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 бакалавра

на тему: «Фрактальне моделювання кристалічних граток»
за освітньою програмою: «12 Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1911»

_____ /Олександр ЛЕТУЧИЙ/
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /проф. Віктор ШИНКАРЕНКО/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Світлана ВОЛКОВА/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

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

Дніпро – 2023 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 Bachelor's Thesis

on the topic: «Fractal modeling of crystal lattices»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ1911: /Oleksandr LETUCHYI/

Scientific Supervisor: /Victor SHINKARENKO/

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2023

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: бакалавр
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 бакалавра
студенту Летучому Олександру Ігоровичу
1. Тема роботи: «Фрактальне моделювання кристалічних ґраток»
Керівник роботи: Шинкаренко Віктор Іванович, професор
затверджені наказом № 1209 ст від 07.12.2022
2. Строк подання студентом роботи: 12.06.2023 р.
3. Вихідні дані до роботи: _____
4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):

Вступ
Огляд предметної галузі
Огляд аналогів
Зовнішнє проектування
Внутрішнє проектування
Математична Модель
Тестування та налагодження
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

Презентація
Відео роботи програми

КАЛЕНДАРНИЙ ПЛАН

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.23 -18.02.23
Робочий проект	Програмування та відлагодження програми.	19.02.23 - 20.05.23
	Тестування програми	20.05.23 - 27.05.23
	Розробка, узгодження і затвердження програмної документації.	27.05.23 - 12.06.23
	Подання кваліфікаційної роботи до кафедри	07.06.2023
	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	26.06.2023

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

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

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра: 65с., 34 рис., 39 табл., 1 додатки, 24 джерела.

Об’єкт розробки –десктопний застосунок для моделювання фрактальної побудови кристалічної ґратки.

Мета роботи – моделювання кристалічних ґраток фрактальними способами.

Методи дослідження – використання фрактальних властивостей при побудові кристалічних структур.

– вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;

– огляд предметної галузі – у цьому розділі аналізується предмета область та описуються основні поняття . Складається з 6 сторінок;

– огляд аналогів – у цьому розділи описуються аналоги програми та література по даній предметній області. Складається з 10 сторінок;

– зовнішнє і внутрішнє проектування – в цих розділах проведений огляд вхідних і вихідних даних, формалізація задачі, розробка фізичного проекту, приводиться опис об’єктно-орієнтованого проектування, проектування інтерфейсу користувача, ескізи форм, аналіз проекту, проектування динаміки системи, вибір мови програмування. Складається з 29 сторінок;

– тестування та налагодження – включає в себе вибір стратегії тестування, опис тестів методами чорної скриньки. Також аналіз помилок їх вплив на систему та вирішення проблеми. Складається з 6 сторінок;

– висновки – підсумок всієї роботи. Складається з 1сторінки;

– список використаних джерел – включає в себе бібліографічний список використаної літератури. Складає 2 сторінки;

– додаток – містить робочий код проекту. Складає 51 сторінку.

Кількість таблиць: 39 штук.

Кількість рисунків: 34 штуки.

Ключові слова: моделювання, кристалічна ґратка, кристалічна структура, фрактал, ґратка Браве, суперкомірка, граф, конструктивно-продукційні структури, Helix Toolkit, MVVM, WPF.

ЗМІСТ

ВСТУП

Актуальність роботи. Кристалічна ґратка є основною структурою багатьох твердих матеріалів, таких як метали, напівпровідники та ізолятори. Моделювання кристалічних ґраток можна використовувати для дослідження існуючих та симуляції уявних матеріалів, вивчення процесів кристалізації. Фрактальна геометрія надає потужний інструмент для моделювання самоподібних та рекурсивних патернів, які відбуваються в кристалічних ґратках. Один із напрямків моделювання є візуалізація за допомогою 3D графіки.

Мета роботи. Створення математичних моделей та інструментів, що дозволяють візуалізувати та аналізувати структуру кристалічної ґратки за допомогою фрактального моделювання.

Експлуатаційне призначення. Візуалізація кристалічних ґраток. Дана програма підходить студентам, викладачам або працівникам з галузі неорганічної хімії, матеріалознавства, кристалографії в науково-освітніх цілях.

1 ОГЛЯД ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Хемоінформатика та геометрична кристалографія

Основна предметна галузь моєї випускної кваліфікаційної роботи – це хемоінформатика та геометрична кристалографія.

Хемоінформатика є науковою галуззю, яка об'єднує принципи хімії та інформатики з метою розв'язання хімічних проблем за допомогою комп'ютерних методів. Вона використовує математичні та статистичні алгоритми для аналізу хімічних даних та побудови моделей хімічних систем. Один з основних напрямків хемоінформатики полягає у побудові моделей хімічних сполук. Це означає, що за допомогою різних математичних методів і комп'ютерних алгоритмів можна створювати віртуальні моделі речовин, які мають певні властивості. Ці моделі можуть бути використані для прогнозування фізико-хімічних властивостей сполук, їх взаємодії з іншими речовинами та біологічною активністю. Ще одним важливим аспектом хемоінформатики є візуалізація хімічних структур. Вона дозволяє графічно представляти хімічні сполуки та їх взаємодії, що спрощує їх розуміння та аналіз. Застосування візуалізації допомагає хімікам вивчати та відкривати нові зв'язки та властивості речовин.

Геометрична кристалографія є галуззю науки, яка вивчає просторову структуру кристалів. Вона використовує методи геометрії, математики та фізики для аналізу та опису геометричних властивостей кристалів, їх симетрії та взаємодії атомів у кристалічній ґратці. Слід наголосити на відмінності понять кристалічної структури та кристалічної ґратки [1]. На відміну від типів кристалічних ґраток, яких налічується чотирнадцять типів, різних кристалічних структур є безмежно багато.

Кристалічна ґратка - це абстрактна геометрична сітка, яка відображає просторове розташування одиниць структури в кристалі. Вона складається з умовних точок (вузлів), розташованих у тривимірному просторі за певними правилами. Кожен вузол ґратки представляє позицію атома, молекули або йону у кристалі. Кристалічна ґратка не враховує самого атомного або молекулярного розміру, а лише вказує на їхній розташування та відносини між ними.

Кристалічна структура включає не тільки інформацію про геометричну організацію атомів у кристалічній ґратці. Вона описує точні положення атомів або

молекул у кристалі, відстані між ними, кути між хімічними зв'язками та інші деталі структури. Кристалічна структура дає повну інформацію про хімічні зв'язки та конфігурацію атомів у кристалі, відображаючи всі хімічні, структурні та фізичні характеристики матеріалу. Приклад кристалічної структури:

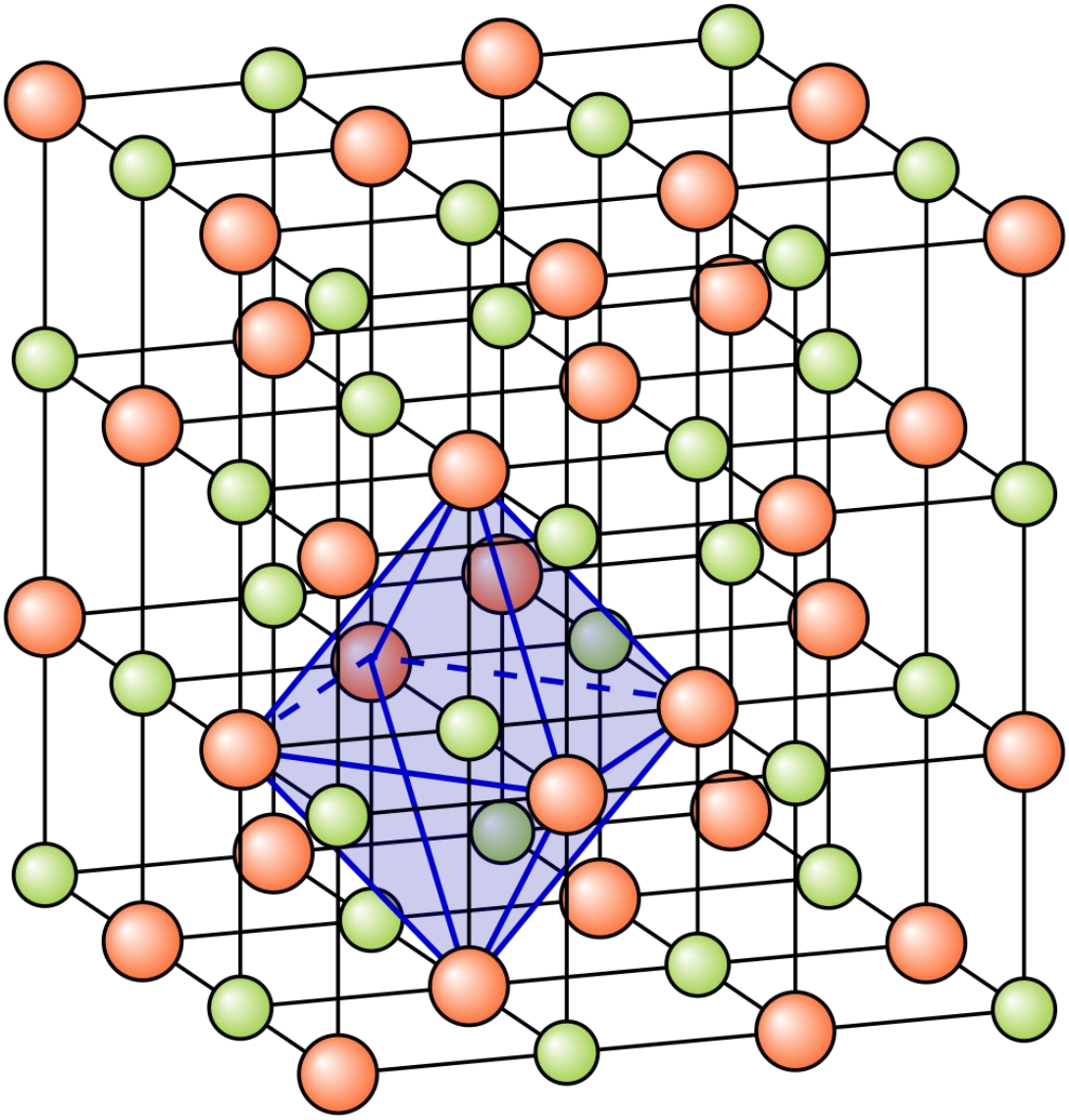


Рисунок 1.1– Кристалічна структура хлориду натрію

В той час як кристалічна структура складається з елементарних комірок (див рис. 1.2), які представлені як правило кристалічними ґратками, вони не завжди мають однаковий вигляд. Для прикладку, кристалічна структура алмазу не співпадає з її геометричною кристалічною ґраткою, бо елементарна комірка алмаза складається з двох атомів (див. рис. 1.3).

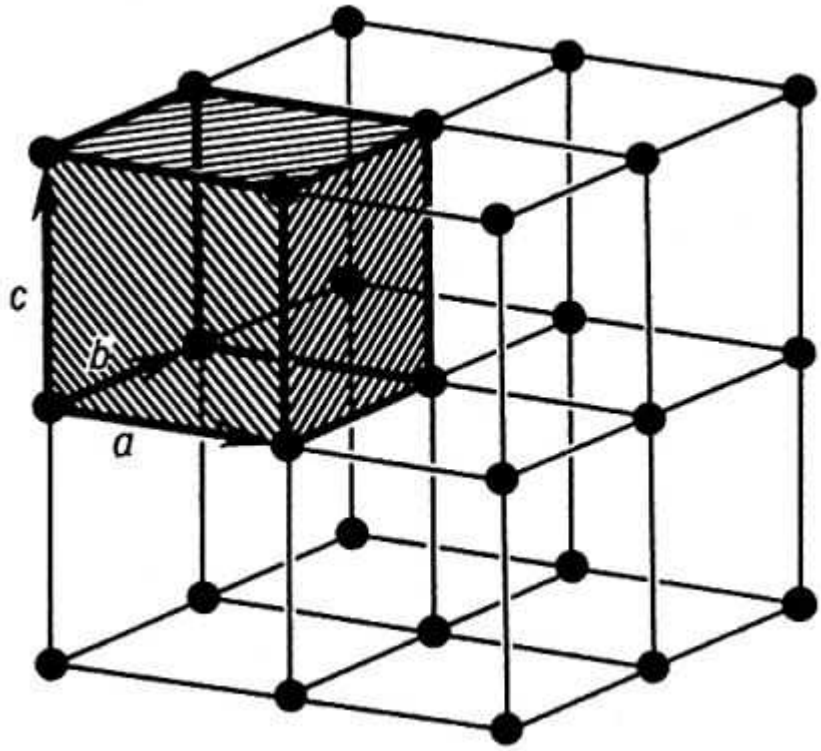


Рисунок 1.2 – Зафарбованим позначається елементарна комірка в кристалічній структурі, а також підписані її ребра *a, b, c*

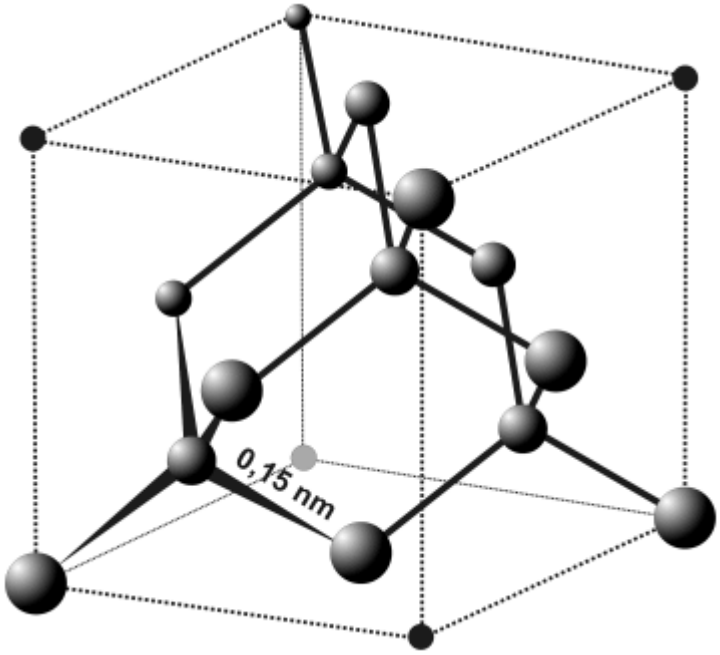


Рисунок 1.3 – Елементарна комірка алмазу

1.2 Застосування ґраток Браве при формуванні кристалічної структури

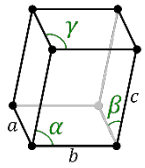
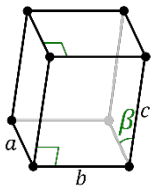
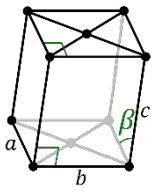
Ґратки Браве дозволяють класифікувати кристалічні ґратки залежно від їхньої симетрії [2]. Ґраткою або системою трансляцій Браве називається набір елементарних трансляцій або трансляційна група, за допомогою яких можна отримати безкінечну

кристалічну ґратку. Залежно від співвідношення між довжинами цих трансляцій *a, b, c* та кутами між ними *α, β, γ* три категорії, шість сингоній та сім кристалічних систем залежно від числа однакових довжин трансляцій:

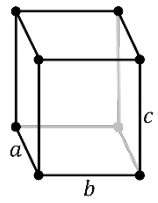
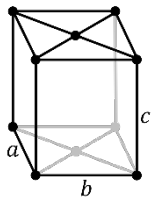
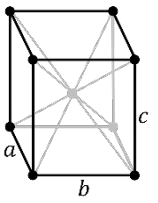
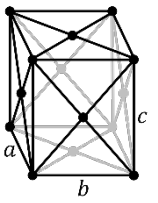
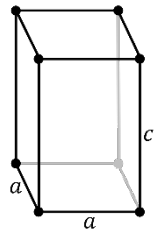
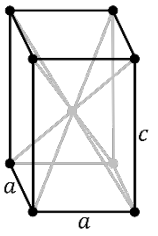
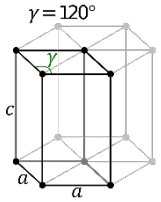
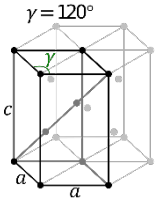
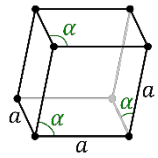
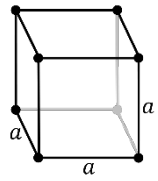
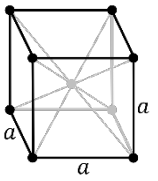
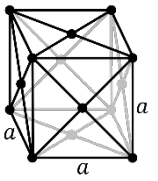
1. Нижча категорія (всі трансляції не рівні один одному)
   - триклінна: $a \neq b \neq c,\ \alpha \neq \beta \neq \gamma \neq 90°$
   - моноклінна: $a \neq b \neq c,\ \alpha = \gamma = 90°,\ \beta \neq 90°$
   - ромбічна: $a \neq b \neq c,\ \alpha = \beta = \gamma = 90°$
2. Середня категорія (дві трансляції 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: примітивна, базоцентрована, об'ємноцентрована, гранецентрована, двічі об'ємноцентрована.

Таблиця 1-1 Різновиди кристалічних ґраток

Кристалічна система	Тип центрування ґратки Браве				
	примітивна	базоцентрована	об'ємноцентрована	гранецентрована	двічі об'ємноцентрована
Триклінна					
Моноклінна					

Продовження таблиці 1.1

Кристалічна система	Тип центрування ґратки Браве				
	примітивна	базоцентро-вана	об'ємноцен-трована	гранецент-рована	двічі об'єм-ноцентро-вана
Ромбічна					
Тетрагона-льна					
Гексагона-льна					
Тригона-льна					
Кубічна					

Ідеальна кристалічна структура може бути розглянута з фрактальної перспективи, оскільки вона відображає самоподібність, яка є ключовою властивістю фракталів. Кожна елементарна ґратка в кристалі має подібну структуру до всього кристалу в цілому. Застосування фрактального підходу в побудові кристалічних структур дає змогу створювати надзвичайно складні та деталізовані структури. За допомогою ітераційного процесу можна відтворити фрактальну структуру на більшому масштабі, створюючи елементи, які повторюються з певною періодичністю в конкретному

масштабі. Ці процеси можна відтворити у комп’ютерній програмі та моделювати штучні кристалічні структури різних кристалів, в тому числі і штучних. Приклад штучного кристалу:

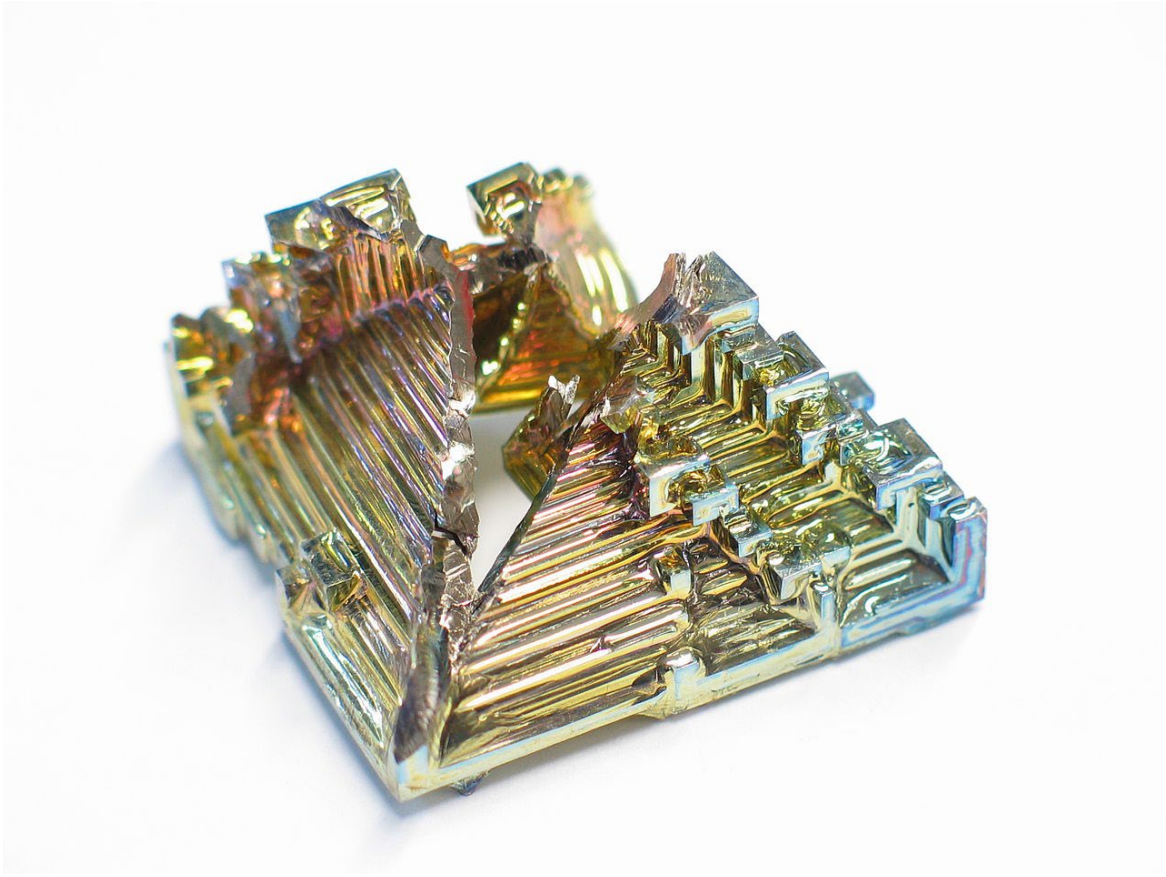


Рисунок 1.4 – Штучний кристал бісмуту

2 ОГЛЯД АНАЛОГІВ

2.1 Використання Virtual Unit Cell як ігри для навчання

Програма, що була написана з метою допомогти студентам в області матеріалознавства, неорганічної хімії та фізики зрозуміти модель кристалічної структури та кристалічної ґратки [3]. Автори програми Sutha Luealamai та Bhinyo Panijpan акцентують увагу на необхідності саме 3D візуалізації, всупереч 2D, та на необхідності мати змогу взаємодіяти з 3D моделями, зокрема на обертанні моделі. Програма складається з трьох частин: верхнє меню та лівого бокового меню, що дають змогу обирати різні моделі, 3D сцена де відображається обрана модель, а також меню з іконок навколо 3D сцени для взаємодії з моделлю.

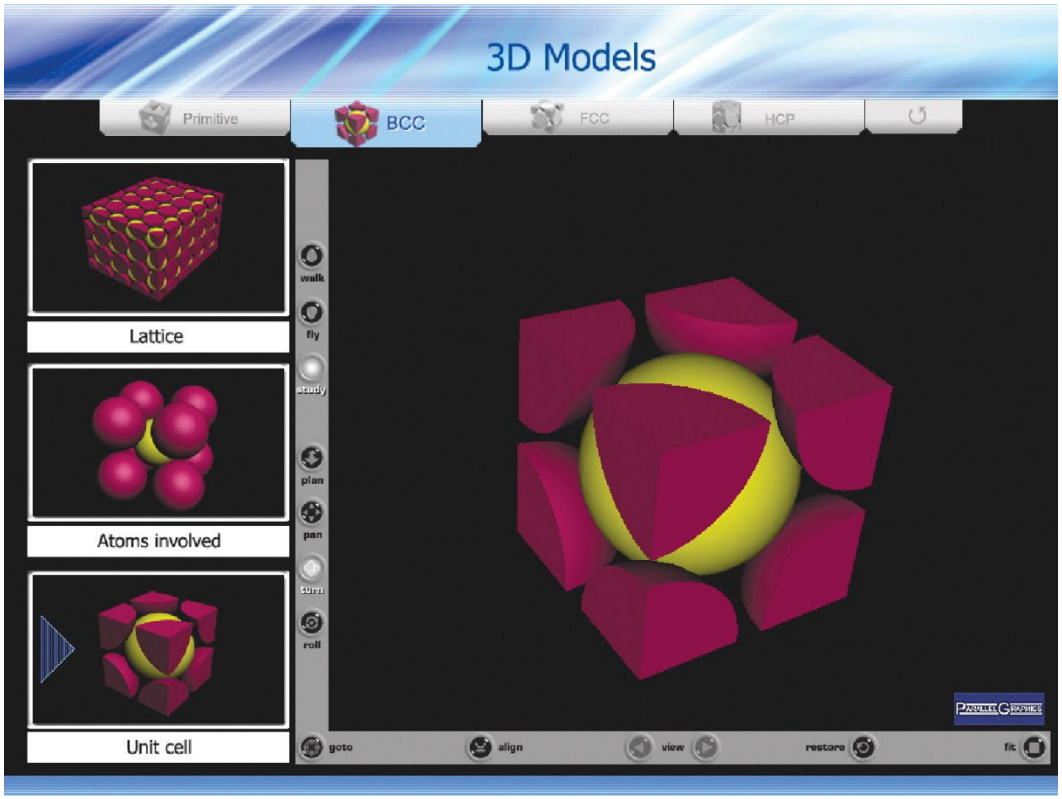


Рисунок 2.1 – Модель елементарної кристалічної комірки

Перевагами програми є простота її інтерфейсу, інтуїтивність та зворотня реакція на дії користувача.

