МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Програмний модуль перегляду та редагування графічних 3D-моделей»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТ-61 Шкура Анастасія Володимирівна

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «___» _____2020 р.

Науковий керівник _____ к.т.н., доц., Ващенко С. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

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

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__» _____ 2020 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Шкура Анастасія Володимирівна

1 Тема роботи *Програмний модуль перегляду та редагування графічних 3D-моделей*

керівник роботи *Ващенко Світлана Михайлівна, к.т.н., доцент*,

затверджені наказом по університету від « 14 » травня 2020 р. № 0576-III

2 Строк подання студентом роботи «1» червня 2020 р.

3 Вхідні дані до роботи технічне завдання на розробку програмного продукту

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, постановка задачі, проектування програмного додатку, розробка програми перегляду та редагування

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність, постановка задачі, аналіз програм-аналогів, порівняльна таблиця програм-аналогів, функціональні вимоги, контекстна модель, діаграма декомпозиції, діаграма варіантів використання, прототип вікна програми, використані технології реалізації, основні елементи вікна, демонстрація можливостей перегляду, демонстрація роботи операції Translate, результата операції Set Texture, висновки, апробація роботи

6 Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7 Дата видачі завдання_____01.10.2019_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Формування завдання роботи	23.03.20 – 24.03.20	
2	Дослідження предметної області	25.03.20 – 26.03.20	
3	Розробка WBS	27.03.20 – 29.03.20	
4	Розробка OBS	30.03.20 – 01.04.20	
5	Створення діаграми Ганта	02.04.20 – 04.04.20	
6	Проведення аналізу ризиків	05.04.20 – 07.04.20	
7	Моделювання в IDEF-0	08.04.20 – 10.04.20	
8	Розробка середовища перегляду	11.04.20 – 01.05.20	
9	Розробка функцій редагування	02.05.20 – 20.05.20	
10	Створення інсталятору	21.05.20 – 22.05.20	
11	Тестування програми	22.05.20 – 24.05.20	
12	Оформлення документації	25.05.20 – 01.06.20	
13	Презентація проекту	10.06.20	

Студент _____ Шкура А.В.
(підпис)

Керівник роботи _____ к.т.н., доц. Ващенко С.М.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Програмний модуль перегляду та редагування графічних 3D-моделей».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 17 найменувань, додатків. Загальний обсяг роботи − 65 сторінок, у тому числі 46 сторінок основного тексту, 2 сторінки списку використаних джерел, 17 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці програми для перегляду та зручного редагування 3D-моделей.

В роботі проведено дослідження останніх дослідження в області 3D-моделювання, існуючих аналогів, зроблена постановка завдання.

У роботі було виконано планування, під час якого розроблено діаграми WBS, OBS та Ганта, проведено аналіз ризиків, а також проектування додатку, в якому наведені діаграми IDEF-0 та Use Case.

Результатом проведеної роботи є створений додаток для редагування 3D-моделей.

Практичне значення роботи полягає у тому, створена програма може бути використана для зручного показу та редагування 3D-об’єктів, а отримані моделі можна використовувати для 3D-печаті.

Ключові слова: WPF, MVVM, HOOPS VISUALIZE, 3D-МОДЕЛЮВАННЯ, ВІЗУАЛІЗАЦІЯ.

ЗМІСТ

ВСТУП

Про надзвичайну актуальність використання програмних засобів у роботі з комп'ютерною графікою, свідчить сучасний розвиток інформаційних технологій. Високі вимоги до візуальної реалізації призводять до швидкого зростання потужності комп'ютерів і можливостей, що надають їх графічні системи. Комп'ютерні технології у свою чергу привертають все більше увагу від фахівців різних напрямків, наприклад інженерів, програмістів, проектувальників середовища віртуальної реальності, розробників концептів комп'ютерних ігор, кінематографістів. Сучасна тривимірна графіка застосовується не лише для створення 2D-зображень, але і для розробки твердотільних об'єктів, таких як прості геометричні фігури, так і складні елементи двигунів чи споруд.

Сучасна індустрія у багатьох сферах не може існувати без 3D-графіки, особливо це проявляється у виробництві. Тривимірна графіка полягає вагоме місце у презентації нового виробу. Щоб почати виробництво доволі складної деталі необхідно спочатку створити 3D-модель об'єкту. Потім на її основі, з використанням технологій швидкого прототипування до яких належать наприклад 3D-друк, лиття силіконових форм або фрезерування, складається реалістичний зразок виробу.

Для створення моделей є надзвичайно важливими редактори 3D-графіки, які в наш час швидко розвиваються. Вони надають можливість створювати та модифікувати вже існуючі моделі для різних цілей.

Таким чином метою виконання даної роботи є створення програмного продукту з можливістю перегляду, а також легкого та швидкого редагування 3D-об'єктів.

Для досягнення мети необхідно виконати такі задачі:

- провести аналіз предметної області;
- провести планування та обрати технології реалізації;
- розробити базовий інтерфейс програми та можливості перегляду моделей;
- реалізувати функціонал для редагування моделей;

- провести тестування;
- розробити інструкцію користувача.

Даний проект є власністю компанії AMC Bridge.

Практичне значення даного проекту в тому, що створена програма може бути використана для зручного показу та редагування 3D-об’єктів у вигляді полігональної сітки в тому числі при роботі зі студентами в університеті. Також отримані моделі можна використовувати для 3D-печаті.

Оприлюднення результатів роботи проводилося на науково-технічній конференції «ІМА 2020» (СумДУ) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Тривимірна графіка або 3D-моделювання – це вид комп’ютерної графіки, що поєднує в собі інструменти та засоби для створення об’ємних моделей в тривимірному просторі [2]. Вона настільки вживана тому, що вона може генерувати реалістичні зображення, створювати відмінні ефекти в іграх і включати гладкі ефекти для призначених для користувача інтерфейсів.

Зі стрімким розвитком інформаційних технологій широко почала використовуватись комп’ютерна графіка в наукових дослідженнях, освіті, комерційній, промисловій, військовій, розважальній та багатьох інших сферах. У різних галузях досліджень методи 3D-моделювання мають свої особливості.

3D-моделювання привертає увагу науковців. Наведемо декілька цікавих прикладів використання 3D-моделювання в науково-дослідних роботах.

Зараз набуває розповсюдження її використання у медицині для створення моделей людини. Для пацієнта такі моделі створюються за допомогою даних комп’ютерної томографії та МРТ, що потім проходять серію етапів постобробки та сегментації зображення для виділення областей, що мають найбільший інтерес в конкретному випадку. Відповідно до рекомендацій щодо адекватності, розробленими SIG, 3D-друковані моделі показали переваги і корисність у пацієнтів зі складними переломами черепа, особи і нижньої щелепи, а також з порушеннями скронево-нижньощелепного суглоба, доброякісними і злоякісними пухлинами [3].

Поширеним є використання 3D-графіки в астрономії. Майже завжди дані або зображення, що отримуються за допомогою приладів є лише двохвимірними. Здатність вивчати астрономічні джерела з усіх боків дає вченим можливість краще зрозуміти, як структуруються космічні об’єкти та їх фізичні властивості. Астрофізики, фахівці в галузі інформатики, інженери створюють нові методи, щоб

вивести візуалізацію за межі 2D-зображень. Це дає можливість вченим переглядати об’єкти з будь-якого кута, а в деяких випадках і практично подорожувати ними [4].

Для моделювання об’єкта необхідно спочатку отримати дані про його розміри. У різних галузях досліджень та застосувань відповідно є різні вимоги. Обладнання та методи моделювання, прийняті в процесі моделювання, також відрізняються [5]. Наприклад, для реконструкції 3D-підводного середовища люди повинні використовувати сонарний та океанічний супутник, щоб отримати дані про підводну місцевість. Щоб поліпшити розуміння оператора людини, що керує підводним дистанційно керованим транспортним засобом, U.Castellani реконструює 3D-підводне середовище за допомогою акустичної камери в режимі реального часу [6].

Щоб ефективно та точно моделювати навколишнє середовище, Toshihiro та Masayuki отримують дані про діапазон та кольори, інтегруючи всеспрямований лазерний далекомір та всенаправлену багатокамерну систему. Для того, щоб стабільно та одночасно реєструвати дані про об’єкти за допомогою вдосконаленого алгоритму ICP, плоскі поверхні, такі як стіни та дороги, витягуються та використовуються в процесі реєстрації. Потім сформована поверхнева модель текстурується на основі всенаправлених зображень, вибраних з урахуванням роздільної здатності та оклюзії [7].

Дослідження 3D-моделювання сприяє розвитку охорони спадщини. Існує декілька проектів охорони спадщини, які використовують лазерне сканування, такі як Stanford Digital Forma Urbis Romae Project або Digital Michelangelo Project [8]. Була використана структурована освітлювальна система ShapeCam для створення детальної 3D-моделі мфея Антонінеї у старовинному місті Сагалассос (Південна Туреччина) [9].

