)((c * (Math.Cos(beta) -
Math.Cos(alpha) * Math.Cos(gamma))) /
Math.Sin(gamma));
            float c_z = (float)(Math.Sqrt(c * c - c_x *
c_x - c_y * c_y));
            var t1 = new Vector3(c_x, c_y, c_z);

            var graph = new Graph();
            var v1 = new Node() { Position = b1, Size
= NodeSize, Color = LeftColor };
            var v2 = new Node() { Position = b2, Size
= NodeSize, Color = RightColor };
            var v3 = new Node() { Position = b3, Size
= NodeSize, Color = RightColor };
            var v4 = new Node() { Position = t1, Size
= NodeSize, Color = RightColor };
            var e1 = new Edge(v1, v2);
            var e2 = new Edge(v1, v3);
            var e3 = new Edge(v1, v4);
            graph.UnsafeAdd(v1, v2, v3, v4);
            graph.UnsafeAdd(e1, e2, e3);
            Graph = graph;
        }
    }
}
```

1.47. ILiraService.cs

```
using FractalGraph.Abstractions;

namespace FractalGraph.Services
{
    public interface ILiraService
    {
        void Create(IGraph graph);
    }
}
```

1.48. LiraService.cs

```
using System;
using System.Collections.Generic;
using LiraSapr;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.Abstractions;

namespace FractalGraph.Services
{
    public class LiraService : ILiraService
    {
        class StiffnessInfo
        {
            public int Id { get; set; }
            public int Type { get; set; }
            public string Parameter { get; set; }
        }

        Dictionary<string, StiffnessInfo>
_elementsByStiffness = new Dictionary<string,
StiffnessInfo>();
        LiraApplication _app;

        LiraApplication App
        {
            get
            {
                if (_app == null)
                    _app = new LiraApplication();
                return _app;
            }
        }

        public void Create(IGraph graph)
        {
            _elementsByStiffness.Clear();
            graph.Reindex();

            LiraDocument doc =
App.CreateNewDocument();
            CreateEmptyTables(doc);
            var tables = doc.AllTables;
```

```
        int i = 0;
        var nodes = graph.Nodes;
        var nodeCoords = new
object[graph.NodesCount(), 4];
        var nodeRestrains = new
object[graph.NodesCount(), 8];
        foreach (var node in nodes)
        {
            nodeCoords[i, 0] = node.Id;
            nodeCoords[i, 1] = node.Position.X;
            nodeCoords[i, 2] = node.Position.Y;
            nodeCoords[i, 3] = node.Position.Z;

            nodeRestrains[i, 0] = node.Id;
            nodeRestrains[i, 1] = node.Restraints.X
== true ?  1 : 0;
            nodeRestrains[i, 2] = node.Restraints.Y
== true ?  1 : 0;
            nodeRestrains[i, 3] = node.Restraints.Z
== true ?  1 : 0;
            nodeRestrains[i, 4] =
node.Restraints.UX == true ? 1 : 0;
            nodeRestrains[i, 5] =
node.Restraints.UY == true ? 1 : 0;
            nodeRestrains[i, 6] =
node.Restraints.UZ == true ? 1 : 0;
            nodeRestrains[i, 7] =
node.Restraints.W == true ?  1 : 0;

            i++;
        }

        int idCounter = 1;
        i = 0;
        var edges = graph.Edges;
        var elTypeNodes = new
object[graph.EdgesCount(), 3];
        foreach (var edge in edges)
        {
            elTypeNodes[i, 0] = i + 1;
            elTypeNodes[i, 1] = 10;
            elTypeNodes[i, 2] = $"{edge.Start.Id},
{edge.End.Id}";
            i++;

            string stiffness =
edge.Stiffness.ToLiraParameter();
            if
(!_elementsByStiffness.TryGetValue(stiffness,
out var stiffnessInfo))
            {
               stiffnessInfo = new StiffnessInfo()
               {
                  Id = idCounter++,
                  Type = edge.Stiffness.LiraType,
                  Parameter = stiffness,
               };
               _elementsByStiffness[stiffness] =
stiffnessInfo;
            }
        }

        LiraTable nodesTable =
GetFirstLiraTable(tables,
LiraTableEnum.kLiraTable_Nodes_Coordinates
);
        nodesTable.SetContents(nodeCoords);

        LiraTable els = GetFirstLiraTable(tables,
LiraTableEnum.kLiraTable_Elements_TypeAnd
Nodes);
        els.SetContents(elTypeNodes);

        nodesTable.Apply(out string strErrors);
        if (!strErrors.IsNullOrEmpty())
           throw new Exception(strErrors);

        els.Apply(out strErrors);
        if (!strErrors.IsNullOrEmpty())
           throw new Exception(strErrors);

        LiraTable restraints =
GetFirstLiraTable(tables,
LiraTableEnum.kLiraTable_Nodes_Restraints);
        restraints.SetContents(nodeRestrains);

        restraints.Apply(out strErrors);
        if (!strErrors.IsNullOrEmpty())
           throw new Exception(strErrors);

        LiraTable stiffnessesTable =
GetFirstLiraTable(tables,
LiraTableEnum.kLiraTable_Stiffnesses);

        var newStiffness = new
object[_elementsByStiffness.Count, 9];
```

```
        i = 0;
        foreach (var pair in
_elementsByStiffness)
        {
           newStiffness[i, 0] = pair.Value.Id;
           newStiffness[i, 1] = pair.Value.Type;
           newStiffness[i, 4] =
pair.Value.Parameter;
           i++;
        }


stiffnessesTable.SetContents(newStiffness);
        stiffnessesTable.Apply(out strErrors);
        if (!strErrors.IsNullOrEmpty())
           throw new Exception(strErrors);

        var elementsCount = edges.Count;
        LiraTable stiffnessesElementsTable =
GetFirstLiraTable(tables,
LiraTableEnum.kLiraTable_Elements_Stiffnesse
s);
        var stiffnessesElements = new
object[elementsCount, 2];

        for (i = 0; i < elementsCount; i++)
        {
           stiffnessesElements[i, 0] = i + 1;
           stiffnessesElements[i, 1] =
_elementsByStiffness[edges[i].Stiffness.ToLiraP
arameter()].Id;
        }

stiffnessesElementsTable.SetContents(stiffnesses
Elements);

        stiffnessesElementsTable.Apply(out
strErrors);
        if (!strErrors.IsNullOrEmpty())
           throw new Exception(strErrors);
     }

     LiraTable GetFirstLiraTable(LiraTables
tables, LiraTableEnum tableType)
     {
        int count = tables.ItemCount;

        for (int j = 0; j < count; j++)
        {
           var groupItem = tables.Item[j];

           var table = groupItem as LiraTable;
           if (table == null)
              continue;

           if (table.Type.HasFlag(tableType))
              return table;
        }

        return null;
     }

     object[,] GetElements(LiraTables tables)
     {

        var elementsTable =
GetFirstLiraTable(tables,
LiraTableEnum.kLiraTable_Elements_TypeAnd
Nodes);
        object elementsObj = null;
        elementsTable.GetContents(ref
elementsObj);
        var elements = elementsObj as object[,];
        return elements;
     }

     void CreateEmptyTables(LiraDocument
doc)
     {
        var values =
Enum.GetValues(typeof(LiraTableEnum));

        for (int i = 0; i < values.Length - 1; i++)
        {
           var type =
(LiraTableEnum)values.GetValue(i);

           if (type ==
LiraTableEnum.kLiraTable_AllTables
              || type ==
LiraTableEnum.kLiraTable_AssemblageStages
              || type ==
LiraTableEnum.kLiraTable_Group
              || type ==
LiraTableEnum.kLiraTable_Cnt
              || type ==
LiraTableEnum.kLiraTable_PRBs
```

```
                || type ==
LiraTableEnum.kLiraTable_StructuralBlocks
                || type ==
LiraTableEnum.kLiraTable_Elements_Materials
                )
                continue;

                doc.AllTables.CreateNewItem(type);
            }
        }
    }
}
```

1.49. RelayCommand.cs

```
using System;
using System.Windows.Input;

namespace FractalGraph.UI.Commands
{
    public class RelayCommand<T> : ICommand
    {
        readonly Action<T> _execute;
        readonly Func<T, bool> _canExecute;

        public event EventHandler
CanExecuteChanged;

        public RelayCommand(Action<T> execute,
Func<T, bool> canExecute = null)
        {
            if (execute == null)
                throw new
ArgumentNullException("execute");

            _execute = execute;
            _canExecute = canExecute;
        }

        public bool CanExecute(object parameter)
        {
            return _canExecute == null ||
_canExecute((T)parameter);
        }

        public void Execute(object parameter)
        {
            _execute((T)parameter);
        }
    }
}
```

1.50. ColorToBrushConverter.cs

```
using System;
using System.Globalization;
using System.Windows.Data;
using SharpDX;
using HelixToolkit.Wpf.SharpDX;
using Media = System.Windows.Media;

namespace FractalGraph.UI.Converters
{
    public class ColorToBrushConverter :
IValueConverter
    {
        public object Convert(object value, Type
targetType, object parameter, CultureInfo
culture)
        {
            if (value is Color4 color)
                return new
Media.SolidColorBrush(color.ToColor());
            return null;
        }

        public object ConvertBack(object value,
Type targetType, object parameter, CultureInfo
culture)
        {
            if (value is Media.SolidColorBrush brush)
                return new
Color4(brush.Color.ToColor4());
            return null;
        }
    }
}
```

1.51. RadiansToDegreesConverter.cs

```
using System;
using System.Globalization;
using System.Windows.Data;

namespace FractalGraph.UI.Converters
{
    public class RadiansToDegreesConverter :
IValueConverter
    {
```

```
        public object Convert(object value, Type
targetType, object parameter, CultureInfo
culture)
        {
            float? val = null;
            if (value is float)
            {
                val = (float)(value);
            }
            else if (value is string str)
            {
                val = float.Parse(str,
CultureInfo.InvariantCulture);
            }
            if (val.HasValue)
                return val * 180 / Math.PI;

            return null;
        }

        public object ConvertBack(object value,
Type targetType, object parameter, CultureInfo
culture)
        {
            float? val = null;
            if (value is float)
            {
                val = (float)(value);
            }
            else if (value is string str)
            {
                val = float.Parse(str,
CultureInfo.InvariantCulture);
            }
            if (val.HasValue)
                return val * Math.PI / 180;

            return null;
        }
    }
}
```

1.52. BaseViewportController.cs

```
using System;
using SharpDX;
using HelixToolkit.Wpf.SharpDX;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.Abstractions;
using FractalGraph.UI.Services;
using FractalGraph.UI.Views;
using FractalGraph.UI.ViewModels;

namespace FractalGraph.UI.Controllers
{
    public class BaseViewportController :
IBaseViewportController
    {
        protected IServiceProvider _services;
        protected IUIService _uiService;
        protected IStateManager _stateManager;

        protected IUIView _viewportView;
        protected ViewportViewModel
_viewportViewModel;

        public IUIView Control => _viewportView;

        public
BaseViewportController(IServiceProvider
services)
        {
            _services = services;
            _uiService =
_services.GetService<IUIService>();
            _stateManager =
_services.GetService<IStateManager>();

            _viewportViewModel = new
ViewportViewModel();
            _viewportView =
_uiService.CreateView(_viewportViewModel);
        }

        public UITerminalEntity SelectedTerminal
        {
            get => _stateManager.UISelected;
            set
            {
                if (value == _stateManager.UISelected)
                    return;

                if (_stateManager.UISelected != null)

_stateManager.UISelected.PostEffects = "";

                _stateManager.UISelected = value;
```

```
            if (_stateManager.UISelected != null)
_stateManager.UISelected.PostEffects =
"highlightSelected";
        }
    }

    public bool IsPrimaryViewport
    {
        get;
        set;
    }


    public bool EnableSelecting
    {
        get;
        set;
    }

    public bool ShowAxis
    {
        get => _viewportViewModel.ShowAxis;
        set => _viewportViewModel.ShowAxis =
value;
    }

    public bool ShowViewCube
    {
        get =>
_viewportViewModel.ShowViewCube;
        set =>
_viewportViewModel.ShowViewCube = value;
    }

    public bool ShowInfoPanel
    {
        get =>
_viewportViewModel.ShowInfoPanel;
        set =>
_viewportViewModel.ShowInfoPanel = value;
    }

    public Vector3 UpDirection
    {
        get => _viewportViewModel == null ?
Vector3.Zero :
_viewportViewModel.UpDirection.ToVector3();
        set
        {
            if (_viewportViewModel == null)
                return;

            _viewportViewModel.UpDirection =
value.ToVector3D();
        }
    }

    public Vector3 CameraPos
    {
        get => _viewportViewModel == null ?
Vector3.Zero :
_viewportViewModel.CameraPos.ToVector3();
        set
        {
            if (_viewportViewModel == null)
                return;

            _viewportViewModel.CameraPos =
value.ToPoint3D();
        }
    }

    public Vector3 CameraLook
    {
        get => _viewportViewModel == null ?
Vector3.Zero :
_viewportViewModel.CameraLook.ToVector3()
;
        set
        {
            if (_viewportViewModel == null)
                return;

            _viewportViewModel.CameraLook =
value.ToVector3D();
        }
    }

    public event EventHandler
ViewportClicked
    {
        remove =>
_viewportViewModel.ViewportClicked -=
value;
        add =>
_viewportViewModel.ViewportClicked +=
value;
```

```
    }
  }
}
```

1.53. GraphViewportController.cs

```
using System;
using System.Collections.Generic;
using SharpDX;
using HelixToolkit.Wpf.SharpDX;
using HelixToolkit.Wpf.SharpDX.Model.Scene;
using FractalGraph.Abstractions;

using Media = System.Windows.Media;

namespace FractalGraph.UI.Controllers
{
  public class GraphViewportController :
BaseViewportController,
IGraphViewportController
  {
    Dictionary<INode, UINode> _uiNodes =
new Dictionary<INode, UINode>();

    public
GraphViewportController(IServiceProvider
services)
      : base(services)
    {
    }

    private GroupNode RootNode =>
_viewportViewModel.RootNode;

    public IGraph Graph { get; set; } = new
Graph();

    public void InvalidateGraphVisual()
    {
      RootNode.Clear();
      _uiNodes.Clear();
      if (Graph == null)
        return;

      AddVisual(Graph.Nodes);
      AddVisual(Graph.Edges);
    }

    public void
AddVisual(IEnumerable<INode> nodes)
    {
      Dictionary<double, Geometry3D>
geometries = new Dictionary<double,
Geometry3D>();

      foreach (var node in nodes)
      {
        if (!geometries.TryGetValue(node.Size,
out var geom))
        {
          var meshBuilder = new
MeshBuilder();

meshBuilder.AddSphere(Vector3.Zero,
node.Size);
          geom =
meshBuilder.ToMeshGeometry3D();
          geom.IsDynamic = true;
          geometries.Add(node.Size, geom);
        }

        var uiNode = new UINode(node) {
Geometry = geom };
        uiNode.ModelMatrix =
Matrix.Translation(node.Position);
        uiNode.Material = new
DiffuseMaterial() { DiffuseColor = node.Color
};

        if (EnableSelecting)
          uiNode.MouseUp +=
MeshNode_MouseUp;

        RootNode.AddChildNode(uiNode);
        _uiNodes[node] = uiNode;
      }
    }

    public void
AddVisual(IEnumerable<IEdge> edges)
    {
      foreach (var edge in edges)
      {
        var meshBuilder = new MeshBuilder();

meshBuilder.AddCylinder(edge.Start.Position,
edge.End.Position, edge.Size, 32);
```

```
            var geom =
meshBuilder.ToMeshGeometry3D();
            geom.IsDynamic = true;

            var uiEdge = new UIEdge(edge) {
Geometry = geom };
            uiEdge.Material = new
DiffuseMaterial() { DiffuseColor =
Defaults.EdgeColor };

            if (EnableSelecting)
               uiEdge.MouseUp +=
MeshNode_MouseUp;

            RootNode.AddChildNode(uiEdge);
         }
      }

      private void MeshNode_MouseUp(object
sender, SceneNodeMouseUpArgs e)
      {
         SelectedTerminal = e.Source as
UITerminalEntity;
      }

      public void SetVisual(IGraph graph)
      {
         AddVisual(graph.Nodes);
         AddVisual(graph.Edges);
      }

      public void
RepaintVisual(IEnumerable<INode> nodes)
      {
         foreach (var node in nodes)
         {
            if (_uiNodes.TryGetValue(node, out
var meshNode))
               meshNode.Material = new
DiffuseMaterial() { DiffuseColor = node.Color
};
         }
      }

      public void RepaintSelected()
      {
         var selected = SelectedTerminal as
UINode;
         if (selected == null)
            return;

         var colorDialog = new
System.Windows.Forms.ColorDialog();
         Color4 newColor;
         if (colorDialog.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
         {
            var col = colorDialog.Color;
            var mediaColor =
Media.Color.FromArgb(col.A, col.R, col.G,
col.B);
            newColor = mediaColor.ToColor4();
            if (selected != null)
               selected.SetColor(newColor);
         }
      }
   }
}
```

1.54. IBaseViewportController.cs

```
using System;
using SharpDX;
using FractalGraph.UI.Views;

namespace FractalGraph.UI.Controllers
{
   public interface IBaseViewportController
   {
      IUIView Control { get; }

      bool IsPrimaryViewport { get; set; }
      bool EnableSelecting { get; set; }

      bool ShowAxis { get; set; }
      bool ShowViewCube { get; set; }
      bool ShowInfoPanel { get; set; }

      Vector3 CameraPos { get; set; }
      Vector3 CameraLook { get; set; }
      Vector3 UpDirection { get; set; }

      event EventHandler ViewportClicked;
   }
}
```

1.55. IGraphViewportController.cs

```
using System;
```

```
using System.Collections.Generic;
using FractalGraph.Abstractions;

namespace FractalGraph.UI.Controllers
{
   public interface IGraphViewportController :
IBaseViewportController
   {
      void InvalidateGraphVisual();
      void AddVisual(IEnumerable<INode>
nodes);
      void AddVisual(IEnumerable<IEdge>
edges);
      void SetVisual(IGraph graph);
      void RepaintVisual(IEnumerable<INode>
nodes);
      void RepaintSelected();