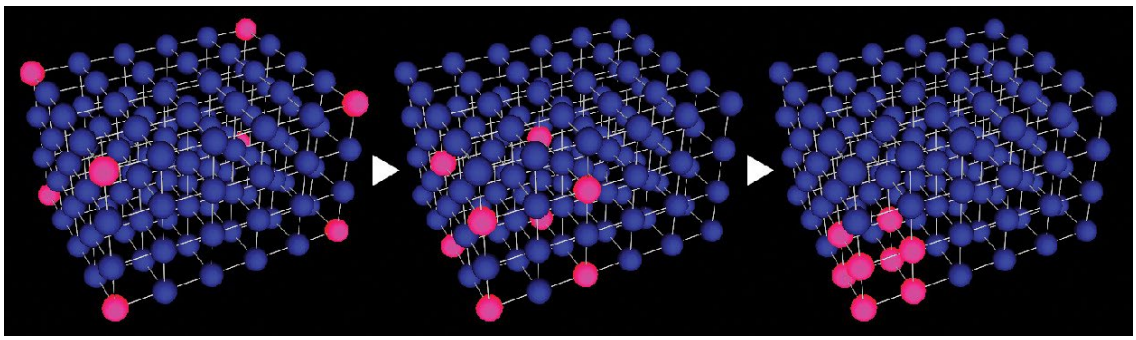


Рисунок 2.2 – Кольорове підсвічення подібності кристалічної структури до кристалічної ґратки, в залежності від обраної кристалічної комірки

2.2 Інтерактивна візуалізація ґратки за допомогою Unit Cell Visualization Tool

Як і попередній аналог “Virtual Unit Cell” розроблений в освітніх цілях з метою допомогти студентам краще зрозуміти та наочно побачити кристалічну структуру та кристалічну ґратку. Автор програми акцентує увагу на патернах, які утворюють елементарні кристалічні структури та повторюванні кристалічні ґратки, з яких вони складаються, зокрема з простих базоцентрованих та об'ємноцентрованих кристалічних ґраток, а також гексагональних [4]. Програма має відкритий вихідний код на Github та написана мові програмування JavaScript. Серед переваг програми є її простота, зручність, та кросплатформеність, так як вона використовує технологію WebGL та може працювати в різних браузерах без необхідності в встановленні спеціальних плагінів.

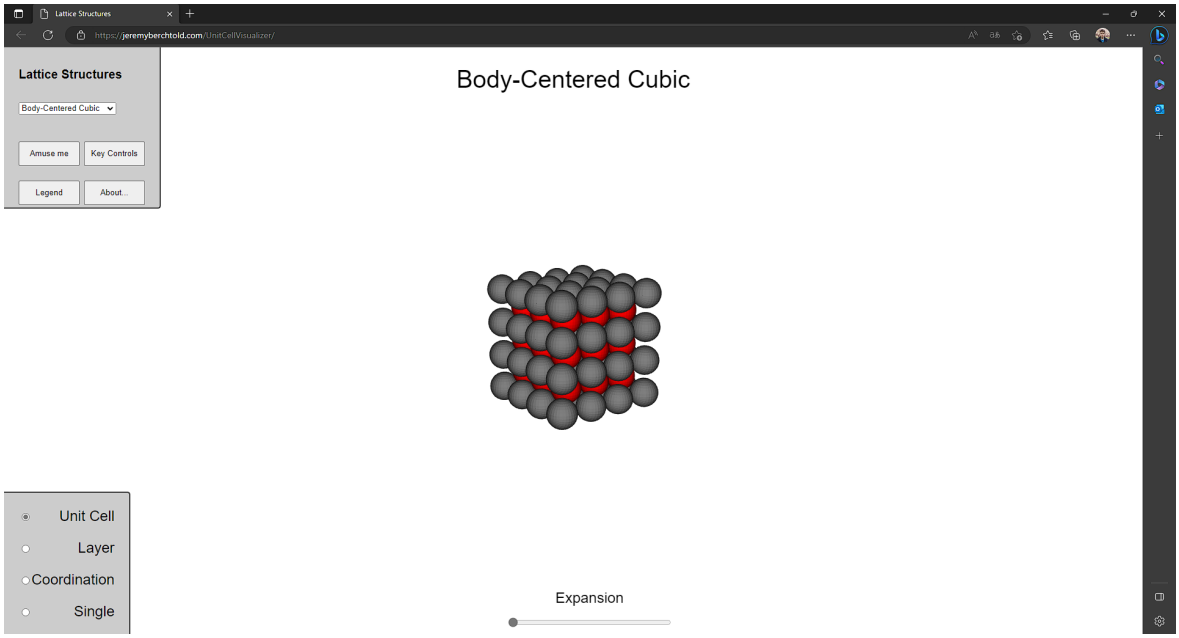


Рисунок 2.3 – Приклад примітивної об’ємноцентрованої кристалічної структури

Програма надає можливість обирати різні кристалічні структури, а також перегляди ці структури в різних режимах: в повному об’ємі, у вигляді одного прошарку,

вигляд однієї елементарної комірки та у вигляді «координації». Автор програми описує режим «координації» як таких, що дозволяє полегшити розуміння таких понять як «координаційне число» та спільних атомів і іонів між сусідніми елементарними комірками. Серед інших можливостей програми є обертання представлених моделей, масштабування у розмірі та можливість розділення кристалічної структури на елементарні комірки з проміжками між ними.

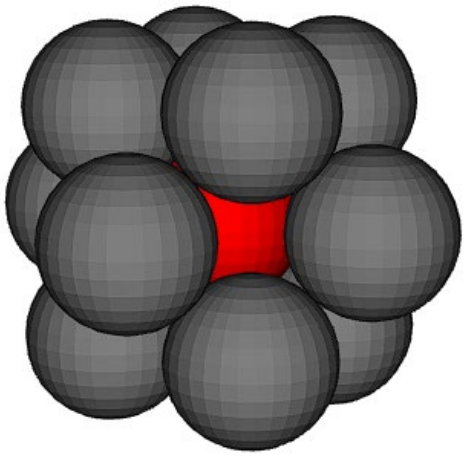


Рисунок 2.4 – Приклад примітивної об’ємноцентрованої кристалічної структури у режимі «координації».

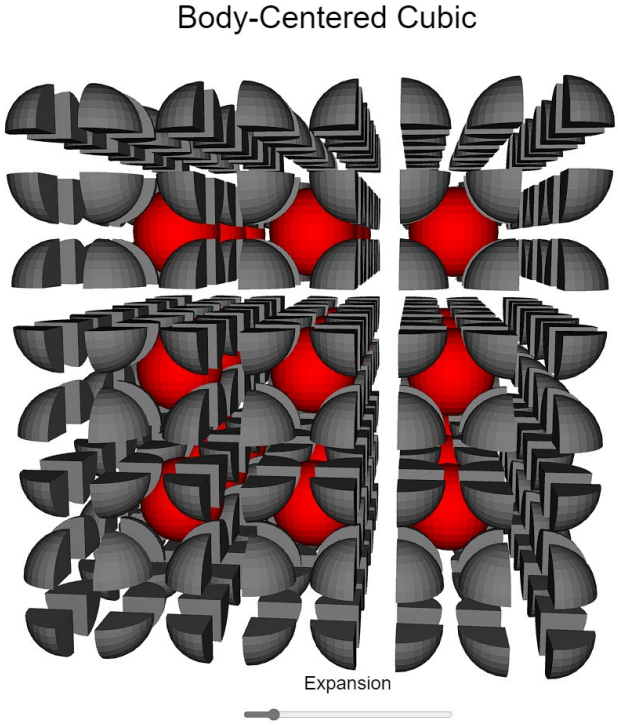


Рисунок 2.5 – Приклад примітивної об’ємноцентрованої кристалічної структури з розділенням на елементарні комірки

2.3 Моделювання молекул та 3DStructGen

3DStructGen [5] є програмою, яка розроблена з використанням стандартних веб-технологій, таких як Hyper Text Markup Language 5 (HTML5), Cascade Style Sheet (CSS) та JavaScript. Ця програма пропонує зручний і універсальний підхід до візуалізації, редагування та побудови молекулярних і кристалічних структур. Однією з ключових можливостей 3DStructGen є можливість відображення та редагування атомів, зв'язків, кутів і дігральних кутів у молекулах. Це дозволяє користувачу вносити зміни в структуру за допомогою інтерактивних дій, що спрощує вивчення та редагування молекулярних систем. Крім того, програма надає широкий набір хемоінформатичних алгоритмів для обробки кристалічних структур. Наприклад, вона дозволяє проводити розділення поверхонь, створювати вакуумні шари та будувати суперкомірки [6].

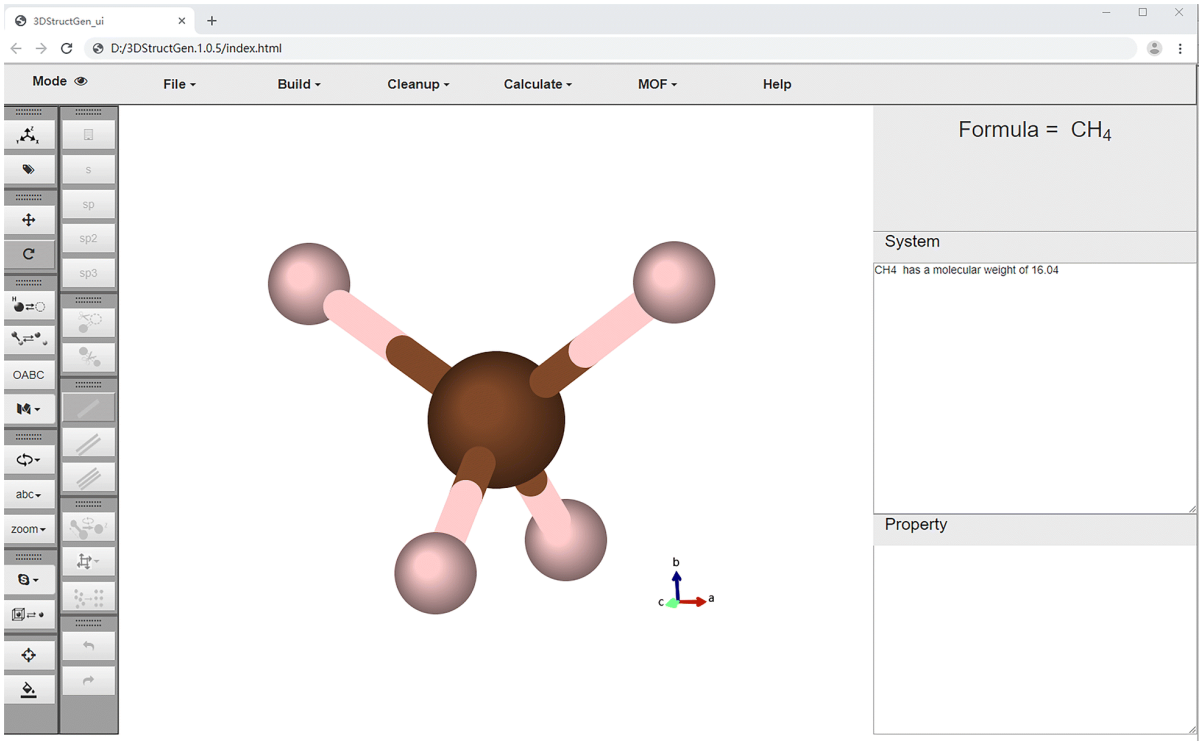


Рисунок 2.6 – Молекула метану візуалізована у 3DSrtuctGen

3DStructGen пропонує чотири стилі відображення, які можуть бути використані для візуалізації елементарної комірки: «примітивна комірка», «оригінальна», «усередині комірки» та «упаковка» (рис. 2.7). Це дозволяє користувачеві отримати більш чітке розуміння кристалічних систем.

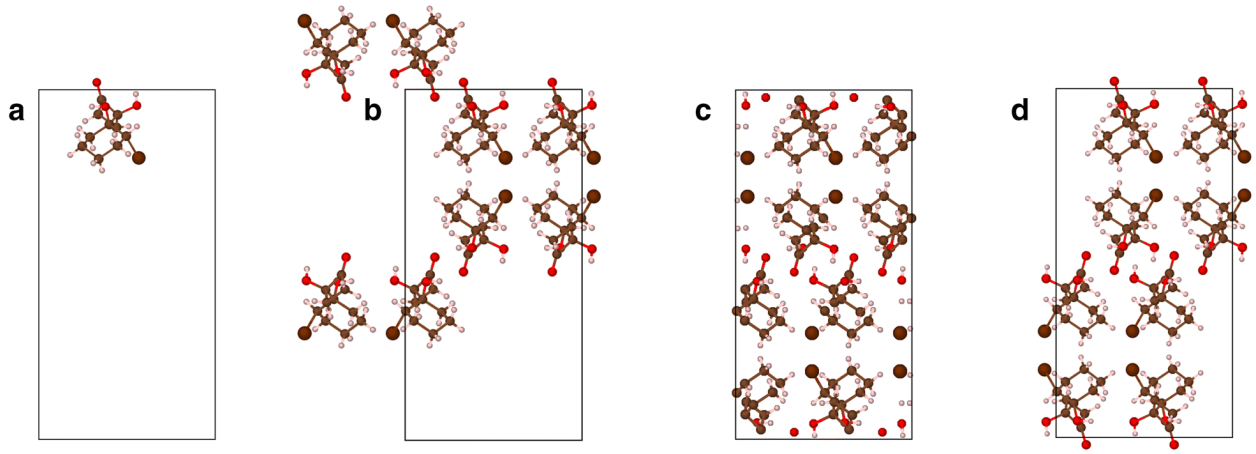


Рисунок 2.7– Зображення елментарної комірки у різних стилях: a) «примітивна комірка» b) «оригінальна» c) «усередині комірки» d) «упаковка»

Завдяки використанню стандартних веб-технологій, 3DStructGen не вимагає значних зусиль для розгортання та використання. Для користування програмою достатньо мати веб-переглядач, який підтримує HTML5. Це дозволяє зручно використовувати програму на будь-якому комп'ютері або пристрої з доступом до Інтернету, не залежно від його конфігурації чи операційної системи.

2.4 Використання CrystalMaker в навчанні

CrystalMaker [7] - це потужна програма для візуалізації і аналізу кристалічних і молекулярних структур. Вона дозволяє створювати тривимірні візуалізації кристалів, включаючи атоми, зв'язки та інші структурні елементи. Програма надає інструменти для побудови кристалічних структур, розрахунку фізичних властивостей кристалів, інтерактивної маніпуляції з структурами, експорту результатів та співпраці з іншими програмами. Окрім моделювання кристалічної структури є моделювання і інших хімічних процесів. Містить базу даних про хімічні сполуки, елементи та процеси, їх властивості. CrystalMaker є корисним інструментом для науковців та викладачів у галузі матеріалознавства, хімії та кристалографії, які вивчають кристалічні структури і їх властивості. Серед переваг варто відзначити кількість наданого програмою функціоналу, а серед недоліків її перевантажений та складний інтерфейс, який потребує значного часу на оволодіння програмою, а також дуже обмежений функціонал демонстраційної версії.

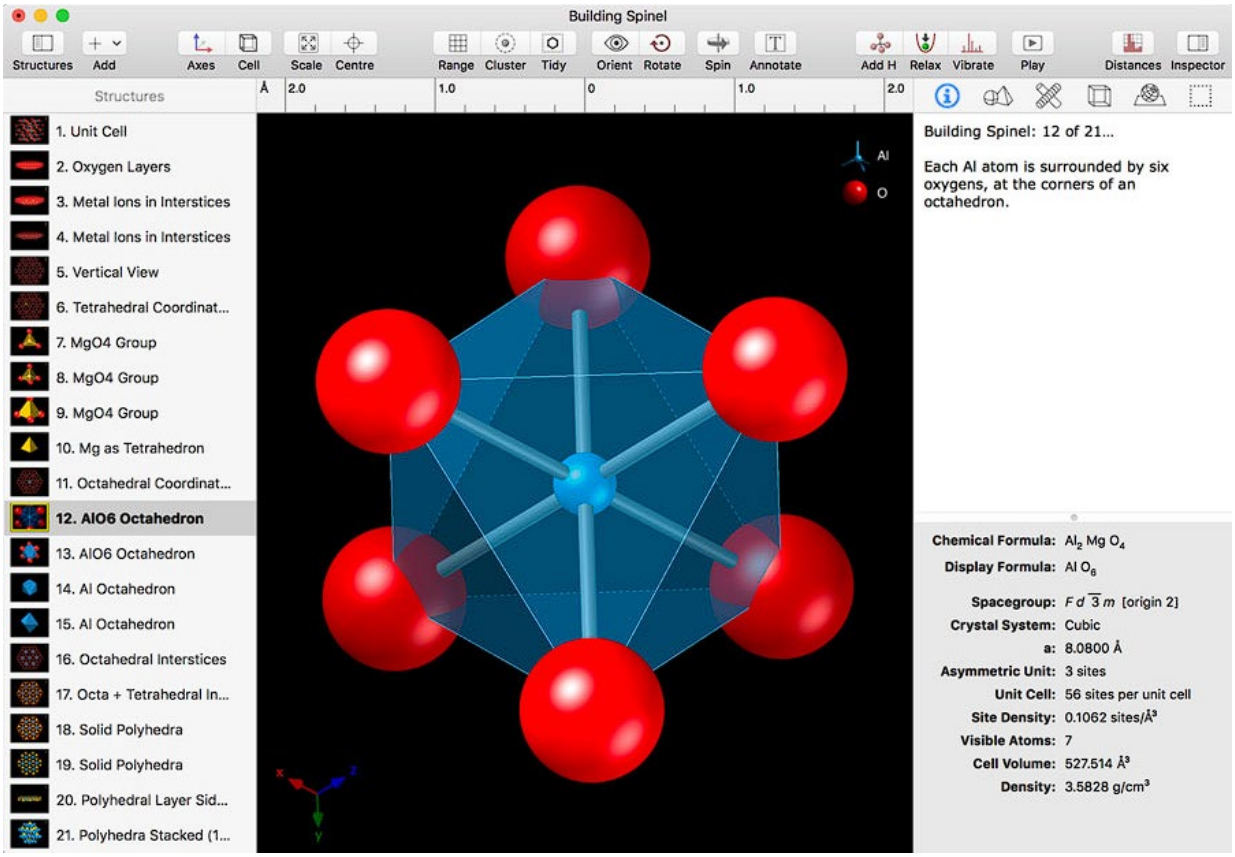


Рисунок 2.8 – Вікно програми, у якому ліворуч відображається список структур, а праворуч — поточна модель структури і примітки до неї

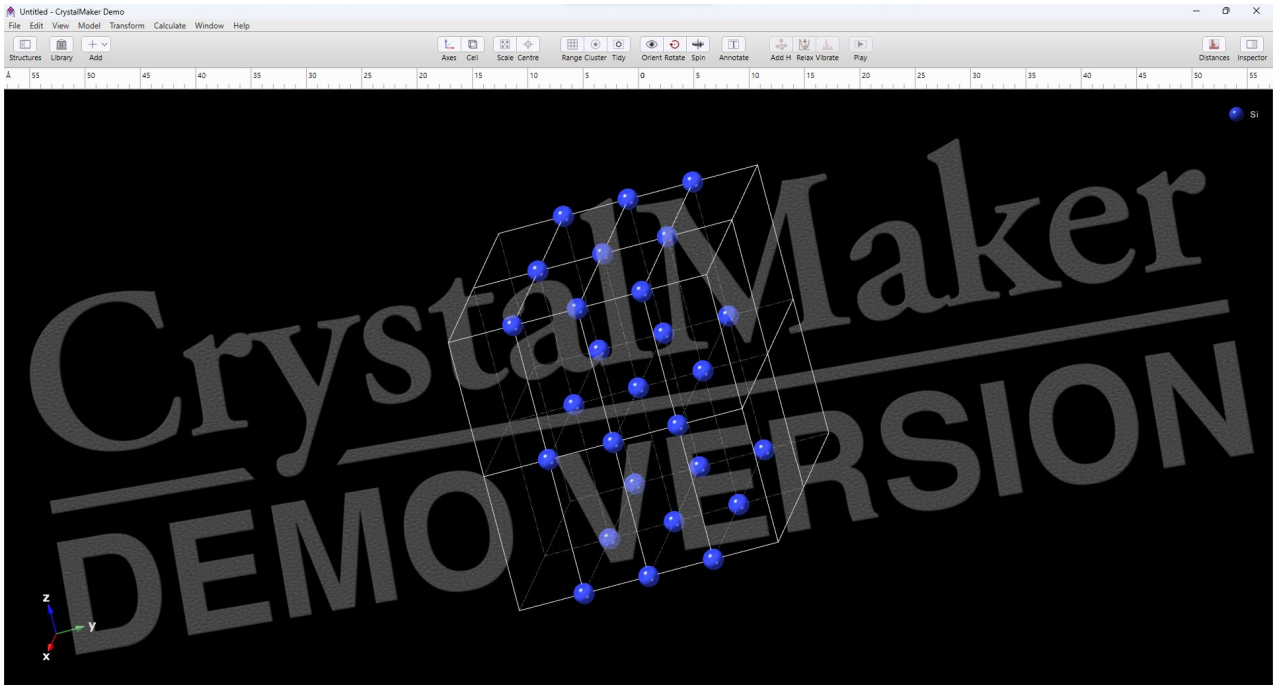


Рисунок 2.9 – Побудова кристалічної структури заданою кристалічною ґраткою у демо-режимі

2.5 VESTA як засіб аналізу кристалічних структур

VESTA [8] - це програмне забезпечення для візуалізації, моделювання та аналізу кристалічних структур. Воно надає розширені можливості для дослідження та розуміння властивостей кристалів, зосереджуючись на візуальному аналізі. VESTA дозволяє завантажувати, відображати і редагувати кристалічні структури в різних форматах. Воно пропонує широкий набір інструментів для розрахунку і візуалізації фізичних властивостей, таких як електронна щільність, розподіл заряду, структурні аналізи та трансформації. VESTA також має вбудовані інструменти для створення тривимірних зображень, анімацій та поверхонь, що дозволяє ефективно представляти та спілкуватися зі структурними даними. Ця програма широко використовується в наукових дослідженнях і навчанні в галузях кристалографії, матеріалознавства, хімії та фізики. Серед інших можливостей є такі як підтримка декількох вкладок і вікон, візуалізація обмежень Ритвельда, експорт графічних зображень високої роздольної здатності. VESTA підтримує багато форматів файлів для введення структурних даних та об’ємних даних і може бути завантажена для Windows, Mac OS X та Linux з офіційного сайту. Серед недоліків програми є перевантаженість інтерфейсу та необхідність часу на опанування програмою, серед переваг варто відміти її безкоштовність та кросплатформеність.

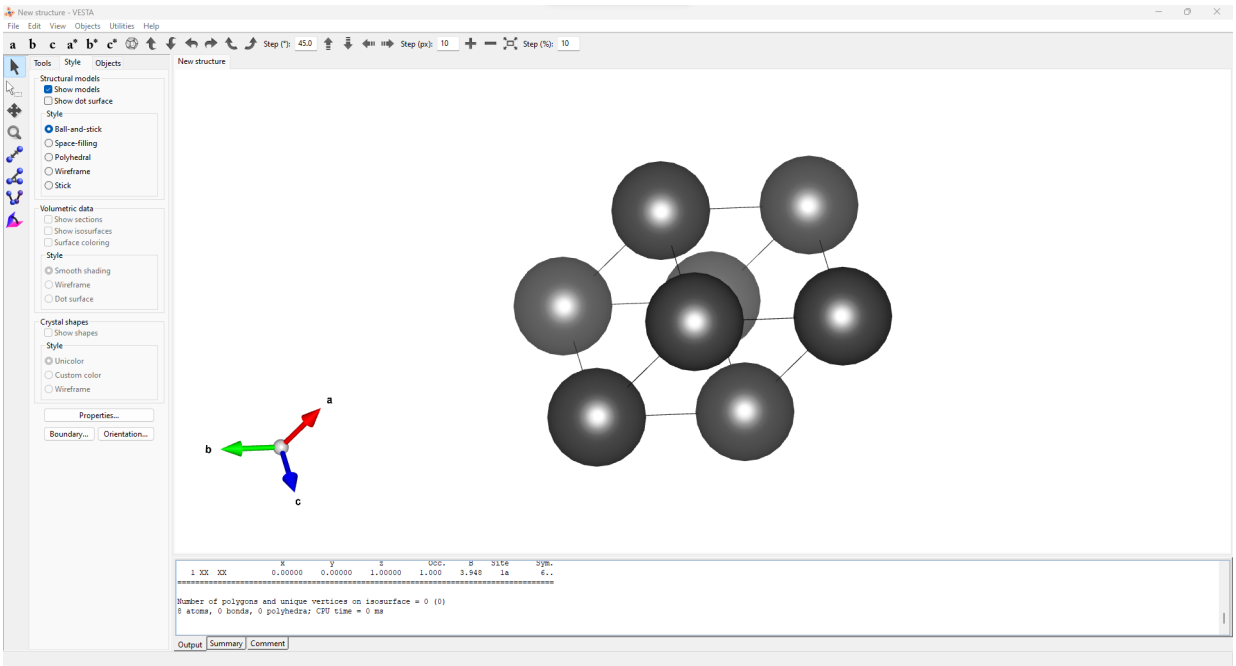


Рисунок 2.10 – Демонстрація кубічної кристалічної ґратки в програмі VESTA

2.6 Моделювання хімічних сполук за допомогою Avogadro

Avogadro [9] - це відкрите програмне забезпечення для моделювання молекул та обчислення хімічних властивостей. Вона надає потужні інструменти для побудови та візуалізації молекулярних структур, включаючи атоми, зв'язки, молекулярні групи та кристалічні структури. Avogadro дозволяє редагувати структури, додавати атоми, видаляти зв'язки, налаштовувати конформації та робити маніпуляції зі структурами. Вона також має вбудований набір інструментів для розрахунку хімічних властивостей, включаючи енергію, геометрію, спектроскопію та термодинамічні властивості. Крім того, Avogadro підтримує інтеграцію з іншими програмами, що дозволяє легко обмінюватися даними та результатами з іншими науковими інструментами. Ця програма широко використовується в хімічних дослідженнях, викладанні та виробничих задачах, надаючи зручне та потужне середовище для вивчення та моделювання хімічних систем. Серед переваг є її відкритий код, кросплатформеність, безкоштовність. Хоча інтерфейс не здається перевантаженим все ж необхідно витратити час на її опанування, а в цьому можуть допомогти документація та навчальний посібник з офіційного сайту програми.