Сьогодні все більше різних дисциплін потребують застосування 3D-моделювання. Є низка напрямків, за якими нам потрібно продовжувати, таких як пошук та автоматичне створення 3D-моделей.

У наш час візуальна якість стає однією з головних точок уваги. Нам варто вивчати та знаходити нові, більш досконалі та швидкі методи оцифрування реального світу. Динамічна модель – наш новий напрямок для майбутньої роботи.

Динамічні моделі можуть імітувати взаємні дії об’єктів, що також дуже корисно для вивчення [5].

1.2 Огляд існуючих програмних продуктів-аналогів

Існують найрізноманітніші програми для 3D-моделювання, розроблені для конкретних цілей. Для інженерного 3D-моделювання найкраще підходять SolidWorks або AutoCAD, для скульптинга – ZBrush, для загального різнопланового моделювання – Blender, Maya, 3ds Max, Mudbox. Деякі з них мають відмінно реалізовані функції анімації. Розглянемо детальніше деякі з них.

Blender – це безкоштовна програма для створення комп’ютерної графіки. Має зрозумілий інтерфейс, що легко адаптується так, щоб найбільш потрібні інструменти завжди були під рукою. Blender включає потужні інструменти для моделювання, візуалізації, анімації, такелажу, відстеження руху, а також створення ігор та відео. Додаток дозволяє здійснювати досить детальне проектування моделей, а також їх обробку. Інструментами для роботи з анімацією можуть бути оброблені окрім достатньо простих дій, таких як цикли пересування персонажа, доволі складні, наприклад анімація обличчя при розмові. Blender – програма з великою спільнотою і пропонує безліч форумів, уроків та навчальних посібників [10].

AutoCAD може створити будь-які 2D-креслення та 3D-моделі, які може уявити людина. Програма також дозволяє користувачеві групувати або шарувати об’єкти, зберігати об’єкти в базі даних для подальшого використання, маніпулювати властивостями об’єктів, такими як розмір, форма та розташування та багато іншого. AutoCAD має безліч застосувань у широкому спектрі областей. Програму можна використовувати як для простих проектів, таких як графіки чи презентації, або складних конструкцій, наприклад, для складання архітектури будівлі або інженерних конструкцій [11].

Додаток SketchUp від Google є досить простим у використанні та може допомогти навчитися створювати досить складні тривимірні моделі будинків, машин, або навіть дуже складних космічних ракет. Google SketchUp – це програма для тих, хто тільки розпочинає навчатися 3D-моделюванню і є відмінною можливістю зрозуміти, що таке взагалі просторове моделювання. Вона дозволяє доволі легко додавати деталі, змінювати розміри та структуру вже існуючих моделей з великою точністю.

Додатки для 3D-графіки розвиваються дуже швидко. Досить довгий період часу на ринку професійних програм для 3D-моделювання провідні позиції займали в основному комерційні програми, такі як SolidWorks або 3DMAX. Це все складні програми для створення та редагування тривимірних об’єктів, для повноцінного вивчення яких потрібно витратити багато часу. Однак зараз вже існує досить багато популярних програм, що постачаються на основі вільної ліцензії, та вони продовжують набирати популярність.

Проведемо порівняння популярних додатків для 3D-моделювання (табл. 1.1).

Таблиця 1.1 – Порівняльна таблиця додатків для 3D-моделювання

№	Критерій	Blender	3ds Max	Maya	SketchUp
1	Операційні системи	Windows, Mac OS X, Linux, Unix	Windows	Windows, Mac OS X, Linux	Windows, Mac OS X
2	Ціна	Безкоштовний	Безкоштовний	Платний	Фрімиум
3	Моделювання	+	+	+	+
4	3D Painting	+	+	+	-
5	Анімація	+	+	+	+
6	Рендеринг	+	+	+	+
7	Позиційний трекінг	+	-	Підтримка припинена	-
8	Композитинг	+	+-	Підтримка припинена	-

Багато з найпопулярніших програм мають дуже дорогі ліцензії, а отже не є досяжними для широкої маси користувачів. Майже всі вони є широкопрофільними,

а отже мають багато функціоналу, про який користувач навіть не буде знати, тому що він йому не потрібний. Це також призводить до того, що такі програми займають дуже багато місця на диску та при роботі потребують велику кількість оперативної пам’яті. Також вони є досить вимогливими до конфігурації та потужності системи.

1.3 Постановка задачі

Метою проекту є створення додатку для перегляду 3D-моделей. Також програма повинна надавати можливість легко редагувати об’єкти, а саме змінювати колір моделі, накладати матеріал, переміщувати, обертати, масштабувати та видаляти об’єкти.

Створена програма має містити інтуїтивно зрозумілий і зручний інтерфейс для користувача. При наявності англомовного інтерфейсу розроблений продукт може бути використаний не залежно від регіону чи країни.

Для досягнення вище зазначеної мети необхідно вирішити такий перелік задач:

- проаналізувати предметну область;
- провести планування;
- розробити та реалізувати базовий інтерфейс програми;
- забезпечити функціонал відкриття та збереження моделей;
- реалізувати базовий функціонал для перегляду моделей (робота з камерою та видом):
  - зміна камери кнопками миші;
  - створення базових видів (спереду, справа, зліва і т.д.);
  - показ сітки;
  - показ NavigationCube та AxisTriad.
- забезпечити можливість виділення об’єктів;

- реалізувати функції для редагування моделей:
  - зміна кольору;
  - накладання матеріалу;
  - видалення об’єктів;
  - переміщення, масштабування та обертання.
- протестувати програму:
  - розробити тест-кейси;
  - провести тестування.
- розробити інструкцію користувача.

Технічне завдання наводиться у додатку А.

Додаток Б відображає хід проведеного планування виконання дипломногопроекту.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

2.1 Моделювання в IDEF0

Процес проектування програми розпочинаємо з розробки контекстної діаграми А-0, адже вона моделює систему у найбільш загальному вигляді та описує її взаємодію з навколишнім середовищем.

Головними елементами даної діаграми є:

– вхід – інформація або об’єкти, що використовуються для отримання результату;
– управління – керуючі, регламентуючі і нормативні дані, якими керується робота;
– вихід – вихідний результат діяльності;
– механізм – ресурси, що виконують роботу.

Провівши аналіз головних елементів системи для контекстної діаграми, що показана на рисунку 2.1, було сформовано такий перелік даних:

– входи: hsf, obj або stl файл, запит на збереження, параметри для редагування;
– управління – документація користувача;
– вихід – збережений файл;
– механізми – користувач, програма.

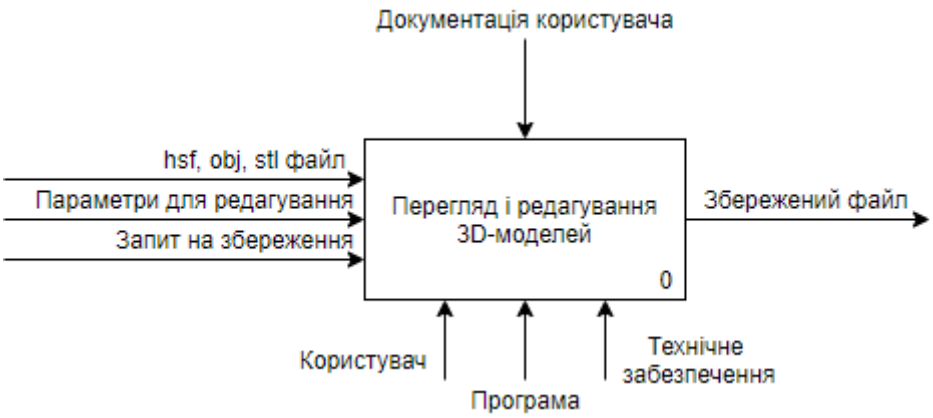


Рисунок 2.1 – Контекстна діаграма IDEF0

Зважаючи на те, що контекстна діаграма описує систему лише у загальному вигляді, з’являється потреба у її декомпозиції, що дозволить більш детально зобразити логіку послідовності робіт.

Розіб’ємо один загальний процес на три різні етапи: відкриття файлу, редагування, збереження. Отже, після проведення декомпозиції була створена діаграма, що наведена на рис. 2.2.