      IGraph Graph { get; set; }

      event EventHandler ViewportClicked;
   }
}
```

1.56. MainController.cs

```
using System;
using System.ComponentModel.Design;
using FractalGraph.Services;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.UI.Services;
using FractalGraph.UI.ViewModels;
using FractalGraph.UI.Views;
using MetricsService;

namespace FractalGraph.UI.Controllers
{
   public class MainController : IServiceProvider
   {
      IUIView _mainView;
      MainWindowViewModel
_mainWindowViewModel;

      IServiceContainer _serviceContainer;
      IUIService _uiService;
      IRuleController _ruleController;
      IStateManager _stateManager;
      ICommandManager _commandManager;
      ILiraService _liraService;
      IMetricsService _metricsService;

      public MainController(IUIView mainView)
      {
         ConfigureServices();
         _mainView = mainView;
         _mainWindowViewModel = new
MainWindowViewModel(_serviceContainer);
         _mainView.ViewModel =
_mainWindowViewModel;

((MainWindow)_mainView).InitServices(_servi
ceContainer);
      }

      void ConfigureServices()
      {
         _serviceContainer = new
ServiceContainer();

         _metricsService = new
MetricsService.MetricsService();

_serviceContainer.AddService<IMetricsService
>(_metricsService);

         _uiService = new
UIService(_serviceContainer);

_serviceContainer.AddService<IUIService>(_ui
Service);

         _liraService = new LiraService();

_serviceContainer.AddService<ILiraService>(_l
iraService);

         _stateManager = new StateManager();

_serviceContainer.AddService<IStateManager>(
_stateManager);

_stateManager.PrimaryViewportController =
_uiService.CreateGraphViewportController();

_stateManager.PrimaryViewportController.Sho
wViewCube = true;
```

```
_stateManager.PrimaryViewportController.Sho
wAxis = true;

_stateManager.PrimaryViewportController.Sho
wInfoPanel = true;

_stateManager.PrimaryViewportController.IsPri
maryViewport = true;

_stateManager.PrimaryViewportController.Enab
leSelecting = true;

        _commandManager = new
CommandManager(_serviceContainer);

_serviceContainer.AddService<ICommandMana
ger>(_commandManager);

        _ruleController = new
RuleController(_serviceContainer);
      }

      public object GetService(Type serviceType)
      {
         return
_serviceContainer.GetService(serviceType);
      }
   }
}
```

1.57. RuleController.cs

```
using System;
using SharpDX;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.Abstractions;
using FractalGraph.UI.Services;
using FractalGraph.UI.ViewModels;
using FractalGraph.UI.Views;
using FractalGraph.Services.LatticeGenerators;

namespace FractalGraph.UI.Controllers
{
   public interface IRuleController
   {
      IUIView Control { get; }
      IUIView ListControl { get; }
   }

   public class RuleController : IRuleController
   {
      IServiceProvider _services;
      IUIService _uiService;
      IStateManager _stateManager;

      IGraphViewportController _leftViewport;
      IGraphViewportController _rightViewport;

      IUIView _ruleView;
      IUIView _rulesListView;
      LeftRuleViewModel _ruleViewModel;
      RulesListViewModel
_rulesListViewModel;
      LatticeGenerator _constructor;
      IRule _rule;

      public IUIView Control => _ruleView;
      public IUIView ListControl =>
_rulesListView;

      public RuleController(IServiceProvider
services)
      {
         _services = services;
         _uiService =
_services.GetService<IUIService>();
         _stateManager =
_services.GetService<IStateManager>();

         _leftViewport =
_uiService.CreateGraphViewportController();
         _rightViewport =
_uiService.CreateGraphViewportController();

         ConfigureDefaultRule();

         _ruleView =
_uiService.CreateView(_ruleViewModel);

         _rulesListViewModel = new
RulesListViewModel(_services);
         _rulesListView =
_uiService.CreateView(_rulesListViewModel);
         _rulesListViewModel.RightRuleSelected
+= _rulesListViewModel_RightRuleSelected;

         _stateManager.RuleView = _ruleView;
```

```
        _stateManager.RulesListView =
_rulesListView;


_leftViewport.AddVisual(_rule.Left.Nodes);
        _leftViewport.CameraPos = new
Vector3(0, -1, 0);
        _leftViewport.CameraLook = new
Vector3(0, 1, 0);


_rightViewport.AddVisual(_rule.Right.Nodes);
        _rightViewport.UpDirection = new
Vector3(0, 1.0f, 0.7f);
        _rightViewport.CameraPos = new
Vector3(0.5f, -1.5f, 2.7f);
        _rightViewport.CameraLook = new
Vector3(0, 2, -2.5f);

        _stateManager.CurrentRule = _rule;
        _stateManager.StateChanged +=
_stateManager_StateChanged;
        _ruleViewModel.LeftPartRepainted +=
_rule_LeftPartRepainted;
        _ruleViewModel.RightPartRepainted +=
_rule_RightPartRepainted;
        DefineRulesList();
      }
      private void
_stateManager_StateChanged(object sender,
StateEventArgs e)
      {
        if (!IsParameter(e.PropertyName))
          return;

        if (_constructor?.Graph != null)
        {
          _constructor.UpdateLatice();
          _rightViewport.Graph =
_constructor.Graph;

_rightViewport.InvalidateGraphVisual();

          _rule.SetRightPart(_constructor.Graph,
_rule.MainRightColor);
        }
      }
      private void _rule_LeftPartRepainted(object
sender, Color4 color)
      {
        _leftViewport.InvalidateGraphVisual();

_rulesListViewModel.SelectedItemViewModel.
Constructor.LeftColor = color;

_rulesListViewModel.SelectedItemViewModel.
ViewportController.InvalidateGraphVisual();

        _constructor.UpdateLatice();
        _rightViewport.Graph =
_constructor.Graph;
        _rightViewport.InvalidateGraphVisual();
      }

      private void
_rule_RightPartRepainted(object sender, Color4
color)
      {
        _rightViewport.InvalidateGraphVisual();

_rulesListViewModel.SelectedItemViewModel.
Constructor.RightColor = color;

_rulesListViewModel.SelectedItemViewModel.
ViewportController.InvalidateGraphVisual();

        _constructor.UpdateLatice();
        _rightViewport.Graph =
_constructor.Graph;
        _rightViewport.InvalidateGraphVisual();
      }


      private void DefineRulesList()
      {
        LatticeGenerator generator = new
TriclinicLatticeSimpleGenerator();
        generator.CreateCell();

_rulesListViewModel.AddRule(generator,
"Триклінна примітивна базова");

        generator = new
TriclinicLatticeFullGenerator();
        generator.CreateCell();
```

```
_rulesListViewModel.AddRule(generator,
"Триклінна примітивна повна");

        generator = new
TriclinicLatticeFullGenerator()
        {
           IEAlpha = false,
           IEGamma = false
        };
        generator.CreateCell();

_rulesListViewModel.AddRule(generator,
"Моноклінна примітивна");

        generator = new
BaseCenteredLatticeGenerator()
        {
           IEAlpha = false,
           IEGamma = false
        };
        generator.CreateCell();

_rulesListViewModel.AddRule(generator,
"Моноклінна базоцентрована");

        generator = new
BodyCenteredLatticeGenerator()
        {
           IEAlpha = false,
           IEBeta = false,
           IEGamma = false
        };
        generator.CreateCell();

_rulesListViewModel.AddRule(generator,
"Ромбічна об'ємноцентрована");

        generator = new
BaseCenteredLatticeGenerator()
        {
           IEAlpha = false,
           IEBeta = false,
           IEGamma = false
        };
        generator.CreateCell();

_rulesListViewModel.AddRule(generator,
"Ромбічна базоцентрована");

        generator = new
FaceCenteredLatticeGenerator()
        {
           IEAlpha = false,
           IEBeta = false,
           IEGamma = false
        };
        generator.CreateCell();

_rulesListViewModel.AddRule(generator,
"Ромбічна гранецентрована");

        generator = new HexagonalGenerator()
        {
           IEGamma = false,
           IEB = false
        };
        generator.CreateCell();

_rulesListViewModel.AddRule(generator,
"Гексагональна");
      }

      private void
_rulesListViewModel_RightRuleSelected(object
sender,
RulesListViewModel.ListItemViewModel e)
      {
        var graph = e.Graph;

        _rule.SetRightPart(graph, e.RightColor);
        _rule.RepaintLeftMain(e.LeftColor);
        _leftViewport.Graph = _rule.Left;
        _leftViewport.InvalidateGraphVisual();
        _constructor = e.Constructor;
        _rightViewport.Graph = graph;
        _rightViewport.InvalidateGraphVisual();
        _stateManager.LatticeGenerator =
_constructor;
        _ruleViewModel.NotifyColorChanged();
      }

      void ConfigureDefaultRule()
      {
        Color4 leftColor = Defaults.LeftColor;
        IGraph left =
Graph.SingleNodeGraph(leftColor);
```

```
        Color4 rightColor = Defaults.RightColor;
        IGraph right =
Graph.SingleNodeGraph(rightColor);

        if (Rule.TryCreateRule(left, right,
leftColor, rightColor, out var rule) ==
RuleErrorCode.Succeed)
        {
            _rule = rule;
        }

        _ruleViewModel = new
LeftRuleViewModel(_rule)
        {
            LeftViewport = _leftViewport.Control,
            RightViewport =
_rightViewport.Control
        };

        _leftViewport.Graph = left;
        _rightViewport.Graph = right;
    }

    private bool IsParameter(string name)
    {
        return name == nameof(_constructor.A)
            || name == nameof(_constructor.B)
            || name == nameof(_constructor.C)
            || name == nameof(_constructor.Alpha)
            || name == nameof(_constructor.Beta)
            || name ==
nameof(_constructor.Gamma);
    }
  }
}
```

1.58. ICommandManager.cs

```
using System;
using System.Windows.Input;

namespace FractalGraph.UI.Services
{
  public class CommandEventArgs : EventArgs
  {
    public object Value { get; }
    public string CommandId { get; }

    public CommandEventArgs(object value,
string commandId)
    {
      Value = value;
      CommandId = commandId;
    }
  }

  public interface ICommandManager
  {
    ICommand ShowSettingsCommand { get; }
    ICommand ShowMetricsCommand { get; }
    ICommand SaveAsCommand { get; }
    ICommand LoadCommand { get; }

    ICommand SubstitutionCommand { get; }
    ICommand
ClearPrimaryViewportCommand { get; }
    ICommand RepaintCommand { get; }

    ICommand FixBottomNodesCommand {
get; }
    ICommand ExportToLiraCommand { get; }

    void OnCommandCall(object sender,
CommandEventArgs args);
    event EventHandler<CommandEventArgs>
CommandCalled;
  }
}
```

1.59. CommandManager.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Input;
using FractalGraph.Abstractions;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.Services;
using FractalGraph.UI.Commands;
using FractalGraph.UI.Controllers;
using MetricsService;

namespace FractalGraph.UI.Services
{
  public class CommandManager :
ICommandManager
  {
    IServiceProvider _services;
```

```
        IStateManager _stateManager;
        ILiraService _liraService;
        IUIService _uiService;
        IMetricsService _metricsService;

        IGraphViewportController
_viewportController;
        IConstructor _constructor;

        RelayCommand<object>
_showSettingsCommand;
        RelayCommand<object>
_showMetricsCommand;
        RelayCommand<object>
_saveAsCommand;
        RelayCommand<object> _loadCommand;

        RelayCommand<object>
_substitutionCommand;
        RelayCommand<object>
_clearPrimaryViewportCommand;
        RelayCommand<object>
_repaintCommand;

        RelayCommand<object>
_exportToLiraCommand;
        RelayCommand<object>
_fixBottomNodesCommand;

        public CommandManager(IServiceProvider
services)
        {
            _services = services;
            _stateManager =
_services.GetService<IStateManager>();
            _liraService =
_services.GetService<ILiraService>();
            _uiService =
_services.GetService<IUIService>();
            _metricsService =
_services.GetService<IMetricsService>();

            var graph =
Graph.SingleNodeGraph(Defaults.LeftColor);
            _constructor = new Constructor(graph);
            _constructor.CurrentRule =
_stateManager.CurrentRule;

_constructor.SetMetrics(_metricsService);
            _stateManager.Constructor =
_constructor;
            _stateManager.StateChanged +=
_stateManager_StateChanged;

            _viewportController =
_stateManager.PrimaryViewportController;

_viewportController.AddVisual(graph.Nodes);
            _constructor.Repainted +=
_constructor_Repainted;
        }

        private void
_stateManager_StateChanged(object sender,
StateEventArgs e)
        {
            if (e.PropertyName ==
nameof(_stateManager.CurrentRule))
                _constructor.CurrentRule =
_stateManager.CurrentRule;
        }

        private void _constructor_Repainted(object
sender, IReadOnlyCollection<INode> e)
        {
            _viewportController.RepaintVisual(e);
        }

        public ICommand ShowSettingsCommand
=>
            _showSettingsCommand ??
(_showSettingsCommand = new
RelayCommand<object>(_ =>
            {
                //TODO
            }));

        public ICommand ShowMetricsCommand
=>
            _showMetricsCommand ??
(_showMetricsCommand = new
RelayCommand<object>(_ =>
            {
                _uiService.ShowMetrics();

            }));

        public ICommand SaveAsCommand =>
```

```
        _saveAsCommand ??
(_saveAsCommand = new
RelayCommand<object>(_ =>
        {
          //TODO
        }));

      public ICommand LoadCommand =>
        _loadCommand ?? (_loadCommand =
new RelayCommand<object>(_ =>
        {
          //TODO
        }));

      public ICommand SubstitutionCommand
=>
        _substitutionCommand ??
(_substitutionCommand = new
RelayCommand<object>(_ =>
        {
          _constructor.PerformSubstitution();

_constructor.ApplyStiffness(_stateManager.Defa
ultStiffness);

_viewportController.AddVisual(_constructor.Ad
dedNodes);

_viewportController.AddVisual(_constructor.Ad
dedEdges);
          _stateManager.SubsCount++;
          UpdateStatistics();
        }));

      public ICommand
ClearPrimaryViewportCommand =>
        _clearPrimaryViewportCommand ??
(_clearPrimaryViewportCommand = new
RelayCommand<object>(_ =>
        {
          _constructor.Invalidate();
          _viewportController.Graph =
_constructor.Graph;

_viewportController.InvalidateGraphVisual();
          _stateManager.SubsCount = 0;
          UpdateStatistics();
        }));

      public ICommand RepaintCommand =>
        _repaintCommand ?? (_repaintCommand
= new RelayCommand<object>(_ =>
        {
          _constructor.Repaint();
        }));


      public ICommand
FixBottomNodesCommand =>
        _fixBottomNodesCommand ??
(_fixBottomNodesCommand = new
RelayCommand<object>(_ =>
        {
          _constructor.FixBottomNodes();

        }));

      public ICommand ExportToLiraCommand
=>
        _exportToLiraCommand ??
(_exportToLiraCommand = new
RelayCommand<object>(_ =>
        {

_liraService.Create(_constructor.Graph);
        }));

      private void UpdateStatistics()
      {
        _stateManager.NodesCount =
_constructor.Graph.Nodes.Count;
        _stateManager.EdgesCount =
_constructor.Graph.Edges.Count;
      }

      public void OnCommandCall(object sender,
CommandEventArgs args)
      {
        CommandCalled?.Invoke(sender, args);
      }

      public event
EventHandler<CommandEventArgs>
CommandCalled;
    }
}
```