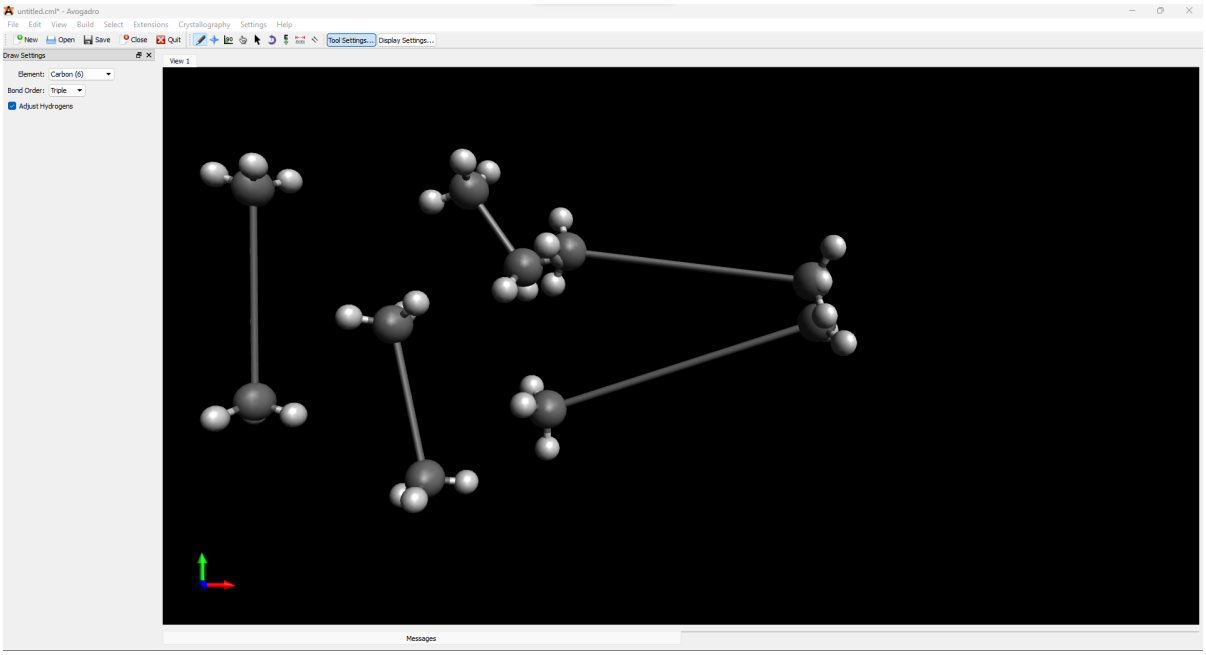


Рисунок 2.11 – Створення зв’язків атомів карбону.

2.7 Застосування Material Studio в науці

Material Studio [10] - це комплексне програмне забезпечення для моделювання та симуляції матеріалів на молекулярному та атомному рівнях. Воно надає вченим та дослідникам потужні інструменти для вивчення структури, властивостей та поведінки матеріалів.

Material Studio включає широкий набір модулів та інструментів, що дозволяють проводити різноманітні види обчислювальних симуляцій. Наприклад, використовуючи модуль CASTEP, можна проводити абінітні обчислення енергії, структури та електронної структури кристалічних матеріалів методом густини функціоналу. Модуль DMol3 дозволяє моделювати хімічні реакції, включаючи реакції на поверхні. Модуль Forcite дозволяє проводити молекулярні динамічні симуляції для вивчення поведінки матеріалів в різних умовах.

Material Studio також має потужні інструменти для візуалізації та аналізу даних. Ви можете відображати структури, проводити аналіз взаємодій, визначати кристалічні поверхні, розраховувати фізичні властивості, відтворювати експериментальні дані та багато іншого. Програма також підтримує інтеграцію з іншими програмами та базами даних.

BIOVIA Materials Studio Visualizer є основним продуктом програмного комплексу Materials Studio, розробленого для підтримки потреб моделювання матеріалів в хімічних та матеріалознавчих галузях промисловості. BIOVIA Materials Studio Visualizer містить основні функціональні можливості моделювання, необхідні для підтримки обчислювальної науки про матеріали (див Рисунок 2.12). Вона допомагає розуміти властивості або процеси, пов'язані з молекулами та матеріалами.

Серед переваг є її дуже потужний функціонал, а серед недоліків висока ціна.

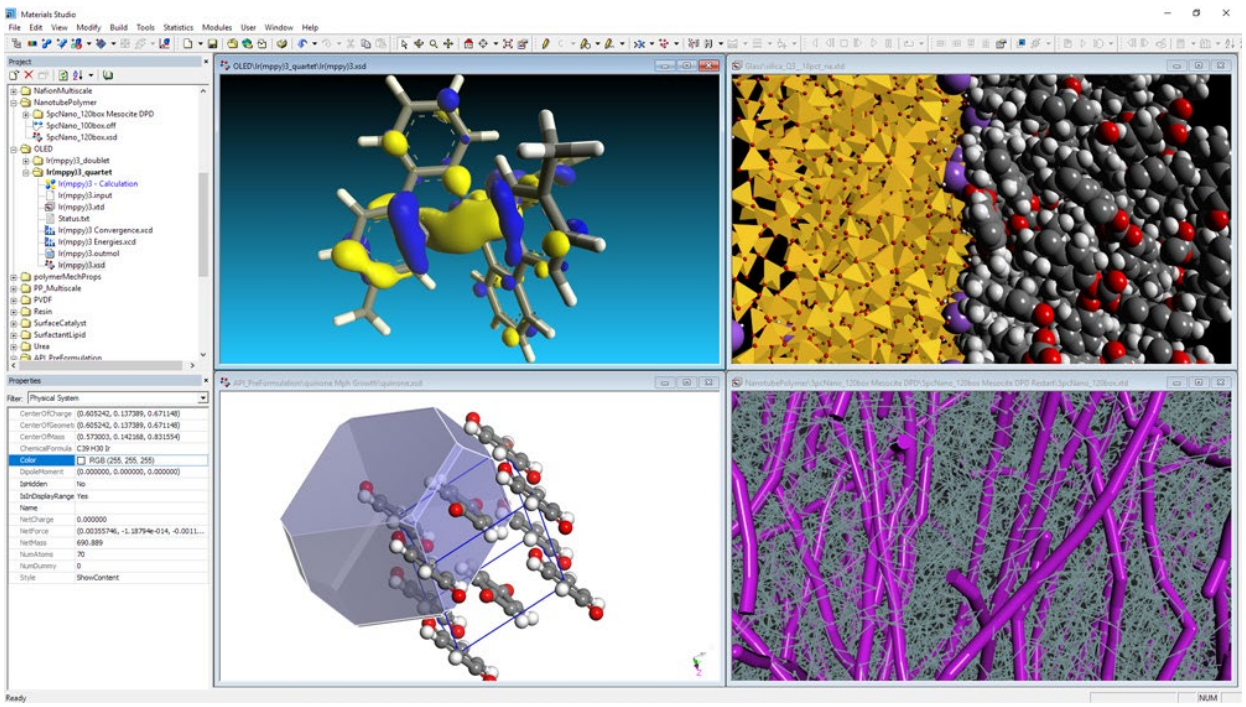


Рисунок 2.12 – Знімок екрана програми BIOVIA Materials Visualizer, основного продукту Materials Studio. BIOVIA Materials Studio може відображати широкий спектр матеріалів, від масивних аморфних полімерів до органічних та неорганічних кристалів.

3 ЗОВНІШНЄ ПРОЕКТУВАННЯ

3.1 Вимоги до функціональних характеристик

Програма повинна мати інтуїтивно зрозумілий та привабливий графічний інтерфейс, який забезпечує зручну взаємодію з користувачем. У графічному інтерфейсі повинні бути доступні всі необхідні поля для введення даних. Користувач може вводити значення різних параметрів, які впливають на виконання програми, в тому числі параметри кристалічної ґратки. Інтерфейс програми має бути поділений декілька розділів, один з яких найголовніший – візуалізатор, який забезпечує відображення кристалічної ґратки у тривимірному форматі. Повинні бути передбачені різні операції з взаємодії з 3D моделлю і збереженням інформації в файл.

.

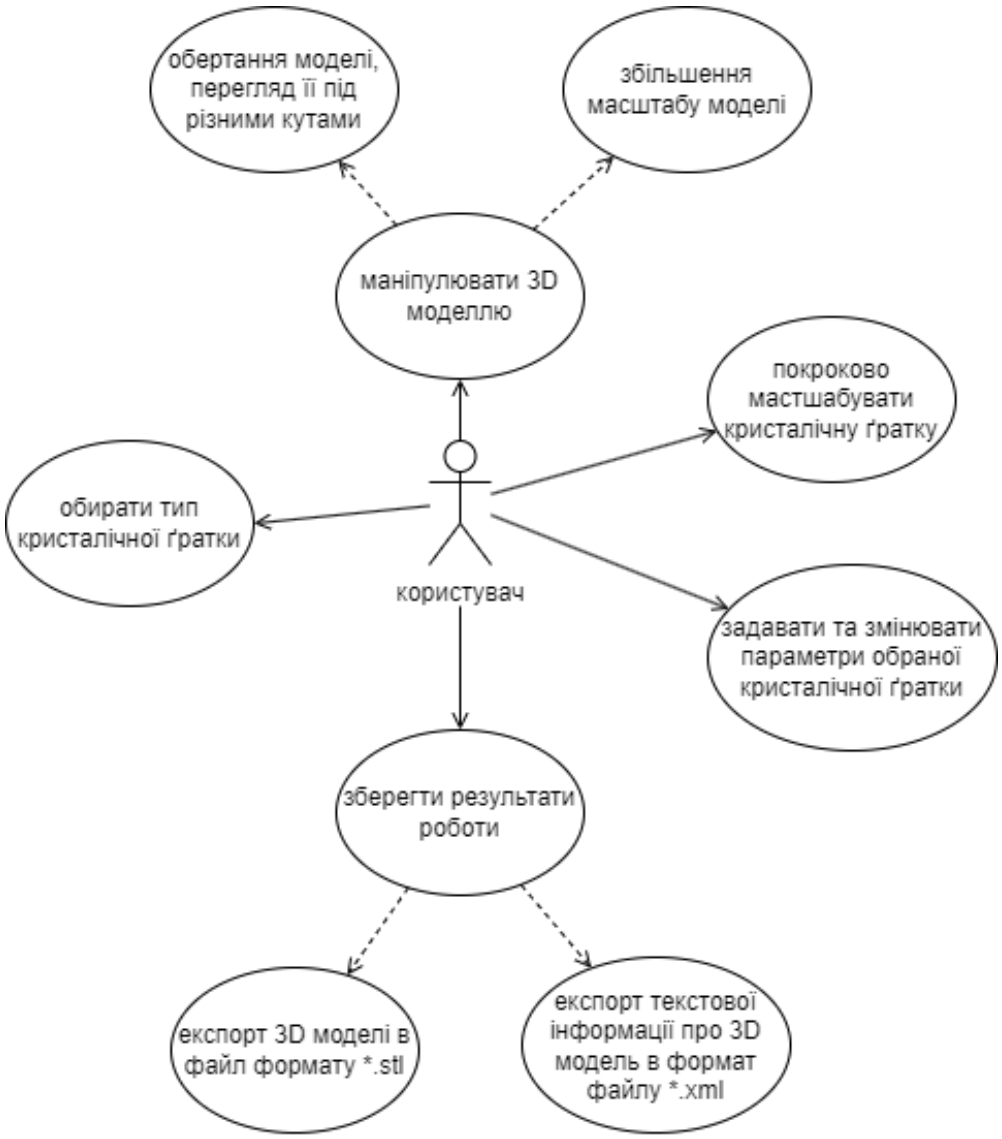


Рисунок 3.1 – Діаграма прецедентів

3.2 Вимоги до складу виконуваних функцій

- редагування параметрів кристалічної ґратки;
- візуалізація 3D моделі кристалічної ґратки;
- огляд 3D моделі під різними кутами та в різному масштабі;
- покрокове створення кристалічної ґратки;
- можливість переглядати стан кожного кроку;
- експорт отриманої фігури в файл формату *.stl;
- експорт інформації про фігуру в файл формату *.xml.

3.3 Вимоги до експлуатаційних характеристик

Ця програма створена з метою дослідження та візуалізації кристалічних ґраток. Вона дозволяє користувачам відтворювати тривимірний вигляд кристалу з усіма його структурними деталями та елементами.

Студенти, викладачі та науковці можуть використовувати цю програму для навчання та дослідження кристалографії. Вони можуть вивчати основні принципи будови кристалічних ґраток, відтворювати кристалічну структуру різних типів кристалів та спостерігати їхні властивості, що допомагає зрозуміти вплив змін в хімічному складі та фізичних умовах на структуру кристалів.

3.4 Вхідні та вихідні дані

Вхідні дані є параметри кристалічної ґратки, такі як довжина ребер, кути між ребрами, які можуть бути вводитись з клавіатури, перелік типів кристалічних ґраток Браве. За допомогою маніпулятора миші можна обертати відображувану фігуру, масштабувати її розмір.

Вихідними даними є графічна 3D модель, що відображається у вікні програми та може буди експортована у форматі *.stl, або її текстовий опис у форматі *xml.

3.5 Вимоги до надійності

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

3.6 Вимоги до складу і параметрів технічних засобів

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

1) Двоядерний процесор Pentium з тактовою частотою 1.6 ГГц;
2) Зовнішній або вбудований 3D відеоадаптер Nvidia, Intel, AMD
3) об’єм оперативної пам’яті 8 ГБ;
4) пам’ять на жорсткому диску 400 Мб.
5) CD-привод або USB-порт для перенесення програми на комп’ютер;
6) маніпулятор «миша»;
7) клавіатура;
8) монітор з роздільною здатністю 1280*720;
9) 3D принтер для роздрукування геометричних фігур.

3.7 Вимоги до інформаційної і програмної сумісності;

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

1) ОС: Windows 10/11;
2) Microsoft .NET Framework версії 4.8.1 та 4.6.2 (веб-інсталятор);
3) мова програмування C#, .NET Framework.

4 ВНУТРІШНЄ ПРОЕКТУВАННЯ

4.1 Огляд підходів та технологій

Процес написання програми вимагає детального передчасного аналізу та планування. Окрім вимог до програми потрібно завжди враховувати власні навички та досвід з вирішення подібної проблеми, а також найнеобхідніше – це час розробки програми та складність її подальшої підтримки. Виходячи з вимог до задачі та власного досвіду потрібно обирати ті інструменти, технології та підходи, які можуть якомога точніше вирішити проблему за оптимальний час та не викликатимуть складнощів під час подальшого супроводу програми.

Проблема, яку повинна вирішувати дана програма – це моделювання та візуалізація процесу побудови кристалічної ґратки. Головні вимоги до програми – мати зручний інтерфейс та бути швидкою. Так як програма створюється з навчально-демонстраційною метою, то інтерфейс програми повинен бути спрощений, інтуїтивно зрозумілий та не вимагати хоч якого часу на його опанування.

Вирішення цієї проблеми не є складним: можна використовувати різні мови програмування, фреймворки та бібліотеки, головне щоб зберігався баланс між швидкість та зручністю розробки. Враховуючи власний досвід з розробки інтерфейсу користувача було обрано об’єктно-орієнтовану мову програмування C# та фреймворк, що її використовує .NET Framework.

.NET Framework – це платформа, яка має широкий спектр інструментів та засобів для вирішення різних задач. Її перевагою також є вбудоване керування пам’яттю, обробка виключень, велика кількість сторонніх бібліотек з вільною ліцензією, що значно спрощує написання програм. Слід зауважити, що використовується саме Framework версія фреймворку, яка може здаватися трохи застарілою, але надає більше можливостей для розробки настільних додатків під операційну систему Windows, ніж її кросплатформена версія .NET.

Для побудови настільного додатку вирішено використовувати технологію WPF – Windows Presentation Platform [11]. За допомогою неї можна писати інтерфейс у декларативному стилі на мові XAML окремо від логіки додатка. Для 3D візуалізації використовується бібліотека Helix Toolkit [12]. Дана бібліотека надає компоненти для

відображання 3D графіки за допомогою процесора або відеокарти. Забігаючи на перед можна сказати, що версія для процесор як основне джерело розрахунків не задовольняє потреби з швидкодії, хоча може використовуватись на більш слабких персональних комп'ютерах. 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.

Є ще інші підходи та технології, які варто відміти, але не були обрані через відсутність досвіду або через великий об'єму часу на опанування. Першим з них є зв'язка мови програмування C++, фреймворку Qt [13] та API Vulkan [14]. Перевага полягає в повному контролі ресурсів ПК, побудова власного рендер движку за рахунок чого можна отримати великий приріст в швидкодії, але побудова власного движку вимагає написання великої кількості коду та часу на розуміння всіх деталей та нюансів з обробки графіки. Другим підходом є використання ігрового движку, час опанування якими також вимагає час, хоча перевага полягає в використанні вже добре налагодженому процесі розробки, а недоліком в ігрових движках є побудова користувацького інтерфейсу. Вони добре орієнтовані на вирішенні проблем з 3D графіки, але зручність та швидкість написання коду для 2D інтерфейсу залишає бажати більшого.

Таким чином враховуючи баланс між швидкістю, зручністю розробки та вимогами до програми стали зв'язка WPF та Helix Toolkit. Використання WPF також зумовлено архітектурним патерном MVVM – Model View ViewModel [15]. Суть даного патерну полягає в його назві: розділення між логікою програми та зовнішнім виглядом. Такий підхід є дуже гнучким та дозволяє ефективно перевикористовувати вже існуючий код. Використання додаткового шару ViewModel дозволяє змінювати поведінку зовнішнього інтерфейсу без необхідності переписувати його під конкретні випадки та не змінювати модель даних звісно. Ще однією з особливостей цього підходу є поняття біндингу [15] – процес за допомогою якого досягається взаємний вплив на відображувані дані зі сторони зовнішнього користувача та всередині коду.

Програма написана в об'єктно-орієнтованому стилі з дотриманням принципів GRASP [16] та SOLID [17], а також з принципом інверсії контролю (англ. «Inversion

of Control») [18]. Проте слідування принципам не дотримано в повній мірі, щоб не роздувати код та штучно не ускладнювати додатковими абстракціями.

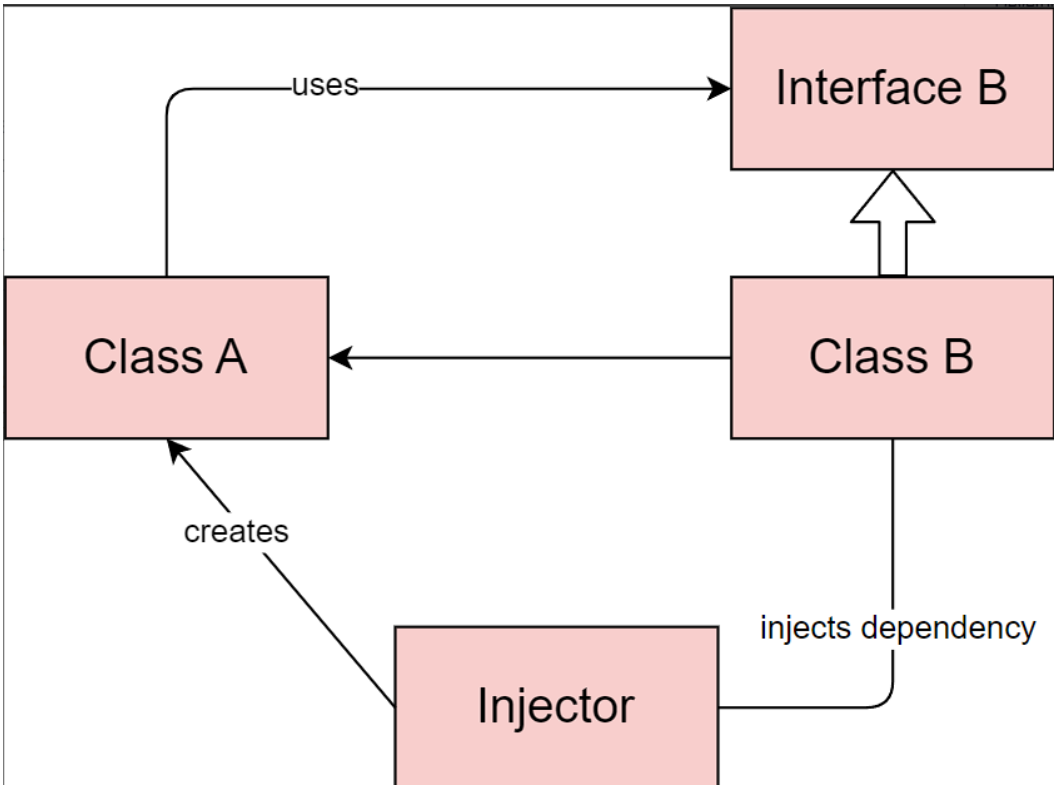


Рисунок 4.1 Приклад «ін'єкції залежності», що є складовою принципа інверсії контролю

4.2 Моделювання сутностей системи

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

Перша група сутностей предметної області: тип кристалічної ґратки, атрибути кристалічної ґратки, вершина, ребро, граф, правило заміни, конструктор, 3D сцена, графічний примітив, графічний об'єкт.

Обов'язки першої групи сутностей:

– тип кристалічної ґратки – тип ґратки Браве, що описує основну різницю між іншими кристалічними ґратками та класифікує ґратку його за її типом центрування і кристалічною системою;

– атрибути кристалічної ґратки – довжини ребер a, b, c, кути між ними, кольори;

– вершина – абстрактна атомарна частина кристалічної ґратки, що має позицію, розмір та колір;

– ребро – абстрактна атомарна частина кристалічної ґратки що поєднує дві вершини між собою, також має розмір;

– граф – сукупність вершин та ребер, що уявляють собою суперкомірку кристалічної ґратки;

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

– конструктор – абстрактний об’єкт що будує графову суперкомірку за обраною кристалічною ґраткою;

– 3D сцена – частина інтерфейсу, яка візуалізує у 3D графіці утворену суперкомірку, тобто граф;

– графічний примітив – форма, яку приймає абстрактна сутність, тобто вершина або ребро, у 3D сцені;

– об’єкт сцени – елемент 3D сцени, що має форму, матеріал, позицію та усю додаткову необхідну інформацію для відображання на 3D сцені;

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

Таблиця 4-1 Сутності, атрибути та методи

Сутність	Атрибути	Методи
1	2	3
тип кристалічної ґратки	являє собою сімейство класів, кожна з яких створює свою графову кристалічну ґратку; містить атрибути кристалічної ґратки, головний лівій та правий кольори, а також атрибути, що позначають можливість редагувати параметри кристалічної ґратки	створити граф кристалічної ґратки.

Продовження таблиці 4.1

вершина (vertice)	Size – розмір вершини, який буде використовуватись для графічного примітиву на 3D сцені – додатне дійсне число; Color – колір вершини, використовується в алгоритмах та для кольору графічного примітиву – структура, яка містить числову інформацію про відтінки кольорів. Position – позиція у світових координатах на 3D сцені – структура Point, яка містить числову інформацію координат x, y, z.	створити вершину; клонувати.
ребро (edge)	Size – розмір вершини, який буде використовуватись для графічного примітиву на 3D сцені – додатне дійсне число; Start – вершина, з якої починається ребро, являє собою сутність vertice; End – вершина, в якій закінчується ребро, являє собою сутність vertice;	створити ребро; клонувати.
граф (graph)	Vertices – вершини графу – список сутностей типу Vertice; Edges – ребра графу – список сутностей типу Edge;	додати вершину; 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;	встановити ліву частину правила; встановити праву частину правила; перемалювати ліву частину;

Продовження таблиці 4.1

	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.	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: AddedVertices – список вершин, які були додані під час заміни, елементи сутності Vertice; AddedEdges – список ребер, які були додані під час заміни, елементи сутності Edge;	виконати заміну; очистити суперкомірку.
3D сцена	Graph – відображує абстрактну сутність графу, яку візуалізує – тип Graph; CameraPos – позиція камери на сцені – сутність Point; CameraLook – напрям, в який дивиться камера, – структура Vector, яка містить числову інформацію координат x, y, z; UpDirection – позначає вісь, яка дивиться в гору, тобто визначає напрямок системи координат – сутність Vector; SceneTree – усі графічні об’єкти, які вона відображує.	відобразити графічну інформацію; методи керування камерою;

Продовження таблиці 4.1

графічний примітив	Сутність, яка має форму, колір і матрицю трансформації, специфікація сутності залежить від використовуваної бібліотеки.	створити графічний примітив.
об’єкт сцени	Специфікація визначається бібліотекою, має усю необхідні атрибути та методи для візуального представлення на 3d сцені.	

Друга група складається з сутностей графічного інтерфейсу та його поділ у вигляді компонентів як пари View, ViewModel як частину патерну MVVM

Друга група сутностей програми – компоненти інтерфейсу. 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:

– системне горизонтальне меню – меню програми, що містить елементи управління з системою, такі як експорт та імпорт файлів;

– меню керування – містить сукупність кнопок та полів, необхідних для виконання основних функцій програми;

– список конструкторів – відображає список конструкторів, які можна обрати та використовувати та побудови суперкомірки

– представлення правила – відображає обрану ліву і праву частину правила, які показують як буде виконуватись підстановка фігур на сцені

– головна сцена програми – відображає 3D сцену.

Третя 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

– CrystalService – інкапсолює роботу основних алгоритмів;

– MainController – конфігрує початкові налаштування програми.

– Defaults – константі значення для всього простору програми.

Четверта група є додатковою та містить класи, що виявились необхідними під час розробки програми. Це статичні класи розширення, класи DTO [19] – Data Transfer Object, за допомогою яких легко імпортувати та експортувати файли, допоміжні класи для користувацького інтерфейсу, так як конвертери даних.