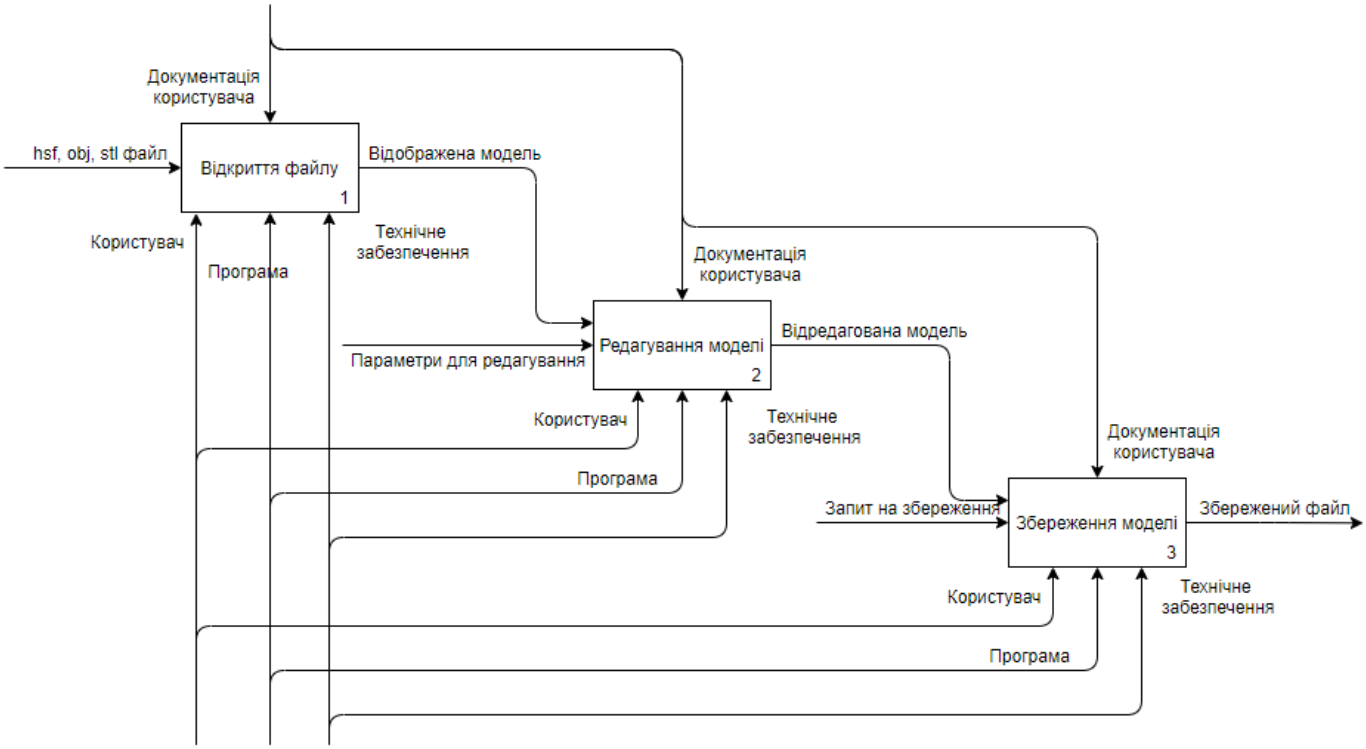


Рисунок 2.2 – Модель декомпозиції

2.2 Моделювання варіантів використання

Для описання дій, що система має можливість виконати, була створена Use-case діаграма. Вона складається з опису варіантів використання, акторів та взаємодії між ними.

Актором системи є просто користувач програми, адже програма не має можливості зареєструватися, тобто не потребує розмежування прав.

Артефактом системи є файл, який містить інформацію про 3D-модель, який може бути завантажений та знову записаний.

До виділених варіантів використання належать:

– ВВ 1 Завантаження файлу – завантаження файлу з файлової системи;
– ВВ 2 Маніпуляція камерою – можливість змінювати камеру за допомогою різних кнопок миші;
– ВВ 3 Виділення об’єктів – можливість змінювати поточний стан виділених об’єктів: додавати до виділення, видаляти, змінювати на протилежний тощо;
– ВВ 4 Редагування об’єктів – зміна характеристик виділених об’єктів за допомогою кнопок на панелі інструментів;
– ВВ 5 Збереження файлу – збереження відредагованої моделі у файл;
– ВВ 6 Перегляд «Про програму» – можливість переглянути інформацію про компанію та версію додатку.

Діаграма варіантів використання наведена на рис. 2.3.

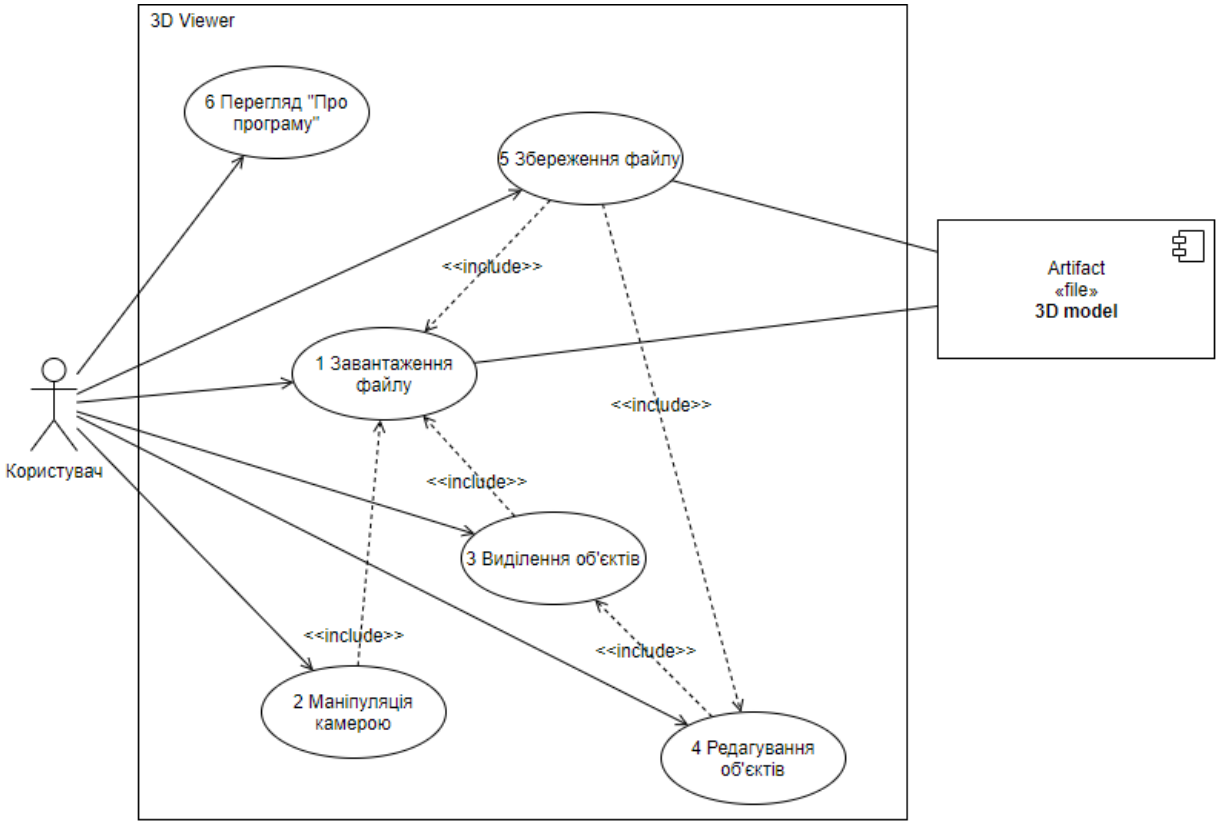


Рисунок 2.3 – Діаграма варіантів використання

3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1 Архітектура програмного додатку

Під час планування проекту було вирішено використовувати паттерн програмування MVVM, адже він добре підходить для розробки програм на C# з використанням системи WPF у командах, збільшує можливості повторного використання коду та значно спрощує роботу з розробкою інтерфейсу, підтримку програми та тестування [12]. Головна ідея цього паттерну полягає в тому, що відбувається розділення розробки представлення, тобто інтерфейсу, від бізнес-логіки [13].

Головними компонентами MVVM паттерну є:

– Представлення (View) – відповідає за візуальний інтерфейс, за допомогою якого користувач буде взаємодіяти з програмою;

– Модель-Представлення (View Model) – служить зв’язком між представленням та моделлю, слідкує за змінами в даних, а також опрацьовує логіку роботи представлення;

– Модель (Model) – є використовувані в програмі дані та бізнес-логіка [13].

Зв’язок цих компонентів графічно представлений на рис. 3.1.