1.60. StateManager.cs

```
using System;
using System.Collections.Generic;
using System.Runtime.CompilerServices;
using FractalGraph.Abstractions;
using FractalGraph.Services.LatticeGenerators;
using FractalGraph.UI.Controllers;
using FractalGraph.UI.Views;

namespace FractalGraph.UI.Services
{
   public class StateEventArgs : EventArgs
   {
      public object Value { get; }
      public string PropertyName { get; }

      public StateEventArgs(object value, string
propertyName)
      {
         Value = value;
         PropertyName = propertyName;
      }
   }

   public interface IStateManager
   {
      float A { get; set; }
      float B { get; set; }
      float C { get; set; }
      float Alpha { get; set; }
      float Beta { get; set; }
      float Gamma { get; set; }

      bool IEA { get; }
      bool IEB { get; }
      bool IEC { get; }
      bool IEAlpha { get; }
      bool IEBeta { get; }
      bool IEGamma { get; }

      int SubsCount { get; set; }
      int NodesCount { get; set; }
      int EdgesCount { get; set; }

      IConstructor Constructor { get; set; }
      LatticeGenerator LatticeGenerator { get; set;
}
      IRule CurrentRule { get; set; }
      IGraphViewportController
PrimaryViewportController { get; set; }
      IUIView RulesListView { get; set; }
      IUIView RuleView { get; set; }

      UITerminalEntity UISelected { get; set; }
      ITerminalEntity SelectedTerminal { get; }

      Stiffness DefaultStiffness { get; }
      Bar DefaultBar { get; }
      SectionI DefaultSectionI { get; }

      event EventHandler<StateEventArgs>
StateChanged;
   }

   public class StateManager : IStateManager
   {
      LatticeGenerator _latticeGenerator;
      IConstructor _constructor;
      IRule _currentRule;
      IGraphViewportController
_primaryViewportController;
      IUIView _rulesListView;
      IUIView _ruleView;
      UITerminalEntity _uiSelected;

      int _subsCount;
      int _nodesCount;
      int _edgesCount;

      bool _enableNotifications = true;
      public event
EventHandler<StateEventArgs> StateChanged;

      public StateManager()
      {
         ConfigureDefaultStiffness();
      }

      void ConfigureDefaultStiffness()
      {
         DefaultBar = new Bar()
         {
            E = 3_000_000,
            V = 0.2,
            B = 20,
            H = 20,
            Ro = 2.5
```

```
        };

        DefaultSectionI = new SectionI()
        {
            E = 210_000_000,
            V = 0.2,
            B = 10,
            H = 40,
            H1 = 10,
            B1 = 20,
            H2 = 10,
            B2 = 20,
            Ro = 7.85,
        };
    }

    bool Set<T>(ref T backingField, T value,
[CallerMemberName] string propertyName = "")
    {
        if
(EqualityComparer<T>.Default.Equals(backing
Field, value))
        {
            return false;
        }

        backingField = value;

        if (_enableNotifications)
            StateChanged?.Invoke(this, new
StateEventArgs(value, propertyName));

        return true;
    }

    void OnStateChanged(object value,
[CallerMemberName] string propertyName = "")
    {
        if (_enableNotifications)
            StateChanged?.Invoke(this, new
StateEventArgs(value, propertyName));
    }

    void OnStateChanged(string propertyName)
    {
        if (_enableNotifications)
            StateChanged?.Invoke(this, new
StateEventArgs(null, propertyName));
    }

    public float A
    {
        get => _latticeGenerator == null ? 0 :
_latticeGenerator.A;
        set
        {
            if (_latticeGenerator == null ||
_latticeGenerator.A == value)
                return;

            if (!IEB)
            {
                _latticeGenerator.B = value;
            }
            if (!IEC)
            {
                _latticeGenerator.C = value;
            }

            _latticeGenerator.A = value;
            OnStateChanged(_latticeGenerator.A);
        }
    }

    public float B
    {
        get => _latticeGenerator == null ? 0 :
_latticeGenerator.B;
        set
        {
            if (_latticeGenerator == null ||
_latticeGenerator.B == value)
                return;

            _latticeGenerator.B = value;
            OnStateChanged(_latticeGenerator.B);
        }
    }

    public float C
    {
        get => _latticeGenerator == null ? 0 :
_latticeGenerator.C;
        set
        {
            if (_latticeGenerator == null ||
_latticeGenerator.C == value)
                return;
```

```
            _latticeGenerator.C = value;
            OnStateChanged(_latticeGenerator.C);
        }
    }

    public float Alpha
    {
        get => _latticeGenerator == null ? 0 :
_latticeGenerator.Alpha;
        set
        {
            if (_latticeGenerator == null ||
_latticeGenerator.Alpha == value)
                return;

            _latticeGenerator.Alpha = value;

            if (!IEBeta && !IEC)
            {
                _latticeGenerator.Beta = value;
                _latticeGenerator.Gamma = value;
            }


OnStateChanged(_latticeGenerator.Alpha);
        }
    }

    public float Beta
    {
        get => _latticeGenerator == null ? 0 :
_latticeGenerator.Beta;
        set
        {
            if (_latticeGenerator == null ||
_latticeGenerator.Beta == value)
                return;

            _latticeGenerator.Beta = value;

OnStateChanged(_latticeGenerator.Beta);
        }
    }

    public float Gamma
    {
        get => _latticeGenerator == null ? 0 :
_latticeGenerator.Gamma;
        set
        {
            if (_latticeGenerator == null ||
_latticeGenerator.Gamma == value)
                return;

            _latticeGenerator.Gamma = value;

OnStateChanged(_latticeGenerator.Gamma);
        }
    }

    public bool IEA
    {
        get => _latticeGenerator == null ? false :
_latticeGenerator.IEA;
    }

    public bool IEB
    {
        get => _latticeGenerator == null ? false :
_latticeGenerator.IEB;
    }

    public bool IEC
    {
        get => _latticeGenerator == null ? false :
_latticeGenerator.IEC;
    }

    public bool IEAlpha
    {
        get => _latticeGenerator == null ? false :
_latticeGenerator.IEAlpha;
    }

    public bool IEBeta
    {
        get => _latticeGenerator == null ? false :
_latticeGenerator.IEBeta;
    }

    public bool IEGamma
    {
        get => _latticeGenerator == null ? false :
_latticeGenerator.IEGamma;
    }

    public IConstructor Constructor
```

```
    {
        get => _constructor;
        set => Set(ref _constructor, value);
    }

    public LatticeGenerator LatticeGenerator
    {
        get => _latticeGenerator;
        set
        {
            if (Set(ref _latticeGenerator, value))
            {
                OnStateChanged(nameof(A));
                OnStateChanged(nameof(B));
                OnStateChanged(nameof(C));
                OnStateChanged(nameof(Alpha));
                OnStateChanged(nameof(Beta));
                OnStateChanged(nameof(Gamma));
                OnStateChanged(nameof(IEA));
                OnStateChanged(nameof(IEB));
                OnStateChanged(nameof(IEC));
                OnStateChanged(nameof(IEAlpha));
                OnStateChanged(nameof(IEBeta));

OnStateChanged(nameof(IEGamma));
            }
        }
    }

    public int SubsCount
    {
        get => _subsCount;
        set => Set(ref _subsCount, value);
    }

    public int NodesCount
    {
        get => _nodesCount;
        set => Set(ref _nodesCount, value);
    }

    public int EdgesCount
    {
        get => _edgesCount;
        set => Set(ref _edgesCount, value);
    }

    public IRule CurrentRule
    {
        get => _currentRule;
        set => Set(ref _currentRule, value);
    }

    public IGraphViewportController
PrimaryViewportController
    {
        get => _primaryViewportController;
        set => Set(ref
_primaryViewportController, value);
    }

    public IUIView RulesListView
    {
        get => _rulesListView;
        set => Set(ref _rulesListView, value);
    }

    public IUIView RuleView
    {
        get => _ruleView;
        set => Set(ref _ruleView, value);
    }

    public UITerminalEntity UISelected
    {
        get => _uiSelected;
        set
        {
            if (Set(ref _uiSelected, value))

OnStateChanged(nameof(SelectedTerminal));
        }
    }

    public ITerminalEntity SelectedTerminal
    {
        get => _uiSelected?.Terminal;
    }

    public Stiffness DefaultStiffness =>
DefaultSectionI;
    public Bar DefaultBar { get; private set; }
    public SectionI DefaultSectionI { get;
private set; }
  }
}
```

1.61. UIService.cs

```
using System;
using System.Collections.Generic;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.UI.Controllers;
using FractalGraph.UI.ViewModels;
using FractalGraph.UI.Views;
using MetricsUI;
using MetricsService;

namespace FractalGraph.UI.Services
{
   public interface IUIService
   {
      IUIView CreateView<T>(T viewModel);
      IGraphViewportController
CreateGraphViewportController();
      void SetMetricsOwner(IMetricsOwner
metricsOwner);
      void ShowMetrics();
   }

   public class UIService : IUIService
   {
      IServiceProvider _services;
      IMetricsService _metricsService;

      IMetricsOwner _metricsOwner;
      MetricsWindow _metricsWindow;


      Dictionary<Type, Func<object, IUIView>>
_viewsFactory
         = new Dictionary<Type, Func<object,
IUIView>>();

      public UIService(IServiceProvider services)
      {
         _services = services;
         _metricsService =
services.GetService<IMetricsService>();
         ConfigureViewsViewModels();
      }

      void ConfigureViewsViewModels()
      {

_viewsFactory.Add(typeof(ViewportViewModel
), viewModel => new ViewportView() {
ViewModel =
(ViewportViewModel)viewModel });

_viewsFactory.Add(typeof(MainViewModel),
viewModel => new MainView() { ViewModel =
(MainViewModel)viewModel });

_viewsFactory.Add(typeof(LeftSideMenuView
Model), viewModel => new
LeftSideMenuView() { ViewModel =
(LeftSideMenuViewModel)viewModel });

_viewsFactory.Add(typeof(RightSideMenuView
Model), viewModel => new
RightSideMenuView() { ViewModel =
(RightSideMenuViewModel)viewModel });

_viewsFactory.Add(typeof(RulesListViewMode
l), viewModel => new RulesListView() {
ViewModel =
(RulesListViewModel)viewModel });

_viewsFactory.Add(typeof(LeftRuleViewModel
), viewModel => new RuleView() { ViewModel
= (LeftRuleViewModel)viewModel });
      }

      public IUIView CreateView<T>(T
viewModel)
      {
         if
(_viewsFactory.TryGetValue(typeof(T), out var
createView))
         {
            return createView(viewModel);
         }
         return null;
      }

      public IGraphViewportController
CreateGraphViewportController()
      {
         return new
GraphViewportController(_services);
      }

      public void ShowMetrics()
      {
```

```
            if (_metricsWindow == null)
            {
                _metricsWindow = new
MetricsWindow(_metricsService,
_metricsOwner);
                _metricsWindow.Closed +=
_metricsWindow_Closed;
                _metricsWindow.Show();
            }
        }

        private void
_metricsWindow_Closed(object sender,
EventArgs e)
        {
            if (_metricsWindow != null)
            {
                _metricsWindow.Closed -=
_metricsWindow_Closed;
                _metricsWindow = null;
            }
        }

        public void
SetMetricsOwner(IMetricsOwner
metricsOwner)
        {
            _metricsOwner = metricsOwner;
        }
    }
}
```

1.62. BaseViewModel.cs

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace FractalGraph.UI.ViewModels
{
    public class BaseViewModel :
INotifyPropertyChanged
    {
        private bool disablePropertyChangedEvent
= false;
        public bool DisablePropertyChangedEvent
        {
            set
            {
                if (disablePropertyChangedEvent ==
value)
                {
                    return;
                }
                disablePropertyChangedEvent = value;
                RaisePropertyChanged();
            }
            get
            {
                return disablePropertyChangedEvent;
            }
        }

        public event PropertyChangedEventHandler
PropertyChanged;
        protected void
RaisePropertyChanged([CallerMemberName]
string propertyName = "")
        {
            if (!DisablePropertyChangedEvent)
                PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
        }

        protected void
RaisePropertyChanged(PropertyChangedEventA
rgs args)
        {
            if (!DisablePropertyChangedEvent)
                PropertyChanged?.Invoke(this, args);
        }

        protected bool Set<T>(ref T backingField,
T value, [CallerMemberName] string
propertyName = "")
        {
            if
(EqualityComparer<T>.Default.Equals(backing
Field, value))
            {
                return false;
            }

            backingField = value;

this.RaisePropertyChanged(propertyName);
            return true;
        }
```

```
        protected bool Set<T>(ref T backingField,
T value, bool raisePropertyChanged,
[CallerMemberName] string propertyName = "")
        {
            if
(EqualityComparer<T>.Default.Equals(backing
Field, value))
            {
                return false;
            }

            backingField = value;
            if (raisePropertyChanged)
            {

this.RaisePropertyChanged(propertyName);
            }
            return true;
        }
    }
}
```

1.63. LeftRuleViewModel.cs

```
using System;
using System.Windows.Input;
using SharpDX;
using HelixToolkit.Wpf.SharpDX;
using FractalGraph.Abstractions;
using FractalGraph.UI.Commands;
using FractalGraph.UI.Views;
using Media = System.Windows.Media;

namespace FractalGraph.UI.ViewModels
{
    public class LeftRuleViewModel :
BaseViewModel
    {
        IUIView _leftViewport;
        IUIView _rightViewport;

        RelayCommand<object>
_leftColorClickedCommand;
        RelayCommand<object>
_rightColorClickedCommand;

        IRule _rule;

        public event EventHandler<Color4>
LeftPartRepainted;
        public event EventHandler<Color4>
RightPartRepainted;

        public LeftRuleViewModel(IRule rule)
        {
            _rule = rule;
        }

        public void NotifyColorChanged()
        {

RaisePropertyChanged(nameof(LeftColor));

RaisePropertyChanged(nameof(RightColor));
        }

        public IUIView LeftViewport
        {
            get => _leftViewport;
            set => Set(ref _leftViewport, value);
        }

        public IUIView RightViewport
        {
            get => _rightViewport;
            set => Set(ref _rightViewport, value);
        }

        public Color4 LeftColor
        {
            get => _rule == null ? Defaults.LeftColor
: _rule.MainLeftColor;
            set
            {
                if (_rule == null || _rule.MainLeftColor
== value)
                    return;

                _rule.RepaintLeftMain(value);
                RaisePropertyChanged();
                LeftPartRepainted.Invoke(this,
_rule.MainLeftColor);
            }
        }

        public Color4 RightColor
        {
```

```
        get => _rule == null ?
Defaults.RightColor : _rule.MainRightColor;
        set
        {
          if (_rule == null ||
_rule.MainRightColor == value)
            return;

          _rule.RepaintRightMain(value);
          RaisePropertyChanged();
          RightPartRepainted.Invoke(this,
_rule.MainRightColor);
        }
      }

      public ICommand
LeftColorClickedCommand =>
        _leftColorClickedCommand ??
(_leftColorClickedCommand = new
RelayCommand<object>(_ =>
        {
          var colorDialog = new
System.Windows.Forms.ColorDialog();
          if (colorDialog.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
          {
            var col = colorDialog.Color;
            var mediaColor =
Media.Color.FromArgb(col.A, col.R, col.G,
col.B);
            LeftColor =
mediaColor.ToColor4();
          }
        }));

      public ICommand
RightColorClickedCommand =>
        _rightColorClickedCommand ??
(_rightColorClickedCommand = new
RelayCommand<object>(_ =>
        {
          var colorDialog = new
System.Windows.Forms.ColorDialog();
          if (colorDialog.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
          {
            var col = colorDialog.Color;
            var mediaColor =
Media.Color.FromArgb(col.A, col.R, col.G,
col.B);
            RightColor =
mediaColor.ToColor4();
          }
        }));
    }
}
```

1.64. LeftSideMenuViewModel.cs

```
using System;
using System.Collections.Generic;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.UI.Services;

namespace FractalGraph.UI.ViewModels
{
  public class LeftSideMenuViewModel :
BaseViewModel
  {
    IServiceProvider _services;
    IStateManager _stateManager;

    Dictionary<string, Action<string, object>>
_reactions
      = new Dictionary<string, Action<string,
object>>();

    public
LeftSideMenuViewModel(IServiceProvider
services)
    {
      _services = services;
      _stateManager =
_services.GetService<IStateManager>();
      _stateManager.StateChanged +=
_stateManager_StateChanged;

      InitReactions();
    }

    private void
_stateManager_StateChanged(object sender,
StateEventArgs e)
    {
```

```
        if
(_reactions.TryGetValue(e.PropertyName, out
var action))
        {
            action.Invoke(e.PropertyName,
e.Value);
        }
    }

    private void InitReactions()
    {
        _reactions[nameof(_stateManager.A)]
            = (_, value) =>
RaisePropertyChanged(nameof(A));

        _reactions[nameof(_stateManager.B)]
            = (_, value) =>
RaisePropertyChanged(nameof(B));

        _reactions[nameof(_stateManager.C)]
            = (_, value) =>
RaisePropertyChanged(nameof(C));


_reactions[nameof(_stateManager.Alpha)]
            = (_, value) =>
RaisePropertyChanged(nameof(Alpha));

        _reactions[nameof(_stateManager.Beta)]
            = (_, value) =>
RaisePropertyChanged(nameof(Beta));


_reactions[nameof(_stateManager.Gamma)]
            = (_, value) =>
RaisePropertyChanged(nameof(Gamma));

        _reactions[nameof(_stateManager.IEA)]
            = (_, value) =>
RaisePropertyChanged(nameof(IEA));

        _reactions[nameof(_stateManager.IEB)]
         = (_, value) =>
RaisePropertyChanged(nameof(IEB));

        _reactions[nameof(_stateManager.IEC)]
         = (_, value) =>
RaisePropertyChanged(nameof(IEC));


_reactions[nameof(_stateManager.IEAlpha)]
        = (_, value) =>
RaisePropertyChanged(nameof(IEAlpha));


_reactions[nameof(_stateManager.IEBeta)]
        = (_, value) =>
RaisePropertyChanged(nameof(IEBeta));


_reactions[nameof(_stateManager.IEGamma)]
        = (_, value) =>
RaisePropertyChanged(nameof(IEGamma));


_reactions[nameof(_stateManager.SubsCount)]
         = (_, value) =>
RaisePropertyChanged(nameof(SubsCount));


_reactions[nameof(_stateManager.NodesCount)]
         = (_, value) =>
RaisePropertyChanged(nameof(NodesCount));


_reactions[nameof(_stateManager.EdgesCount)]
         = (_, value) =>
RaisePropertyChanged(nameof(EdgesCount));
    }

    public float A
    {
        get => _stateManager.A;
        set => _stateManager.A = value;
    }

    public float B
    {
        get => _stateManager.B;
        set => _stateManager.B = value;

    }

    public float C
    {
        get => _stateManager.C;
        set => _stateManager.C = value;
    }
```

```
        public float Alpha
        {
            get => _stateManager.Alpha;
            set => _stateManager.Alpha = value;
        }

        public float Beta
        {
            get => _stateManager.Beta;
            set => _stateManager.Beta = value;
        }

        public float Gamma
        {
            get => _stateManager.Gamma;
            set => _stateManager.Gamma = value;
        }

        public bool IEA => _stateManager.IEA;

        public bool IEB => _stateManager.IEB;

        public bool IEC => _stateManager.IEC;

        public bool IEAlpha =>
_stateManager.IEAlpha;

        public bool IEBeta =>
_stateManager.IEBeta;

        public bool IEGamma =>
_stateManager.IEGamma;

        public int SubsCount
        {
            get => _stateManager.SubsCount;
            set => _stateManager.SubsCount = value;
        }
        public int NodesCount
        {
            get => _stateManager.NodesCount;
            set => _stateManager.NodesCount =
value;
        }
        public int EdgesCount
        {
            get => _stateManager.EdgesCount;
            set => _stateManager.EdgesCount =
value;
        }
    }
}
```

1.65. MainViewModel.cs

```
using System;
using System.Windows.Input;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.UI.Services;
using FractalGraph.UI.Views;

namespace FractalGraph.UI.ViewModels
{
    class MainViewModel : BaseViewModel
    {
        IServiceProvider _services;
        IUIService _uiService;
        IStateManager _stateManager;
        ICommandManager _commandManager;