Визначивши сутності програми представимо залежності між ними, складемо основі діаграми класів, а також представимо їх вигляді CRC-карток [] (Class - Responsibility- Collaboration), так як фізично відобразити зв’язки між ними всіма на одній діаграмі не є можливим. Для уникнення непорозуміння будуть використовуватись англійські назви класів, які більш точно відображають сутності.

Результат залежностей представлено у наступній таблиці:

Таблиця 4-2 Моделювання залежностей

Клас, який зв'язується	Клас, з яким зв’язуються	Тип зв’язку
MainController	IUIService	Асоціація
	ICrystalService	Асоціація
	IStateManager	Асоціація
	ICommandManager	Асоціація

Продовження таблиці 4.2

	IRuleController	Асоціація
	IUIView	Асоціація
UIService	IUIService	Реалізація
	IUIView	Асоціація
	ViewportView	Асоціація
	ViewportViewModel	Асоціація
	MainView	Асоціація
	MainViewModel	Асоціація
	SideMunuView	Асоціація
	SideMunuViewModel	Асоціація
	RulesListView	Асоціація
	RulesListViewModel	Асоціація
	RuleView	Асоціація
	RuleViewModel	Асоціація
CrystalService	ICrystalService	Реалізація
	IStateManager	Агрегація
	ICommandManager	Агрегація
	IConstructor	Агрегація
	IViewportControler	Агрегація
	Defaults	Залежність
	StateEventArgs	Залежінсть
	CommandEventArgs	Залежінсть
	Graph	Залежінсть
	GraphDto	Залежінсть
StateManager	IStateManager	Реалізація
	LatticeConstructor	Агрегація
	IConstructor	Агрегація
	IRule	Агрегація

Продовження таблиці 4.2

	IViewportController	Агрегація
	IView	Агрегація
	StateEventArgs	Залежінсть
CommandManager	ICommandManager	Реалізація
	CommandEventArgs	Залежінсть
RuleController	IRuleController	Реалізація
	IUIService	Агрегація
	IStateManager	Агрегація
	IViewportController	Агрегація
	IUIView	Агрегація
	IRule	Агрегація
	RulesListViewModel	Композиція
	RuleViewModel	Композиція
	LatticeConstructor	Асоціація
	StateEventArgs	Залежінсть
	ListItemViewModel	Залежінсть
	TriclinicLatticeSimpleConstructor	Залежінсть
	TriclinicLatticeFullConstructor	Залежінсть
	BodyCenteredLatticeConstructor	Залежінсть
	BaseCenteredLatticeConstructor	Залежінсть
	FaceCenteredLatticeConstructor	Залежінсть
	HexagonalConstructor	Залежінсть
MainVeiwModel	BaseViewModel	Узагальнення
	IStateManager	Агрегація
	IUIService	Агрегація
	IUIView	Агрегація
	SideMenuVeiewModel	Композиція

Продовження таблиці 4.2

MainWindowViewModel	BaseViewModel	Узагальнення
	IStateManager	Агрегація
	ICommandManager	Агрегація
	IUIService	Агрегація
	IUIView	Агрегація
	MainVeiwModel	Композиція
	RelayCommand	Асоціація
RulesListViewModel	BaseViewModel	Узагальнення
	IUIService	Агрегація
	ListItemViewModel	Композиція
	LatticeConstructor	Залежність
	ViewportController	Залежність
ListItemViewModel	ViewportController	Агрегація
	LatticeConstructor	Агрегація
	IUIView	Асоціація
	IGraph	Асоціація
RuleViewModel	BaseViewModel	Узагальнення
	IUIView	Агрегація
	RelayCommand	Агрегація
	IRule	Агрегація
SideMenuViewModel	BaseViewModel	Узагальнення
	IStateManager	Агрегація
	ICommandManager	Агрегація
	RelayCommand	Агрегація
	StateEventArgs	Залежність
ViewportViewModel	BaseViewModel	Узагальнення
MainView	IUIView	Реалізція
	BaseViewModel	Агрегація

Продовження таблиці 4.2

MainWindowView	IUIView	Реалізція
	BaseViewModel	Агрегація
RulesListView	IUIView	Реалізція
	BaseViewModel	Агрегація
RuleView	IUIView	Реалізція
	BaseViewModel	Агрегація
SideMenuView	IUIView	Реалізція
	BaseViewModel	Агрегація
ViewportView	IUIView	Реалізція
	BaseViewModel	Агрегація
LatticeConstructor	IGraph	Агрегація
TriclinicLatticeSimpleConstructor	LatticeConstructor	Узагальнення
	IGraph	Агрегація
	Vertice	Залежність
	Edge	Залежність
TriclinicLatticeFullConstructor	LatticeConstructor	Узагальнення
	IGraph	Агрегація
	Vertice	Залежність
	Edge	Залежність
BodyCenteredLatticeConstructor	LatticeConstructor	Узагальнення
	IGraph	Агрегація
	Vertice	Залежність
	Edge	Залежність
BaseCenteredLatticeConstructor	LatticeConstructor	Узагальнення
	IGraph	Агрегація
	Vertice	Залежність
	Edge	Залежність
FaceCenteredLatticeConstructor	LatticeConstructor	Узагальнення

Продовження таблиці 4.2

	IGraph	Агрегація
	Vertice	Залежність
	Edge	Залежність
HexagonalConstructor	LatticeConstructor	Узагальнення
	IGraph	Агрегація
	Vertice	Залежність
	Edge	Залежність
Constructor	IConstructor	Реалізцаія
	IGraph	Агрегація
	IRule	Агрегація
	Vertice	Агрегація
	Edge	Агрегація
	SubstitutionEventArgs	Залежність
Rule	IRule	Реалізцаія
	IGraph	Агрегація
	RuleEventArgs	Залежність
Graph	IGraph	Реалізцаія
	Edge	Агрегація
	Vertice	Агрегація
	GraphDto	Залежність
Edge	Vertice	Агрегація
GraphDto	VerticeDto	Агрегація
	EdgeDto	Агрегація
	Vertice	Залежність
	Edge	Залежність
VerticeDto	Vertice	Залежність

Встановивши зв’язки, представимо заключний результат у вигляді CRC карток:

Таблиця 4.3 CRC кратка для класу MainController

Базовий клас	Похідні класи (нащадки)
відсутній	відсутній
Обов’язки	Зв’язки
Головний контролер програми, виконує реєстрацію та ініціалізацію сервісів.	IUIService, ICrystalService IStateManager, ICommandManager, IRuleController, IUIView

Таблиця 4.4 CRC кратка для класу UIService

Базовий клас	Похідні класи (нащадки)
відсутній	відсутній
Обов’язки	Зв’язки
Реєструє пари View-ViewModel, для необхідної ViewModel створює відповідній примірник View. Також окремо створює примірник IViewportController, який необхідній для списку конструкторів та для головної сцени програми.	IUIService, IUIView, ViewportView, ViewportViewModel, MainView, MainViewModel, SideMunuView, SideMunuViewModel, RulesListView, RulesListViewModel, RuleView, RuleViewModel

Таблиця 4.5 CRC кратка для класу CrystalService

Базовий клас	Похідні класи (нащадки)
ICrystalService	відсутній
Обов’язки	Зв’язки
Інкапсулює процес створення суперкомірки, використовуючи IRule та Constructor, спілкується з RuleController через StateManager.	IStateManager, ICommandManager, IConstructor, IViewportControler, Defaults, StateEventArgs, CommandEventArgs, Graph, GraphDto

Таблиця 4.6 CRC кратка для класу StateManager

Базовий клас	Похідні класи (нащадки)
IStateManager	відсутній
Обов’язки	Зв’язки
Містить спільний для програми стан, виступаючи посередником для обміну даними між класами, які не повинні напряму бути взаємопов’язані.	LatticeConstructor, IConstructor, IRule, IViewportController, IView, StateEventArgs,

Таблиця 4.7 CRC кратка для класу CommandManager

Базовий клас	Похідні класи (нащадки)
ICommandManager	відсутній
Обов’язки	Зв’язки
Допомагає реагувати на дії користувача у декількох місцях програми за одну атомарну дію, наприклад, клік по кнопці тощо.	CommandEventArgs

Таблиця 4.8 CRC кратка для класу RuleController

Базовий клас	Похідні класи (нащадки)
IRuleController	відсутній
Обов’язки	Зв’язки
Інкапсулює роботу з вибору правила та його редагування, а також роботу з конструкторами. Дозволяє змінювати кольори правила, їх графи, обираючи відповідний конструктор.	IUIService, IStateManagerv, IViewportController, IUIView, IRule, RulesListViewModel, RuleViewModel, LatticeConstructor, StateEventArgs, ListItemViewModel, TriclinicLatticeSimpleConstructor, TriclinicLatticeFullConstructor,

Продовження таблиці 4.9

	BodyCenteredLatticeConstructor, BaseCenteredLatticeConstructor, FaceCenteredLatticeConstructor, HexagonalConstructor

Таблиця 4.10 CRC кратка для класу MainViewModel

Базовий клас	Похідні класи (нащадки)
BaseViewModel	відсутній
Обов’язки	Зв’язки
Прошарок між моделлю та інтерфейсом.	IStateManager, IUIService, IUIView, SideMenuVeiewModel

Таблиця 4.11 CRC кратка для класу MainWindowViewModel

Базовий клас	Похідні класи (нащадки)
BaseViewModel	відсутній
Обов’язки	Зв’язки
Прошарок між моделлю та інтерфейсом.	IStateManager, ICommandManager, IUIService, IUIView, MainViewModel, RelayCommand

Таблиця 4.12 CRC кратка для класу RulesListViewModel

Базовий клас	Похідні класи (нащадки)
BaseViewModel	відсутній
Обов’язки	Зв’язки
Містить інформацію для відображання списку конструкторів.	IUIService, ListItemViewModel, LatticeConstructor, ViewportController

Таблиця 4.13 CRC кратка для класу ListItemViewModel

Базовий клас	Похідні класи (нащадки)
відсутній	відсутній
Обов’язки	Зв’язки
Використовується як цілісна частина списку конструкторів.	ViewportController, LatticeConstructor, IUIView IGraph

Таблиця 4-14 CRC кратка для класу RuleViewModel

Базовий клас	Похідні класи (нащадки)
BaseViewModel	відсутній
Обов’язки	Зв’язки
Прошарок між логікою та інтерфейсом користувача.	IUIView, RelayCommand, IRule

Таблиця 4.15 CRC кратка для класу SideMenuViewModel

Базовий клас	Похідні класи (нащадки)
BaseViewModel	відсутній
Обов’язки	Зв’язки
Прошарок між інтерфейсом користувача та логікою програми.	IStateManager, ICommandManager, RelayCommand, StateEventArgs

Таблиця 4.16 CRC кратка для класу ViewportViewModel

Базовий клас	Похідні класи (нащадки)
BaseViewModel	відсутній
Обов’язки	Зв’язки
За допомогою цього класу можливо впливати на 3D сцену, але через атрибути і	UIService

Продовження таблиці 4.15

методи що він надає, слугує свого роду інтерфейсом до сцени.	

Таблиця 4.17 CRC кратка для класу MainView

Базовий клас	Похідні класи (нащадки)
IUIView	відсутній
Обов’язки	Зв’язки
Інтерфейсна частина програми, яка містить усі інші компоненти інтерфейсу, окрім головного меню програми.	BaseViewModel

Таблиця 4.18 CRC кратка для класу MainWindowView

Базовий клас	Похідні класи (нащадки)
IUIView	відсутній
Обов’язки	Зв’язки
Весь інтерфейс програми, містить усі інші інтерфейсі компоненти.	BaseViewModel

Таблиця 4.19 CRC кратка для класу RulesListView

Базовий клас	Похідні класи (нащадки)
IUIView	відсутній
Обов’язки	Зв’язки
Відображає список конструкторів.	BaseViewModel

Таблиця 4.20 CRC кратка для класу RuleView

Базовий клас	Похідні класи (нащадки)
IUIView	відсутній
Обов’язки	Зв’язки
Відображає інформацію про правило, яке використовується на інтерфейсі.	BaseViewModel

Таблиця 4.21 CRC кратка для класу SideMenuView

Базовий клас	Похідні класи (нащадки)
IUIView	відсутній
Обов’язки	Зв’язки
Меню для управління процесом створення суперкомірки, також відображає додаткову інформацію.	BaseViewModel

Таблиця 4.22 CRC кратка для класу ViewportView

Базовий клас	Похідні класи (нащадки)
IUIView	відсутній
Обов’язки	Зв’язки
Показує 3D сцену.	BaseViewModel

Таблиця 4.23 CRC кратка для класу LatticeConstructor

Базовий клас	Похідні класи (нащадки)
відсутній	TriclinicLatticeSimpleConstructor, TriclinicLatticeFullConstructor, BodyCenteredLatticeConstructor, BaseCenteredLatticeConstructor, FaceCenteredLatticeConstructor, HexagonalConstructor

Продовження таблиці 4.22

Обов’язки	Зв’язки
Є базовим абстрактним класом для всіх інших конструкторів, містить інформацію про спільні для них атрибути.	IGraph

Таблиця 4-24 CRC кратка для класу TriclinicLatticeSimpleConstructor

Базовий клас	Похідні класи (нащадки)
LatticeConstructor	відсутній
Обов’язки	Зв’язки
Конструктор неповної триклинної кристалічної ґратки.	IGraph, Vertice, Edge

Таблиця 4.25 CRC кратка для класу TriclinicLatticeFullConstructor

Базовий клас	Похідні класи (нащадки)
LatticeConstructor	відсутній
Обов’язки	Зв’язки
Конструктор повної триклинної кристалічної ґратки.	IGraph, Vertice, Edge

Таблиця 4.26 CRC картка для класу BodyCenteredLatticeConstructor

Базовий клас	Похідні класи (нащадки)
LatticeConstructor	відсутній
Обов’язки	Зв’язки
Конструктор повної об’ємноцентрованої кристалічної ґратки.	IGraph, Vertice, Edge

Таблиця 4.27 CRC картка для класу BaseCenteredLatticeConstructor

Базовий клас	Похідні класи (нащадки)
LatticeConstructor	відсутній
Обов’язки	Зв’язки
Конструктор повної базоцентрованої кристалічної ґратки.	IGraph, Vertice, Edge

Таблиця 4-28 CRC картка для класу FaceCenteredLatticeConstructor

Базовий клас	Похідні класи (нащадки)
LatticeConstructor	відсутній
Обов’язки	Зв’язки
Конструктор повної гранецентрованої кристалічної ґратки.	IGraph, Vertice, Edge

Таблиця 4.29 CRC картка для класу HexagonalConstructor

Базовий клас	Похідні класи (нащадки)
LatticeConstructor	відсутній
Обов’язки	Зв’язки
Конструктор повної гексагональної кристалічної ґратки.	IGraph, Vertice, Edge

Таблиця 4.30 CRC картка для класу Constructor

Базовий клас	Похідні класи (нащадки)
IConstructor	відсутній
Обов’язки	Зв’язки
Виконує підстановку усіх елементів відповідно до правила.	IGraph, IRule, Vertice, Edge, SubstitutionEventArgs

Таблиця 4.31 CRC картка для класу Rule

Базовий клас	Похідні класи (нащадки)
IRule	відсутній
Обов’язки	Зв’язки
Визначає яким чином повинна змінюватись суперкомірка.	IGraph, RuleEventArgs

Таблиця 4.32 CRC картка для класу Graph

Базовий клас	Похідні класи (нащадки)
IGraph	відсутній
Обов’язки	Зв’язки
Сукупність вершин та ребер, що можуть являти собою кристалічну ґратку, структуру або суперкомірку.	Vertice, Edge, GraphDto

Таблиця 4.33 CRC картка для класу Edge

Базовий клас	Похідні класи (нащадки)
відсутній	відсутній
Обов’язки	Зв’язки
Поєднує дві вершини.	Vertice

Таблиця 4.34 CRC картка для класу Vertice

Базовий клас	Похідні класи (нащадки)
відсутній	відсутній
Обов’язки	Зв’язки
Представляє інформацію про атомарну частину кристалічної ґратки.	VerticeDto

Таблиця 4.35 CRC картка для класу GraphDto

Базовий клас	Похідні класи (нащадки)
відсутній	відсутній
Обов’язки	Зв’язки
Використовується для імпорту/експорту до файлу з форматом *.xml.	VerticeDto, EdgeDto, Vertice, Edge

Таблиця 4-36 CRC картка для класу EdgeDto

Базовий клас	Похідні класи (нащадки)
відсутній	відсутній
Обов’язки	Зв’язки
Використовується для імпорту/експорту до файлу з форматом *.xml.	GraphDto

Таблиця 4.37 CRC картка для класу VerticeDto

Базовий клас	Похідні класи (нащадки)
відсутній	відсутній
Обов’язки	Зв’язки
Використовується для імпорту/експорту до файлу з форматом *.xml.	GraphDto, Vertice

Таблиця 4.38 CRC картка для класу BaseViewModel

Базовий клас	Похідні класи (нащадки)
відсутній	MainViewModel, MainWindowView-Model, RulesListViewModel, RuleViewModel, SideMenuView-Model, ViewportViewModel
INotifyPropertyChanged	Зв’язки

Продовження таблиці 4.37

Інкапсулює спільні для всіх ViewModel атрибути та методи.	

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

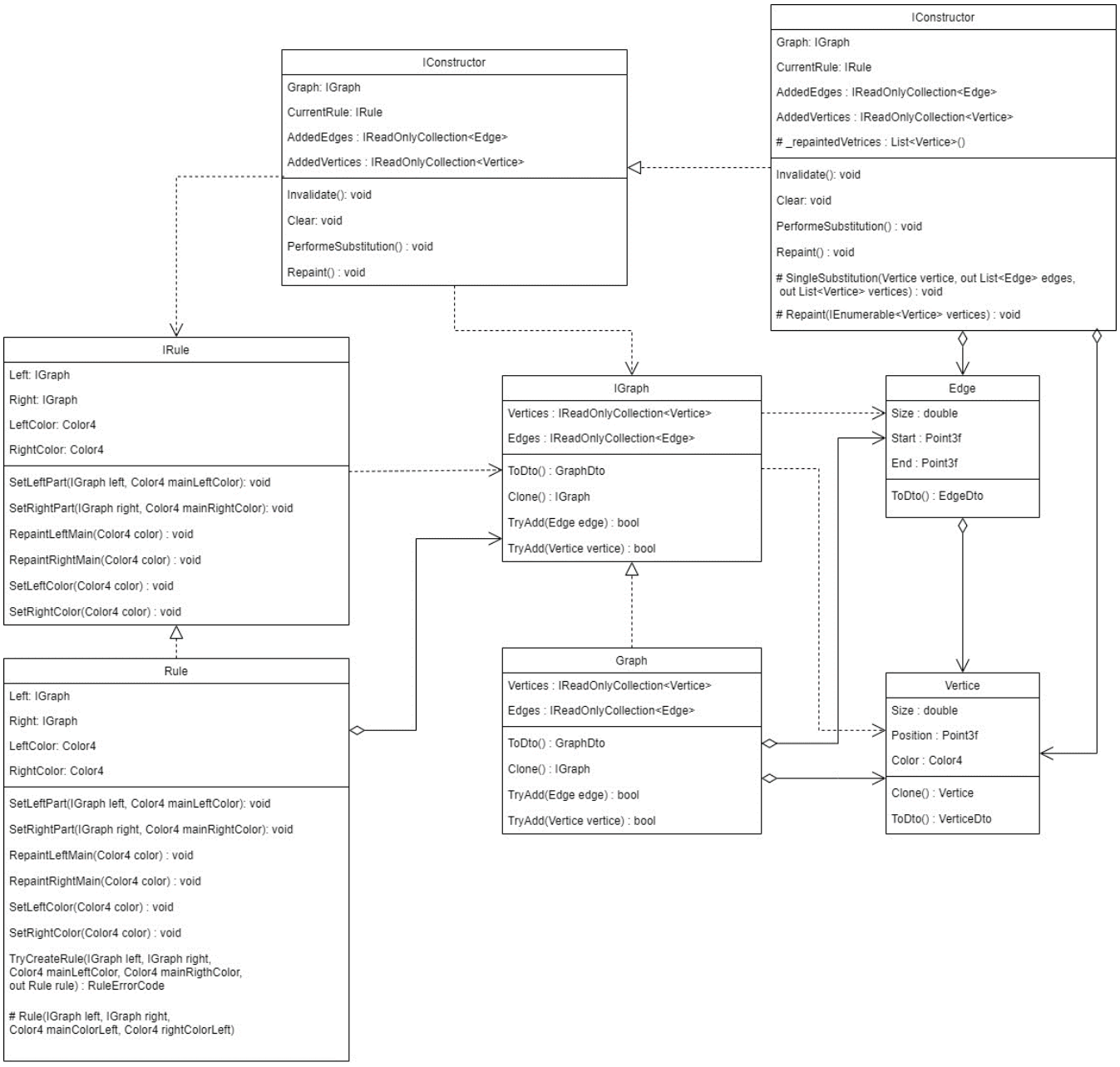


Рисунок 4.2 Діаграма класів абстрактних сутностей, що становлять логіку програми

4.3 Проектування інтерфейсу користувача

Для побудови інтерфейсу користувача було обрано технологію WPF, так як вона дозволяє швидко та зручно його описувати а також за допомогою нею легко реалізувати патерн MVVM. Спочатку було побудовано демонстраційну версію програми, з простим макетом. Метою демонстраційної версії було протестувати зручність користувацького інтерфейсу та виокремлення спільних за призначення елементів інтерфейсу.

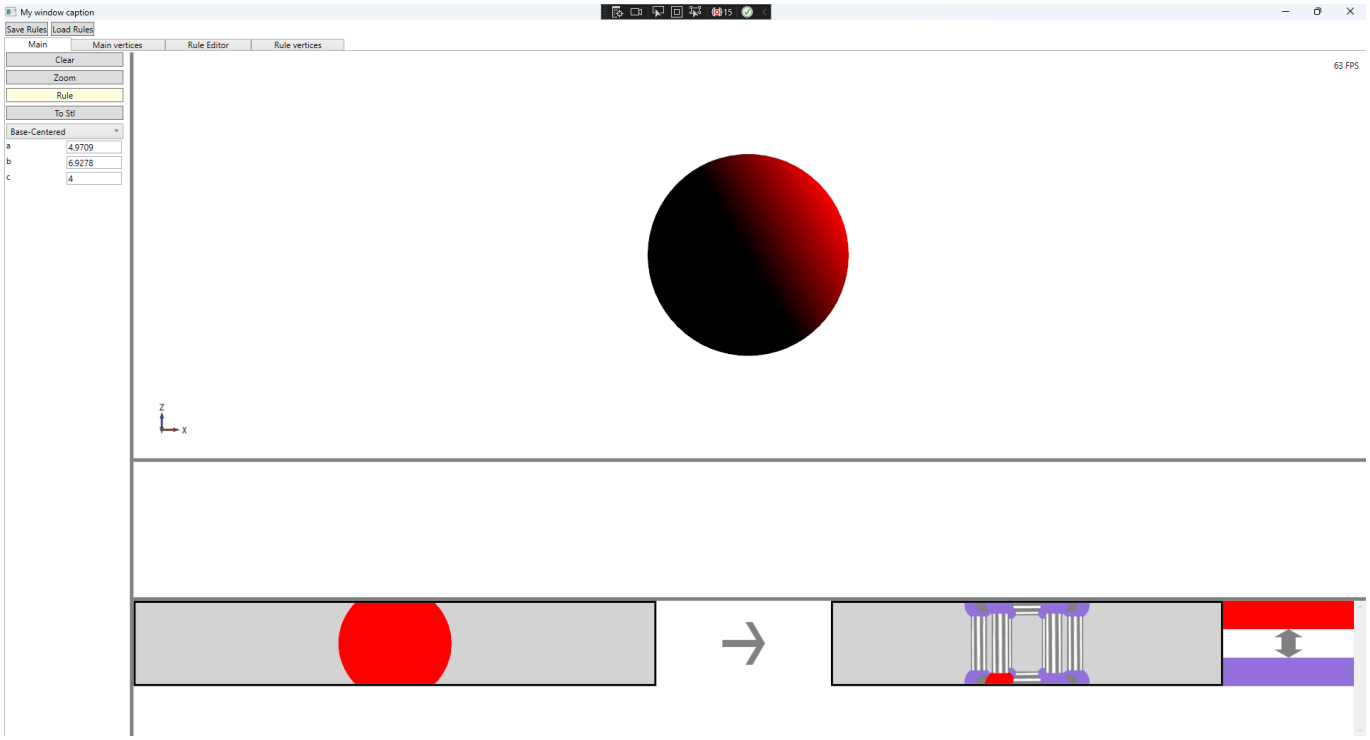


Рисунок 4.3 Демонстраційна версія програми та її інтерфейс

В результаті тестування інтерфейсу в демоверсії було прийнято рішення відмовитись від горизонтального меню, як складається з вкладок «Main», «Main vertices», «Rule editor», «Rule vertices», а також зробити перерозподіл місця між більш структурними елементами через бажання зробити досвід користувача більш приємним, а також позбутися зайвого функціоналу, який дублює вже існуючий.

В якості головної теми програми було обрано Material Design Theme [21], так як вона досить популярна і користувачі, скоріш за все, мали досвід з її використанням. Також використання готової теми скорочує час на розробку додатку, що допомагає сконцентрувати увагу на функціоналі програми.

Програма використовує наступну кольорову палітру:

Рисунок 4.4 Кольорова палітра програми

Містить наступні кольорои:

1. Темно-сірий – головний фоновий колір програми, код hex: #303030;
2. Світло-сірий – другорядний фоновий колір програми, код hex: #939393;
3. Фіолетовий – головний колір елементів управління, код hex: #9370DB;
4. Світло-зелений – другорядний колір елементів управління, код hex: #90EE90.