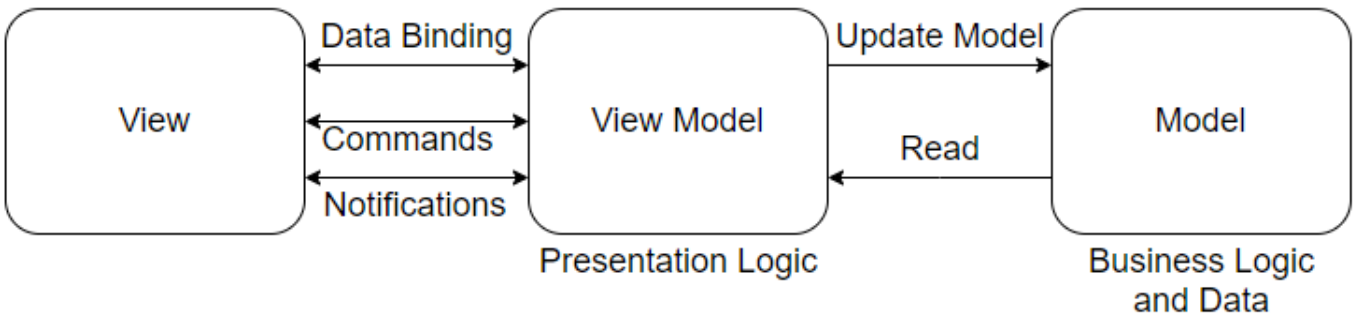


Рисунок 3.1 – Відношення складових паттерну MVVM

Таким чином для головного вікна програми, а також для кожної операції редагування, які реалізовані у вигляді користувацьких UserControl, були створені по окремій трійці компонентів.

Модель вікна програми є головним компонентом програми, що пов’язує всі інші його елементи. Крім цього вона реалізує паттерн одинак (Singleton), який гарантує, що деякий клас може мати лише один екземпляр, а також надає глобальний простий доступ до цього екземпляру [14]. Таким чином окрім того, що модель головного вікна присутня в відповідній моделі-представлення, до неї також будуть мати доступ моделі операцій для редагування 3D-об’єктів.

Відношення головних компонентів програми наведено на діаграмі на рисунку 3.2.

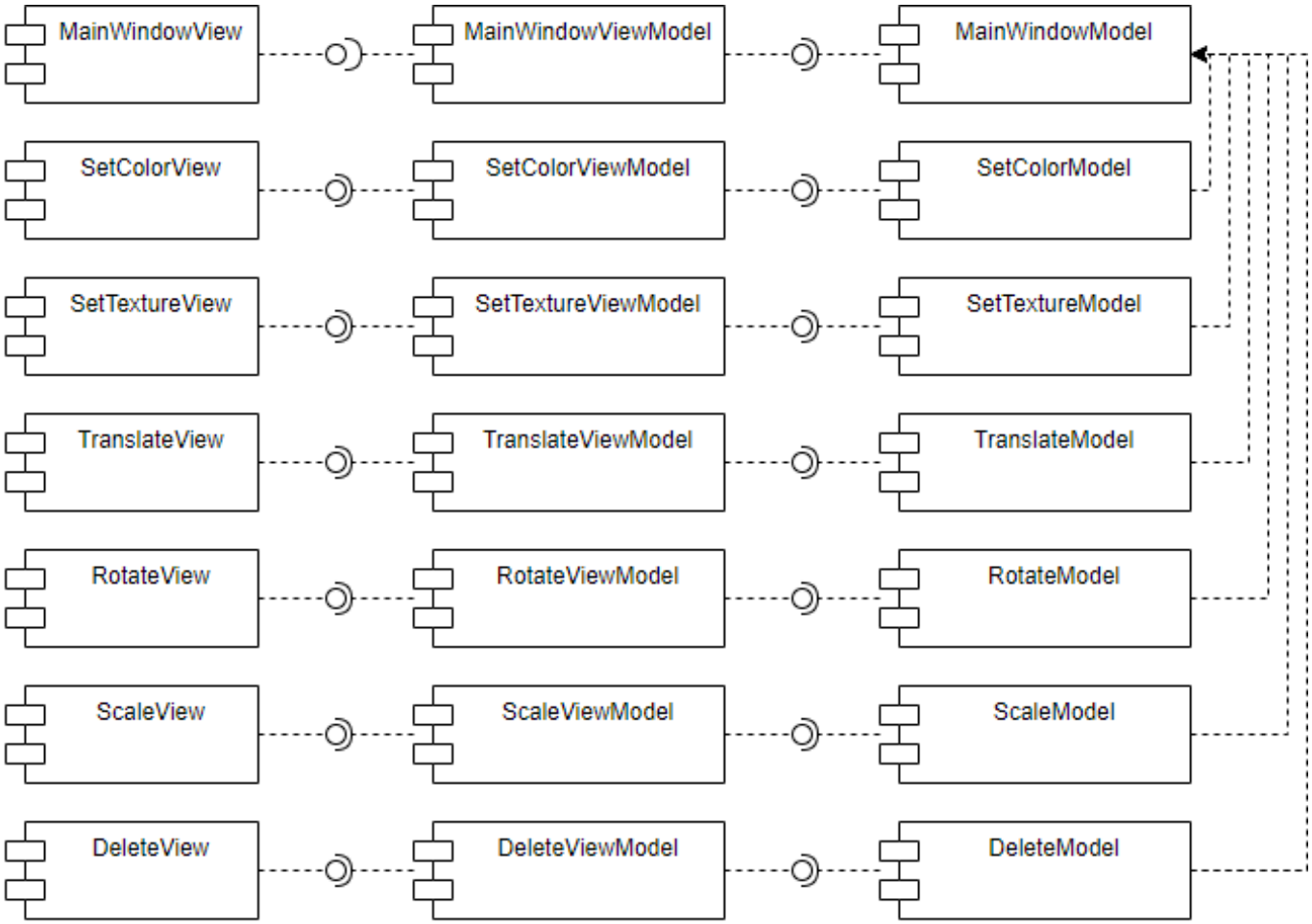


Рисунок 3.2 – Діаграма компонентів

3.2 Програмна реалізація

3.2.1 Створення вікна програми та полотна

Розпочинається розробка програми з того, що спершу реалізовується представлення головного вікна MainWindow за допомогою декларативної мови розмітки XAML згідно макету наведеного в Технічному завданні. Створюються панелі меню, інструментів та логування за допомогою стандартних інструментів WPF, а для панелі перегляду резервується місце тегом Border, якому прив’язується полотно (canvas) з бібліотеки HPS. На рис. 3.3 представлено процес редагування інтерфейсу програми за допомогою конструктору.