        LeftSideMenuViewModel
_leftSideMenuViewModel;
        RightSideMenuViewModel
_rightSideMenuViewModel;

        public MainViewModel(IServiceProvider
services)
        {
            _services = services;
            _uiService =
_services.GetService<IUIService>();
            _stateManager =
_services.GetService<IStateManager>();
            _commandManager =
_services.GetService<ICommandManager>();
            SetViews();
        }

        void SetViews()
        {
            _leftSideMenuViewModel = new
LeftSideMenuViewModel(_services);
            LeftSideMenu =
_uiService.CreateView(_leftSideMenuViewMod
el);
```

```
        _rightSideMenuViewModel = new
RightSideMenuViewModel(_services);
        RightSideMenu =
_uiService.CreateView(_rightSideMenuViewM
odel);

        PrimaryViewportView =
_stateManager.PrimaryViewportController.Cont
rol;
        RulesListView =
_stateManager.RulesListView;
        RuleView = _stateManager.RuleView;
      }
      public IUIView LeftSideMenu { get; private
set; }
      public IUIView RightSideMenu { get;
private set; }
      public IUIView PrimaryViewportView {
get; private set; }
      public IUIView RulesListView { get;
private set; }
      public IUIView RuleView { get; private set;
}

      public ICommand ShowSettingsCommand
=> _commandManager.ShowSettingsCommand;
      public ICommand ShowMetricsCommand
=> _commandManager.ShowMetricsCommand;
      public ICommand SaveAsCommand =>
_commandManager.SaveAsCommand;
      public ICommand LoadCommand =>
_commandManager.LoadCommand;

      public ICommand SubstitutionCommand
=> _commandManager.SubstitutionCommand;
      public ICommand
ClearPrimaryViewportCommand =>
_commandManager.ClearPrimaryViewportCom
mand;
      public ICommand RepaintCommand =>
_commandManager.RepaintCommand;

      public ICommand
FixBottomNodesCommand =>
_commandManager.FixBottomNodesCommand
;
      public ICommand ExportToLiraCommand
=> _commandManager.ExportToLiraCommand;
   }
}
```

1.66. MainWindowViewModel.cs

```
using System;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.UI.Services;
using FractalGraph.UI.Views;

namespace FractalGraph.UI.ViewModels
{
   public class MainWindowViewModel :
BaseViewModel
   {
      IServiceProvider _services;
      IUIService _uiService;
      IStateManager _stateManager;
      ICommandManager _commandManager;
      MainViewModel _mainViewModel;

      public
MainWindowViewModel(IServiceProvider
services)
      {
         _services = services;
         _uiService =
_services.GetService<IUIService>();
         _mainViewModel = new
MainViewModel(_services);
         MainView =
_uiService.CreateView(_mainViewModel);
         _stateManager =
_services.GetService<IStateManager>();
         _commandManager =
_services.GetService<ICommandManager>();
      }

      public IUIView MainView { get; }
   }
}
```

1.67. RightSideMenuViewModel.cs

```
using System;
using FractalGraph.Abstractions;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.UI.Services;

namespace FractalGraph.UI.ViewModels
{
```

```
   public class RightSideMenuViewModel :
BaseViewModel
   {
      IServiceProvider _services;
      IStateManager _stateManager;
      public
RightSideMenuViewModel(IServiceProvider
services)
      {
         _services = services;
         _stateManager =
_services.GetService<IStateManager>();
         _stateManager.StateChanged +=
_stateManager_StateChanged;
      }

      private void
_stateManager_StateChanged(object sender,
StateEventArgs e)
      {
         if(e.PropertyName ==
nameof(SelectedTerminal))

RaisePropertyChanged(nameof(SelectedTermina
l));
      }

      public ITerminalEntity SelectedTerminal
=> _stateManager.SelectedTerminal;
   }
}
```

1.68. RulesListViewModel.cs

```
using System;
using System.Collections.ObjectModel;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.Abstractions;
using FractalGraph.UI.Controllers;
using FractalGraph.UI.Services;
using FractalGraph.UI.Views;
using SharpDX;
using FractalGraph.Services.LatticeGenerators;

namespace FractalGraph.UI.ViewModels
{
   public class RulesListViewModel :
BaseViewModel
   {
      public class ListItemViewModel
      {
         public LatticeGenerator Constructor {
get; }
         public IGraphViewportController
ViewportController { get; }
         public IUIView ViewportView =>
ViewportController.Control;
         public string Title { get; }
         public IGraph Graph =>
Constructor.Graph;
         public Color4 LeftColor =>
Constructor.LeftColor;
         public Color4 RightColor =>
Constructor.RightColor;

         public
ListItemViewModel(IGraphViewportController
viewportController, string title, LatticeGenerator
constructor)
         {
            ViewportController =
viewportController;
            Title = title;
            Constructor = constructor;
            Constructor.LaticeUpdated +=
Constructor_LaticeUpdated;
         }

         private void
Constructor_LaticeUpdated(object sender,
IGraph e)
         {
            ViewportController.Graph = Graph;

ViewportController.InvalidateGraphVisual();
         }
      }

      IUIService _uiService;

      ListItemViewModel
_selectedItemViewModel;
      public event
EventHandler<ListItemViewModel>
RightRuleSelected;
      public ListItemViewModel
SelectedItemViewModel
      {
```

```
        get => _selectedItemViewModel;
        set
        {
            if (Set(ref _selectedItemViewModel,
value))
                RightRuleSelected?.Invoke(this,
_selectedItemViewModel);
        }
    }

    public
ObservableCollection<ListItemViewModel>
ItemsViewModels { get; }
        = new
ObservableCollection<ListItemViewModel>();

    public
RulesListViewModel(IServiceProvider services)
    {
        _uiService =
services.GetService<IUIService>();
    }

    public void AddRule(LatticeGenerator
constructor, string title)
    {
        var viewport =
_uiService.CreateGraphViewportController();

viewport.AddVisual(constructor.Graph.Nodes);

viewport.AddVisual(constructor.Graph.Edges);
        var item = new
ListItemViewModel(viewport, title, constructor);
        viewport.CameraPos = new Vector3(0.5f,
-3, 0.5f);
        viewport.CameraLook = new Vector3(0,
3, 0);
        viewport.ViewportClicked += (_, e) =>
SelectedItemViewModel = item;
        ItemsViewModels.Add(item);

        //added first elemt
        if (ItemsViewModels.Count == 1)
            SelectedItemViewModel = item;
    }
  }
}
```

1.69. ViewportViewModel.cs

```
using System;
using System.Windows.Media;
using System.Windows.Input;
using HelixToolkit.Wpf.SharpDX;
using HelixToolkit.Wpf.SharpDX.Model.Scene;
using FractalGraph.Abstractions.Extensions;
using FractalGraph.UI.Commands;
using Media3D =
System.Windows.Media.Media3D;

namespace FractalGraph.UI.ViewModels
{
  public class ViewportViewModel :
BaseViewModel
  {
    string _title;
    string _subTitle;
    bool _showInfoPanel;
    bool _showAxis;
    bool _showViewCube;

    IEffectsManager _effectsManager;

    //camera
    CameraType _cameraModel;
    Camera _camera;
    Media3D.Vector3D _upDirection = new
Media3D.Vector3D(0, 0, 1);

    //ligth
    Media3D.Vector3D
_directionalLightDirection;
    Color _directionalLightColor;
    Color _ambientLightColor;
    Color _backgroundColor;

    SceneNodeGroupModel3D _sceneTree =
new SceneNodeGroupModel3D();

    OrthographicCamera
_defaultOrthographicCamera;
    PerspectiveCamera
_defaultPerspectiveCamera;

    RelayCommand<object>
_viewportClickedCommand;
```

```
        public event EventHandler
ViewportClicked;

        #region properties
        public string Title
        {
            get => _title;
            set => Set(ref _title, value);
        }

        public string SubTitle
        {
            get => _subTitle;
            set => Set(ref _subTitle, value);
        }

        public IEffectsManager EffectsManager
        {
            get => _effectsManager;
            set => Set(ref _effectsManager, value);
        }

        public CameraType CameraModel
        {
            get => _cameraModel;
            set
            {
                if (_cameraModel == value)
                    return;

                if (_cameraModel ==
CameraType.Perspective)
                    Camera =
_defaultPerspectiveCamera;
                else if (_cameraModel ==
CameraType.Orthographic)
                    Camera =
_defaultOrthographicCamera;
            }
        }

        public Camera Camera
        {
            get => _camera;
            private set => Set(ref _camera, value);
        }

        public Media3D.Vector3D CameraLook
        {
            get => Camera.LookDirection;
            set => Camera.LookDirection = value;
        }

        public Media3D.Point3D CameraPos
        {
            get => Camera.Position;
            set => Camera.Position = value;
        }

        public Media3D.Vector3D UpDirection
        {
            get => _upDirection;
            set => Set(ref _upDirection, value);
        }

        public Media3D.Vector3D
DirectionalLightDirection
        {
            get => _directionalLightDirection;
            set => Set(ref _directionalLightDirection,
value);
        }

        public Color DirectionalLightColor
        {
            get => _directionalLightColor;
            set => Set(ref _directionalLightColor,
value);
        }

        public Color AmbientLightColor
        {
            get => _ambientLightColor;
            set => Set(ref _ambientLightColor,
value);
        }

        public Color BackgroundColor
        {
            get => _backgroundColor;
            set => Set(ref _backgroundColor, value);
        }

        public SceneNodeGroupModel3D
SceneTree
        {
            get => _sceneTree;
        }
```

```
    public GroupNode RootNode
    {
      get => _sceneTree.GroupNode;
    }

    public bool ShowInfoPanel
    {
      get => _showInfoPanel;
      set => Set(ref _showInfoPanel, value);
    }

    public bool ShowAxis
    {
      get => _showAxis;
      set => Set(ref _showAxis, value);
    }

    public bool ShowViewCube
    {
      get => _showViewCube;
      set => Set(ref _showViewCube, value);
    }
    #endregion

    public ICommand
ViewportClickedCommand =>
        _viewportClickedCommand ??
(_viewportClickedCommand = new
RelayCommand<object>(arg =>
        {
          ViewportClicked?.Invoke(this,
EventArgs.Empty);
        }));

    public ViewportViewModel()
    {
      ConfigureDefaultCameras();

      _effectsManager = new
DefaultEffectsManager();
      _camera = _defaultPerspectiveCamera;

      // setup lighting
      _ambientLightColor = Colors.DimGray;
      _directionalLightColor = Colors.White;
      _directionalLightDirection = new
Media3D.Vector3D(-2, -5, -2);
    }

    void ConfigureDefaultCameras()
    {
      _defaultOrthographicCamera = new
OrthographicCamera
      {
        Position = new Media3D.Point3D(0, -
5, 0),
        LookDirection = new
Media3D.Vector3D(-0, 5, -0),
        UpDirection = new
Media3D.Vector3D(0, 0, 1),
        NearPlaneDistance = 1,
        FarPlaneDistance = 5000000
      };

      _defaultPerspectiveCamera = new
PerspectiveCamera
      {
        Position = new Media3D.Point3D(0, -
5, 0),
        LookDirection = new
Media3D.Vector3D(-0, 5, -0),
        UpDirection = new
Media3D.Vector3D(0, 0, 1),
        NearPlaneDistance = 0.5,
        FarPlaneDistance = 5000000
      };
    }
  }
}
```

1.70. IUIView.cs

```
using FractalGraph.UI.ViewModels;

namespace FractalGraph.UI.Views
{
  public interface IUIView
  {
    BaseViewModel ViewModel { get; set; }
  }
}
```

1.71. LeftSideMenuView.xaml

```
<UserControl
x:Class="FractalGraph.UI.Views.LeftSideMenu
View"
```

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:c="clr-namespace:FractalGraph.UI.Converters"
        xmlns:local="clr-namespace:FractalGraph.UI.Views"
        mc:Ignorable="d"
        d:DesignHeight="450" d:DesignWidth="800">
  <UserControl.Resources>
    <c:RadiansToDegreesConverter x:Key="RadToDegConverter"/>
  </UserControl.Resources>
  <Grid>
    <StackPanel>
      <GroupBox Header="Загальна інформація">
        <StackPanel>
          <TextBlock>
        <Run Text="Кількіть замін: "/>
        <Run Text="{Binding SubsCount}"/>
          </TextBlock>
          <TextBlock>
        <Run Text="Кількіть вузлів: "/>
        <Run Text="{Binding NodesCount}"/>
          </TextBlock>
          <TextBlock>
        <Run Text="Кількіть ребер: "/>
        <Run Text="{Binding EdgesCount}"/>
          </TextBlock>
        </StackPanel>
      </GroupBox>

      <GroupBox Header="Довжини ребер">
        <Grid>
          <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto"/>
            <ColumnDefinition Width="*"/>
          </Grid.ColumnDefinitions>
          <Grid.RowDefinitions>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
          </Grid.RowDefinitions>

          <Label Grid.Row="0" Grid.Column="0" Content="a"/>
          <TextBox Grid.Row="0" Grid.Column="1"  BorderThickness="0,0,0,1" VerticalAlignment="Center"

PreviewTextInput="NumberValidationTextBox"
                   Text="{Binding A, StringFormat=N3, UpdateSourceTrigger=PropertyChanged}">
            <TextBox.Style>
              <Style TargetType="TextBox">
                <Style.Triggers>
                  <DataTrigger Binding="{Binding IEA}" Value="False">
                    <Setter Property="IsEnabled" Value="False"/>
                  </DataTrigger>
                </Style.Triggers>
              </Style>
            </TextBox.Style>
          </TextBox>

          <Label Grid.Row="1" Grid.Column="0" Content="b"/>
          <TextBox Grid.Row="1" Grid.Column="1" BorderThickness="0,0,0,1" VerticalAlignment="Center"

PreviewTextInput="NumberValidationTextBox"
                   Text="{Binding B, StringFormat=N3, UpdateSourceTrigger=PropertyChanged}">
            <TextBox.Style>
              <Style TargetType="TextBox">
                <Style.Triggers>
                  <DataTrigger Binding="{Binding IEB}" Value="False">
                    <Setter Property="IsEnabled" Value="False"/>
                  </DataTrigger>
                </Style.Triggers>
```

```
                </Style>
              </TextBox.Style>
            </TextBox>

            <Label Grid.Row="2"
Grid.Column="0" Content="c"/>
            <TextBox Grid.Row="2"
Grid.Column="1" BorderThickness="0,0,0,1"
VerticalAlignment="Center"

PreviewTextInput="NumberValidationTextBox"
                Text="{Binding C,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}">
              <TextBox.Style>
                <Style TargetType="TextBox">
                  <Style.Triggers>
                    <DataTrigger
Binding="{Binding IEC}" Value="False">
                      <Setter
Property="IsEnabled" Value="False"/>
                    </DataTrigger>
                  </Style.Triggers>
                </Style>
              </TextBox.Style>
            </TextBox>

          </Grid>
        </GroupBox>

        <GroupBox Header="Кути">
          <Grid>
            <Grid.ColumnDefinitions>
              <ColumnDefinition
Width="auto"/>
              <ColumnDefinition Width="*"/>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
            </Grid.RowDefinitions>

            <Label Grid.Row="0"
Grid.Column="0" Content="alpha"/>
            <TextBox Grid.Row="0"
Grid.Column="1" BorderThickness="0,0,0,1"
VerticalAlignment="Center"
PreviewTextInput="NumberValidationTextBox"
                Text="{Binding Alpha,
StringFormat=N2,
UpdateSourceTrigger=PropertyChanged,
Converter={StaticResource
RadToDegConverter}}">
              <TextBox.Style>
                <Style TargetType="TextBox">
                  <Style.Triggers>
                    <DataTrigger
Binding="{Binding IEAlpha}" Value="False">
                      <Setter
Property="IsEnabled" Value="False"/>
                    </DataTrigger>
                  </Style.Triggers>
                </Style>
              </TextBox.Style>
            </TextBox>

            <Label Grid.Row="1"
Grid.Column="0" Content="beta"/>
            <TextBox Grid.Row="1"
Grid.Column="1" BorderThickness="0,0,0,1"
VerticalAlignment="Center"

PreviewTextInput="NumberValidationTextBox"
                Text="{Binding Beta,
StringFormat=N2,
UpdateSourceTrigger=PropertyChanged,
Converter={StaticResource
RadToDegConverter}}">
              <TextBox.Style>
                <Style TargetType="TextBox">
                  <Style.Triggers>
                    <DataTrigger
Binding="{Binding IEBeta}" Value="False">
                      <Setter
Property="IsEnabled" Value="False"/>
                    </DataTrigger>
                  </Style.Triggers>
                </Style>
              </TextBox.Style>
            </TextBox>

            <Label Grid.Row="2"
Grid.Column="0" Content="gamma"/>
```

```
            <TextBox Grid.Row="2"
Grid.Column="1" BorderThickness="0,0,0,1"
VerticalAlignment="Center"

PreviewTextInput="NumberValidationTextBox"
                  Text="{Binding Gamma,
StringFormat=N2,
UpdateSourceTrigger=PropertyChanged,
Converter={StaticResource
RadToDegConverter}}">
                <TextBox.Style>
                  <Style TargetType="TextBox">
                    <Style.Triggers>
                      <DataTrigger
Binding="{Binding IEGamma}"
Value="False">
                        <Setter
Property="IsEnabled" Value="False"/>
                      </DataTrigger>
                    </Style.Triggers>
                  </Style>
                </TextBox.Style>
              </TextBox>
            </Grid>
          </GroupBox>
        </StackPanel>
    </Grid>
</UserControl>
```

1.72. LeftSideMenuView.xaml.cs

```
using System.Text.RegularExpressions;
using System.Windows.Controls;
using System.Windows.Input;
using FractalGraph.UI.ViewModels;

namespace FractalGraph.UI.Views
{
    /// <summary>
    /// Interaction logic for
LeftSideMenuView.xaml
    /// </summary>
    public partial class LeftSideMenuView :
UserControl, IUIView
    {
        public LeftSideMenuView()
        {
            InitializeComponent();
        }

        public BaseViewModel ViewModel
        {
            get => DataContext as BaseViewModel;
            set
            {
                if (DataContext == value)
                    return;

                DataContext = value;
            }
        }

        private void
NumberValidationTextBox(object sender,
TextCompositionEventArgs e)
        {
            Regex regex = new Regex("[0-9]+");
            var isMatch = regex.IsMatch(e.Text);
            e.Handled = !isMatch;
        }
    }
}
```