Визначившись з кольоровою палітрою, темою програми, а також врахувавши усі недоліки, було побудовано наступний інтерфейс користувача:

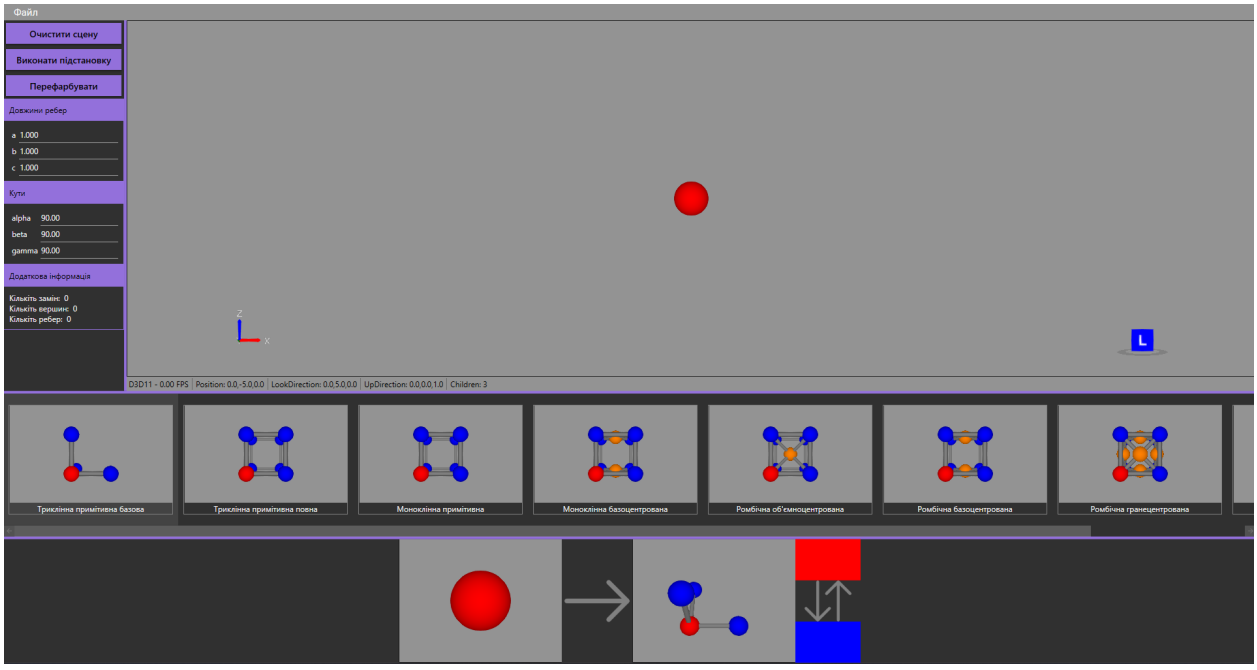


Рисунок 4.5 Кінцевий інтерфейс програми

Як видно, інтерфейс складається з 3D сцени, меню управління програмо, що містить додаткову інформацію про стан програми, горизонтальний список

конструкторів, а також інформаційної частини правила. Необхідності в діаграмі проектуванні стану систему не має багату сенсу, так як вихід з програми можливий з будь-якого стану програму, а усі сценарії роботи з програмою розроблені спеціально з найменшою кількістю послідовних дій.

4.4 Проектування динаміки системи

На основі use case діаграми побудуємо діаграму послідовностей для сценарію «обрати тип кристалічної ґратки», за допомогою якої можна зрозуміти, як вдалося створити зручний та компактний інтерфейс користувача:

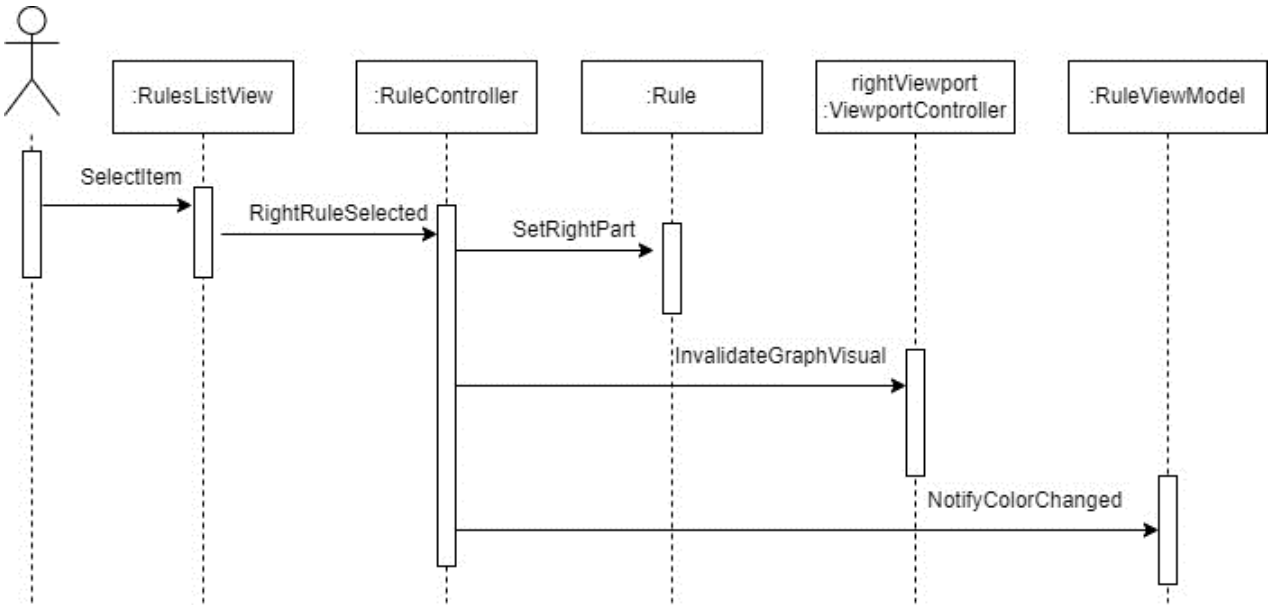


Рисунок 4.6 Діаграма послідовностей для сценарію «обрати тип кристалічної ґратки»

Як видно з діаграми послідовностей дії користувача перенаправляюсь в відповідний контролер, який змінює стан необхідних компонентів. Таким чином програма реагує і на інші дії користувача – делегує дію контролеру, який вже в свою чергу вирішує що і як повинно бути змінено.

5 МАТЕМАТИЧНА МОДЕЛЬ

5.1 Вибір підходу вирішення проблему

Існують різні підходи для вирішення проблеми з моделювання сутностей що подібні самі до себе, що є однією властивістю фракталів. До таким елементів зрозуміло належать фрактали, а також кристалічна структура. Для моделювання фракталів можна використовувати деякі з наступних підходів: алгоритмічний, функціонально-алгоритмічний із застосуванням системи ітерованих функцій на основі сукупності стискаючих відображень, L-систем, стискаючих афінних автоматів. Іншим підходом є використання конструктивно-продукційної структури КПС [21] [22], що заснована на формальних граматик. КПС дозволяє моделювати різноманітні конструкції та конструктивні процеси у різних сферах людської діяльності, таких як: інформаційних технологій, машинобудування, робототехніки, біології та інших, в тому числі хімії та її відгалуженням.

КПС зручно використовувати для моделювання та покрокової деталізації кристалічної структури, що складається з елементарних комірок та повторюються безліч разів і утворюють суперкомірку, так як особливе місце в цьому підході займають атрибути елементів та їх форма, що в даному разі характерно для кристалічної ґратки Браве: довжини ребер, кут між ними, та інші допоміжні атрибути.

Опис математичної наведено у роботі «Constructive-Synthesizing Structures and Their Grammatical Interpretations» [21], де детально описуються поняття конструкторів та їх властивості:

«We call a generalized constructive-synthesizing structure (GCSS) a triple

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

where M is the 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 a constructive axiomatics that includes a set of definitions, axioms, rules, properties, instructions, etc. The sets M and Σ are defined by the axiomatics»

5.2 Конструктор кристалічної ґратки

Виконаємо конкретизацію конструктора $C_{crystal}$, призначену на формування кристалічної структури.

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

$\Lambda_{crystal}$ визначає в даному випадку:

- вершина (термінал) – v, з атрибутами розміру $size = 1$, та координатами $pos = (x, y, z)$, атрибутом кольору $col = red$
- ребро (термінал) – e, з атрибутами початку $\text{start} = (x, y, z)$, та кінця $end = (x, y, z)$,
- граф (нетермінал) – graph, складаєтья з вершин v та ребер e;
- кристалічна ґратка (нетермінал) – cell, складаєтья з графу та має власні атрибути: довжини ребер *a, b, c,* кути між ребрами α, β, γ;
- початкові умови – граф, що має одну вершину v_0, з атрибутами ${}_{size}v_0 = 1$, ${}_{pos}v_0 = (0, 0, 0)$, ${}_{col}v_0 = red$.
- правило підстановки $\psi =< s, g >$, в якому операція складається з єдиного відношення підстановки, яке представлене рис. 5.1 та операцій над атрибутами $g = < col(left, right) >$, права частина підстановки повинна містити хоча б одну вершину з кольором як у лівому правилі.
- умова завершення задана користувачем кількістю ітерацій.

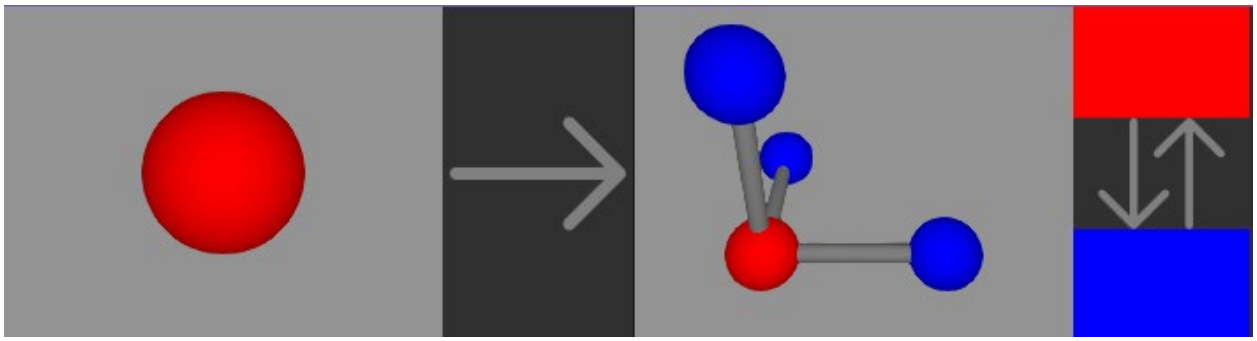


Рисунок 5.1 Відношення підстановки конструктора $C_{crystal}$ та операція над атрибутами кольору

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

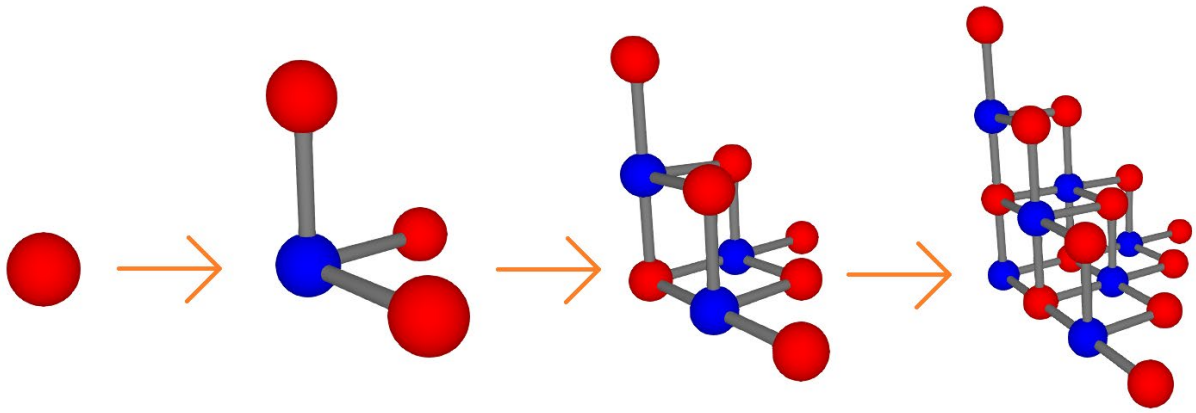


Рисунок 5.2 Послідовність конструювання кристалічних ґраток

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. 5.1), одноразово застосовуючи операцію підстановки, перетворює в f_1:
- виконуємо операцію над атрибутам кольору: взаємне перефарбування вершин з кольором left на right і навпаки.

Форма f_3 (рис 5.2) отримана операцією часткового виведення з триразовим застосуванням операції підстановки з подальшою операцією над атрибутами кольорів.

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

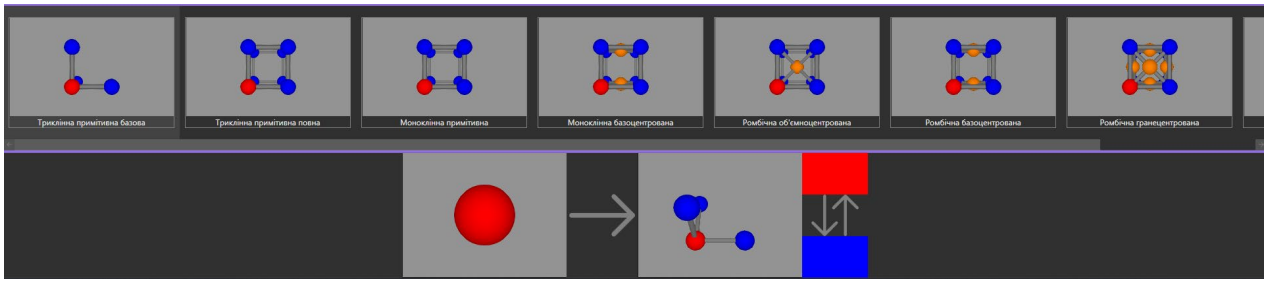


Рисунок 5.3 Правило підстановки та список можливих її модифікацій

Таким чином можливо поєднувати різні типи кристалічних ґраток, як наприклад, поєднання ромбічну об’ємноцентровану та ромбічну гранецентровану:

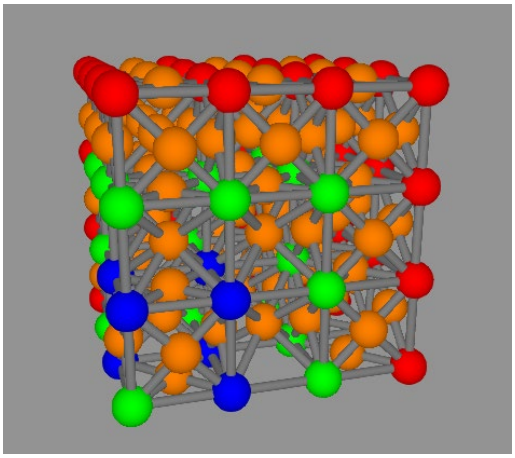


Рисунок 5.4 Поєднання різних кристалічних ґраток

До модифікації правила підстановки належать не лише типи кристалічних ґраток, а й їх параметри. Як приклад, візьмемо триклінну кристалічну ґратку та на кожному кроці будемо змінювати її атрибути довжин ребер та кутів між ребрами:

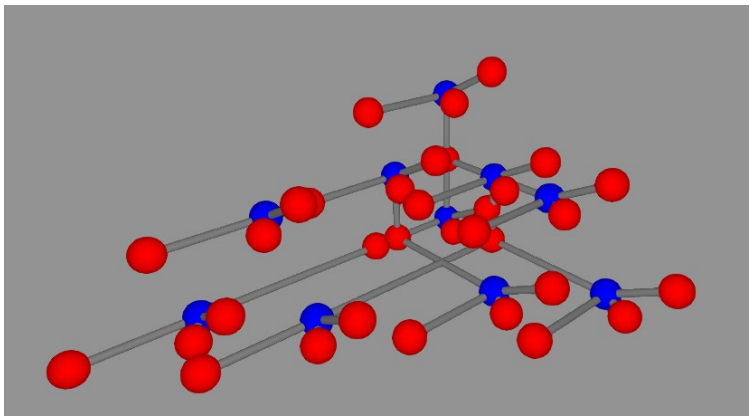


Рисунок 5.5 Кристалічна структура складена з різних триклінних кристалічних ґраток

6 ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

6.1 Тестування чорною скринькою

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

Метод чорної скриньки дозволяє перевірити фунціонал програми не значюи внутрішнього коду, проте щоб таке тестування було ефективним потрібно мати чітку документацію, тобто розуміти, що буде отримано на виході, які дані потрібні на вході і чи не впивають на роботу цих функції, що тестується, виклики інших функцій. Тестування чорною скринькою дозволяє перевірити очікувані результати проте не надає інформації про її причини. Коли мова йде про тестування програми з 3D графікою важко сказати, чи повністю функція відповідає своїм специфікаціям, чи немає прихованих помилок в роботі 3D движка. Так як використовується спеціальна бібліотека Helix Tookit[], що реалізовує функціонал 3D движку, можуть виникати збої та помилки, які розробники цієї бібліотеки не задокументували або не передбачили їх ймовірність.

Окрім явних помилок та збої, які можуть виникнути, можна також виявити небажані ефекти в результаті роботи програми, або приховані помилки, що мають накопичувальний ефект. Одним з таких недоліків є втрата швидкодії при збільшенні графічних об’єктів. Хоча це може бути очевидним, проте коли втрата швидкої відбувається дуже швидко це унеможливлює користування програми. Такий ефект виявився в одній із варіацій бібліотеки Helix, яка працює за допомогою DirectX 9, проте в іншій варіації Helix SharpDX [12], яка використовує DirectX 11 [23], такого стрімкого падіння швидкої не виявилось.

За допомогою методів чорної скриньки, а саме методів припущення про помилку, було перевірено дві основні функції програми: перефарбування та виконання підстановки одних графічних елементів на інші.

Специфікуємо сутності, які будуть використовуватись в методах:

- Point3D – структура даних, що містить три дійсних числа, які позначають глобальні координати на сцені;

• Color4b – структура, яка містить числове значення кольору, та містить чотири числові значення типу байт: R, G, B, A;

• Vertice – вершина, що має наступні властивості: Position (позиція) – тип Point3D, Color (колір) – тип Color4b, Size (розмір) – тип дійсне число;

• Edge – ребро, що складається з двох значень типу Point3D – початок і кінець;

• Graph – граф, що містить список вершин та список ребер.

• Rule – правило підстановки, складається з лівого кольору заміни LeftMainColor та правого кольору заміни RightMainColor – тип Color4b, а також з лівого графа Left і правого Right – типу Graph.

Для функції перефарбування вхідними даними є:

• колекція вершин, що представлена на 3D сцені, кожна з яких має колір, колір заміни;

• колір, який змінюється

• колір, на який потрібно замінити.

Вихідними даними є:

• Колекція вершин зі зміненим кольором;

• Колекція вершин зі станом, яка подалась на вході

Хоча функція здається простою, проте вона має важливе значення для роботи інших функцій. Можливі лише два наслідки для цієї функції: є заміна кольору або нічого не змінюється. Проте важливо знати, чи буде відбуватися заміна, якщо відтінок кольору трохи відрізняється від наявного кольору, що показують вершини на сцені чи ні?

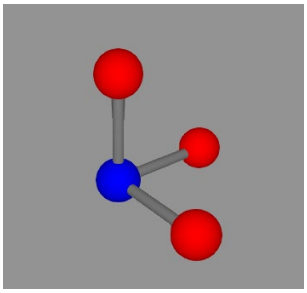


Рисунок 6.1 Вхідні дані для тестування заміни кольорів

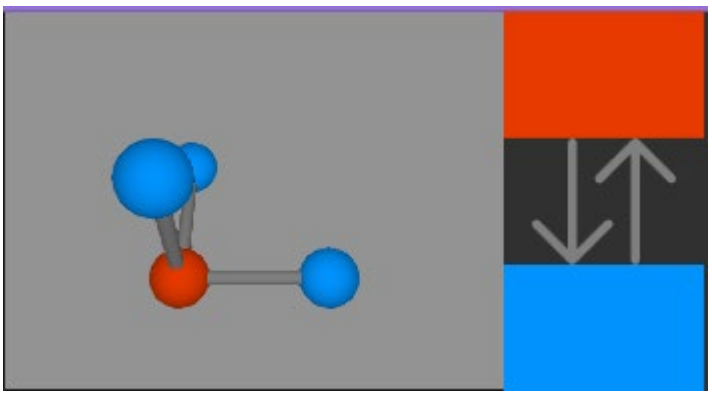


Рисунок 6.2 Правило заміни кольорів

Як і очікувалось заміни не відбулося, бо стосовно специфікації, колір заміни повинен повністю відповідати кольору фігури, а це можливе, якщо два кольори мають однакове число значення.

Для функції підстановки визначимо даними є:

- граф, що представлено на 3D сцені;
- правило підстановки, що включає в себе кольори взаємної підстановки, ліва частина правила, що позначає які об’єкти потрібно замінити, права частина правила, яка позначає на що саме потрібно замінити.

Вихідними даними є: змінений граф на 3D сцені

- Змінений граф на 3D сцені;
- Граф на 3D сцені як і був на вході.

Як і в випадку з функцією перефарбування наслідком може бути або виконання заміни, або її не виконання. Проте чи відбудеться заміна, якщо після підстановки виконати повторне перефарбування? Перефарбування є повторним, так як під час заміни завжди виконується перефарбування.

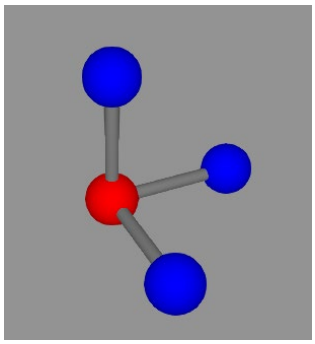


Рисунок 6.3 Стан головної 3D сцени для тестування підстановки після повторної операції заміни кольорів

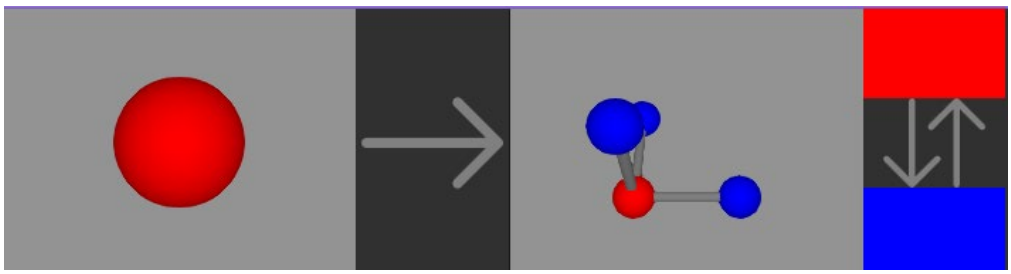


Рисунок 6.4 Правило підстановки для тестування підстановки після повторної операції заміни кольорів

Як результат підстановки не виконується. Це і не дивно, так форма, що представлена на сцені вже знаходиться в стані, що відображає права частина правила, тому для правила підстановки завжди виконується операція з перефарбування.

6.2 Налагодження

Кількість помилок та похибок під час розробки програми була мінімальною, так як попередньо було побудована демоверсія та враховані усі недоліки архітектури, та алгоритмів. Однак декілька суттєвих помилок все ж виявилося, але їх вирішення не потребувало значного часу. Для їх усунення було використано режим відладки в Visual Studio 2022.

Результат налагодження програми представлені в наступній таблиці:

Таблиця 6.1 Протокол налагодження програми

Опис помилки	Опис ситуації	Дії, що були застосовані для усунення
Неправильне виконання операції підстановки	Після операції з очищення сцени та виконання операції підстановки заміни відбувалися на місці видалених об’єктів	Очищення відбувалося на 3D сцені, проте не на логічному рівні, де виконуються алгоритми заміни та очищення
Вибір кольору ігнорувався	Вибір кольору для правила підстановки не застосувався після його вибору	Помилка пов’язана з конвертацією даних між бібліотечним типом Color4b та кольором вибору з діалогу типу Color.

Продовження таблиці 6.1

		Помилка в коді бібліотеки, яка не передбачає повторну конвертацію кольору. Було написано власний метод конвертації між типами.
Неправильний експорт моделі в формат файлу *.stl.	Бібліотечна версія Helix з конвертації моделі до файлу конвертує лише один з групової моделі.	Написання власного методу для експорту файлу з урахуванням внутрішньої будови бібліотеки Heli.x
Правило підстановки не відбувалось.	Заміна кольору яке буде перефарбовуватись замінювалось лише в лівій частині правила, проте в правій воно не змінювало колір, через це не виконувалась умова з пошуку вершини з лівим кольором у правій частині правила.	Оновлення та перефарбування правої частини правила під час заміні лівого кольору.