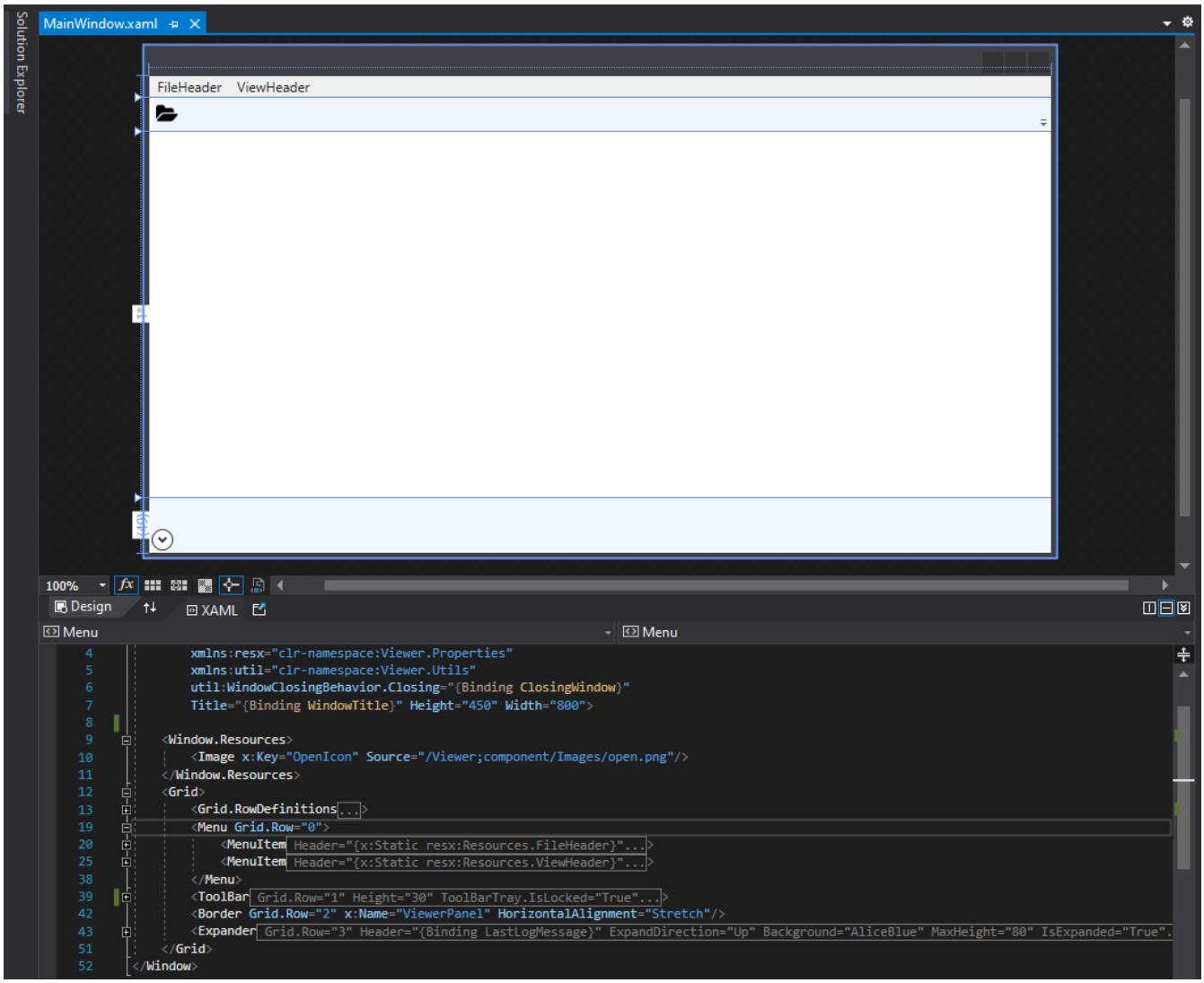


Рисунок 3.3 – Редагування інтерфейсу в конструкторі

Для того, щоб на місці тегу Border відображалось полотно для перегляду 3D-об’єктів необхідно перед відкриттям головного вікна створити це полотно під назвою ViewerHost, яке буде зберігатись у моделі ViewerModel. Тому спочатку створюється екземпляр ViewerModel, що зберігається у ViewerViewModel. Потім ініціалізується вікно MainWindow, якому присвоюється контекст ViewerViewModel, а користувацькому елементу Border присвоюється створений ViewerHost.

ViewerHost відповідає за рендеринг зображення, а також реєстрацію подій миші та клавіатури. Він наслідується від System.Windows.Controls.Image, а джерелом (Source) картинки слугує властивість HPS.D3DImageCanvas, який використовує рушій DirectX11.

Для того щоб в майбутньому можливо було обробляти події миші потрібно перевизначити такі події як OnMouseDown, OnMouseUp, OnMouseMove, та OnMouseWheel. В них визначається положення миші в пікселях і конвертується в координати віконного простору (window space), які клавіші були активовані та яка саме дія відбулась. На основі цих даних було побудовано HPS.MouseEvent, а потім вносено (injected) в диспетчер подій. Код однієї з подій наведений на рис. 3.4.

ViewerModel після створення HPS.D3DImageCanvas може виглядати приблизно таким чином як наведено на рис. 3.5.

```
0 references | AnastasiiaShadow, 47 days ago | 1 author, 2 changes | 1 work item
protected override void OnMouseWheel(MouseWheelEventArgs e)
{
    base.OnMouseWheel(e);
    Point location = e.GetPosition(this);
    HPS.MouseEvent mouseEvent = BuildMouseEvent(HPS.MouseEvent.Action.Scroll, new HPS.MouseButtons(), (int)location.X, (int)location.Y, e.Delta, 0);
    Canvas.GetWindowKey().GetEventDispatcher().InjectEvent(mouseEvent);
}
#endregion

#region Event Helpers
4 references | AnastasiiaShadow, 47 days ago | 1 author, 1 change | 1 work item
private HPS.MouseEvent BuildMouseEvent(HPS.MouseEvent.Action action, HPS.MouseButtons buttons, int x, int y, float scalar, ulong click_count)
{
    // Convert location to window space.
    HPS.Point point = new HPS.Point(x, y, 0);
    Canvas.GetWindowKey().ConvertCoordinate(HPS.Coordinate.Space.Pixel, point, HPS.Coordinate.Space.Window, out point);
    HPS.WindowPoint windowPoint = new HPS.WindowPoint(point.x, point.y, point.z);

    if(action == HPS.MouseEvent.Action.Scroll)
    {
        return new HPS.MouseEvent(action, scalar, windowPoint, MapModifierKeys());
    }
    else
    {
        return new HPS.MouseEvent(action, windowPoint, buttons, MapModifierKeys(), click_count);
    }
}
```

Рисунок 3.4 – Перевизначення події миші

```
using HPS;
using Viewer.Views;

namespace Viewer.Models
{
    8 references | AnastasiiaShadow, 47 days ago | 1 author, 2 changes | 1 work item
    public class ViewerModel
    {
        private static ViewerModel instance;

        1 reference | AnastasiiaShadow, 47 days ago | 1 author, 2 changes | 1 work item
        public ViewerModel()
        {
            ViewerHost = new ViewerHost();
            View = Canvas.GetFrontView();
            Model model = Factory.CreateModel();
            View.AttachModel(model);
            View.Update();
        }

        #region Properties
        1 reference | AnastasiiaShadow, 48 days ago | 1 author, 1 change | 1 work item
        public static ViewerModel Instance...

        3 references | AnastasiiaShadow, 48 days ago | 1 author, 1 change | 1 work item
        public ViewerHost ViewerHost { get; private set; }

        1 reference | AnastasiiaShadow, 47 days ago | 1 author, 1 change | 1 work item
        public D3DImageCanvas Canvas => ViewerHost.Canvas;

        3 references | AnastasiiaShadow, 48 days ago | 1 author, 1 change | 1 work item
        private View View { get; set; }
        #endregion
    }
}
```

Рисунок 3.5 – ViewerModel із властивістю D3DImageCanvas

Для того, аби далі було більш зрозуміло, як працює D3DImageCanvas, потрібно описати ієрархію видів (view hierarchy), що реалізована в Hoops Visualize. В цій ієрархії знаходиться 4 основних класи, які існують як вкладені один в одного контейнери:

- HPS.Canvas (полотно) є аналогом клієнтської області графічного інтерфейсу програми;
- HPS.Layout (макет) – визначає яким чином види розміщені у вікні;
- HPS.View (вид) – представляє вид камери на модель;
- HPS.Model (модель) – частина сцени, що відповідає за логічну 3D-модель [15].

Кожен контейнер знаходиться всередині свого батька. Усі відносини є один до одного за винятком макета – HPS.Layout, який має можливість містити кілька видів (рис. 3.6).

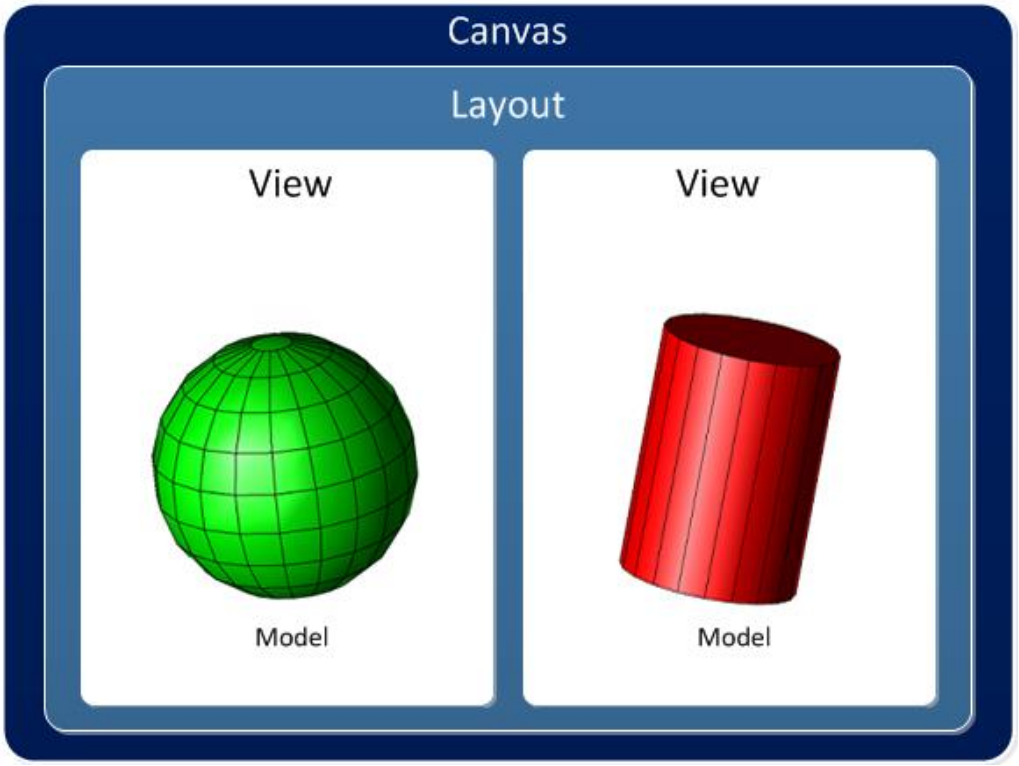


Рисунок 3.6 – Приклад ієрархії сцени з декількома видами

Таким чином, за допомогою властивості Canvas, що міститься у ViewerModel, можна отримати доступ до будь-якої частини сцени. Для того, щоб відображалось полотно потрібно його оновити на одному з рівнів ієрархії.