1.73. MainView.xaml

```
<UserControl
x:Class="FractalGraph.UI.Views.MainView"

xmlns="http://schemas.microsoft.com/winfx/200
6/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2
006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/
markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expressi
on/blend/2008"
        xmlns:local="clr-
namespace:FractalGraph.UI.Views"
        mc:Ignorable="d"
        d:DesignHeight="450"
d:DesignWidth="800">
    <DockPanel>
        <StackPanel DockPanel.Dock="Top"
Orientation="Horizontal">
```

```
        <Button Margin="4"
Content="Налаштування"
Command="{Binding
ShowSettingsCommand}" />
        <Button Margin="4"
Content="Метрики" Command="{Binding
ShowMetricsCommand}" />
        <Button Margin="4" Content="Зберегти
як" Command="{Binding SaveAsCommand}"
/>
        <Button Margin="4"
Content="Завантажити" Command="{Binding
LoadCommand}" />

        <Button Margin="4"
Content="Очистити сцену"
Command="{Binding
ClearPrimaryViewportCommand}" />
        <Button Margin="4"
Content="Виконати підстановку"
Command="{Binding SubstitutionCommand}"
/>
        <Button Margin="4"
Content="Перефарбувати"
Command="{Binding RepaintCommand}" />

        <Button Margin="4"
Content="Закріпити нижчі вузли"
Command="{Binding
FixBottomNodesCommand }"  />
        <Button Margin="4"
Content="Перенести до ліри"
Command="{Binding ExportToLiraCommand
}" />
      </StackPanel>
      <Grid>
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="auto"/>
          <ColumnDefinition Width="*"/>
          <ColumnDefinition Width="auto"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
          <RowDefinition Height="3*"/>
          <RowDefinition Height="1.2*"/>
          <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <Border Grid.Row="0"
BorderThickness="0 0 4 0"
BorderBrush="Gray">
          <ContentPresenter Content="{Binding
LeftSideMenu}"/>
        </Border>

        <ContentPresenter Grid.Column="1"
Grid.Row="0" Content="{Binding
PrimaryViewportView}"/>

        <Border Grid.ColumnSpan="3"
Grid.Row="1" BorderThickness="0 4 0 4"
BorderBrush="Gray">
          <ContentPresenter Content="{Binding
RulesListView}"/>
        </Border>

        <Border Grid.ColumnSpan="3"
Grid.Row="2">
          <ContentPresenter Content="{Binding
RuleView}" HorizontalAlignment="Center"/>
        </Border>

        <Border Grid.Row="0"
Grid.Column="2" BorderThickness="4 0 0 0"
BorderBrush="Gray">
          <ContentPresenter Content="{Binding
RightSideMenu}"/>
        </Border>
      </Grid>
    </DockPanel>
</UserControl>
```

1.74. MainView.xaml.cs

```
using System.Windows.Controls;
using FractalGraph.UI.ViewModels;

namespace FractalGraph.UI.Views
{
    /// <summary>
    /// Interaction logic for MainView.xaml
    /// </summary>
    public partial class MainView : UserControl,
IUIView
    {
        public MainView()
        {
```

```
        InitializeComponent();
    }

    public BaseViewModel ViewModel
    {
        get => DataContext as BaseViewModel;
        set
        {
            if (DataContext == value)
                return;

            DataContext = value;
        }
    }
  }
}
```

1.75. RightSideMenuView.xaml

```
<UserControl
x:Class="FractalGraph.UI.Views.RightSideMen
uView"

xmlns="http://schemas.microsoft.com/winfx/200
6/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2
006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/
markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expressi
on/blend/2008"
        xmlns:local="clr-
namespace:FractalGraph.UI.Views"

xmlns:b="http://schemas.microsoft.com/xaml/be
haviors"
        xmlns:abstract="clr-
namespace:FractalGraph.Abstractions;assembly
=FractalGraph.Abstractions"
        mc:Ignorable="d"
        d:DesignHeight="450"
d:DesignWidth="800">
  <UserControl.Resources>

    <DataTemplate DataType="{x:Type
abstract:Bar}">
      <StackPanel>
        <Image Source="/Views/Bar.jpg" />
        <Grid HorizontalAlignment="Center">
          <Grid.ColumnDefinitions>
            <ColumnDefinition
Width="auto"/>
            <ColumnDefinition
Width="150"/>
            <ColumnDefinition
Width="auto"/>
          </Grid.ColumnDefinitions>
          <Grid.RowDefinitions>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
          </Grid.RowDefinitions>

          <TextBlock Grid.Row="0"
Grid.Column="0" Text="E" />
          <TextBox Grid.Row="0"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
              Text="{Binding E,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
          <TextBlock Grid.Row="0"
Grid.Column="3" Text="T/M²" />

          <TextBlock Grid.Row="1"
Grid.Column="0" Text="V" />
          <TextBox Grid.Row="1"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
              Text="{Binding V,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />

          <TextBlock Grid.Row="2"
Grid.Column="0" Text="B" />
          <TextBox Grid.Row="2"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"
```

```
PreviewTextInput="NumberValidationTextBox"
                    Text="{Binding B,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
            <TextBlock Grid.Row="2"
Grid.Column="3" Text="CM" />

            <TextBlock Grid.Row="3"
Grid.Column="0" Text="H" />
            <TextBox Grid.Row="3"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                    Text="{Binding H,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
            <TextBlock Grid.Row="3"
Grid.Column="3" Text="CM" />

            <TextBlock Grid.Row="4"
Grid.Column="0" Text="Ro" />
            <TextBox Grid.Row="4"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                    Text="{Binding Ro,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
            <TextBlock Grid.Row="4"
Grid.Column="3" Text="T/M³" />
          </Grid>
        </StackPanel>
      </DataTemplate>

      <DataTemplate DataType="{x:Type
abstract:SectionI}">
        <StackPanel>
          <Image Source="/Views/SectionI.jpg"
/>
          <Grid HorizontalAlignment="Center">
            <Grid.ColumnDefinitions>
              <ColumnDefinition
Width="auto"/>
              <ColumnDefinition
Width="150"/>
              <ColumnDefinition
Width="auto"/>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
            </Grid.RowDefinitions>

            <TextBlock Grid.Row="0"
Grid.Column="0" Text="E" />
            <TextBox Grid.Row="0"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                    Text="{Binding E,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
            <TextBlock Grid.Row="0"
Grid.Column="3" Text="T/M²" />

            <TextBlock Grid.Row="1"
Grid.Column="0" Text="V" />
            <TextBox Grid.Row="1"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                    Text="{Binding V,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />

            <TextBlock Grid.Row="2"
Grid.Column="0" Text="B" />
            <TextBox Grid.Row="2"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                    Text="{Binding B,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
```

```
            <TextBlock Grid.Row="2"
Grid.Column="3" Text="CM" />

            <TextBlock Grid.Row="3"
Grid.Column="0" Text="H" />
            <TextBox Grid.Row="3"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                  Text="{Binding H,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
            <TextBlock Grid.Row="3"
Grid.Column="3" Text="CM" />

            <TextBlock Grid.Row="4"
Grid.Column="0" Text="B1" />
            <TextBox Grid.Row="4"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                  Text="{Binding B1,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
            <TextBlock Grid.Row="4"
Grid.Column="3" Text="CM" />

            <TextBlock Grid.Row="5"
Grid.Column="0" Text="H1" />
            <TextBox Grid.Row="5"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                  Text="{Binding H1,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
            <TextBlock Grid.Row="5"
Grid.Column="3" Text="CM" />

            <TextBlock Grid.Row="6"
Grid.Column="0" Text="B2" />
            <TextBox Grid.Row="6"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                  Text="{Binding B2,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
            <TextBlock Grid.Row="6"
Grid.Column="3" Text="CM" />

            <TextBlock Grid.Row="7"
Grid.Column="0" Text="H2" />
            <TextBox Grid.Row="7"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                  Text="{Binding H2,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
            <TextBlock Grid.Row="7"
Grid.Column="3" Text="CM" />

            <TextBlock Grid.Row="8"
Grid.Column="0" Text="Ro" />
            <TextBox Grid.Row="8"
Grid.Column="1" BorderThickness="0,0,0,1"
Margin="2,0,0,2"

PreviewTextInput="NumberValidationTextBox"
                  Text="{Binding Ro,
StringFormat=N3,
UpdateSourceTrigger=PropertyChanged}" />
            <TextBlock Grid.Row="8"
Grid.Column="3" Text="T/M³" />
          </Grid>
        </StackPanel>
      </DataTemplate>

      <DataTemplate DataType="{x:Type
abstract:Edge}">
        <StackPanel>
          <ComboBox>
            <ComboBoxItem>
              <TextBlock>Брус</TextBlock>
            </ComboBoxItem>
            <ComboBoxItem>

<TextBlock>Двотавр</TextBlock>
            </ComboBoxItem>
          </ComboBox>
          <ContentControl Content="{Binding
Stiffness}"/>
```

```
        </StackPanel>
      </DataTemplate>

      <DataTemplate DataType="{x:Type
abstract:Node}">
        <StackPanel>
          <TextBlock Text="Вузол:"/>
          <Grid HorizontalAlignment="Center">
            <Grid.ColumnDefinitions>
              <ColumnDefinition
Width="auto"/>
              <ColumnDefinition
Width="auto"/>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
              <RowDefinition Height="auto"/>
            </Grid.RowDefinitions>

            <TextBlock Grid.Row="0"
Grid.Column="0" Text="X" />
            <CheckBox  Grid.Row="0"
Grid.Column="1" IsChecked="{Binding
Restraints.X}" />

            <TextBlock Grid.Row="1"
Grid.Column="0" Text="Y" />
            <CheckBox  Grid.Row="1"
Grid.Column="1" IsChecked="{Binding
Restraints.Y}" />

            <TextBlock Grid.Row="2"
Grid.Column="0" Text="Z" />
            <CheckBox  Grid.Row="2"
Grid.Column="1" IsChecked="{Binding
Restraints.Z}" />

            <TextBlock Grid.Row="3"
Grid.Column="0" Text="UX" />
            <CheckBox  Grid.Row="3"
Grid.Column="1" IsChecked="{Binding
Restraints.UX}" />

            <TextBlock Grid.Row="4"
Grid.Column="0" Text="UY" />
            <CheckBox  Grid.Row="4"
Grid.Column="1" IsChecked="{Binding
Restraints.UY}" />

            <TextBlock Grid.Row="5"
Grid.Column="0" Text="UZ" />
            <CheckBox  Grid.Row="5"
Grid.Column="1" IsChecked="{Binding
Restraints.UZ}" />

            <TextBlock Grid.Row="6"
Grid.Column="0" Text="W" />
            <CheckBox  Grid.Row="6"
Grid.Column="1" IsChecked="{Binding
Restraints.W}" />
          </Grid>
        </StackPanel>
      </DataTemplate>

    </UserControl.Resources>
    <Grid MinWidth="300">
      <Grid.RowDefinitions>
        <RowDefinition Height="*"/>
      </Grid.RowDefinitions>
      <ContentControl MinWidth="300"
Content="{Binding SelectedTerminal}" />
    </Grid>
</UserControl>
```

1.76. RightSideMenuView.xaml.cs

```
using System.Windows.Controls;
using System.Windows.Input;
using System.Text.RegularExpressions;
using FractalGraph.UI.ViewModels;

namespace FractalGraph.UI.Views
{
    /// <summary>
    /// Interaction logic for
RightSideMenuView.xaml
    /// </summary>
    public partial class RightSideMenuView :
UserControl, IUIView
    {
        public RightSideMenuView()
        {
```

```
        InitializeComponent();
    }

    public BaseViewModel ViewModel
    {
        get => DataContext as BaseViewModel;
        set
        {
            if (DataContext == value)
                return;

            DataContext = value;
        }
    }

    private void
NumberValidationTextBox(object sender,
TextCompositionEventArgs e)
    {
        Regex regex = new Regex("[0-9]+");
        var isMatch = regex.IsMatch(e.Text);
        e.Handled = !isMatch;
    }
  }
}
```

1.77. RulesListView.xaml

```
<UserControl
x:Class="FractalGraph.UI.Views.RulesListView
"

xmlns="http://schemas.microsoft.com/winfx/200
6/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2
006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/
markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expressi
on/blend/2008"
        xmlns:local="clr-
namespace:FractalGraph.UI.Views"
        mc:Ignorable="d"
        d:DesignHeight="450"
d:DesignWidth="800">
  <Grid>
    <ListBox ItemsSource="{Binding
ItemsViewModels}" SelectedItem="{Binding
SelectedItemViewModel}">
      <ListBox.ItemTemplate>
        <DataTemplate>
          <Border BorderBrush="Gray"
BorderThickness="1">
            <Grid>
              <Grid.RowDefinitions>
                <RowDefinition
Height="*"/>
                <RowDefinition
Height="auto"/>
              </Grid.RowDefinitions>
              <Grid Width="250"
Height="150">
                <ContentPresenter
Content="{Binding ViewportView}"/>
              </Grid>
              <TextBlock Grid.Row="1"
Text="{Binding Title}"
HorizontalAlignment="Center"/>
            </Grid>
          </Border>
        </DataTemplate>
      </ListBox.ItemTemplate>

      <ListBox.ItemsPanel>
        <ItemsPanelTemplate>
          <VirtualizingStackPanel
IsItemsHost="True" Orientation="Horizontal"/>
        </ItemsPanelTemplate>
      </ListBox.ItemsPanel>

      <ListBox.Template>
        <ControlTemplate
TargetType="ItemsControl">
          <ScrollViewer
HorizontalScrollBarVisibility="Visible"
VerticalScrollBarVisibility="Disabled">
            <ItemsPresenter/>
          </ScrollViewer>
        </ControlTemplate>
      </ListBox.Template>
    </ListBox>

  </Grid>
</UserControl>
```

1.78. RulesListView.xaml.cs

```
using FractalGraph.UI.ViewModels;
using System.Windows.Controls;

namespace FractalGraph.UI.Views
{
   /// <summary>
   /// Interaction logic for RulesListView.xaml
   /// </summary>
   public partial class RulesListView :
UserControl, IUIView
   {
      public RulesListView()
      {
         InitializeComponent();
      }

      public BaseViewModel ViewModel
      {
         get => DataContext as BaseViewModel;
         set
         {
            if (DataContext == value)
               return;

            DataContext = value;
         }
      }
   }
}
```

1.79. RuleView.xaml

```
<UserControl
x:Class="FractalGraph.UI.Views.RuleView"

xmlns="http://schemas.microsoft.com/winfx/200
6/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2
006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/
markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expressi
on/blend/2008"
xmlns:b="http://schemas.microsoft.com/xaml/be
haviors"
         xmlns:c="clr-
namespace:FractalGraph.UI.Converters"
         xmlns:local="clr-
namespace:FractalGraph.UI.Views"
         mc:Ignorable="d"
         d:DesignHeight="450"
d:DesignWidth="800">
   <UserControl.Resources>
      <c:ColorToBrushConverter
x:Key="ColorConverter"/>
   </UserControl.Resources>
   <UniformGrid Rows="1" Columns="2"
HorizontalAlignment="Stretch">
      <Grid  HorizontalAlignment="Center">
         <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto" />
            <ColumnDefinition Width="auto" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="auto" />
            </Grid.ColumnDefinitions>
         <TextBlock Grid.Column="0" Text="1)"
FontSize="128" FontFamily="Fira Code"
Foreground="Gray"
VerticalAlignment="Center"/>
         <Grid Grid.Column="1" Width="250"
Height="auto">
            <ContentPresenter Content="{Binding
LeftViewport}"/>
         </Grid>
         <Viewbox Grid.Column="2"
Stretch="Uniform" Width="100" Margin="4 0 4
0">
            <Path
               Fill="Gray"
               Stretch="Fill"
               HorizontalAlignment="Center"
               Data="M1 8a.5.5 0 0 1 .5-
.5h11.793l-3.147-3.146a.5.5 0 0 1 .708-.708l4
4a.5.5 0 0 1 0 .708l-4 4a.5.5 0 0 1-.708-
.708L13.293 8.5H1.5A.5.5 0 0 1 1 8z"
               />
         </Viewbox>
         <Grid Grid.Column="3" Width="250"
Height="auto">
            <ContentPresenter Content="{Binding
RightViewport}"/>
```

```
        </Grid>
    </Grid>

    <!--<Grid HorizontalAlignment="Center"
Grid.Column="1" Width="100">-->
    <Grid  HorizontalAlignment="Center">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <TextBlock Grid.Column="0" Text="2)"
FontSize="128" FontFamily="Fira Code"
Foreground="Gray"
VerticalAlignment="Center"/>
        <Rectangle Grid.Column="1"
Name="LeftRect" Fill="{Binding LeftColor,
Converter={StaticResource ColorConverter}}"
MinWidth="125">
            <b:Interaction.Triggers>
                <b:EventTrigger
EventName="MouseDown">
                    <b:InvokeCommandAction
Command="{Binding
LeftColorClickedCommand}"/>
                </b:EventTrigger>
            </b:Interaction.Triggers>
        </Rectangle>

        <Viewbox Grid.Column="2"
Stretch="Uniform">
            <Path Fill="Gray"
                Stretch="Fill"
                HorizontalAlignment="Center"
                Data="M11.5 15a.5.5 0 0 0 .5-
.5V2.707l3.146 3.147a.5.5 0 0 0 .708-.708l-4-
4a.5.5 0 0 0-.708 0l-4 4a.5.5 0 1 0 .708.708L11
2.707V14.5a.5.5 0 0 0 .5.5zm-7-14a.5.5 0 0 1
.5.5v11.793l3.146-3.147a.5.5 0 0 1 .708.708l-4
4a.5.5 0 0 1-.708 0l-4-4a.5.5 0 0 1 .708-.708L4
13.293V1.5a.5.5 0 0 1 .5-.5zZ">
                <Path.LayoutTransform>
                    <TransformGroup>
                        <RotateTransform Angle="90"
/>
                        <ScaleTransform ScaleY="-
1"/>
                    </TransformGroup>
                </Path.LayoutTransform>
            </Path>
        </Viewbox>

        <Rectangle Grid.Column="3"
Fill="{Binding RightColor,
Converter={StaticResource ColorConverter}}"
MinWidth="125">
            <b:Interaction.Triggers>
                <b:EventTrigger
EventName="MouseDown">
                    <b:InvokeCommandAction
Command="{Binding
RightColorClickedCommand}"/>
                </b:EventTrigger>
            </b:Interaction.Triggers>
        </Rectangle>
    </Grid>
  </UniformGrid>
</UserControl>
```