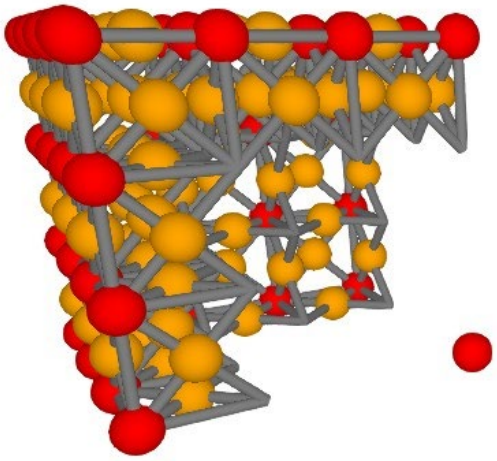


Рисунок 6.5 Хибна побудова кристалічної ґратки після очищення сцени

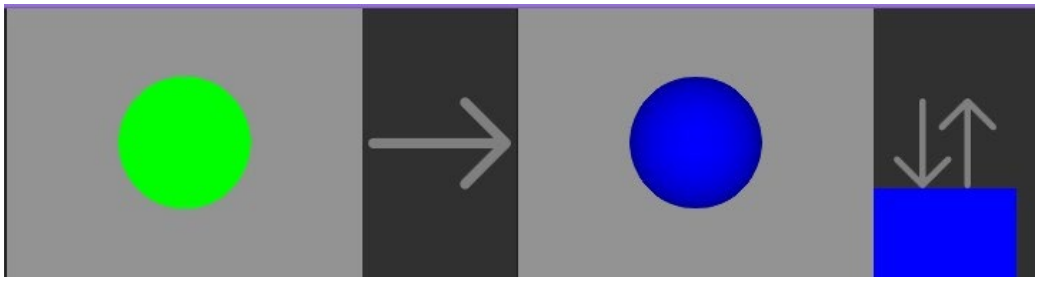


Рисунок 6.6 Ігнорування вибору кольору

Завдяки досвіду з налагодження програм, методам дедукції та індукції можливо швидко вирішити помилки, навіть, якщо вони спричинені сторонніми розробниками, які не передбачили усіх можливих наслідків використання їх коду. Також запобіганню помилок сприяє наявність специфікації методів та інформації про призначення класів.

Висновки

Результатом роботи є програма з моделювання кристалічних ґраток, яка може застосовуватись в освітніх цілях в галузі неорганічної хімії та кристалографії. Особливість програми полягає в методах формування кристалічної структури, її візуалізація, можливості редагування а також можливості комбінування різних типів кристалічних ґраток.

Головна частина успіху в побудові ПЗ залежить в розстановці пріоритетів на більш ранні етапи розробки, такі як огляд предметної галузі, складання експлуатаційних і функціональних специфікацій, проектування архітектуру ПЗ. Таким чином на етапі огляду аналогів було сформоване остаточне бачення програми, сформовані її вимоги, що відобразилось в use-case діаграмі.

На етапі внутрішнього проектування потрібно обрати оптимальні інструменти та технології, за допомогою яких можна написати ПЗ, що задовольняє вимогам, проте при цьому потрібно враховувати власні навички та досвід, що у свою чергу безпосередньо впливає на швидкість та якість розробки.

КПС дозволяє будувати гнучкі алгоритмічні системи, які підлягають легкій модифікації та спеціалізації, що дає змогу побудувати більш складну систему, використовуючи створену, для подальшого вивчення кристалічних структур.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unit Cell. [Virtual Resource] Access Mode: URL: https://chem.libretexts.org/Bookshelves/Introductory_Chemistry/Introductory_Chemistry_(CK-12)/13%3A_States_of_Matter/13.14%3A_Unit_Cells#:~:text=A%20unit%20cell%20is%20the,Figure%2013.14. Date of Access: 14 May 2023 p.
2. Bravias Lattice [Virtual Resource] Access Mode: URL: https://byjus.com/chemistry/bravais-lattice/#14-Types-of-Bravais-Lattices. Date of Access: 15 May 2023 p.
3. Luealamai, S., & Panijpan, B. (2012). Learning About the Unit Cell and Crystal Lattice With Computerized Simulations and Games: A Pilot Study. Simulation & Gaming, 43(1), 67–84. https://doi.org/10.1177/1046878110378704
4. Interactive Unit Cell Visualization Tool for Crystal Lattice Structures. Corbin Gruber, Alec James, Jeremy T. Berchtold, Zoe J. Wood, Gregory E. Scott, and Zahra Alghoul Journal of Chemical Education 2020 97 (7), 2020-2024 DOI: 10.1021/acs.jchemed.9b01207
5. Chen, P., Wang, Y., Yan, H. et al. 3DStructGen: an interactive web-based 3D structure generation for non-periodic molecule and crystal. J Cheminform 12, 7 (2020). https://doi.org/10.1186/s13321-020-0411-2
6. Supercell (crystal). [Virtual Resource] Access Mode: URL: https://en.wikipedia.org/wiki/Supercell_(crystal). Date of Access: 16 May 2023 p.
7. Crystal & Molecular Structures: Modelling and Diffraction [Virtual Resource] Access Mode: URL : https://crystalmaker.com. Date of Access: 16 May 2023 p.
8. VESTA: Visualization for Electronic and STructual Analysis [Virtual Resource] Access Mode: URL https://jp-minerals.org/vesta/en/. Date of Access: 17 May 2023 p.
9. Avogadro. [Virtual Resource] Режим доступу: URL https://avogadro.cc/. Date of Access: 17 May 2023 p.
10. Biovia Materials Studio visualizer datasheet. [Virtual Resource] Access Mode: URL https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/BIOVIA/PDF/materials-studio-visualizer.pdf. Date of Access: 18 May 2023 p.
11. 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: 5 June 2023 p.
12. Helix Toolkit. [Virtual Resource] Access Mode: URL: https://github.com/helix-toolkit/helix-toolkit. Date of Access: 5 June 2023 p.
13. Qt Framework. [Virtual Resource] Access Mode: URL: https://www.qt.io/product/framework. Date of Access: 1 June 2023 p.

14. Vulkan API. [Virtual Resource] Access Mode: URL: https://www.vulkan.org/. Date of Access: 4 червня 2023 р.
15. Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming [TEXT] WPF Notifications, Validations, Commands, and MVVM / Phillip Japikse, Andrew Troelsen, 2021. – 1143 – 1177 p.
16. GRASP Design Principles By Danya Rao. [Virtual Resource] Access Mode: URL: https://home.cs.colorado.edu/~kena/classes/5448/f12/presentation-materials/rao.pdf Date of Access: 4 June 2023.
17. A Solid Guide to SOLID Principles. [Virtual Resource] Access Mode: URL: https://www.baeldung.com/solid-principles. Date of Access: 2 June 2023.
18. What is Inversion of Control. [Virtual Resource] Access Mode: URL: https://www.educative.io/answers/what-is-inversion-of-control. Date of Access: 2 June 2023.
19. The DTO Pattern (Data Transfer Object) [Virtual Resource] Access Mode: URL: https://www.baeldung.com/java-dto-pattern. Date of Access: 2 June 2023.
20. Class, Responsibility, and Collaboration. [Virtual Resource] Access Mode: URL: https://wwwcgi.teach.cs.toronto.edu/~csc207h/winter/lectures/L0201/w5/CRC.pdf. Date of Access: 3 June 2023
21. Material Design In XAML Toolkit. [Virtual Resource] Access Mode: URL: http://materialdesigninxaml.net/ Date of Access: 16 May 2023.
22. V. I. Shynkarenko, V. M. Ilman. Constructive-Synthesizing Structures and Their Grammatical Interpretations. I. Generalized Formal Constructive-Synthesizing Structure / Cybernetics and Systems Analysis, Volume 50, Issue 5, pp 655-662
23. V. I. Shynkarenko, V. M. Ilman. Constructive-Synthesizing Structures and Their Grammatical Interpretations. II. Refining Transformations / Cybernetics and Systems Analysis, Volume 50, Issue 6, pp 829-841.
24. Direct3D 11 Reference. [Virtual Resource] Access Mode: URL: https://learn.microsoft.com/en-us/windows/win32/direct3d11/d3d11-graphics-reference. Date of Access: 3 June 2023

Код робочого проекту

Додаток А

Текст програми

Файл MainWindow.xaml

```
<Window x:Class="Crystal.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Crystal"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
        TextElement.Foreground="{StaticResource MaterialDesignBody}"
        Background="{StaticResource MaterialDesignPaper}"
        mc:Ignorable="d"
        Title="Crystal" Height="450" Width="800">
    <DockPanel>
            <Menu DockPanel.Dock="Top" materialDesign:MenuAssist.TopLevelMenuItemHeight="25">
                <Menu.Background>
                    <SolidColorBrush Opacity="0.5" Color="WhiteSmoke" />
                </Menu.Background>
                <MenuItem Header="Файл">
                <MenuItem Command="{Binding ExportToStlCommand}" Header="Експорт до Stl" />
                <MenuItem Command="{Binding ExportToXmlCommand}" Header="Експорт до xml" />
                <MenuItem Command="{Binding ImportFromXmlCommand}" Header="Імпорт з xml" />
            </MenuItem>
            </Menu>
        <ContentPresenter Content="{Binding MainView}"/>
    </DockPanel>
</Window>
```

Файл MainWindow.xaml.cs

```
using Crystal.Controllers;
using Crystal.ViewModels;
using Crystal.Views;
using System.Windows;

namespace Crystal
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window, IUIView
    {
        MainController _mainController;
        public MainWindow()
        {
            _mainController = new MainController(this);
            InitializeComponent();
        }

        public BaseViewModel ViewModel
        {
            get => DataContext as BaseViewModel;
            set
            {
                if (DataContext == value)
                    return;

                DataContext = value;
            }
        }
    }
}
```

Файл App.xaml

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

namespace Crystal
{
   public static class Defaults
   {
      public static readonly Color4 LeftColor = Color.Red;
      public static readonly Color4 RightColor = Color.Blue;
      public static readonly Color4 EdgeColor = Color.Gray;
      public static readonly Color4 NeutralColor = Media.Color.FromArgb(255, 255, 128, 0).ToColor4(); // orange

      public static readonly string SubstitutionCommandId = "Substituion";
      public static readonly string RepaintCommandId = "Repaint";
      public static readonly string ClearPrimaryViewportCommandId = "ClearPrimaryViewport";
      public static readonly string ExportToXmlCommandId = "ExportToXml";
      public static readonly string ImportFromXmlCommandId = "ImportFromXm";

      public const double DefaultVerticeSize = 0.2;
      public const double DefaultEdgeSize = 0.1;
   }
}
```

Файл Defaults.cs

```
<Application x:Class="Crystal.App"
          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:local="clr-namespace:Crystal"
          xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
          StartupUri="MainWindow.xaml">
   <Application.Resources>
      <ResourceDictionary>
         <ResourceDictionary.MergedDictionaries>
            <materialDesign:CustomColorTheme BaseTheme="Dark" PrimaryColor="MediumPurple" SecondaryColor="LightGreen" />
            <ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesignTheme.Defaults.xaml" />
         </ResourceDictionary.MergedDictionaries>
      </ResourceDictionary>
   </Application.Resources>
</Application>
```

Файл IUIView.cs

```
using Crystal.ViewModels;

namespace Crystal.Views
{
   public interface IUIView
   {
      BaseViewModel ViewModel { get; set; }
   }
}
```

Файл MainController.cs

```
using System.ComponentModel.Design;
using Crystal.Extensions;
using Crystal.Services;
using Crystal.ViewModels;
using Crystal.Views;


namespace Crystal.Controllers
{
   public class MainController
   {
      IUIView _mainView;
      MainWindowViewModel _mainWindowViewModel;

      IServiceContainer _serviceContainer;
      IUIService _uiService;
      ICrystalService _crystalService;
      IRuleController _ruleController;
      IStateManager _stateManager;
      ICommandManager _commandManager;
```

```
        public MainController(IUIView mainView)
        {
            ConfigureServices();
            _mainView = mainView;
            _mainWindowViewModel = new MainWindowViewModel(_serviceContainer);
            _mainView.ViewModel = _mainWindowViewModel;
        }

        void ConfigureServices()
        {
            _serviceContainer = new ServiceContainer();

            _uiService = new UIService(_serviceContainer);
            _serviceContainer.AddService<IUIService>(_uiService);

            _stateManager = new StateManager();
            _serviceContainer.AddService<IStateManager>(_stateManager);
            _stateManager.PrimaryViewportController = _uiService.CreateViewportController();
            _stateManager.PrimaryViewportController.ShowViewCube = true;
            _stateManager.PrimaryViewportController.ShowAxis = true;
            _stateManager.PrimaryViewportController.ShowInfoPanel = true;

            _ruleController = new RuleController(_serviceContainer);

            _commandManager = new CommandManager();
            _serviceContainer.AddService<ICommandManager>(_commandManager);

            _crystalService = new CrystalService(_serviceContainer);
            _serviceContainer.AddService<ICrystalService>(_crystalService);
        }
    }
}
```

Файл CommandManager.cs

```
using System;

namespace Crystal.Controllers
{

    public class CommandEventArgs : EventArgs
    {
        public object Value { get; }
        public string CommandId { get; }

        public CommandEventArgs(object value, string commandId)
        {
            Value = value;
            CommandId = commandId;
        }
    }

    interface ICommandManager
    {
        void OnCommandCall(object sender, CommandEventArgs args);
        event EventHandler<CommandEventArgs> CommandCalled;
    }

    public class CommandManager : ICommandManager
    {
        public void OnCommandCall(object sender, CommandEventArgs args)
        {
            CommandCalled?.Invoke(sender, args);
        }

        public event EventHandler<CommandEventArgs> CommandCalled;
    }
}
```

Файл StateManager.cs

```
using Crystal.Services.CrystalServiceComponents.Abstraction;
using Crystal.Services.CrystalServiceComponents.LatticeContructors;
using Crystal.Views;
using System;
using System.Collections.Generic;
using System.Runtime.CompilerServices;
```

```
namespace Crystal.Controllers
{
    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 ParameterA { get; set; }
        float ParameterB { get; set; }
        float ParameterC { get; set; }
        float ParameterAlpha { get; set; }
        float ParameterBeta { get; set; }
        float ParameterGamma { get; set; }

        bool InterEditParameterA { get; }
        bool InterEditParameterB { get; }
        bool InterEditParameterC { get; }
        bool InterEditParameterAlpha { get; }
        bool InterEditParameterBeta { get;  }
        bool InterEditParameterGamma { get; }

        int SubstitionsCount { get; set; }
        int VerticesCount { get; set; }
        int EdgesCount { get; set; }


        IConstructor Constructor { get; set; }
        LatticeConstructor LatticeConstructor { get; set; }
        IRule CurrentRule { get; set; }
        IViewportController PrimaryViewportController { get; set; }
        IUIView RulesListView { get; set; }
        IUIView RuleView { get; set; }
        event EventHandler<StateEventArgs> StateChanged;
    }

    public class StateManager : IStateManager
    {
        LatticeConstructor _latticeConstructor;
        IConstructor _constructor;
        IRule _currentRule;
        IViewportController _primaryViewportController;
        IUIView _rulesListView;
        IUIView _ruleView;

        int _substitionsCount;
        int _verticesCount;
        int _edgesCount;

        bool _enableNotifications = true;
        public event EventHandler<StateEventArgs> StateChanged;

        public float ParameterA
        {
            get => _latticeConstructor == null ? 0 : _latticeConstructor.ParameterA;
            set
            {
                if (_latticeConstructor == null || _latticeConstructor.ParameterA == value)
                    return;

                if (!InterEditParameterB )
                {
                    _latticeConstructor.ParameterB =  value;
                }
                if (!InterEditParameterC)
                {
                    _latticeConstructor.ParameterC =  value;
                }
```

```
            _latticeConstructor.ParameterA = value;
            OnStateChagned(_latticeConstructor.ParameterA);
        }
    }

    public float ParameterB
    {
        get => _latticeConstructor == null ? 0 : _latticeConstructor.ParameterB;
        set
        {
            if (_latticeConstructor == null || _latticeConstructor.ParameterB == value)
                return;

            _latticeConstructor.ParameterB = value;
            OnStateChagned(_latticeConstructor.ParameterB);
        }
    }

    public float ParameterC
    {
        get => _latticeConstructor == null ? 0 : _latticeConstructor.ParameterC;
        set
        {
            if (_latticeConstructor == null || _latticeConstructor.ParameterC == value)
                return;

            _latticeConstructor.ParameterC = value;
            OnStateChagned(_latticeConstructor.ParameterC);
        }
    }

    public float ParameterAlpha
    {
        get => _latticeConstructor == null ? 0 : _latticeConstructor.ParameterAlpha;
        set
        {
            if (_latticeConstructor == null || _latticeConstructor.ParameterAlpha == value)
                return;

            _latticeConstructor.ParameterAlpha = value;

            if (!InterEditParameterBeta && !InterEditParameterC)
            {
                _latticeConstructor.ParameterBeta = value;
                _latticeConstructor.ParameterGamma = value;
            }

            OnStateChagned(_latticeConstructor.ParameterAlpha);
        }
    }

    public float ParameterBeta
    {
        get => _latticeConstructor == null ? 0 : _latticeConstructor.ParameterBeta;
        set
        {
            if (_latticeConstructor == null || _latticeConstructor.ParameterBeta == value)
                return;

            _latticeConstructor.ParameterBeta = value;
            OnStateChagned(_latticeConstructor.ParameterBeta);
        }
    }

    public float ParameterGamma
    {
        get => _latticeConstructor == null ? 0 : _latticeConstructor.ParameterGamma;
        set
        {
            if (_latticeConstructor == null || _latticeConstructor.ParameterGamma == value)
                return;

            _latticeConstructor.ParameterGamma = value;
            OnStateChagned(_latticeConstructor.ParameterGamma);
        }
    }

    public bool InterEditParameterA
```

```
{
    get => _latticeConstructor == null ? false : _latticeConstructor.InterEditParameterA;
}

public bool InterEditParameterB
{
    get => _latticeConstructor == null ? false : _latticeConstructor.InterEditParameterB;
}

public bool InterEditParameterC
{
    get => _latticeConstructor == null ? false : _latticeConstructor.InterEditParameterC;
}

public bool InterEditParameterAlpha
{
    get => _latticeConstructor == null ? false : _latticeConstructor.InterEditParameterAlpha;
}

public bool InterEditParameterBeta
{
    get => _latticeConstructor == null ? false : _latticeConstructor.InterEditParameterBeta;
}

public bool InterEditParameterGamma
{
    get => _latticeConstructor == null ? false : _latticeConstructor.InterEditParameterGamma;
}

public IConstructor Constructor
{
    get => _constructor;
    set => Set(ref _constructor, value);
}

public LatticeConstructor LatticeConstructor
{
    get => _latticeConstructor;
    set
    {
        if(Set(ref _latticeConstructor, value))
        {
            OnStateChagned(nameof(ParameterA));
            OnStateChagned(nameof(ParameterB));
            OnStateChagned(nameof(ParameterC));
            OnStateChagned(nameof(ParameterAlpha));
            OnStateChagned(nameof(ParameterBeta));
            OnStateChagned(nameof(ParameterGamma));
            OnStateChagned(nameof(InterEditParameterA));
            OnStateChagned(nameof(InterEditParameterB));
            OnStateChagned(nameof(InterEditParameterC));
            OnStateChagned(nameof(InterEditParameterAlpha));
            OnStateChagned(nameof(InterEditParameterBeta));
            OnStateChagned(nameof(InterEditParameterGamma));
        }
    }
}

public int SubstitionsCount
{
    get => _substitionsCount;
    set => Set(ref _substitionsCount, value);
}

public int VerticesCount
{
    get => _verticesCount;
    set => Set(ref _verticesCount, value);
}

public int EdgesCount
{
    get => _edgesCount;
    set => Set(ref _edgesCount, value);
}

public IRule CurrentRule
{
    get => _currentRule;
```

```
            set => Set(ref _currentRule, value);
        }

        public IViewportController 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);
        }

        bool Set<T>(ref T backingField, T value, [CallerMemberName] string propertyName = "")
        {
            if (EqualityComparer<T>.Default.Equals(backingField, value))
            {
                return false;
            }

            backingField = value;

            if (_enableNotifications)
                StateChanged?.Invoke(this, new StateEventArgs(value, propertyName));

            return true;
        }

        void OnStateChagned(object value, [CallerMemberName] string propertyName = "")
        {
            if (_enableNotifications)
                StateChanged?.Invoke(this, new StateEventArgs(value, propertyName));
        }

        void OnStateChagned(string propertyName)
        {
            if (_enableNotifications)
                StateChanged?.Invoke(this, new StateEventArgs(null, propertyName));
        }
    }
}
```

Файл RuleController.cs

```
using Crystal.Extensions;
using Crystal.Services;
using Crystal.ViewModels;
using Crystal.Views;
using Crystal.Services.CrystalServiceComponents.Abstraction;
using Crystal.Services.CrystalServiceComponents.LatticeContructors;
using SharpDX;
using System;
using Crystall.Services.CrystalServiceComponents.LatticeContructors;

namespace Crystal.Controllers
{
    public interface IRuleController
    {
        IUIView Control { get; }
        IUIView ListControl { get; }
    }

    public class RuleController : IRuleController
    {
        IServiceProvider _services;
        IUIService _uiService;
        IStateManager _stateManager;

        IViewportController _leftViewport;
        IViewportController _rightViewport;
```

```
IUIView _ruleView;
IUIView _rulesListView;
RuleViewModel _ruleViewModel;
RulesListViewModel _rulesListViewModel;
LatticeConstructor _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.CreateViewportController();
   _rightViewport = _uiService.CreateViewportController();

   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.Vertices);
   _leftViewport.CameraPos = new Vector3(0, -1, 0);
   _leftViewport.CameraLook = new Vector3(0, 1, 0);

   _rightViewport.AddVisual(_rule.Right.Vertices);
   _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 _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 _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 bool IsParameter(string name)
    {
        return name == nameof(_constructor.ParameterA)
            || name == nameof(_constructor.ParameterB)
            || name == nameof(_constructor.ParameterC)
            || name == nameof(_constructor.ParameterAlpha)
            || name == nameof(_constructor.ParameterBeta)
            || name == nameof(_constructor.ParameterGamma);
    }

    private void DefineRulesList()
    {
        LatticeConstructor constructor = new TriclinicLatticeSimpleConstructor();
        constructor.CreateCell();
        _rulesListViewModel.AddRule(constructor, "Триклінна примітивна базова");

        constructor = new TriclinicLatticeFullConstructor();
        constructor.CreateCell();
        _rulesListViewModel.AddRule(constructor, "Триклінна примітивна повна");

        //моноклінна
        constructor = new TriclinicLatticeFullConstructor()
        {
            InterEditParameterAlpha = false,
            InterEditParameterGamma = false
        };
        constructor.CreateCell();
        _rulesListViewModel.AddRule(constructor, "Моноклінна примітивна");

        constructor = new BaseCenteredLatticeConstructor()
        {
            InterEditParameterAlpha = false,
            InterEditParameterGamma = false
        };
        constructor.CreateCell();
        _rulesListViewModel.AddRule(constructor, "Моноклінна базоцентрована");

        //ромбічна
        constructor = new BodyCenteredLatticeConstructor()
        {
            InterEditParameterAlpha = false,
            InterEditParameterBeta = false,
            InterEditParameterGamma = false
        };
        constructor.CreateCell();
        _rulesListViewModel.AddRule(constructor, "Ромбічна об'ємноцентрована");

        constructor = new BaseCenteredLatticeConstructor()
        {
            InterEditParameterAlpha = false,
            InterEditParameterBeta = false,
            InterEditParameterGamma = false
        };
        constructor.CreateCell();
        _rulesListViewModel.AddRule(constructor, "Ромбічна базоцентрована");

        constructor = new FaceCenteredLatticeConstructor()
        {
            InterEditParameterAlpha = false,
            InterEditParameterBeta = false,
            InterEditParameterGamma = false
        };
        constructor.CreateCell();
        _rulesListViewModel.AddRule(constructor, "Ромбічна гранецентрована");

        //хексогональна
        constructor = new HexagonalConstructor()
        {
            InterEditParameterGamma = false,
            InterEditParameterB = false
        };
        constructor.CreateCell();
        _rulesListViewModel.AddRule(constructor, "Хексагональна");
    }
```

```
        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.LatticeConstructor = _constructor;
            _ruleViewModel.NotifyColorChanged();
        }

        void ConfigureDefaultRule()
        {
            Color4 leftColor = Defaults.LeftColor;
            IGraph left = Graph.SingleVerticeGraph(leftColor);

            Color4 rightColor = Defaults.RightColor;
            IGraph right = Graph.SingleVerticeGraph(rightColor);

            if (Rule.TryCreateRule(left, right, leftColor, rightColor, out var rule) == Extensions.RuleErrorCode.Succeed)
            {
                _rule = rule;
            }

            _ruleViewModel = new RuleViewModel(_rule)
            {
                LeftViewport = _leftViewport.Control,
                RightViewport = _rightViewport.Control
            };

            _leftViewport.Graph = left;
            _rightViewport.Graph = right;
        }
    }
}
```

Файл CrystalService.cs

```
using Crystal.Controllers;
using Crystal.Extensions;
using Crystal.Services.CrystalServiceComponents.Abstraction;
using Crystall.Services.CrystalServiceComponents.Abstraction;
using System;
using System.Collections.Generic;
using System.IO;
using System.Xml.Serialization;

namespace Crystal.Services
{
    public interface ICrystalService
    {
        void ExportToXml(string fileName);
        void ImportFromXml(string fileName);
    }

    public class CrystalService : ICrystalService
    {
        IServiceProvider _services;
        IStateManager _stateManager;
        ICommandManager _commandManager;
        IConstructor _constructor;
        IViewportController _viewportController;

        public CrystalService(IServiceProvider services)
        {
            _services = services;
            _stateManager = _services.GetService<IStateManager>();
            _commandManager = _services.GetService<ICommandManager>();

            var graph = Graph.SingleVerticeGraph(Defaults.LeftColor);
            _constructor = new Constructor(graph);
            _constructor.CurrentRule = _stateManager.CurrentRule;
            _stateManager.Constructor = _constructor;
            _stateManager.StateChanged += _stateManager_StateChanged;
```

```
    _viewportController = _stateManager.PrimaryViewportController;
    _commandManager.CommandCalled += _commandManager_CommandCalled;
    _viewportController.AddVisual(graph.Vertices);
    _constructor.Repainted += _constructor_Repainted;