Також для того, щоб була можливість додавати повідомлення та попередження до панелі логування було використано елемент Expander та створено у ViewerViewModel дві властивості для зв’язування відображуваного тексту. В ViewerModel були реалізовані методи для додавання повідомлень.

Вигляд вікна з відображеним полотном представлений на рис. 3.7.

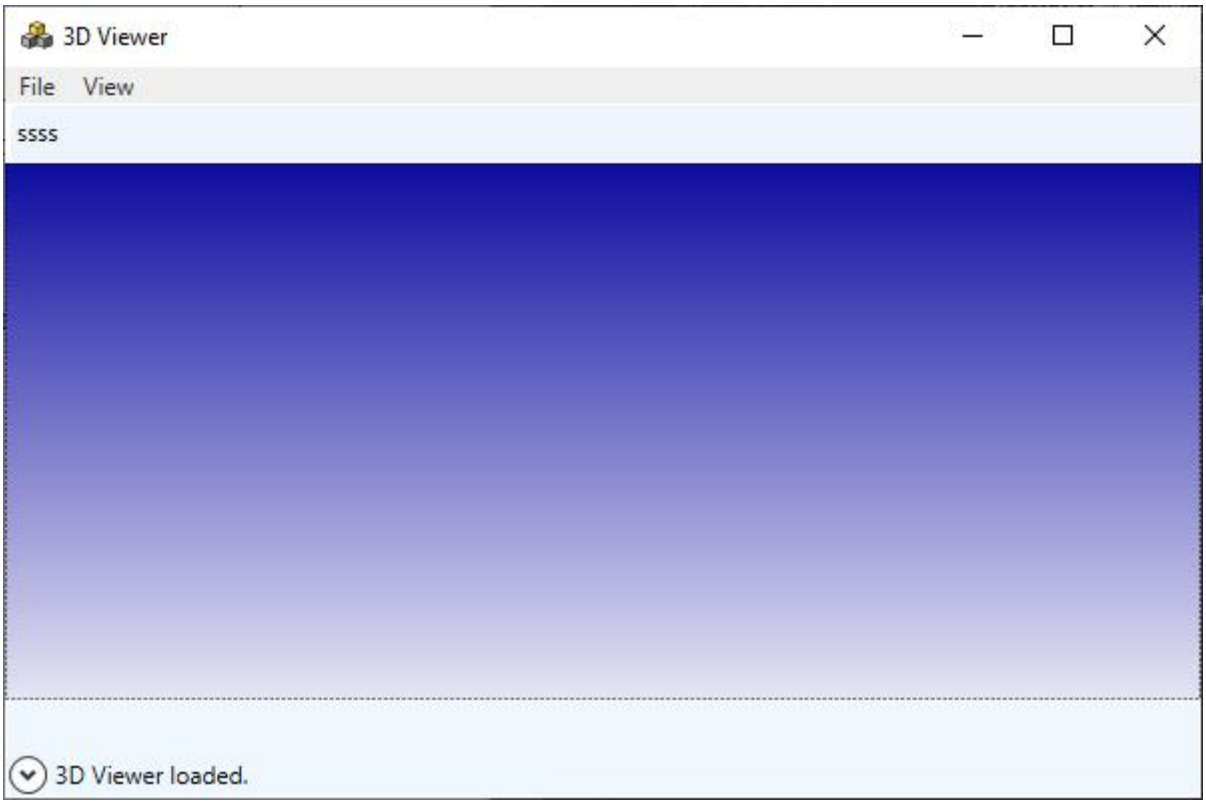


Рисунок 3.7 – Вигляд вікна програми

3.2.2 Реалізація можливості відкриття програми

До панелей інструментів та меню було додано кнопку відкриття документу, а також створено KeyBinding для того, щоб можна було відкривати файли за допомогою комбінації клавіш.

Для того, щоб підтримувати паттерн MVVM бажано всюди, де це можливо, замість подій використовувати команди. Для цього було створено допоміжний клас RelayCommand, що буде наслідувати ICommand, та реалізовувати два основних метода CanExecute та Execute.

Таким чином, тепер є можливість викликати метод відкриття за допомогою команд. В коді XAML це виглядає таким чином:

<Button Command="{Binding OpenFile}" ToolTip="Open Document (Ctrl+N)" Content="{StaticResource OpenIcon}"/>

У ViewModel було створено об’єкт RelayCommand, якому передано делегат, що виконує команду Open (рис. 3.8).

```
84      private RelayCommand openFile;
        0 references | AnastasiiaShadow, 31 days ago | 1 author, 1 change | 1 work item
85      public RelayCommand OpenFile
86      {
87          get
88          {
89              if(openFile == null)
90              {
91                  openFile = new RelayCommand(o => Open());
92              }
93              return openFile;
94          }
95      }
        1 reference | AnastasiiaShadow, 31 days ago | 1 author, 1 change | 1 work item
96      private void Open()
97      {
98          viewer.OpenDocument();
99          WindowTitle = DocumentTitle;
100         OnPropertyChanged(nameof(IsDocumentOpen));
101     }
102
        0 references | AnastasiiaShadow, 31 days ago | 1 author, 1 change | 1 work item
103     public RelayCommand ClosingWindow
104     {
105         get
106         {
107             if(closingWindow == null)
108             {
109                 closingWindow = new RelayCommand(o => CloseWindow());
110             }
111             return closingWindow;
112         }
113     }
```

Рисунок 3.8 – Реалізація команди Open

Для зберігання інформації про поточно відкритий файл було створено клас Document, який також буде містити методи відкриття та збереження.

Для вибору файлу з диску було використано клас OpenFileDialog. Наявна можливість відкрити файли трьох типів: STL, OBJ та HSF. Після того, як користувач обрав файл, починається процес безпосереднього завантаження файлу. Для кожного нового файлу створюється новий екземпляр класу Model, в якому виділяється сегмент, що буде містити відкритий об’єкт. Сегменти – це вузли графу сцени, вони можуть зберігати геометрію, інші сегменти, атрибути тощо. Даний сегмент вказується як початковий вузол, в який буде завантажуватись модель. Далі викликається метод Import. В разі спроби завантажити файл HSF також проводиться спроба прочитати камеру за замовчування для файлу. Код для завантаження

наведено на рис. 3.9. В разі помилки при відкритті, View повертається до свого початкового вигляду.

```
1 reference | AnastasiiaShadow, 31 days ago | 1 author, 1 change | 1 work item
public static Document Import(string filePath, SegmentKey geometrySegment, out CameraKit defaultCamera)
{
    defaultCamera = null;
    Enum.TryParse(Path.GetExtension(filePath).TrimStart('.'), true, out Format fileExtension);
    IONotifier importNotifier;
    if(fileExtension == Format.OBJ)
    {
        OBJ.ImportOptionsKit objOptionsKit = new OBJ.ImportOptionsKit();
        objOptionsKit.SetSegment(geometrySegment);
        importNotifier = OBJ.File.Import(filePath, objOptionsKit);
    }
    else if(fileExtension == Format.STL)
    {
        STL.ImportOptionsKit stlOptionsKit = new STL.ImportOptionsKit();
        stlOptionsKit.SetSegment(geometrySegment);
        importNotifier = STL.File.Import(filePath, stlOptionsKit);
    }
    else
    {
        HPS.Stream.ImportOptionsKit hsfOptionsKit = new HPS.Stream.ImportOptionsKit();
        hsfOptionsKit.SetSegment(geometrySegment);
        HPS.Stream.ImportNotifier hpsImportNotifier = HPS.Stream.File.Import(filePath, hsfOptionsKit);
        HPS.Stream.ImportResultsKit importResults = hpsImportNotifier.GetResults();
        importResults.ShowDefaultCamera(out defaultCamera);
        importNotifier = hpsImportNotifier;
    }

    importNotifier.Wait();
    if(importNotifier.Status() == IOResult.Failure || importNotifier.Status() == IOResult.UnableToOpenFile)
    {
        throw new HPS.IOException("Can't open the file.", IOResult.Canceled);
    }

    return new Document(filePath);
}
```

Рисунок 3.9 – Код завантаження файлу

Також проводяться деякі невеликі налаштування полотна для більш приємного перегляду моделі. В контейнер View додано джерело світла, відображемо навігаційний куб та трійку осей, обрано тип освітлення тощо.

Знову ж таки, щоб модель відобразилася, необхідно оновити один з рівнів ієрархії. В даному випадку, а також в подальшому оновлення відбувається на рівні View, адже він відповідає за вид камери на модель, а рівнем нижче ніяких змін до ієрархії проводитися не буде.