1.80. RuleView.xaml.cs

```
using System.Windows.Controls;
using FractalGraph.UI.ViewModels;

namespace FractalGraph.UI.Views
{
  /// <summary>
  /// Interaction logic for RuleView.xaml
  /// </summary>
  public partial class RuleView : UserControl,
IUIView
  {
    public RuleView()
    {
      InitializeComponent();
    }

    public BaseViewModel ViewModel
    {
      get => DataContext as BaseViewModel;
      set
      {
        if (DataContext == value)
          return;

        DataContext = value;
```

```
        }
      }
    }
}
```

1.81. ViewportView.xaml

```
<UserControl
x:Class="FractalGraph.UI.Views.ViewportView
"

xmlns="http://schemas.microsoft.com/winfx/200
6/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2
006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/
markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expressi
on/blend/2008"
        xmlns:hx="http://helix-
toolkit.org/wpf/SharpDX"

xmlns:b="http://schemas.microsoft.com/xaml/be
haviors"
        xmlns:local="clr-
namespace:FractalGraph.UI.Views"
        mc:Ignorable="d"
        d:DesignHeight="450"
d:DesignWidth="800">
  <UserControl.Resources>
    <BooleanToVisibilityConverter
x:Key="VisibleIfTrueConverter"/>
  </UserControl.Resources>
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="*"/>
      <RowDefinition Height="auto"/>
    </Grid.RowDefinitions>
    <Border BorderThickness="0.5">
      <hx:Viewport3DX
        x:Name="Viewport"
        Title="{Binding Title}"
        BackgroundColor="White"
        Camera="{Binding Camera}"
CoordinateSystemLabelForeground="LightGray
"
        EffectsManager="{Binding
EffectsManager}"
        EnableDesignModeRendering="False"
        FXAALevel="Low"
EnableSwapChainRendering="True"
        EnableD2DRendering = "False"
        ModelUpDirection="{Binding
UpDirection}"
        ShowCoordinateSystem="{Binding
ShowAxis}"
        SubTitle="{Binding SubTitle}"
        TextBrush="Black"
        UseDefaultGestures="False"
        ShowFrameRate="True"
        ShowFrameDetails="True"
        ShowViewCube="{Binding
ShowViewCube}">
        <hx:Viewport3DX.InputBindings>
          <KeyBinding Key="B"
Command="hx:ViewportCommands.BackView
" />
          <KeyBinding Key="F"
Command="hx:ViewportCommands.FrontView
" />
          <KeyBinding Key="U"
Command="hx:ViewportCommands.TopView"
/>
          <KeyBinding Key="D"
Command="hx:ViewportCommands.BottomVie
w" />
          <KeyBinding Key="L"
Command="hx:ViewportCommands.LeftView"
/>
          <KeyBinding Key="R"
Command="hx:ViewportCommands.RightView
" />
          <KeyBinding
Command="hx:ViewportCommands.ZoomExte
nts" Gesture="Control+E" />
          <MouseBinding
Command="hx:ViewportCommands.Rotate"
Gesture="RightClick" />
          <MouseBinding
Command="hx:ViewportCommands.Zoom"
Gesture="MiddleClick" />
```

```
            <MouseBinding
Command="hx:ViewportCommands.Pan"
Gesture="LeftClick" />
          </hx:Viewport3DX.InputBindings>
          <hx:AmbientLight3D
Color="{Binding AmbientLightColor}" />
          <hx:DirectionalLight3D
Direction="{Binding Camera.LookDirection}"
          Color="{Binding
DirectionalLightColor}" />

          <hx:PostEffectMeshBorderHighlight
            EffectName="highlightSelected"
            Color="Red" />

        </hx:Viewport3DX>

        <b:Interaction.Triggers>
          <b:EventTrigger
EventName="PreviewMouseDown">
            <b:InvokeCommandAction
Command="{Binding
ViewportClickedCommand}"/>
          </b:EventTrigger>
        </b:Interaction.Triggers>
      </Border>
      <StatusBar Grid.Row="1"
VerticalAlignment="Bottom"
Visibility="{Binding ShowInfoPanel,
        Converter={StaticResource
VisibleIfTrueConverter}}">
        <StatusBar.Background>
          <SolidColorBrush Opacity="0.5"
Color="WhiteSmoke" />
        </StatusBar.Background>
        <StatusBarItem>
          <TextBlock Grid.Row="0"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Text="{Binding FrameRate,
ElementName=Viewport, StringFormat=D3D11
- \{0:0.00\} FPS}" />
        </StatusBarItem>
        <Separator />
        <StatusBarItem>
          <TextBlock Text="{Binding
Camera.Position, StringFormat=Position:
\{0:0.0\}}" />
        </StatusBarItem>
        <Separator />
        <StatusBarItem>
          <TextBlock Text="{Binding
Camera.LookDirection,
StringFormat=LookDirection: \{0:0.0\}}" />
        </StatusBarItem>
        <Separator />
        <StatusBarItem>
          <TextBlock Text="{Binding
Camera.UpDirection,
StringFormat=UpDirection: \{0:0.0\}}" />
        </StatusBarItem>
        <Separator />
        <StatusBarItem>
          <TextBlock Text="{Binding
Items.Count, ElementName=Viewport,
StringFormat=Children: \{0\}}" />
        </StatusBarItem>
      </StatusBar>
    </Grid>
</UserControl>
```

1.82. ViewportView.xaml.cs

```
using FractalGraph.UI.ViewModels;
using System.Windows.Controls;

namespace FractalGraph.UI.Views
{
  /// <summary>
  /// Interaction logic for ViewportView.xaml
  /// </summary>
  public partial class ViewportView :
UserControl, IUIView
  {
    public ViewportView()
    {
      InitializeComponent();
    }

    public BaseViewModel ViewModel
    {
      get => DataContext as BaseViewModel;
      set
      {
        if (DataContext == value)
          return;

        DataContext = value;
```

```
            if (value is ViewportViewModel
viewModel)
            {

Viewport.Items.Add(viewModel.SceneTree);
            }
         }
      }
   }
}
```

1.83. App.xaml

```
<Application x:Class="FractalGraph.UI.App"

xmlns="http://schemas.microsoft.com/winfx/200
6/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2
006/xaml"
         xmlns:local="clr-
namespace:FractalGraph.UI"
         StartupUri="MainWindow.xaml">
   <Application.Resources>

   </Application.Resources>
</Application>
```

1.84. App.xaml.cs

```
using System.Windows;

namespace FractalGraph.UI
{
   /// <summary>
   /// Interaction logic for App.xaml
   /// </summary>
   public partial class App : Application
   {
   }
}
```

1.85. MainWindow.xaml

```
<Window
x:Class="FractalGraph.UI.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/200
6/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2
006/xaml"

xmlns:d="http://schemas.microsoft.com/expressi
on/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/
markup-compatibility/2006"
      xmlns:local="clr-
namespace:FractalGraph.UI"
      mc:Ignorable="d"
      Title="MainWindow" Height="450"
Width="800">
   <ContentPresenter Content="{Binding
MainView}"/>
</Window>
```

1.86. MainWindow.xaml.cs

```
using System;
using System.Windows;
using FractalGraph.UI.Controllers;
using FractalGraph.UI.Services;
using FractalGraph.UI.ViewModels;
using FractalGraph.UI.Views;
using FractalGraph.Abstractions.Extensions;
using MetricsService;

namespace FractalGraph.UI
{
   /// <summary>
   /// Interaction logic for MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window,
IUIView, IMetricsOwner
   {
      MainController _mainController;
      IServiceProvider _services;
      IUIService _uiService;

      public MainWindow()
      {
         _mainController = new
MainController(this);
         InitializeComponent();
      }

      public BaseViewModel ViewModel
```

```
        {
            get => DataContext as BaseViewModel;
            set
            {
                if (DataContext == value)
                    return;

                DataContext = value;
            }
        }

        public void InitServices(IServiceProvider
services)
        {
            _services = services;
            _uiService =
services.GetService<IUIService>();
            _uiService.SetMetricsOwner(this);
        }
    }
}
```

ДОДАТОК Г

Тези

ЗАТВЕРДЖУЮ

Проректор Українського державного

університету науки і технологій

Анатолій РАДКЕВИЧ

«КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ МУЛЬТИАТРИБУ-ТИВНИХ ПРОСТОРОВИХ ГРАФОВИХ ФРАКТАЛІВ»

Тези

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Віктор ШИНКАРЕНКО
Виконавець
_____Олександр ЛЕТУЧИЙ
Нормоконтролер
_____Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.01443

«КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ МУЛЬТИАТРИБУТИВНИХ ПРОСТОРОВИХ ГРАФОВИХ ФРАКТАЛІВ»

Тези

44165850.01443

Листів

2025

Міністерство освіти і науки України Український державний університет науки і технологій

ТЕЗИ

XVIII Міжнародної науково-практичної конференції

«СУЧАСНІ ІНФОРМАЦІЙНІ ТА КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ НА ТРАНСПОРТІ, В ПРОМИСЛОВОСТІ І ОСВІТІ»

Присвячено пам'яті Владислава СКАЛОЗУБА

ABSTRACTS

of the XVIII International Conference

«MODERN INFORMATION AND COMMUNICATION TECHNOLOGIES ON A TRANSPORT, IN INDUSTRY AND EDUCATION»

Dedicated to the memory of Vladislav SKALOZUB

12.12.2024 – 13.12.2024

Дніпро
2024

А Використання ЛІРИ-САПР у клієнт-серверному додатку

Летучий О. І., Шинкаренко В. І. Український державний університет науки і технологій, Україна

Система автоматизованого проєктування (САПР) це програма або комплекс програм, які автоматизують процес створення, аналізу та вдосконалення проєктів у різних галузях: будівництво, машинобудування, суднобудування, авіація, електроніка, медицина тощо. САПР допомагає проєктувати складні об'єкти, виконувати розрахунки, моделювати поведінку систем і створювати креслення або 3D-моделі.

Серед САПР програм ЛІРА-САПР є одним із найпотужніших інструментів для моделювання, розрахунку та оптимізації несучих систем і споруд. ЛІРА-САПР надає функції розрахунку міцності, стійкості та деформацій, що дозволяє аналізувати напруження, переміщення та стабільність конструкцій. Для отримання результатів деформацій використовується моделювання впливу вітрових, сейсмічних і експлуатаційних навантажень.

Клієнт-серверна архітектура є розподіленою моделлю, у якій функції програм розділено між клієнтами, що ініціюють запити, і серверами, які забезпечують необхідний функціонал або ресурси. Такий підхід сприяє ефективному використанню ресурсів, дозволяє масштабувати систему за рахунок додавання серверів і забезпечує централізоване управління даними та безпекою.

ЛІРИ-САПР використовується як сервер за допомогою технології Component Object Model (COM). Такий підхід має переваги незалежності від мови програмування у клієнтській програми. За допомогою COM можна обгорнути усі доступні можливості програми для серверу, що у свою чергу створює умови для ефективної роботи "тонких" клієнтів, які передають значну частину обчислень на сервер, дозволяючи задіяти менш продуктивні пристрої для клієнтської сторони.

Як приклад програми клієнту наведемо програму зі створення мульти-атрибутивних графових фракталів. Програма написана мові C#, у об’єктно-орієнтованому стилі, на фреймворку .NET Framework 4.7.2. Для розробки зовнішнього інтерфейсу використано фреймворк WPF, для 3D графіки використовується бібліотека Helix. Для відображення даних використовується патерн MVVM із запровадженням ін’єкції залежностей у вигляді сервісів та менеджерів даних.

Для формування фракталів використовується підхід конструктивно-продукційної моделювання (КПМ) на основі формальних граматик.

Для розширення графового фракталу задіяні операції підстановки та виведення, для здійснення яких необхідна інформація про вузли та ребра, тобто атрибути позиції вершин, їх розмір, колір тощо. До даних атрибутів можна додати тип матеріалу, жорсткість, тип навантаження, напрямок навантаження, спосіб з’єднання вузлів з ребрами; усі ці атрибути характерні для САПР програм.

В ЛІРА-САПР передається інформація про усі вузли, ребра та їх атрибути, проводиться аналіз за допомогою методів скінченних елементів, а результат аналізу повертається в програму-клієнт. Використання ЛІРА-САПР можливе як допоміжний інструмент з формування графових фракталів або для дослідження отриманої графової структури як конструкцію, або ж її частину, будівлі.

У свою чергу клієнт-серверної взаємодія дозволяє більш зручно оперувати самою Лірою: швидке повторення виконання аналів на попередніх результатах зі змінами від програми-клієнта, автоматизація користувальницьких сценаріїв використання ЛІРА-САПР. Користувальницькі сценарії можуть бути представлені у вигляді консольних команд, без потреби написання окремого інтерфейсу користувача.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МІНІСТЕРСТВО ІНФРАСТРУКТУРИ УКРАЇНИ
УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ ТА ТЕХНОЛОГІЙ
СХІДНИЙ НАУКОВИЙ ЦЕНТР ТРАНСПОРТНОЇ АКАДЕМІЇ НАУК

ABSTRACTS
OF THE XVII INTERNATIONAL CONFERENCE
«MODERN INFORMATION AND COMMUNICATION TECHNOLOGIES ON A TRANSPORT, IN INDUSTRY AND EDUCATION»
13-14, December, 2023

СУЧАСНІ ІНФОРМАЦІЙНІ ТА КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ НА ТРАНСПОРТІ, В ПРОМИСЛОВОСТІ ТА ОСВІТІ

ТЕЗИ

XVII МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

13-14 ГРУДНЯ 2023

ДНІПРО
2023

Засоби формування графових 3D фракталів

Летучий О. І., Шинкаренко В. І. Український державний університет науки і технологій, Україна

Фрактали — це складні геометричні фігури, які демонструють самоподібність у різних масштабах, тобто при зміні масштабу, можна побачити повторювані схожі візерунки або патерни. Вони генеруються за допомогою рекурсивних математичних формул і ітерацій. Фрактали можуть бути використані для створення складних візуальних ефектів у комп'ютерній графіці, включаючи створення складних текстур, або для генерації та моделювання різних структур, таких як річкові мережі, гірські ландшафти.

Для моделювання фракталів можна використовувати наступні підходи: алгоритмічний, функціонально-алгоритмічний із застосуванням системи ітерованих функцій на основі сукупності стискаючих відображень, афінних автоматів, L-систем. Нами представляється підхід конструктивно-продукційної моделювання (КПМ) на основі формальних граматик.

Розроблені конструктори на основі узагальненого здатні генерувати фрактальні сутності. Конструктор складається з трьох частин: неоднорідний розширюваний носій; відношення та відповідні операції: над атрибутами, зв’язування, підстановки, виведення; множина (формальних і неформальних) тверджень інформаційного забезпечення конструювання, яке включає в себе: онтологію, мету, правила, обмеження, початкові умови та умови завершення конструювання.

Конструювання передбачає низку уточнюючих перетворень: спеціалізацію, інтерпретація, конкретизація та реалізація.

Конструктивно-продукційних моделювання було застосовано для генерації кристалічної ґратки, на основі патернів Браве.

Спеціалізація конструктора стосується формування графових геометричних фігур у 3D просторі. На цьому ж етапі визначаються поняття форми, атрибутики, операції над формами та атрибутами, зокрема операції підстановки.

Інтерпретація описує основні алгоритми, які виконують згадані вище операції.

Конкретизація конструктора для кристалічної ґратки включає в себе: вершину з атрибутами кольору та розміру; кристалічну ґратку з атрибутами довжин ребер a, b, c і кути між ними; початкову умову у вигляді однієї вершини червоного кольору з розміром один, умову підстановки та операцію підстановки. Правило підстановки складається с послідовних операції заміни форми та взаємного перефарбування вершин. Використання кольору в якості умови дає змогу побачити, над якими елементами буде виконуватись заміна на ту чи іншу форму, яку можна обирати в процесі формування ґратки, контролюючи таким чином процес її формування.

Реалізація виконується програмними інструментальними засобами, у об’єктно орієнтованому парадигмі мовою C#, з фреймворком .NET Framework 4.7.2. Для розробки зовнішнього інтерфейсу використано WPF з компонентами Material Design, для 3D графіки використовується бібліотека Helix. Конструктор та сутності представлені у вигляді класів, їх атрибути у вигляд полів та властивостей класів. Усі алгоритми для формування фракталу належать конструктору, які реалізовані у вигляді методів, та сервісу-класу, який передає інформацію в 3D сцену.

Іншим можливим застосуванням графових 3D фракталів може бути моделювання композитних матеріалів із фрактальним розподілом волокон та отримання методу визначення загальних властивостей композиційних матеріалів із частинками, який враховує вплив розподілу розмірів і взаємодії між частинками на їх механічні властивості.

Фрактальні графові структури дозволяють моделювати деякі будівлі та композиційні будівельні матеріали з метою покращення їх характеристик стійкості та міцності.