}

public void ExportToXml(string fileName)
{
    XmlSerializer x = new XmlSerializer(typeof(GraphDto));
    using (FileStream fs = File.Create(fileName))
    {
        x.Serialize(fs, _constructor.Graph.ToDto());
    }
}

public void ImportFromXml(string fileName)
{
    XmlSerializer x = new XmlSerializer(typeof(GraphDto));
    using (FileStream fs = File.Open(fileName, FileMode.Open))
    {
        var graph = (GraphDto)x.Deserialize(fs);
        _constructor.Clear();
        _constructor.Graph = graph.FromDto();
        _viewportController.Graph = _constructor.Graph;
        _viewportController.InvalidateGraphVisual();
    }
}


private void _stateManager_StateChanged(object sender, StateEventArgs e)
{
    if (e.PropertyName == nameof(_stateManager.CurrentRule))
        _constructor.CurrentRule = _stateManager.CurrentRule;
}

private void _commandManager_CommandCalled(object sender, CommandEventArgs e)
{
    if (e.CommandId == Defaults.SubstitutionCommandId)
    {
        _constructor.PerformeSubstitution();
        _viewportController.AddVisual(_constructor.AddedVertices);
        _viewportController.AddVisual(_constructor.AddedEdges);
        _stateManager.SubstitionsCount++;
        UpdateStatistics();
    }
    else if(e.CommandId == Defaults.RepaintCommandId)
    {
        _constructor.Repaint();

    }
    else if (e.CommandId == Defaults.ClearPrimaryViewportCommandId)
    {
        _constructor.Invalidate();
        _viewportController.Graph = _constructor.Graph;
        _viewportController.InvalidateGraphVisual();
        _stateManager.SubstitionsCount = 0;
        UpdateStatistics();
    }
    else if(e.CommandId == Defaults.ExportToXmlCommandId)
    {
        ExportToXml((string)e.Value);
    }
    else if (e.CommandId == Defaults.ImportFromXmlCommandId)
    {
        ImportFromXml((string)e.Value);
        _stateManager.SubstitionsCount = 0;
        UpdateStatistics();
    }
}

private void _constructor_Repainted(object sender, IReadOnlyCollection<Vertice> e)
{
    _viewportController.RepaintVisual(e);
}

private void UpdateStatistics()
{
```

```
            _stateManager.VerticesCount = _constructor.Graph.Vertices.Count;
            _stateManager.EdgesCount = _constructor.Graph.Edges.Count;
        }
    }
}
```

Файл UIService.cs

```
using System;
using System.Collections.Generic;
using Crystal.ViewModels;
using Crystal.Views;
using Crystal.Controllers;

namespace Crystal.Services
{
    public interface IUIService
    {
        IUIView CreateView<T>(T viewModel);
        IViewportController CreateViewportController();
    }

    public class UIService : IUIService
    {
        IServiceProvider _services;

        Dictionary<Type, Func<object, IUIView>> _viewsFactory
            = new Dictionary<Type, Func<object, IUIView>>();

        public UIService(IServiceProvider services)
        {
            _services = services;
            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(SideMenuViewModel), viewModel => new SideMenuView() {ViewModel = (SideMenuViewModel)viewModel });
            _viewsFactory.Add(typeof(RulesListViewModel), viewModel => new RulesListView() {ViewModel = (RulesListViewModel)viewModel });
            _viewsFactory.Add(typeof(RuleViewModel), viewModel => new RuleView() {ViewModel = (RuleViewModel)viewModel });
        }

        public IUIView CreateView<T>(T viewModel)
        {
            if(_viewsFactory.TryGetValue(typeof(T), out var createView))
            {
                return createView(viewModel);
            }
            return null;
        }

        public IViewportController CreateViewportController()
        {
            return new ViewportController(_services);
        }
    }
}
```

Файл ViewportController.cs

```
using Crystal.Extensions;
using Crystal.Services;
using Crystal.Utils;
using Crystal.ViewModels;
using Crystal.Views;
using Crystal.Services.CrystalServiceComponents.Abstraction;
using HelixToolkit.Wpf.SharpDX;
using HelixToolkit.Wpf.SharpDX.Model.Scene;
using SharpDX;
using System;
using System.Collections.Generic;
using System.IO;

namespace Crystal.Controllers
{
    public interface IViewportController
    {
```

```
    IUIView Control { get; }
    bool ShowAxis { get; set; }
    bool ShowViewCube { get; set; }
    bool ShowInfoPanel { get; set; }

    void InvalidateGraphVisual();
    void AddVisual(IEnumerable<Vertice> vertices);
    void AddVisual(IEnumerable<Edge> edges);
    void RepaintVisual(IEnumerable<Vertice> vertices);

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

    IGraph Graph { get; set; }
    void ExportToStl(string fileName);

    event EventHandler ViewportClicked;
}


public class ViewportController : IViewportController
{
    IServiceProvider _services;
    IUIService _uiService;

    IUIView _viewportView;
    ViewportViewModel _viewportViewModel;

    IGraph _graph;
    Dictionary<Vertice, MeshNode> _verticeVisuals = new Dictionary<Vertice, MeshNode>();
    Color4 _defaultEdgeColor = Defaults.EdgeColor;

    public IUIView Control => _viewportView;

    public ViewportController(IServiceProvider services)
    {
        _services = services;
        _uiService = _services.GetService<IUIService>();
        _graph = new Graph();

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

    private GroupNode RootNode => _viewportViewModel.RootNode;

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

    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;
}

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

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

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

   foreach(var vertice in vertices)
   {
      if (!geometries.TryGetValue(vertice.Size, out var geom))
      {
         var meshBuilder = new MeshBuilder();
         meshBuilder.AddSphere(Vector3.Zero, vertice.Size);
         geom = meshBuilder.ToMeshGeometry3D();
         geom.IsDynamic = true;
         geometries.Add(vertice.Size, geom);
      }

      var meshNode = new MeshNode() { Geometry = geom };
      meshNode.ModelMatrix = Matrix.Translation(vertice.Position);
      meshNode.Material = new DiffuseMaterial() { DiffuseColor = vertice.Color };
      RootNode.AddChildNode(meshNode);
      _verticeVisuals[vertice] = meshNode;


   }
}

public void AddVisual(IEnumerable<Edge> 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 meshNode = new MeshNode() { Geometry = geom };
      meshNode.Material = new DiffuseMaterial() { DiffuseColor = _defaultEdgeColor };
      RootNode.AddChildNode(meshNode);
```

```
            }
        }

        public void RepaintVisual(IEnumerable<Vertice> vertices)
        {
            foreach(var vertice in vertices)
            {
                if (_verticeVisuals.TryGetValue(vertice, out var meshNode))
                    meshNode.Material = new DiffuseMaterial() { DiffuseColor = vertice.Color };
            }
        }


        public void ExportToStl(string filePath)
        {
            var stlExporter = new StlExporter();
            using (FileStream fs = File.Create(filePath))
            {
                stlExporter.Export(RootNode, fs);
            }
        }
    }
}
```

Файл Constructor.cs

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

namespace Crystal.Services.CrystalServiceComponents.Abstraction
{
    public interface IConstructor
    {
        IGraph Graph { get; set; }
        IRule CurrentRule { get; set; }

        IReadOnlyCollection<Edge> AddedEdges { get; }
        IReadOnlyCollection<Vertice> AddedVertices { get; }

        void Invalidate();
        void PerformeSubstitution();
        void Clear();
        void Repaint();

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

    public class SubstitutionEventArgs : EventArgs
    {
        IReadOnlyCollection<Edge> AddedEdges { get; }
        IReadOnlyCollection<Vertice> AddedVertices { get; }

        public SubstitutionEventArgs(IReadOnlyCollection<Edge> addedEdges, IReadOnlyCollection<Vertice> addedVertices)
        {
            AddedEdges = addedEdges;
            AddedVertices = addedVertices;
        }
    }

    public class Constructor : IConstructor
    {
        List<Edge> _addedEdges = new List<Edge>();
        List<Vertice> _addedVertices = new List<Vertice>();
        List<Vertice> _repaintedVetrices = new List<Vertice>();
        IGraph _graph;

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

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

        public IRule CurrentRule { get; set; }

        public IReadOnlyCollection<Edge> AddedEdges => _addedEdges.AsReadOnly();
```

```
public IReadOnlyCollection<Vertice> AddedVertices => _addedVertices.AsReadOnly();

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

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

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

public void Repaint()
{
    Repaint(Graph.Vertices);
}

private void Repaint(IEnumerable<Vertice> vertices)
{
    foreach (var vertice in vertices)
    {
        if (vertice.Color == CurrentRule.MainLeftColor)
        {
            vertice.Color = CurrentRule.MainRightColor;
            _repaintedVetrices.Add(vertice);
        }
        else if (vertice.Color == CurrentRule.MainRightColor)
        {
            vertice.Color = CurrentRule.MainLeftColor;
            _repaintedVetrices.Add(vertice);
        }
    }
    Repainted(this, _repaintedVetrices);
}

public void PerformeSubstitution()
{
    var verticesLeftColor = Graph.Vertices.Where(vertice => vertice.Color == CurrentRule.MainLeftColor).ToArray();

    _addedEdges.Clear();
    _addedVertices.Clear();

    foreach (var vertice in verticesLeftColor)
    {
        SingleSubstitution(vertice, out var edges, out var vertices);

        if(!edges.IsNullOrEmpty())
            foreach (var edge in edges)
                if (Graph.TryAdd(edge))
                    _addedEdges.Add(edge);

        if (!vertices.IsNullOrEmpty())
            foreach (var vert in vertices)
                if (Graph.TryAdd(vert))
                    _addedVertices.Add(vert);
    }

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

private void SingleSubstitution(Vertice vertice, out List<Edge> edges, out List<Vertice> vertices)
{
    var pos = vertice.Position;

    var rightVertices = CurrentRule.Right.Vertices.ToList();
    var rightEdges = CurrentRule.Right.Edges.ToList();
    var mainRightVertice = rightVertices.FirstOrDefault(vert => vert.Color == CurrentRule.MainLeftColor);

    if(mainRightVertice is null)
    {
        edges = null;
```

```
            vertices = null;
            return;
        }

        var newEdges = new List<Edge>();
        var newVertices = new List<Vertice>();

        foreach(var vert in rightVertices)
        {
            var distance = vert.Position - mainRightVertice.Position;

            newVertices.Add(new Vertice()
            {
                Position = (pos + distance).Round(3),
                Color = vert.Color,
                Size = vert.Size
            });
        }

        foreach (var edge in rightEdges)
        {
            var distanceStart = edge.Start.Position - mainRightVertice.Position;
            var distanceEnd = edge.End.Position - mainRightVertice.Position;

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

            var start = newVertices.FirstOrDefault(vert => newStart == vert.Position);
            var end = newVertices.FirstOrDefault(vert => newEnd == vert.Position);

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

        edges = newEdges;
        vertices = newVertices;
    }
  }
}
```

Файл DtoObjects.cs

```
using Crystal.Services.CrystalServiceComponents.Abstraction;
using HelixToolkit.Wpf.SharpDX;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Media;
using System.Windows.Media.Media3D;

namespace Crystall.Services.CrystalServiceComponents.Abstraction
{
  public class GraphDto
  {
    public List<VerticeDto> Vertices { get; set; } = new List<VerticeDto>();
    public List<EdgeDto> Edges { get; set; } = new List<EdgeDto>();
    public Graph FromDto()
    {
      var graph = new Graph();
      foreach(var vertDto in Vertices)
      {
        graph.TryAdd(vertDto.FromDto());
      }
      foreach (var edgeDto in Edges)
      {
        var st = edgeDto.Start.ToVector3();
        var start = graph.Vertices.FirstOrDefault(vt => vt.Position == st);
        var en = edgeDto.End.ToVector3();
        var end = graph.Vertices.FirstOrDefault(vt => vt.Position == en);
        graph.TryAdd(new Edge(start, end) { Size = edgeDto.Size });
      }
      return graph;
    }
  }

  public class VerticeDto
  {
    public Point3D Position { get; set; }
```

```
        public double Size { get; set; }
        public Color Color { get; set; }

        public Vertice FromDto()
        {
            return new Vertice() { Size = this.Size, Color = this.Color.ToColor4(), Position = this.Position.ToVector3() };
        }
    }

    public class EdgeDto
    {
        public Point3D Start { get; set; }
        public Point3D End { get; set; }
        public double Size { get; set; }

    }
}
```

Файл Edge.cs

```
using Crystall.Services.CrystalServiceComponents.Abstraction;
using HelixToolkit.Wpf.SharpDX;
using SharpDX;

namespace Crystal.Services.CrystalServiceComponents.Abstraction
{
    public class Edge
    {
        public Edge(Vertice start, Vertice end)
        {
            Start = start;
            End = end;
        }
        public Vertice Start { get; set; }
        public Vertice End { get; set; }
        public double Size { get; set; } = Defaults.DefaultEdgeSize;
        public Vector3 Direction => Start.Position - End.Position;

        public EdgeDto ToDto()
        {
            return new EdgeDto() { Size = this.Size, Start = Start.Position.ToPoint3D(), End = End.Position.ToPoint3D() };
        }
    }
}
```

Файл Graph.cs

```
using Crystal.Extensions;
using Crystall.Services.CrystalServiceComponents.Abstraction;
using SharpDX;
using System.Collections.Generic;
using System.Linq;
using System.Xml;
using System.Xml.Serialization;

namespace Crystal.Services.CrystalServiceComponents.Abstraction
{
    public interface IGraph
    {
        IReadOnlyCollection<Vertice> Vertices { get; }
        IReadOnlyCollection<Edge> Edges { get; }
        bool TryAdd(Edge edge);
        bool TryAdd(Vertice vertice);
        void UnsafeAdd(params Vertice[] vertice);
        void UnsafeAdd(params Edge[] edge);
        void UnsafeAdd(IEnumerable<Vertice> vertice);
        void UnsafeAdd(IEnumerable<Edge> edge);
        IGraph Clone();
        GraphDto ToDto();
    }

    public class Graph : IGraph
    {
        Dictionary<Vector3, Vertice> _vertices = new Dictionary<Vector3, Vertice>();

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

        [XmlAttribute]
        public IReadOnlyCollection<Vertice> Vertices => _vertices.Values;
```

```
[XmlAttribute]
public IReadOnlyCollection<Edge> Edges => _edges.Values;

public bool TryAdd(Edge 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(Vertice vertice)
{
   if (_vertices.ContainsKey(vertice.Position))
       return false;

   _vertices.Add(vertice.Position, vertice);
   return true;
}

public void UnsafeAdd(params Vertice[] vertice)
{
   foreach (var vert in vertice)
       _vertices.Add(vert.Position, vert);
}

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

public void UnsafeAdd(IEnumerable<Vertice> vertice)
{
   foreach (var vert in vertice)
       _vertices.Add(vert.Position, vert);
}

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


public static IGraph SingleVerticeGraph(Color4 color, double size = Defaults.DefaultVerticeSize)
{
   var graph = new Graph();
   graph.TryAdd(new Vertice() { Color = color, Size = size });
   return graph;
}

public IGraph Clone()
{
   var originRefCloned = new HashSet<Vertice>();
   var clonedVertices = new HashSet<Vertice>();
   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);
       clonedVertices.TryAdd(startClone);
       clonedVertices.TryAdd(endClone);
   }

   var verticesToClone = Vertices.Except(originRefCloned);
   var verticesUnclonedClones = verticesToClone.Select(vert => vert.Clone());

   var graph = new Graph();
   graph.UnsafeAdd(clonedVertices);
   graph.UnsafeAdd(verticesUnclonedClones);
```

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

        public GraphDto ToDto()
        {
            var dto = new GraphDto();
            foreach (var v in Vertices)
                dto.Vertices.Add(v.ToDto());

            foreach (var e in Edges)
                dto.Edges.Add(e.ToDto());

            return dto;
        }
    }
}
```

Файл Rule.cs

```
using Crystal.Extensions;
using SharpDX;
using System;
using System.Linq;

namespace Crystal.Services.CrystalServiceComponents.Abstraction
{
    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.Vertices.FirstOrDefault(vertice => vertice.Color == mainLeftColor);
            if(colorOccurrence is null)
            {
                rule = null;
                return RuleErrorCode.WrongLeftColor;
```

```
            }

            colorOccurrence = right.Vertices.FirstOrDefault(vertice => vertice.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 vert in Left.Vertices)
                if (vert.Color == MainLeftColor)
                    vert.Color = color;

            MainLeftColor = color;
        }

        public void RepaintRightMain(Color4 color)
        {
            foreach (var vert in Right.Vertices)
                if (vert.Color == MainRightColor)
                    vert.Color = color;

            MainRightColor = color;
        }

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

Файл Vertice.cs

```
using Crystall.Services.CrystalServiceComponents.Abstraction;
using HelixToolkit.Wpf.SharpDX;
using SharpDX;

namespace Crystal.Services.CrystalServiceComponents.Abstraction
{
    using Point3 = SharpDX.Vector3;

    public class Vertice
    {
        public double Size { get; set; } = Defaults.DefaultVerticeSize;
        public Point3 Position { get; set; }
        public Color4 Color { get; set; }

        public Vertice Clone()
        {
```

```
            return new Vertice()
            {
                Color = this.Color,
                Size = this.Size,
                Position = this.Position
            };
        }

        public VerticeDto ToDto()
        {
            return new VerticeDto() { Position = this.Position.ToPoint3D(), Size = this.Size, Color = this.Color.ToColor() };
        }
    }
}
```

Файл CameraType.cs

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

Файл CollectionExtensions.cs

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

namespace Crystal.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;
        }
    }
}
```

Файл ErrorCode.cs

```
namespace Crystal.Extensions
{
    public enum RuleErrorCode
    {
```

```
        Succeed = 0,
        WrongLeftColor = 1,
        WrongRightColor = 2,
    }
}
```

Файл MathExtenstion.cs

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

namespace Crystal.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);
        }
    }
}
```

Файл ServiceProviderExtensions.cs

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

namespace Crystal.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);
        }
    }
}
```

Файл ColorToBrushConverter.cs

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

namespace Crystal.Utils
{
    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;
        }
    }
}
```

Файл RadiansToDegreesConverter.cs

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

namespace Crystal.Utils
{
    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;
        }
    }
}
```

Файл RelayCommand.cs

```
using System;
```

```
using System.Windows.Input;

namespace Crystal.Utils
{
   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);
      }
   }
}
```

Файл StlExporter.cs

```
using System.IO;
using System.Linq;
using Media3D = System.Windows.Media.Media3D;
using HelixToolkit.Wpf.SharpDX;
using HelixToolkit.Wpf.SharpDX.Model;
using HelixToolkit.Wpf.SharpDX.Model.Scene;

namespace Crystal.Utils
{
   public class StlExporter
   {
      public void Export(GroupNode groupNode, FileStream stream)
      {
         var writer = new BinaryWriter(stream);

         int triangleIndicesCount = 0;

         foreach (MeshNode meshNode in groupNode.Traverse().Skip(1))
         {
            triangleIndicesCount += meshNode.Geometry.Indices.Count;
         }
         ExportHeader(writer, triangleIndicesCount / 3);

         foreach (MeshNode meshNode in groupNode.Traverse().Skip(1))
         {
            ExportMeshNode(writer, meshNode);
         }
      }

      void ExportMeshNode(BinaryWriter writer, MeshNode meshNode)
      {

         var mesh = (MeshGeometry3D)meshNode.Geometry;
         var normals = mesh.Normals;

         var transform = new Media3D.MatrixTransform3D(meshNode.ModelMatrix.ToMatrix3D());

         var matrix = meshNode.ModelMatrix;
         matrix.M41 = 0;
         matrix.M42 = 0;
         matrix.M43 = 0;
         var normalTransform = new Media3D.MatrixTransform3D(matrix.ToMatrix3D());

         var material = (DiffuseMaterialCore)meshNode.Material;
         var color = material.DiffuseColor.ToColor();
```

```
            byte r = color.R;
            byte g = color.G;
            byte b = color.B;
            ushort attribute = (ushort)((1 << 15) | ((r >> 3) << 10) | ((g >> 3) << 5) | (b >> 3));

            for (int i = 0; i < mesh.TriangleIndices.Count; i += 3)
            {
                int i0 = mesh.TriangleIndices[i + 0];
                int i1 = mesh.TriangleIndices[i + 1];
                int i2 = mesh.TriangleIndices[i + 2];

                // Normal
                var faceNormal = normalTransform.Transform((normals[i0] + normals[i1] + normals[i2]).ToVector3D());
                faceNormal.Normalize();
                WriteVector(writer, faceNormal);

                // Vertices
                WriteVertex(writer, transform.Transform(mesh.Positions[i0].ToPoint3D()));
                WriteVertex(writer, transform.Transform(mesh.Positions[i1].ToPoint3D()));
                WriteVertex(writer, transform.Transform(mesh.Positions[i2].ToPoint3D()));

                // Attributes
                writer.Write(attribute);
            }
        }

        private static void WriteVector(BinaryWriter writer, Media3D.Vector3D normal)
        {
            writer.Write((float)normal.X);
            writer.Write((float)normal.Y);
            writer.Write((float)normal.Z);
        }

        private static void WriteVertex(BinaryWriter writer, Media3D.Point3D p)
        {
            writer.Write((float)p.X);
            writer.Write((float)p.Y);
            writer.Write((float)p.Z);
        }

        void ExportHeader(BinaryWriter writer, int triangleCount)
        {
            ExportHeader(writer);
            writer.Write(triangleCount);
        }

        void ExportHeader(BinaryWriter writer)
        {
            writer.Write(new byte[80]);
        }
    }
}
```

Файл BaseCenteredLatticeConstructor.cs

```
using System;
using SharpDX;
using Crystal;
using Crystal.Services.CrystalServiceComponents.LatticeContructors;
using Crystal.Services.CrystalServiceComponents.Abstraction;

namespace Crystall.Services.CrystalServiceComponents.LatticeContructors
{
    public class BaseCenteredLatticeConstructor : LatticeConstructor
    {
        public override void CreateCell()
        {
            float a = ParameterA;
            float b = ParameterB;
            float c = ParameterC;
            float alpha = ParameterAlpha;
            float beta = ParameterBeta;
            float gamma = ParameterGamma;

            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 Vertice() { Position = b1, Size = VerticeSize, Color = LeftColor };
            var v2 = new Vertice() { Position = b2, Size = VerticeSize, Color = RightColor };
            var v3 = new Vertice() { Position = b3, Size = VerticeSize, Color = RightColor };
            var v4 = new Vertice() { Position = b4, Size = VerticeSize, Color = RightColor };
            var v5 = new Vertice() { Position = t1, Size = VerticeSize, Color = RightColor };
            var v6 = new Vertice() { Position = t2, Size = VerticeSize, Color = RightColor };
            var v7 = new Vertice() { Position = t3, Size = VerticeSize, Color = RightColor };
            var v8 = new Vertice() { Position = t4, Size = VerticeSize, 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 Vertice() { Position = botCent, Size = VerticeSize, 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 Vertice() { Position = topCent, Size = VerticeSize, 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;
        }
    }
}
```

Файл BodyCenteredLatticeConstructor.cs

```
using Crystal.Services.CrystalServiceComponents.Abstraction;
using Crystal;
using Crystal.Services.CrystalServiceComponents.LatticeContructors;
using System;
using SharpDX;

namespace Crystall.Services.CrystalServiceComponents.LatticeContructors
{
    internal class BodyCenteredLatticeConstructor : LatticeConstructor
    {
```

```
        public override void CreateCell()
        {
            float a = ParameterA;
            float b = ParameterB;
            float c = ParameterC;
            float alpha = ParameterAlpha;
            float beta = ParameterBeta;
            float gamma = ParameterGamma;

            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 Vertice() { Position = b1, Size = VerticeSize, Color = LeftColor };
            var v2 = new Vertice() { Position = b2, Size = VerticeSize, Color = RightColor };
            var v3 = new Vertice() { Position = b3, Size = VerticeSize, Color = RightColor };
            var v4 = new Vertice() { Position = b4, Size = VerticeSize, Color = RightColor };
            var v5 = new Vertice() { Position = t1, Size = VerticeSize, Color = RightColor };
            var v6 = new Vertice() { Position = t2, Size = VerticeSize, Color = RightColor };
            var v7 = new Vertice() { Position = t3, Size = VerticeSize, Color = RightColor };
            var v8 = new Vertice() { Position = t4, Size = VerticeSize, 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 Vertice() { Position = cent, Size = VerticeSize, 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;
        }
    }
}
```

Файл FaceCenteredLatticeConstructor.cs

```
using System;
using SharpDX;
using Crystal;
using Crystal.Services.CrystalServiceComponents.LatticeContructors;
```

```
using Crystal.Services.CrystalServiceComponents.Abstraction;

namespace Crystall.Services.CrystalServiceComponents.LatticeContructors
{
   public class FaceCenteredLatticeConstructor : LatticeConstructor
   {
      public override void CreateCell()
      {
         float a = ParameterA;
         float b = ParameterB;
         float c = ParameterC;
         float alpha = ParameterAlpha;
         float beta = ParameterBeta;
         float gamma = ParameterGamma;

         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 Vertice() { Position = b1, Size = VerticeSize, Color = LeftColor };
         var v2 = new Vertice() { Position = b2, Size = VerticeSize, Color = RightColor };
         var v3 = new Vertice() { Position = b3, Size = VerticeSize, Color = RightColor };
         var v4 = new Vertice() { Position = b4, Size = VerticeSize, Color = RightColor };
         var v5 = new Vertice() { Position = t1, Size = VerticeSize, Color = RightColor };
         var v6 = new Vertice() { Position = t2, Size = VerticeSize, Color = RightColor };
         var v7 = new Vertice() { Position = t3, Size = VerticeSize, Color = RightColor };
         var v8 = new Vertice() { Position = t4, Size = VerticeSize, 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 Vertice() { Position = botCent, Size = VerticeSize, 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 Vertice() { Position = topCent, Size = VerticeSize, 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 Vertice() { Position = sideCent1, Size = VerticeSize, 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 Vertice() { Position = sideCent2, Size = VerticeSize, 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 Vertice() { Position = sideCent3, Size = VerticeSize, 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 Vertice() { Position = sideCent4, Size = VerticeSize, 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;
        }
    }
}
```

Файл HexagonalConstructor.cs