Відкритий файл розширення HSF показано на рис. 3.10. На ньому також крім моделі можна побачити відображені куб та осі, а на панелі логування є повідомлення про те, що файл був успішно відкритий.

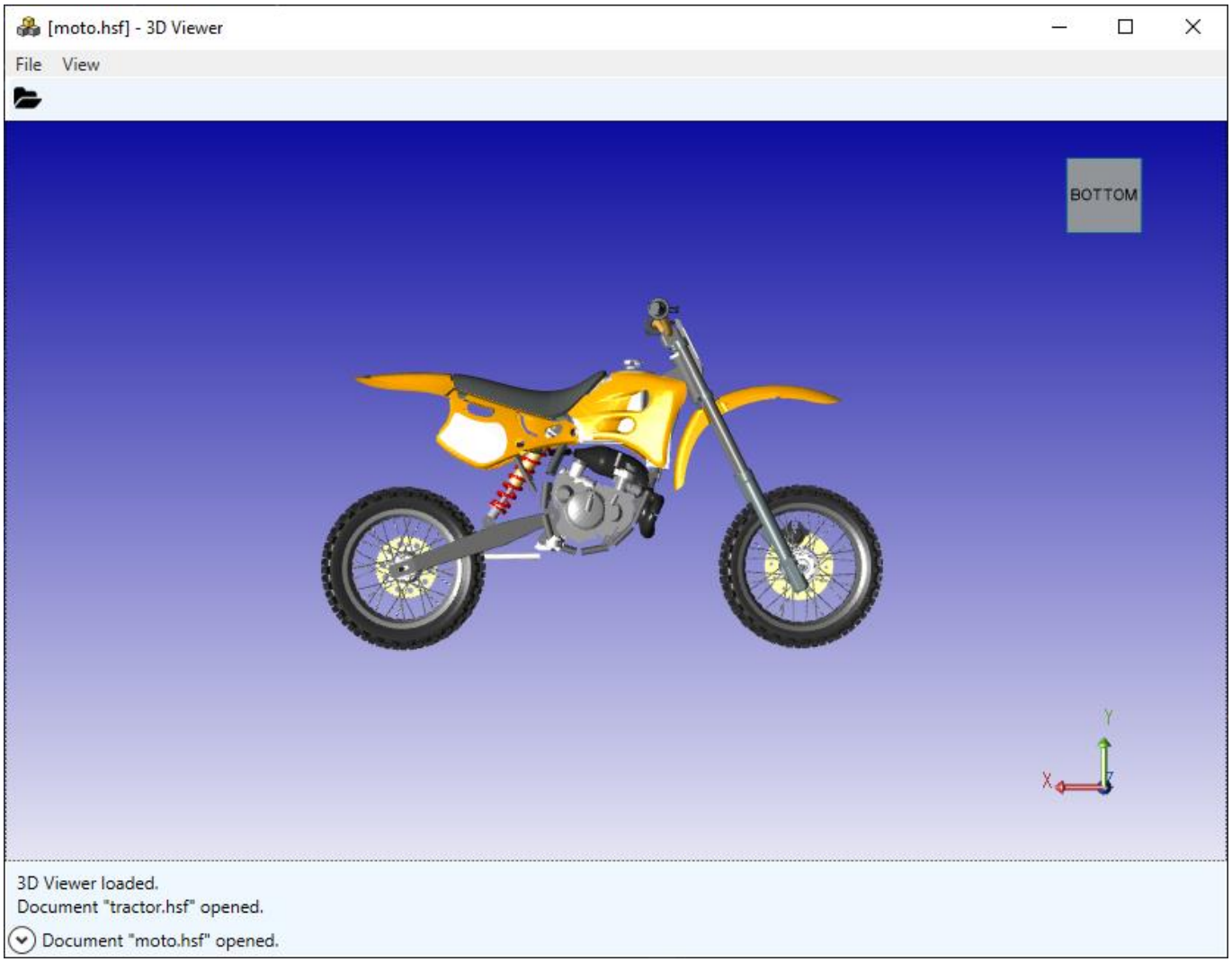


Рисунок 3.10 – Відображена модель у програмі

3.2.3 Створення операцій миші для роботи з камерою

В Hoops Visualize оператори завжди асоціюються з View екземпляром. Тобто для того, щоб активувати конкретний оператор, необхідно закріпити його за допомогою OperatorControl конкретного View. Наприклад:

```
PanOperator panOperator = new PanOperator(MouseButtons.ButtonLeft());
view.GetOperatorControl().Push(panOperator);
```

Після виконання цього коду повинна з’явитись можливість переміщувати камеру натискаючи ліву клавішу миші. Таким же чином, щоб від’єднати оператор, це можна зробити методом Pop класу OperatorControl, або за допомогою panOperator.DetachView().

Для роботи з камерою було створено такі оператори:

- Pan – по натисканню правої клавіші миші;
- Orbit – середня клавіша;
- Zoom – обертання колеса миші.

Для взаємодії користувача Hoops Visualize пропонує досить широкий набір попередньо побудованих операторів, враховуючи і ті, що можуть бути використані в даній роботі, а саме PanOperator, OrbitOperator / RelativeOrbitOperator та MouseWheelOperator / ZoomOperator.

Проте оператори обертання не задовольняють вимогам, тому що обертання проводиться відносно початкового положення миші або поточної цілі камери. Тому виникає необхідність створити CustomOrbitOperator, що буде обертати камеру відносно центру обмежувальної рамки (bounding box) моделі.

Також не задовольняють вимоги оператори масштабування. ZoomOperator – тому що він працює лише по натисканню правої клавіші миші, а MouseWheelOperator – тому що при масштабуванні він обертає камеру.

Таким чином, єдиним оператором, який буде використано в програмі, є PanOperator.

При створенні CustomOrbitOperator унаслідуємо його від HPS.Operator. Для його повноцінної роботи необхідно перевизначити методи OnMouseDown, OnMouseMove та OnMouseUp. При натисканні середньої клавіші, тобто на OnMouseDown, активується оператор, знаходиться центр поточного bounding box, а також положення миші.

При виконанні OnMouseMove камера переноситься таким чином, щоб центр обертання співпав з центром вінка, потім рахується відстань, що пройшла миша, та кути обертання відносно осі обертання. За цими кутами проводиться обертання. Потім камера переноситься за таким же вектором, що і спочатку, проте зі знаком мінус. Код методу наведено на рис. 3.11.

При виконанні OnMouseUp оператор деактивується.

```
1 reference | AnastasiiaShadow, 31 days ago | 1 author, 1 change | 1 work item
77  private bool RelativeOrbit(WindowPoint inLocation, KeyPath eventPath)
78  {
79      viewKey.ShowCamera(out CameraKit camera);
80      camera.ShowTarget(out Point target);
81      eventPath.ConvertCoordinate(Coordinate.Space.World, centerOfRotation, Coordinate.Space.Camera, out Point convertedCenterOfRotation);
82      eventPath.ConvertCoordinate(Coordinate.Space.World, target, Coordinate.Space.Camera, out Point convertedCameraTarget);
83
84      Vector dollyDelta = convertedCenterOfRotation - convertedCameraTarget;
85      WindowPoint newPoint = inLocation;
86      OperatorUtility.ScreenToSphereMousePoint(newPoint, out Vector newSpherePosition);
87      Vector rotationAxis = startSpherePosition.Cross(newSpherePosition);
88      Vector mouseMoveVector = newPoint - startPoint;
89      float distance = (float)mouseMoveVector.Length() * ORBIT_INTENSITY_MAGNIFIER;
90      float tolerance = 0.0000001f;
91      bool isRotationChanged = Math.Abs(rotationAxis.x) > tolerance
92          || Math.Abs(rotationAxis.y) > tolerance
93          || Math.Abs(rotationAxis.z) > tolerance;
94
95      if(isRotationChanged)
96      {
97          if(viewKey.GetDrawingAttributeControl().ShowWorldHandedness(out Drawing.Handedness worldHandedness)
98              && worldHandedness == Drawing.Handedness.Right)
99          {
00              rotationAxis.y *= -1;
01              rotationAxis.z *= -1;
02          }
03          rotationAxis.Normalize();
04
05          float angle = OperatorUtility.CalculateAngleOrbitOnPlane(rotationAxis, OperatorUtility.ProjectedPlane.Plane_YZ);
06          float panPhi = rotationAxis.x < 0 ? -angle * distance : angle * distance;
07          angle = OperatorUtility.CalculateAngleOrbitOnPlane(rotationAxis, OperatorUtility.ProjectedPlane.Plane_XZ);
08          float panTheta = rotationAxis.y < 0 ? angle * distance : -angle * distance;
09          angle = OperatorUtility.CalculateAngleOrbitOnPlane(rotationAxis, OperatorUtility.ProjectedPlane.Plane_XY);
10          float rollTheta = rotationAxis.z < 0 ? angle * distance : -angle * distance;
11
12          camera.Dolly(dollyDelta.x, dollyDelta.y, dollyDelta.z);
13          camera.Orbit(panTheta, panPhi);
14          camera.Roll(rollTheta);
15          camera.Dolly(-dollyDelta.x, -dollyDelta.y, -dollyDelta.z);
16          viewKey.SetCamera(camera);
17          GetAttachedView().Update();
18      }
19
20      startSpherePosition = newSpherePosition;
21      startPoint = newPoint;
22      return true;
23  }
```