Міністерство освіти і науки
Український державний університет науки і технологій
Дніпровський державний технічний університет
Дніпровський національний університет імені Олеся Гончара
Національний технічний університет «Дніпровська політехніка»
Криворізький національний університет
Харківський національний університет радіоелектроніки
Херсонський національний технічний університет
Чорноморський державний університет імені П. Могили
Aalto University (Університет Аалто, Фінляндія)
American University of Central Asia (Бішкек, Киргизстан)
Tallinna Tehnikaülikool (Таллінн, Естонія)
AGH University of Science and Technology (Краків, Польща)
Politechnika Rzeszowska (Жешув, Польща)
Ariel University (Аріель, Ізраїль)
Michigan State University (Іст-Лансінґ, США)
Leibniz Universitat Hannover, Institute of Photogrammetry and Geoinformation
(Ганновер, Німеччина)

МАТЕРІАЛИ
Міжнародної науково-технічної конференції
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ в
МЕТАЛУРГІЇ та МАШИНОБУДУВАННІ

MATERIALS
of Scientific and Technical International Conference
INFORMATION TECHNOLOGY IN
METALLURGY AND MACHINE ENGINEERING

10 - квітня 2024 року

м. Дніпро

DOI: 10.34185/1991-7848.itmm.2024.01.049

ГРАФОВІ ФРАКТАЛИ З ВАРІАТИВНІСТЮ ПРОЦЕСУ ФОРМУВАННЯ

Летучий О. І., Шинкаренко В. І.

Український державний університет науки і технологій, Україна

Анотація. *Фрактали, які характеризуються своєю самоподібністю в різних масштабах, є складними геометричними сутностями, створеними за допомогою рекурсивних алгоритмів. Вони знаходять широке застосування в комп'ютерній графіці для створення складних візуальних ефектів і моделювання природних явищ, таких як річкові мережі та гірські ландшафти. Графові фрактали поєднують властивості фракталів і графових структур та можуть використовуватись для досліджень у таких галузях, як комп'ютерні мережі чи медицина. У цій роботі представлено підхід конструктивно-продукційного моделювання, заснованого на формальних граматиках, для генерації графових фракталів із варіативністю процесу формування.*

Ключові слова. *фрактал, граф, конструктивно-продукційне моделювання, комп'ютерне моделювання, ітераційні алгоритми, формування патернів*

Вступ. Фрактали – це складні геометричні фігури, які демонструють самоподібність у різних масштабах [1], тобто при зміні масштабу, можна побачити повторювані схожі візерунки або патерни. Вони генеруються за допомогою рекурсивних афінних перетворень і ітерацій. Фрактали можуть бути використані для створення складних візуальних ефектів у комп'ютерній графіці, включаючи створення складних текстур, або для генерації та моделювання різних структур, таких як річкові мережі, гірські ландшафти [2].

Графові фрактали [3-4] поєднують фрактальні властивості та графові структури, такі як, наприклад, комп'ютерні мережі [5], кристалічні ґратки [5], структури раку легень [6]. Особливістю таких фракталів є графові вершини, які можуть бути навантажені координатами у просторі, різними фізичними або хімічними властивостями.

Для моделювання фракталів можна використовувати наступні підходи: алгоритмічний, функціонально-алгоритмічний із застосуванням системи ітерованих функцій на основі сукупності стискаючих відображень, афінних автоматів, L-систем. Нами представляється підхід конструктивно-продукційної моделювання (КПМ) на основі формальних граматик.

Основна частина. Основним елементом конструктивно-продукційного моделювання є поняття узагальненого конструктор [7, 8], який описується наступним чином

$$C = \langle M_{GI}, \Sigma_{GI}, \Lambda_{GI} \rangle, \quad (1)$$

де M – поновлюваний неоднорідний носій;

Σ – сигнатура відносин та пов'язаних з ними операцій;

Λ – інформаційне забезпечення конструювання: призначення, умови початку та завершення конструювання, правила та обмеження та онтологія.

Опис інтерпретації та спеціалізації конструктора для графових фракталів в 3D просторі наведений в [4]. Проведемо наступну конкретизацію конструктора C_{graph}, призначеного для формування графового фракталу:

$$C = \langle M_{GI}, \Sigma_{GI}, \Lambda_{GI} \rangle \ {}_K\circledR \ C_{graph} = \langle M_{GI}, \Sigma_{GI}, \Lambda_{graph} \rangle, \quad (2)$$

Λ_{graph} визначає:

- вершини – v_i, з атрибутами розміру $size$, координатами $pos = (x, y, z)$, кольору col;
- граф – g, з атрибутами довжини ребер a;
- початкові умови: граф з однією вершиною v_0, з атрибутами $size \lfloor v_0 = 1$, $pos \lfloor v_0 = (0,0,0)$, $col \lfloor v_0 = red$;
- правило підстановки: $\psi = \langle s, g \rangle$, в якому операція складається з одного відношення підстановки s (приклад у лівій частині на рис. 1) і операції над атрибутами g (приклад у правій частині рис. 1) для взаємного перефарбування вершин;

обмеження конструювання: права частина підстановки повинна містити хоча б одну вершину кольору, як у лівому правилі,

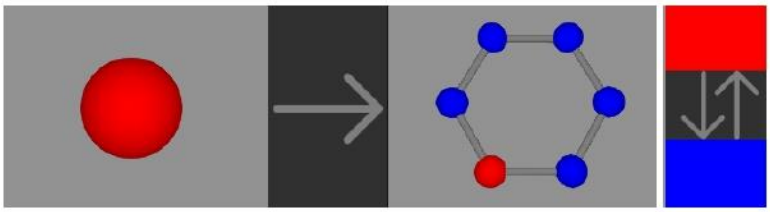


Рисунок 01 – Відношення підстановки конструктора C_{graph} з заданими кольорами заміни

Представимо на рис. 2 реалізацію у вигляді послідовності сентенційних форм f_0, f_1, f_2, f_3.

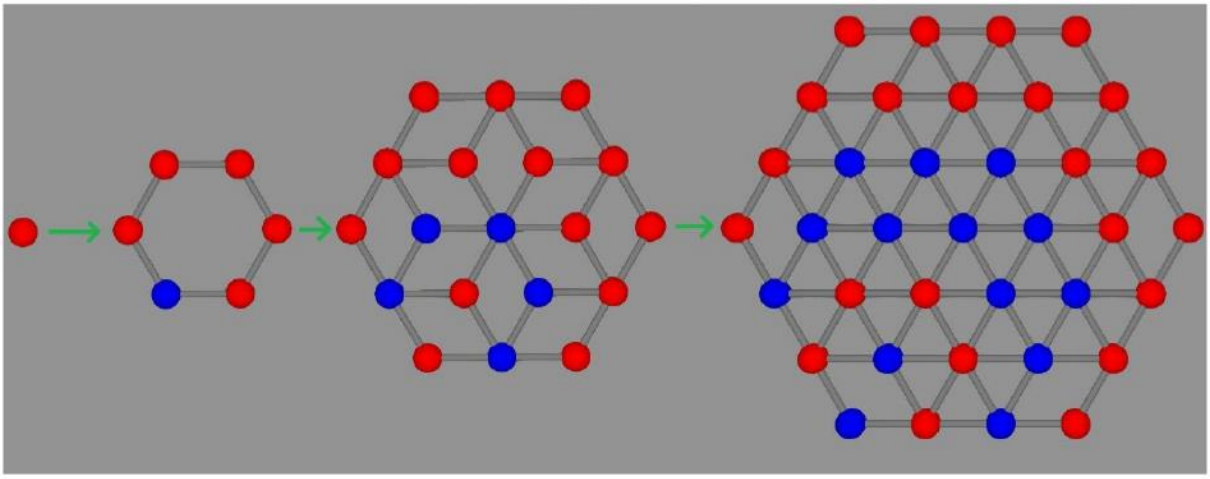


Рисунок 2 – Послідовність сентенційних форм

Реалізація здійснюється в такий спосіб:

- початкова форма f_0, відповідно до відношення підстановки на рис. 1, застосовуючи операцію заміни один раз, перетворюється на f_1;
- виконується операція з атрибутами кольору: взаємне перефарбування вершин;
- повторюються дії попереднії двох пунктів з перетворенням форм f_i у f_{i+1}.

Розглянемо приклад з варіативністю правил підстановки під час формування графових фракталів. Для цього початок формування графового фракталу виконаємо як на рис. 2 та отримаємо форму f_3. Змінимо правило підстановки як на рис. 3 а). Отримаємо граф як на рис. 4 у лівій частині. Далі будемо ітеративно застосовувати правило рис. 3 б). У правій частині рис 4. наведено двократне застосування правила рис. 3 б).

В результаті двостадійного перетворення отримуємо граф з двома ознаками фрактальності.

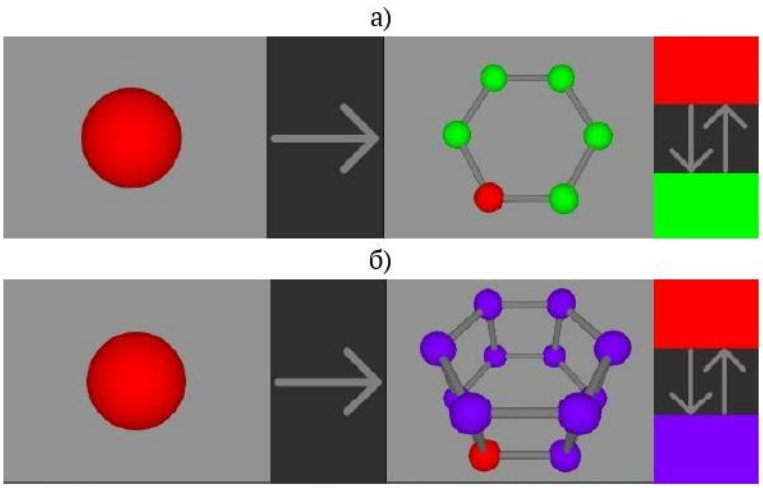


Рисунок 3 – Правила для другої стадії формування фрактального графу

На першій стадії формується фрактальний граф з самоподібністю забезпечену застосуванням правила рис 1. На другій стадії цей фрактальний граф є основою для формування об'ємного графу з новою властивістю самоподібності.

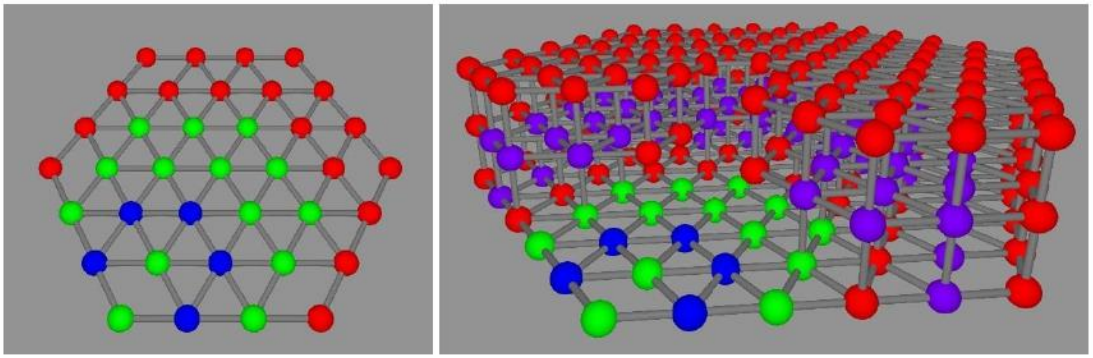


Рисунок 4 – Графи другої стадії формування

Висновок. Розроблена конструктивно-продукційна модель формування фрактальних графових структур та відповідне програмне забезпечення. Показана можливість моделювання графових фракталів з варіативністю правил підстановки та зміни атрибутів вершин у процесі формування. Таким чином проявляється самоподібність різних частин утвореного фракталу по мірі його формування.

Приклад представлений у роботі не є вичерпним. Правила підстановки та фарбування вершин може багаторазово змінюватись під час формування досить значних значень сентенційних форм f_M.

Застосований підхід дозволяє формувати моделі сутностей, які точніше представляють об'єкти природи.

ЛІТЕРАТУРА / REFERENCE

1. Husain A., Nanda M.N., Chowdary M.S., Sajid M. Fractals: An Eclectic Survey, Part-I. Fractal and Fractional. – 2022. Vol. 6. P. 2504 – 3110. DOI: https://doi.org/10.3390/fractalfract6020089
2. Frankhauser P., Pumain D. Fractals and Geography. Machine Learning and the City, S. Carta (Ed.). – 2022. P. 31-55. DOI: https://doi.org/10.1002/9781119815075.ch3
3. Ille P., Woodrow R. Fractal graphs. Journal of Graph Theory. – 2019. No. 91 (1), P. 53-72 DOI: https://doi.org/10.1002/jgt.22420
4. Skums P., Bunimovich L., Graph fractal dimension and the structure of fractal networks. J Complex Netw. – 2020. Vol. 8 DOI: https://doi.org/10.1093/comnet/cnaa037
5. Shynkarenko V., Letuchyi O., Chyhir R. Constructive-synthesizing modeling of fractal crystal lattices, 18th IEEE International Conference on Computer Science and Information Technologies (CSIT), – 2023. DOI: https://doi.org/10.1109/CSIT61576.2023.10324251
6. Babič M.; Mihelič J. Calì, M. Complex Network Characterization Using Graph Theory and Fractal Geometry: The Case Study of Lung Cancer DNA Sequences. Applied Sciences. – 2020, Vol 10. No. 9. P. 3037. DOI: https://doi.org/10.3390/app10093037
7. Skalozub V., Ilman V. Shynkarenko V. Ontological support formation for constructive-synthesizing modeling of information systems development processes, Eastern-European Journal of Enterprise Technologies. – 2018. Vol. 5. No. 4 (95). P. 55–63. DOI: https://doi.org/10.15587/1729-4061.2018.143968
8. Shynkarenko V. I. Constructive-Synthesizing Representation of Geometric Fractals, Cybernetics and Systems Analysis. – 2019. Vol. 55, P. 186-199. DOI: https://doi.org/10.1007/s10559-019-00123-w

GRAPH FRACTALS WITH THE VARIABILITY OF THE FORMATION PROCESS

Oleksandr Letuchyi, Viktor Shynkarenko

Abstract. *Fractals, which are characterized by their self-similarity at different scales, are complex geometric entities created using recursive algorithms. They are widely used in computer graphics to create complex visual effects and to model natural phenomena such as river networks and mountain landscapes. Graph fractals combine the properties of fractals and graph structures and can be used for research in fields such as computer networks or medicine. This work presents the approach of constructive-production modeling, based on formal grammars, for the generation of graph fractals with the variability of the formation process.*

Keywords: *fractals, graph, constructive-synthesizing modeling, computer modeling, iterative algorithms, pattern formation*

2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT 2023)

Lviv, Ukraine
19-21 October 2023

IEEE

IEEE Catalog Number: CFP23D36-POD
ISBN: 979-8-3503-6047-9

Constructive-synthesizing modeling of fractal crystal lattices

Viktor Shynkarenko, Oleksandr Letuchyi, Robert Chyhir
Department of Computer Information Technology
Ukrainian State University of Science and Technologies
Dnipro, Ukraine
oleksandrletuchyi@gmail.com, shinkarenko_vi@ua.fm, robertchigir@ukr.net

***Abstract* — A crystal lattice is the basic structure of many solid materials, such as metals, semiconductors, and insulators. Modeling of crystal lattices can be used for research of existing and simulation of new materials, study of crystallization processes. Constructive-synthesizing modeling tools are used to model crystal lattices. That uses the attributes of the elemental base, which allows to naturally build fractal spatial graphs according to the rules of substitution. It becomes possible to specify which vertices of the graph and to which graph should be changed at each step of graph generation. Crystal lattices are considered as partial cases of fractal spatial graphs.**

Keywords — constructive-synthesizing modeling, crystal lattice, crystal structure, Brave lattice, supercell, fractal, spatial graph

I. INTRODUCTION

The crystal structure of metals, semiconductors, and diamonds is determined by the crystal lattice. Modeling of crystal structures is widely used in various fields of science and industry. The study of crystal structures is important for the creation of new materials with improved properties. For example, in electronics, modeling crystal structures enhances the ability to design more efficient semiconductor devices.

In this work, attention is paid to the modeling of the geometric properties of the crystal structure using Brave lattices [1] and its subsequent visualization. The obtained graphic model of the crystal structure can be used for educational and scientific purposes for a better understanding of the structure of the lattice and the connections between the atoms of the crystal.

II. RELATED WORKS

The connection between the crystal structure and fractals is highlighted and proven in [2]. The crystal structure model is identified with the Cayley graph. A crystal structure can be viewed from a fractal perspective, as it reflects self-similarity, which is a key property of fractals. Each elementary lattice in a crystal has a similar structure to the entire crystal as a whole. Different models of crystal growth are proposed: static, dynamic, population, asymptotic, slice. Special attention is drawn to the dynamic model, the essence of which is the step-by-step formation of the crystal with the help of generators and relators. As the authors themselves define, this process of crystal formation is very similar to the use of L-systems, which are known to be widely used to form fractals.

Some works [3, 4] focus on the visualization of the crystal structure and the possibilities of viewing it from different angles, but they do not explain the growth methods of the crystal lattice.

The generation of crystals using Brave lattices and the Miller index is considered in [5]. Various approaches to viewing the crystal structure are proposed, each of which has its own characteristics: "asymmetric", "original", "in-cell", "packing". Unfortunately, the program described in this work does not provide functionality for iterative crystal growth, but an immediately ready crystal structure in a given volume. Using the Miller index, you can also display of crystal morphologies [6].

There is also the simplest way to generate crystals using repetitions of the initial crystal lattice over cell vectors. The use of such an approach is primitive, but it minimizes computational cost.

III. TECHNICAL DETAILS

The program is developed in the C# language in an object-oriented style using the MVVM pattern [7]. WPF is used to display the interface, and the Helix Toolkit library [8] is used to display 3D graphics.

IV. CRYSTAL MODELING

The models, developed and described in this paper, allow for the formation of a diverse class of 3D graph fractals. The task of this study is to simulate crystal lattices using Bravais patterns.

The so-called “dynamic” growth model and the use of generators and relators proposed in this model, as well as initial conditions, will be used for crystal formation.

There are different approaches to solving the problem of modeling entities that are similar to themselves: algorithmic, functional-algorithmic with the use of a system of iterated functions based on a set of compressive mappings, L-systems, compressive affine automata. Another approach is the use of constructive-synthesizing modeling (CSM), which is based on formal grammars.