```
using Crystal.Services.CrystalServiceComponents.Abstraction;
using Crystal.Services.CrystalServiceComponents.LatticeContructors;
using SharpDX;
using Crystal.Extensions;
using System;

namespace Crystall.Services.CrystalServiceComponents.LatticeContructors
{
    public class HexagonalConstructor : LatticeConstructor
    {
        public HexagonalConstructor()
        {
            //120 градусів
            ParameterGamma = (float)(Math.PI / 3 * 2);
        }

        public override void CreateCell()
        {
            float a = ParameterA;
            float b = ParameterB;
            float c = ParameterC;
            float alpha = ParameterAlpha;
            float beta = ParameterBeta;
            float gamma = ParameterGamma;

            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 Vertice() { Position = b1, Size = VerticeSize, Color = LeftColor };
            var v2b = new Vertice() { Position = b2, Size = VerticeSize, Color = RightColor };
            var v3b = new Vertice() { Position = b3, Size = VerticeSize, Color = RightColor };
            var v4b = new Vertice() { Position = b4, Size = VerticeSize, Color = RightColor };
            var v5b = new Vertice() { Position = b5, Size = VerticeSize, Color = RightColor };
            var v6b = new Vertice() { Position = b6, Size = VerticeSize, 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 Vertice() { Position = t1, Size = VerticeSize, Color = RightColor };
            var v2t = new Vertice() { Position = t2, Size = VerticeSize, Color = RightColor };
            var v3t = new Vertice() { Position = t3, Size = VerticeSize, Color = RightColor };
            var v4t = new Vertice() { Position = t4, Size = VerticeSize, Color = RightColor };
            var v5t = new Vertice() { Position = t5, Size = VerticeSize, Color = RightColor };
            var v6t = new Vertice() { Position = t6, Size = VerticeSize, 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;
        }
    }
}
```

Файл LatticeConstructor.cs

```
using Crystal.Extensions;
using Crystal.Services.CrystalServiceComponents.Abstraction;
using SharpDX;
using System;

namespace Crystal.Services.CrystalServiceComponents.LatticeContructors
{

   public abstract class LatticeConstructor
   {
      public LatticeConstructor()
      {
      }

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

      public abstract void CreateCell();

      public event EventHandler<IGraph> LaticeUpdated;

      public IGraph Graph { get; protected set;  }

      public LatticeConstructor(float a, float b, float c,
         float alpha, float beta, float gamma, double verticeSize = 1)
      {
         ParameterA = a;
         ParameterB = b;
         ParameterC = c;
         ParameterAlpha = alpha;
         ParameterBeta = beta;
         ParameterGamma = gamma;
         VerticeSize = verticeSize;
      }

      public Color4 LeftColor { get; set; } = Defaults.LeftColor;
      public Color4 RightColor { get; set; } = Defaults.RightColor;
      public float ParameterA { get; set; } = 1.0f;
      public float ParameterB { get; set; } = 1.0f;
      public float ParameterC { get; set; } = 1.0f;

      public float ParameterAlpha { get; set; } = MathExtenstion.ToRadians(90);
      public float ParameterBeta { get; set; } = MathExtenstion.ToRadians(90);
      public float ParameterGamma { get; set; } = MathExtenstion.ToRadians(90);

      public double VerticeSize { get; set; } = Defaults.DefaultVerticeSize;

      //InterEdit stands for interactively editable
      public bool InterEditParameterA { get; set; } = true;
      public bool InterEditParameterB { get; set; } = true;
      public bool InterEditParameterC { get; set; } = true;
      public bool InterEditParameterAlpha { get; set; } = true;
      public bool InterEditParameterBeta  { get; set; } = true;
      public bool InterEditParameterGamma { get; set; } = true;
   }
}
```

Файл TriclinicLatticeFullConstructor.cs

```
using Crystal.Services.CrystalServiceComponents.Abstraction;
using SharpDX;
using System;

namespace Crystal.Services.CrystalServiceComponents.LatticeContructors
{
   public class TriclinicLatticeFullConstructor : LatticeConstructor
   {
      public TriclinicLatticeFullConstructor()
      {
      }

      public TriclinicLatticeFullConstructor(float a, float b, float c,
         float alpha, float beta, float gamma, double verticeSize = 1)
```

```
        : base(a, b, c, alpha, beta, gamma, verticeSize)
    {
    }

    public override void CreateCell()
    {
        float a = ParameterA;
        float b = ParameterB;
        float c = ParameterC;
        float alpha = ParameterAlpha;
        float beta = ParameterBeta;
        float gamma = ParameterGamma;

        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 Vertice() { Position = b1, Size = VerticeSize, Color = LeftColor };
        var v2 = new Vertice() { Position = b2, Size = VerticeSize, Color = RightColor };
        var v3 = new Vertice() { Position = b3, Size = VerticeSize, Color = RightColor };
        var v4 = new Vertice() { Position = b4, Size = VerticeSize, Color = RightColor };
        var v5 = new Vertice() { Position = t1, Size = VerticeSize, Color = RightColor };
        var v6 = new Vertice() { Position = t2, Size = VerticeSize, Color = RightColor };
        var v7 = new Vertice() { Position = t3, Size = VerticeSize, Color = RightColor };
        var v8 = new Vertice() { Position = t4, Size = VerticeSize, 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;
    }
  }
}
```

Файл TriclinicLatticeSimpleConstructor.cs

```
using System;
using SharpDX;
using Crystal.Services.CrystalServiceComponents.Abstraction;

namespace Crystal.Services.CrystalServiceComponents.LatticeContructors
{
  public class TriclinicLatticeSimpleConstructor : LatticeConstructor
  {
    public TriclinicLatticeSimpleConstructor()
    {
    }

    public TriclinicLatticeSimpleConstructor(float a, float b, float c,
      float alpha, float beta, float gamma, double verticeSize = 1)
```

```
        : base(a, b, c, alpha, beta, gamma, verticeSize)
    {
    }

    public override void CreateCell()
    {
        float a  = ParameterA;
        float b  = ParameterB;
        float c  = ParameterC;
        float alpha = ParameterAlpha;
        float beta =  ParameterBeta;
        float gamma = ParameterGamma;

        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 Vertice() { Position = b1, Size = VerticeSize, Color = LeftColor };
        var v2 = new Vertice() { Position = b2, Size = VerticeSize, Color = RightColor };
        var v3 = new Vertice() { Position = b3, Size = VerticeSize, Color = RightColor };
        var v4 = new Vertice() { Position = t1, Size = VerticeSize, 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;
    }
  }
}
```

Файл BaseViewModel.cs

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace Crystal.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(PropertyChangedEventArgs args)
```

```
        {
            if (!DisablePropertyChangedEvent)
                PropertyChanged?.Invoke(this, args);
        }

        protected bool Set<T>(ref T backingField, T value, [CallerMemberName] string propertyName = "")
        {
            if (EqualityComparer<T>.Default.Equals(backingField, 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(backingField, value))
            {
                return false;
            }

            backingField = value;
            if (raisePropertyChanged)
            {
                this.RaisePropertyChanged(propertyName);
            }
            return true;
        }
    }
}
```

Файл MainViewModel.cs

```
using Crystal.Controllers;
using Crystal.Extensions;
using Crystal.Services;
using Crystal.Views;
using System;

namespace Crystal.ViewModels
{
    public class MainViewModel : BaseViewModel
    {
        IServiceProvider _services;
        IUIService _uiService;
        IStateManager _stateManager;

        SideMenuViewModel _sideMenuViewModel { get; set; }

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

        void SetViews()
        {
            _sideMenuViewModel = new SideMenuViewModel(_services);
            SideMenu = _uiService.CreateView(_sideMenuViewModel);
            PrimaryViewportView = _stateManager.PrimaryViewportController.Control;
            RulesListView = _stateManager.RulesListView;
            RuleView = _stateManager.RuleView;
        }

        public IUIView SideMenu { get; private set; }
        public IUIView PrimaryViewportView { get; private set; }
        public IUIView RulesListView { get; private set; }
        public IUIView RuleView { get; private set; }
    }
}
```

Файл MainWindowViewModel.cs

```
using Crystal.Controllers;
```

```
using Crystal.Extensions;
using Crystal.Services;
using Crystal.Utils;
using Crystal.Views;
using Microsoft.Win32;
using System;
using System.Windows.Input;

namespace Crystal.ViewModels
{
    public class MainWindowViewModel : BaseViewModel
    {
        IServiceProvider _services;
        IUIService _uiService;
        IStateManager _stateManager;
        ICommandManager _commandManager;
        MainViewModel _mainViewModel;
        RelayCommand<object> _exportToStlCommand;
        RelayCommand<object> _exportToXmlCommand;
        RelayCommand<object> _importFromXmlCommand;

        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; }

        public ICommand ExportToStlCommand =>
            _exportToStlCommand ?? (_exportToStlCommand = new RelayCommand<object>( _ =>
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "Crystal (*.stl)|*.stl";
            if (saveFileDialog.ShowDialog() == true)
            {
                //var stateManager = _services.GetService<IStateManager>();
                _stateManager.PrimaryViewportController.ExportToStl(saveFileDialog.FileName);
            }
        }));


        public ICommand ExportToXmlCommand =>
            _exportToXmlCommand ?? (_exportToXmlCommand = new RelayCommand<object>(_ =>
            {
                SaveFileDialog saveFileDialog = new SaveFileDialog();
                saveFileDialog.Filter = "Crystal (*.xml)|*.xml";
                if (saveFileDialog.ShowDialog() == true)
                {
                    _commandManager.OnCommandCall(this, new CommandEventArgs(saveFileDialog.FileName, Defaults.ExportToXmlCommandId));
                }
            }));


        public ICommand ImportFromXmlCommand =>
            _importFromXmlCommand ?? (_importFromXmlCommand = new RelayCommand<object>(_ =>
            {
                OpenFileDialog openFileDialog = new OpenFileDialog();
                openFileDialog.Filter = "Crystal (*.xml)|*.xml";
                if (openFileDialog.ShowDialog() == true)
                {
                    _commandManager.OnCommandCall(this, new CommandEventArgs(openFileDialog.FileName, Defaults.ImportFromXmlCommandId));
                }
            }));
    }
}
```

Файл RulesListViewModel.cs

```
using System;
using System.Collections.ObjectModel;
using Crystal.Controllers;
using Crystal.Extensions;
using Crystal.Services;
using Crystal.Views;
```

```
using Crystal.Services.CrystalServiceComponents.Abstraction;
using Crystal.Services.CrystalServiceComponents.LatticeContructors;
using SharpDX;

namespace Crystal.ViewModels
{
   public class RulesListViewModel : BaseViewModel
   {
      public class ListItemViewModel
      {
         public LatticeConstructor Constructor { get; }
         public IViewportController 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(IViewportController viewportController, string title, LatticeConstructor 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(LatticeConstructor constructor, string title)
      {
         var viewport = _uiService.CreateViewportController();
         viewport.AddVisual(constructor.Graph.Vertices);
         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);

         //перший доданий елемент
         if (ItemsViewModels.Count == 1)
            SelectedItemViewModel = item;
      }
   }
}
```

Файл RuleViewModel.cs

```
using System;
using System.Windows.Input;
using Crystal.Services.CrystalServiceComponents.Abstraction;
using Crystal.Utils;
```

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

namespace Crystal.ViewModels
{
    public class RuleViewModel : BaseViewModel
    {
        IUIView _leftViewport;
        IUIView _rightViewport;

        RelayCommand<object> _leftColorClickedCommand;
        RelayCommand<object> _rightColorClickedCommand;

        IRule _rule;

        public event EventHandler<Color4> LeftPartRepainted;
        public event EventHandler<Color4> RightPartRepainted;

        public RuleViewModel(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();
            }
        }));
  }
}
```

Файл SideMenuViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Input;
using Crystal.Controllers;
using Crystal.Extensions;
using Crystal.Utils;

namespace Crystal.ViewModels
{
  public class SideMenuViewModel : BaseViewModel
  {
    IServiceProvider _services;
    IStateManager _stateManager;
    ICommandManager _commandManager;

    RelayCommand<object> _substituionCommand;
    RelayCommand<object> _repaintCommand;
    RelayCommand<object> _clearPrimaryViewportCommand;

    Dictionary<string, Action<string, object>> _reactions
      = new Dictionary<string, Action<string, object>>();

    public SideMenuViewModel(IServiceProvider services)
    {
      _services = services;
      _commandManager = _services.GetService<ICommandManager>();
      _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.ParameterA)]
        = (_, value) => RaisePropertyChanged(nameof(ParameterA));

      _reactions[nameof(_stateManager.ParameterB)]
        = (_, value) => RaisePropertyChanged(nameof(ParameterB));

      _reactions[nameof(_stateManager.ParameterC)]
        = (_, value) => RaisePropertyChanged(nameof(ParameterC));

      _reactions[nameof(_stateManager.ParameterAlpha)]
        = (_, value) => RaisePropertyChanged(nameof(ParameterAlpha));

      _reactions[nameof(_stateManager.ParameterBeta)]
        = (_, value) => RaisePropertyChanged(nameof(ParameterBeta));

      _reactions[nameof(_stateManager.ParameterGamma)]
```

```
        = (_, value) => RaisePropertyChanged(nameof(ParameterGamma));

    _reactions[nameof(_stateManager.InterEditParameterA)]
        = (_, value) => RaisePropertyChanged(nameof(InterEditParameterA));

    _reactions[nameof(_stateManager.InterEditParameterB)]
     = (_, value) => RaisePropertyChanged(nameof(InterEditParameterB));

    _reactions[nameof(_stateManager.InterEditParameterC)]
     = (_, value) => RaisePropertyChanged(nameof(InterEditParameterC));

    _reactions[nameof(_stateManager.InterEditParameterAlpha)]
     = (_, value) => RaisePropertyChanged(nameof(InterEditParameterAlpha));

    _reactions[nameof(_stateManager.InterEditParameterBeta)]
     = (_, value) => RaisePropertyChanged(nameof(InterEditParameterBeta));

    _reactions[nameof(_stateManager.InterEditParameterGamma)]
     = (_, value) => RaisePropertyChanged(nameof(InterEditParameterGamma));

    _reactions[nameof(_stateManager.SubstitionsCount)]
        = (_, value) => RaisePropertyChanged(nameof(SubstitionsCount));

    _reactions[nameof(_stateManager.VerticesCount)]
        = (_, value) => RaisePropertyChanged(nameof(VerticesCount));

    _reactions[nameof(_stateManager.EdgesCount)]
        = (_, value) => RaisePropertyChanged(nameof(EdgesCount));
}

public float ParameterA
{
    get => _stateManager.ParameterA;
    set => _stateManager.ParameterA = value;
}

public float ParameterB
{
    get => _stateManager.ParameterB;
    set => _stateManager.ParameterB = value;

}

public float ParameterC
{
    get => _stateManager.ParameterC;
    set => _stateManager.ParameterC = value;
}

public float ParameterAlpha
{
    get => _stateManager.ParameterAlpha;
    set => _stateManager.ParameterAlpha = value;
}

public float ParameterBeta
{
    get => _stateManager.ParameterBeta;
    set => _stateManager.ParameterBeta = value;
}

public float ParameterGamma
{
    get => _stateManager.ParameterGamma;
    set => _stateManager.ParameterGamma = value;
}

public bool InterEditParameterA => _stateManager.InterEditParameterA;

public bool InterEditParameterB => _stateManager.InterEditParameterB;

public bool InterEditParameterC => _stateManager.InterEditParameterC;

public bool InterEditParameterAlpha => _stateManager.InterEditParameterAlpha;

public bool InterEditParameterBeta => _stateManager.InterEditParameterBeta;

public bool InterEditParameterGamma => _stateManager.InterEditParameterGamma;
```

```
        public int SubstitionsCount
        {
            get => _stateManager.SubstitionsCount;
            set => _stateManager.SubstitionsCount = value;
        }
        public int VerticesCount
        {
            get => _stateManager.VerticesCount;
            set => _stateManager.VerticesCount = value;
        }
        public int EdgesCount
        {
            get => _stateManager.EdgesCount;
            set => _stateManager.EdgesCount = value;
        }

        public ICommand SubstituionCommand =>
                _substituionCommand ?? (_substituionCommand = new RelayCommand<object>(
                    _ => _commandManager.OnCommandCall(this, new CommandEventArgs(null, Defaults.SubstitutionCommandId))));

        public ICommand ClearPrimaryViewportCommand =>
            _clearPrimaryViewportCommand ?? (_clearPrimaryViewportCommand = new RelayCommand<object>(
                _ => _commandManager.OnCommandCall(this, new CommandEventArgs(null, Defaults.ClearPrimaryViewportCommandId))));

        public ICommand RepaintCommand =>
            _repaintCommand ?? (_repaintCommand = new RelayCommand<object>(
                _ => _commandManager.OnCommandCall(this, new CommandEventArgs(null, Defaults.RepaintCommandId))));
    }
}
```

Файл ViewportViewModel.cs

```
using System;
using HelixToolkit.Wpf.SharpDX;
using System.Windows.Media;
using Media3D = System.Windows.Media.Media3D;
using Crystal.Extensions;
using HelixToolkit.Wpf.SharpDX.Model.Scene;
using System.Windows.Input;
using Crystal.Utils;

namespace Crystal.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>(_ =>
            {
                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
            };
        }
    }
}
```

Файл MainView.xaml.

```
<UserControl x:Class="Crystal.Views.MainView"
             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:Crystal.Views"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto"/>
```

```
        <ColumnDefinition Width="*"/>
      </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="MediumPurple">
        <ContentPresenter Content="{Binding SideMenu}"/>
      </Border>

      <ContentPresenter Grid.Column="1" Grid.Row="0" Content="{Binding PrimaryViewportView}"/>

      <Border Grid.ColumnSpan="2" Grid.Row="1" BorderThickness="0 4 0 4" BorderBrush="MediumPurple">
        <ContentPresenter Content="{Binding RulesListView}"/>
      </Border>

      <Border Grid.ColumnSpan="2" Grid.Row="2">
        <ContentPresenter Content="{Binding RuleView}" HorizontalAlignment="Center"/>
      </Border>
    </Grid>
</UserControl>
```

Файл MainView.xaml.cs

```
using Crystal.ViewModels;
using System.Windows.Controls;

namespace Crystal.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;
            }
        }
    }
}
```

Файл RulesListView.xaml

```
<UserControl x:Class="Crystal.Views.RulesListView"
             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:Crystal.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>
```

Файл RulesListView.xaml.cs

```
using Crystal.ViewModels;
using System.Windows.Controls;

namespace Crystal.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;
            }
        }
    }
}
```

Файл RuleView.xaml

```
<UserControl x:Class="Crystal.Views.RuleView"
             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:b="http://schemas.microsoft.com/xaml/behaviors"
             xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
             xmlns:local="clr-namespace:Crystal.Views"
             xmlns:utils="clr-namespace:Crystal.Utils"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800">
    <UserControl.Resources>
        <utils:ColorToBrushConverter x:Key="ColorConverter"/>
    </UserControl.Resources>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="auto" />
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="auto" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <Grid Grid.Column="0" Width="250" Height="auto">
```

```
            <ContentPresenter Content="{Binding LeftViewport}"/>
        </Grid>

        <Viewbox Grid.Column="1" 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="2" Width="250" Height="auto">
            <ContentPresenter Content="{Binding RightViewport}"/>
        </Grid>

        <Grid Grid.Column="3" Width="100">
            <Grid.RowDefinitions>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>

            <Rectangle Grid.Row="0" Name="LeftRect" Fill="{Binding LeftColor, Converter={StaticResource ColorConverter}}" MinHeight="25">
                <b:Interaction.Triggers>
                    <b:EventTrigger EventName="MouseDown">
                        <b:InvokeCommandAction Command="{Binding LeftColorClickedCommand}"/>
                    </b:EventTrigger>
                </b:Interaction.Triggers>
            </Rectangle>

            <Viewbox Grid.Row="1" 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"
                />
            </Viewbox>

            <Rectangle Grid.Row="2" Fill="{Binding RightColor, Converter={StaticResource ColorConverter}}" MinHeight="25">
                <b:Interaction.Triggers>
                    <b:EventTrigger EventName="MouseDown">
                        <b:InvokeCommandAction Command="{Binding RightColorClickedCommand}"/>
                    </b:EventTrigger>
                </b:Interaction.Triggers>
            </Rectangle>
        </Grid>
    </Grid>
</UserControl>
```

Файл RuleView.xaml.cs

```
using Crystal.ViewModels;
using System.Windows.Controls;

namespace Crystal.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;
```

```
        }
    }
  }
}
```

Файл SideMenuView.xaml

```
<UserControl x:Class="Crystal.Views.SideMenuView"
        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:utils="clr-namespace:Crystal.Utils"
        xmlns:local="clr-namespace:Crystal.Views"
        xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
        mc:Ignorable="d"
        d:DesignHeight="450" d:DesignWidth="800">
  <UserControl.Resources>
    <utils:RadiansToDegreesConverter x:Key="RadToDegConverter"/>
  </UserControl.Resources>
  <Grid>
    <StackPanel>
      <Button Content="Очистити сцену" Command="{Binding ClearPrimaryViewportCommand}" Margin="4"
          materialDesign:ButtonAssist.CornerRadius="0"/>
      <Button Content="Виконати підстановку" Command="{Binding SubstituionCommand}" Margin="4"
          materialDesign:ButtonAssist.CornerRadius="0"/>
      <Button Content="Перефарбувати" Command="{Binding RepaintCommand}" Margin="4"
          materialDesign:ButtonAssist.CornerRadius="0"/>
      <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"
              PreviewTextInput="NumberValidationTextBox"
              Text="{Binding ParameterA, StringFormat=N3, UpdateSourceTrigger=PropertyChanged}">
            <TextBox.Style>
              <Style BasedOn="{StaticResource MaterialDesignTextBox}" TargetType="TextBox">
                <Style.Triggers>
                  <DataTrigger Binding="{Binding InterEditParameterA}" 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"
              PreviewTextInput="NumberValidationTextBox"
              Text="{Binding ParameterB, StringFormat=N3, UpdateSourceTrigger=PropertyChanged}">
            <TextBox.Style>
              <Style BasedOn="{StaticResource MaterialDesignTextBox}" TargetType="TextBox">
                <Style.Triggers>
                  <DataTrigger Binding="{Binding InterEditParameterB}" 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"
              PreviewTextInput="NumberValidationTextBox"
              Text="{Binding ParameterC, StringFormat=N3, UpdateSourceTrigger=PropertyChanged}">
            <TextBox.Style>
              <Style BasedOn="{StaticResource MaterialDesignTextBox}" TargetType="TextBox">
                <Style.Triggers>
                  <DataTrigger Binding="{Binding InterEditParameterC}" 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"
                  PreviewTextInput="NumberValidationTextBox"
                  Text="{Binding ParameterAlpha, StringFormat=N2, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource RadToDegCon-
verter}}">
              <TextBox.Style>
                <Style BasedOn="{StaticResource MaterialDesignTextBox}" TargetType="TextBox">
                  <Style.Triggers>
                    <DataTrigger Binding="{Binding InterEditParameterAlpha}" 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"
                  PreviewTextInput="NumberValidationTextBox"
                  Text="{Binding ParameterBeta, StringFormat=N2, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource RadToDegCon-
verter}}">
              <TextBox.Style>
                <Style BasedOn="{StaticResource MaterialDesignTextBox}" TargetType="TextBox">
                  <Style.Triggers>
                    <DataTrigger Binding="{Binding InterEditParameterBeta}" 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"
                  PreviewTextInput="NumberValidationTextBox"
                  Text="{Binding ParameterGamma, StringFormat=N2, UpdateSourceTrigger=PropertyChanged, Converter={StaticResource RadToDegCon-
verter}}">
              <TextBox.Style>
                <Style BasedOn="{StaticResource MaterialDesignTextBox}" TargetType="TextBox">
                  <Style.Triggers>
                    <DataTrigger Binding="{Binding InterEditParameterGamma}" Value="False">
                      <Setter Property="IsEnabled" Value="False"/>
                    </DataTrigger>
                  </Style.Triggers>
                </Style>
              </TextBox.Style>
            </TextBox>
          </Grid>
        </GroupBox>

        <GroupBox Header="Додаткова інформація">
          <StackPanel>
            <TextBlock>
              <Run Text="Кількіть замін: "/>
              <Run Text="{Binding SubstitionsCount}"/>
            </TextBlock>
            <TextBlock>
              <Run Text="Кількіть вершин: "/>
```

```
                <Run Text="{Binding VerticesCount}"/>
              </TextBlock>
              <TextBlock>
                <Run Text="Кількіть ребер: "/>
                <Run Text="{Binding EdgesCount}"/>
              </TextBlock>
            </StackPanel>
          </GroupBox>
        </StackPanel>
    </Grid>
</UserControl>
```

Файл SideMenuView.xaml.cs

```
using Crystal.ViewModels;
using System.Text.RegularExpressions;
using System.Windows.Controls;
using System.Windows.Input;

namespace Crystal.Views
{
    /// <summary>
    /// Interaction logic for SideMenuView.xaml
    /// </summary>
    public partial class SideMenuView : UserControl, IUIView
    {
        public SideMenuView()
        {
            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;
        }
    }
}
```

Файл ViewportView.xaml

```
<UserControl x:Class="Crystal.Views.ViewportView"
             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:hx="http://helix-toolkit.org/wpf/SharpDX"
             xmlns:b="http://schemas.microsoft.com/xaml/behaviors"
             xmlns:local="clr-namespace:Crystal.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="#939393"
                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.BottomView" />
              <KeyBinding Key="L" Command="hx:ViewportCommands.LeftView" />
              <KeyBinding Key="R" Command="hx:ViewportCommands.RightView" />
              <KeyBinding Command="hx:ViewportCommands.ZoomExtents" 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: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>
```

Файл ViewportView.xaml.cs

```
using Crystal.ViewModels;
using System.Windows.Controls;

namespace Crystal.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);
                }
            }
        }
    }
}
```