The use of CSM is effective in iterative processes, where the detailing of fractals or similar entities is performed. The main idea is to create a constructor capable of generating these entities based on their characteristic attributes with the help of special rules and algorithms.

The crystal structure consists of unit cells, described as Brave lattices, which are repeated many times to form a supercell. As attributes of the supercell, it is appropriate to use the characteristics of the Brave crystal lattice: the lengths of the edges, the angle between them, the type of lattices. However, since the crystal structure can be considered as a special case of the Cayley graph, a color attribute can be added to show the growth of the crystal.

The description of CSM is given in [9, 10], where the concepts of constructors and their properties are described in detail:

"We use a generalized constructor as triple

$$C = \langle M, \Sigma, \Lambda \rangle,$$

where M is the replenished carrier of the structure, Σ is its signature consisting of sets of possible names of many-placed operations and relations such as linking, substitution, inference, and auxiliary operations; Λ is information provision of construction, which includes the goal, initial conditions, conditions for completing construction of the rule and restrictions. The sets M and Σ are defined by the axiomatics."

The constructor can be considered as a generator, and substitution rules and operations on attributes as relators.

The specialization of this designer is the formation of graph geometric shapes in space $\boldsymbol{R}^3$:

$$\mathrm{C} = \langle M, \Sigma, \Lambda \rangle_s \mapsto C_G = \langle M_G, \Sigma_g, \Lambda_G \rangle,$$

Where $\Lambda_G = \Lambda \cup \Lambda_1 \cup \Lambda_2$, $\Lambda_1 = \{M_G \supset \boldsymbol{R}^3 \cup T_G \cup F_G \cup K_G$, $\Sigma_G = \{\Xi_G, \Theta_G, \Phi_G, \{\rightarrow, \hookleftarrow\}\}$, $\Phi_G = \{\gg, *, /, +, -, \odot\}$, $\Theta_G = \{\Rightarrow, |\Rightarrow, ||\Rightarrow\}\}$.

Described sets and notations:

- T_G – conditionally indivisible graphic design elements;
- F_G – intermediate forms;
- K_G – structures;
- Ξ_G – relations of mutual arrangement of geometric figures, which are specified graphically;
- Θ_G – substitution and subtraction operations;
- Φ_G – operations on attributes;
- $\rightarrow$ – substitution relation;
- $\hookleftarrow$ – attributiveness.

The component Λ_2 contains the following definitions, additions, and restrictions that specify the alphabet, media attributes, substitution relations, and specify the specifics of substitution and inference operations.

A connected compact set in $\boldsymbol{R}^3$. will be called a figure or an object.

The presence of the w in the element of the carrier m will be denoted as ${}_w m$ (identifier m з with the attribute w), and the fact that w is an attribute of the identifier m is $w \hookleftarrow m$.

Terminal alphabet T_G consists of a set of geometric objects with attributes.

Form ${}_w f$ with attribute w is called a set of terminals, which are united by linking operations.

The final form is called a construction, intermediate forms are sentential forms.

The substitution rules look like $\psi_r: \langle s_r, g_r \rangle \in \Psi$, where s_r – one or more substitution relations, $s_r = \langle s_{r_1}, s_{r_2} \dots s_{r_n} \rangle$, $g_r = \langle g_{r_1}, g_{r_2} \dots g_{r_n} \rangle$ – a set of operations on attributes.

The two-place partial inference operation $f^* = (| \Rightarrow (\Psi, f))$ (here f,f^* are forms before and after the partial selection operation) consists in the alternate selection of all possible substitution rules for the current sentential form $\psi_r: \langle s_r, g_r \rangle \in \Psi$, with substitution relations s_r and given operations on attributes g_r, which will be executed according to the last execution order attribute.

The operation of full inferenceconsists of sequentially performing the operation of partial inference, starting from the starting terminal, and ending with a construction that satisfies the inference termination condition.

In these constructors, the following binding operations are provided for attributes:

- the duplication of a set $f_i \gg f_j$, a set f_i к is copied into f_j;
- the operation of multiplication of real numbers $*(a, b)$, implemented as $a := a * b$;
- the operation of division of real numbers $/(a, b)$, implemented as $a := a/b$;
- the addition operation $+(a, b)$, implemented as $a := a + b$;
- the subtraction operation $-(a, b)$, implemented as $a := a - b$;
- swap of colors operation $\odot(col_1, col_2)$, implemented as $col_x := col_2, col_2 := col_1, col_1 = col_x$.

The condition for completion of an inference is the execution of the Nth partial inference operation is specified by user.

To interpret construction operations, it is necessary to set a basic algorithmic structure:

$$C_A = \langle M_A,\ V_A,\ \Sigma_A,\ \Lambda_A \rangle$$

$V_\mathrm{A} \subset M_A$ a set of basic algorithms of the internal executor $V_\mathrm{A} \supset \{A_1|^{f_j}_{l_h, l_q, f_i}, A_2|^{f_j}_{f_i, \Psi}, A_3|^{f_j}_{f_i, \Psi}, A_4|^{a}_{a,b}, A_5|^{a}_{a,b}, A_6|^{a}_{a,b}, A_7|^{a}_{a,b}, A_8|^{\mathrm{col}_1}_{\mathrm{col}_1, \mathrm{col}_2}\}$.

Algorithms implement the following operations:

- $A_1|^{f_j}_{l_h, l_q, f_i}$ – are substitution operations;;
- $A_2|^{f_j}_{f_i, \Psi}, A_3|^{\bar{\Omega}}_{\sigma, \Psi}$ – are operations of partial and complete inference. f_i, f_j are forms, σ is the initial terminal, $\bar{\Omega}$ – is the set of formed constructions;
- $A_4|^{a}_{a,b}$, $A_5|^{a}_{a,b}$, $A_6|^{a}_{a,b}$ $A_7|^{a}_{a,b}$– multiplication, division, addition and subtraction of real numbers;
- $A_8|^{\mathrm{col}_1, \mathrm{col}_2}_{\mathrm{col}_1, \mathrm{col}_2}$ is an implementation of the operation $\odot(col_1, col_2)$.

Interpretation of construction operations for a specialized designer:

$$\langle C_G = \langle M_G, \Sigma_G, \Lambda_G \rangle, C_A = \langle M_A,\ V_A,\ \Sigma_A,\ \Lambda_A \rangle \rangle_I \mapsto C_{GI} = \langle M_{GI}, \Sigma_{GI}, \Lambda_{GI} \rangle,$$

where $I \mapsto$ – interpretation operation; $\Lambda_{GI} = \Lambda_G \cup \Lambda_A \cup \Lambda_3$,$\Lambda_3 = \{ (A_1|^{f_j}_{l_h, l_q, f_i} \hookleftarrow \Rightarrow), (A_2|^{f_j}_{f_i, \Psi} \hookleftarrow | \Rightarrow), (A_3|^{\bar{\Omega}}_{\sigma, \Psi} \hookleftarrow || \Rightarrow), (A_5|^{a}_{a,b} \hookleftarrow /), (A_6|^{a}_{a,b} \hookleftarrow +), (A_7|^{c}_{a,b} \hookleftarrow -), (A_8|^{\mathrm{col}_1, \mathrm{col}_2}_{\mathrm{col}_1, \mathrm{col}_2} \hookleftarrow \odot)\}$

We perform the following concretization of the constructor $C_{crystal}$, destined for forming of a crystal structure

$$C_{GI} = \langle M_{GI}, \Sigma_{GI}, \Lambda_{GI} \rangle_K \mapsto C_{crystal} = \langle M_{GI}, \Sigma_{GI}, \Lambda_{crystal} \rangle,$$

$\Lambda_{crystal}$ specifies:

- vertice – v, with attributes of size $size = 1$, coordinates $pos = (x, y, z)$, color $col = red$;
- unit cell – uc, a graph with attributes of edge length *a, b, c* and angles between the edges α, β, γ;
- initial conditions – onve vertice v_0, with attributes ${}_{size}v_0 = 1$, ${}_{pos}v_0 = (0,0,0)$, ${}_{col}v_0 = red$;
- substitution rule $\psi = < s, g >$, in which the operation consists of a single substitution relation Fig. 1 and operations on attributes $g = < \odot(left, right) >$ Fig. 2 for mutual repainting of vertices;
- the right part of the substitution must contain at least one vertex with the color as in the left rule;

Fig. 1. Constructor substitution relation $C_{crystal}$

Fig. 2. Display of colors that participate in the mutual repainting operation

We will present the implementation in the form of a sequence of sentential forms f_1, f_2, f_3, f_4 ...:

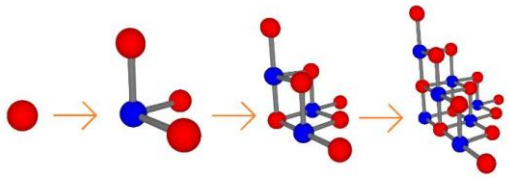


Fig. 3. The sequence of construction of crystal lattices

The implementation is carried out in the following way:

- the partial inference operation performs form transformation of f_i into f_{i+1};
- initial form f_0, according to the substitution Fig. 1, by applying the substitution operation once, transforms into f_1;
- we perform an operation on color attributes: mutual repainting of vertices with the color left to right and vice versa.

Form f_3 Fig. 3 obtained by a partial subtraction operation with three substitution operations followed by an operation on color attributes.

Brave crystal lattices and the crystal structure can be represented as a Cayley graph, but vertices, not vectors, are colored. As an example, we can consider A) incomplete triclinic, B) full triclinic, C) Face-centered Cubic, D) Body-centered Cubic E) full hexagonal: lattices

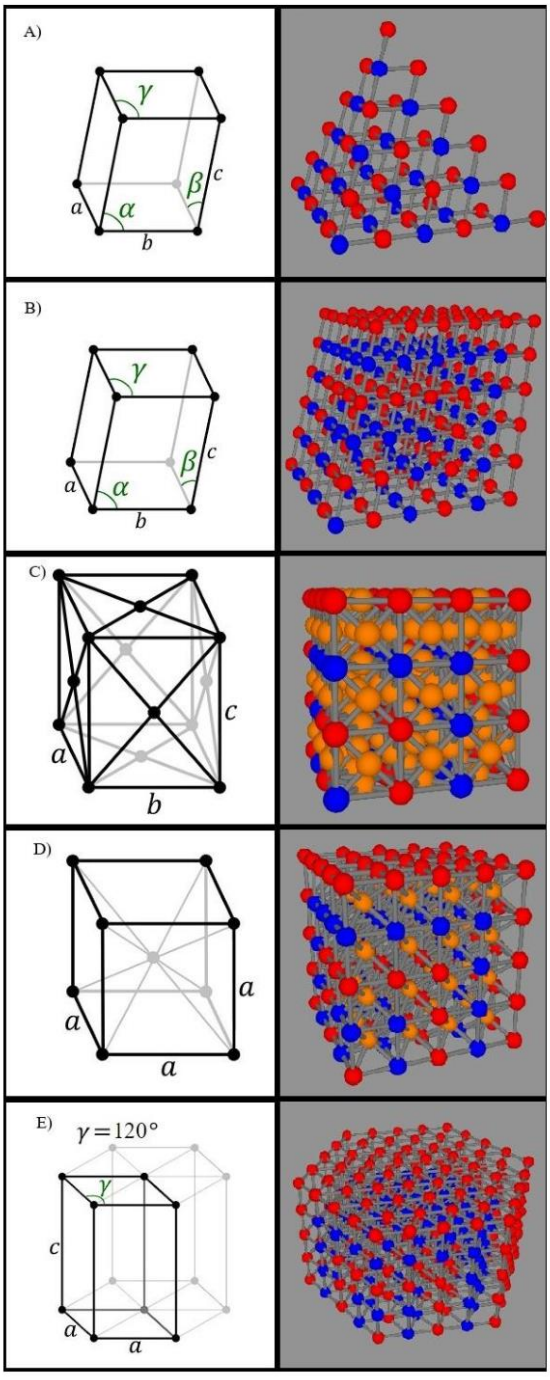


Fig. 4 Creating a crystal structure based on different Brave lattices. a, b, c – length of edges, α, β, γ – angles between edges

Such crystals can be called Cayley crystals [2]. Again, it is worth paying attention to the color of Fig. 4, C: if blue or red were used instead of orange, then a completely different crystal structure would be obtained.

It is worth noting that instead of using color, you can use other attributes of vertices: their size or position. It is possible to enter shape attributes for each vertex: pyramid, cube, ellipsoid, as well as parameters describing these shapes: radius, height, width, etc. CSM allow you to use several attributes and set their priority.

To change the shape of the crystal structure, there are separate operations on the attributes for modifying the crystal lattice:

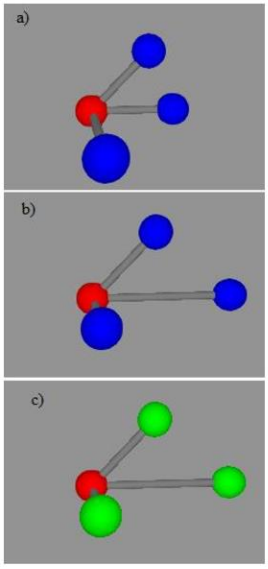


Fig. 5 Changed crystal lattice a) operation on angle attributes b) changing attributes of the lengths of the edges c) changing the color of the vertices

By changing the crystal lattice during the formation of the crystal structure, it is possible to display the moment when the shape of the crystal changes

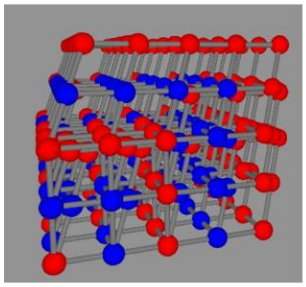


Fig. 6. Changing the crystal lattice during the construction of the crystal

However, using only color does not allow you to see the desired result. It is possible to modify the crystal structure with an imaginary attribute, a restriction that would prevent replacing a vertex with a crystal lattice if its result would cross or be very close to an already existing crystal cell.

RESULTS AND DISCUSSION

The generated 3D model can be printed by any 3d printer that supports *.stl file format. They can be used to study new crystal lattices in a visual form.

CSM illustrates the fractal nature of crystals and demonstrate the process of crystal formation, in contrast to well-known models of crystal lattices [2, 4, 5, 6].

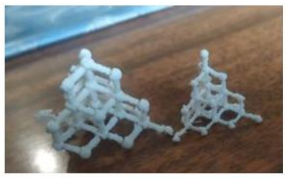


Fig. 7. 3d Printed models of crystal structure generated by incomplete triclinic Bravais lattice

CONCLUSIONS

As a result of the work, a program was developed that allows us to form spatial graphs and, as a special case, crystal structures, using CSM methods. The key role is played by the attributes of the elements and related algorithmic operations.

Constructive-synthesizing modeling allows building flexible algorithmic systems that are subject to easy modification and specialization, which makes it possible to build a more complex system using the created one for further study of crystal structures or to use it in another area.

For example, replacing the algorithm for displaying balls (graph vertices) with an algorithm for displaying a dynamic object that demonstrates the rotation of electrons around the nucleus, we obtain a means for an in-depth study of crystal structures; it's easy to replace balls with cubes, rigid ligaments with elastic, etc.

REFERENCES

[1] X. A. Jo'rayevich, "Regarding the study of crystal structures of substances", Int. J. Innov. Eng. Res. and Technol., vol. 9(11), 2022, pp. 93-100. doi: 10.17605/OSF.IO/TW2ZN

[2] P. Nasr, H. Leung, F. I. Auzanneau, and M. A. Rogers, "Supramolecular Fractal Growth of Self-Assembled Fibrillar Networks". Gels, vol. 7(2), 2021, pp 46. doi: 10.3390/gels7020046.

[3] P. R. Spackman, M. J. Turner, J. J. McKinnon, S. K.Wolff, D. J. Grimwood, D. Jayatilaka, and M. A. Spackman, "CrystalExplorer: A program for Hirshfeld surface analysis, visualization and quantitative analysis of molecular crystals", J. Applied Crystallography, vol. 54(3), 2021, pp. 1006-1011.

[4] C. Gruber, A. James, J. T. Berchtold, Z. J. Wood, G. E. Scott, and Z. Alghoul, "Interactive Unit Cell Visualization Tool for Crystal Lattice Structures", J. Chem. Educ., vol. 97, 2020, pp. 2020-2024, doi: 10.1021/acs.jchemed.9b01207.

[5] P.Chen, Y. Wang, H. Yan, S. Gao, Z. Xu, Y. Li, Q. Mo, J. Huang, J. Tao, G. Pan, J. Li and Y. Du, "3DStructGen: an interactive web-based 3D structure generation for non-periodic molecule and crystal", J. Cheminform., vol. 12, 2020, pp. 7-17, doi: 10.1186/s13321-020-0411-2.

[6] S. Sun, X. Zhang, J. Cui, and S. Liang, "Identification of the Miller indices of a crystallographic plane: a tutorial and a comprehensive review on fundamental theory, universal methods based on different case studies and matters needing attention", Nanoscale, vol. 12(32), 2020, pp. 16657-16677. doi: 10.1039/D0NR03637D.

[7] A. Troelsen, P. Japikse "WPF Notifications, Validations, Commands, and MVVM" in *Pro C# 9 with .NET 5*, J. Murray, L. Berendson, M. Powers, Eds., 10th ed. Berkeley, CA, USA: Apress, 2021, pp. 1143-1152.

[8] Helix Toolkit. Accessed: Jul. 29, 2023. [Online]. Available: https://github.com/helix-toolkit.

[9] V. Skalozub, V. Ilman and V. Shynkarenko, "Ontological support formation for constructive-synthesizing modeling of information systems development processes", East.-Eur. J. Enterprise Technol., vol. 5, issue 4 (95), 2018, pp. 55–63. doi: 10.15587/1729-4061.2018.143968

[10] V. I. Shynkarenko, "Constructive-Synthesizing Representation of Geometric Fractals", Cybern. Syst. Anal., vol. 55, pp. 186-199, Mar. 2019, doi: 10.1007/s10559-019-001