Рисунок 3.11 – Метод обертання камери

При створенні оператору масштабування важливо установити межі, за які масштабування буде неможливим. Це можна зробити отримавши bounding box поточної моделі, визначити довжину її діагоналі та помножити на деякі коефіцієнти (рис. 3.12).

```
2 references | AnastasiiaShadow, 31 days ago | 1 author, 1 change | 1 work item
101 private void UpdateZoomLimit()
102 {
103     view.GetAttachedModel().GetSegmentKey().ShowBounding(out BoundingKit modelBounding);
104     modelBounding.ShowVolume(out SimpleSphere _, out SimpleCuboid boundingCuboid);
105     float boundingDiagonalLength = (float)(boundingCuboid.max - boundingCuboid.min).Length(),
106         minZoomLimitCoefficient = 0.001f,
107         maxZoomLimitCoefficient = 200;
108     minZoomLimit = boundingDiagonalLength * minZoomLimitCoefficient;
109     maxZoomLimit = boundingDiagonalLength * maxZoomLimitCoefficient;
110 }
```

Рисунок 3.12 – Встановлення меж масштабування

При обертанні колеса миші, розраховуються нові значення позиції, цілі, та поля видимості камери. Ціль рахується відносно того, чи знаходиться якась геометрія під курсором миші. Якщо ні, то залишається початкова ціль. Якщо так, то переноситься в сторону курсора.

Ширина та висота поля множиться на співвідношення довжин векторів напрямку попередньої та нової камер (рис. 3.13).

```
1 reference | AnastasiiaShadow, 31 days ago | 1 author, 1 change | 1 work item
135     private CameraField ComputeNewField(Point newTarget, CameraKit oldCamera)
136     {
137         CameraField field;
138         oldCamera.ShowTarget(out Point target);
139         oldCamera.ShowPosition(out Point position);
140         oldCamera.ShowField(out field.Width, out field.Height);
141
142         Vector oldDirection = target - position,
143             newDirection = new Vector(newTarget) - new Vector(position);
144         double oldLength = oldDirection.Length(),
145             newLength = newDirection.Length();
146         float ratio = (float)(newLength / oldLength);
147         field *= ratio;
148         return field;
149     }
```

Рисунок 3.13 – Розрахунок поля видимості

Якщо розміри поля занадто великі чи малі, то процес масштабування переривається. В іншому разі проводиться його масштабування.

Останнім кроком розраховується позиція камери після чого зміни застосовуються.

Таким чином, було використано два створених оператори та один стандартний. Для того, щоб почати їх використовувати, при відкритті моделі проводиться їх активація (рис. 3.14).

```
29                  view.GetOperatorControl().
30                      Push(new CustomOrbitOperator(MouseButtons.ButtonMiddle())).
31                      Push(new PanOperator(MouseButtons.ButtonRight())).
32                      Push(new CustomZoomOperator());
```

Рисунок 3.14 – Активація операторів

Результат використання операторів наведений на рис. 3.15, де камера була віддалена, переміщена праворуч та обернута.

Рисунок 3.15 – Результат роботи операцій

3.2.4 Відображення сітки

Сітка додається до сегменту, що знаходиться на View. Вона прораховується беручи до уваги розміри моделі таким чином, щоб помістити деяку кількість квадратів відносно центру в обидві сторони та при цьому, щоб вона мала розміри дещо більші за розміри моделі. Також відображаються дві осі симетрії.

Додавання до сегментів будь якої геометрії проводиться за допомогою спеціальних кітів, наприклад для лінії це LineKit, для сітки – GridKit.

Код додавання сітки до сегменту наведено на рис. 3.16, а вигляд вікна на рис. 3.17.

```
142             GridKit gridKit = new GridKit();
143             gridKit.SetFirstPoint(firstPoint).SetOrigin(origin).SetSecondPoint(secondPoint).
144                 SetFirstCount(-halfGridSquaresNumber).SetSecondCount(-halfGridSquaresNumber).SetPriority(1);
145             GridSegment.InsertGrid(gridKit);
```

Рисунок 3.16 – Додавання сітки

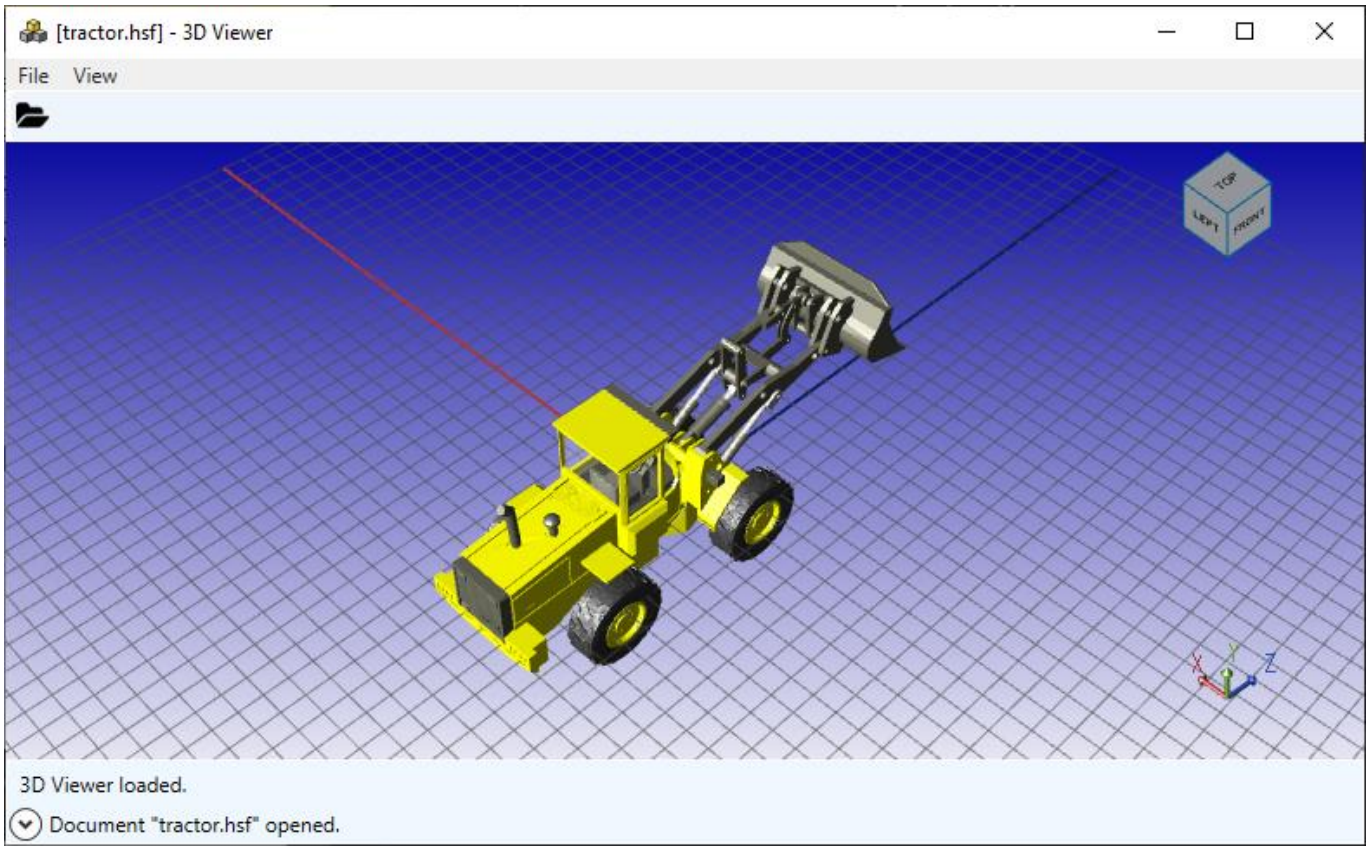


Рисунок 3.17 – Вигляд сітки

3.2.5 Виділення об’єктів

Реалізація виділення об’єктів є напевно одним з найбільш складних етапів розробки, який можна поділити на такі стадії:

- створення структур для збереження одного виділеного об’єкту;
- створення колекції, що міститиме поточне виділення;
- реалізація оператору виділення;
- реалізація методів для виділення та його зміни.

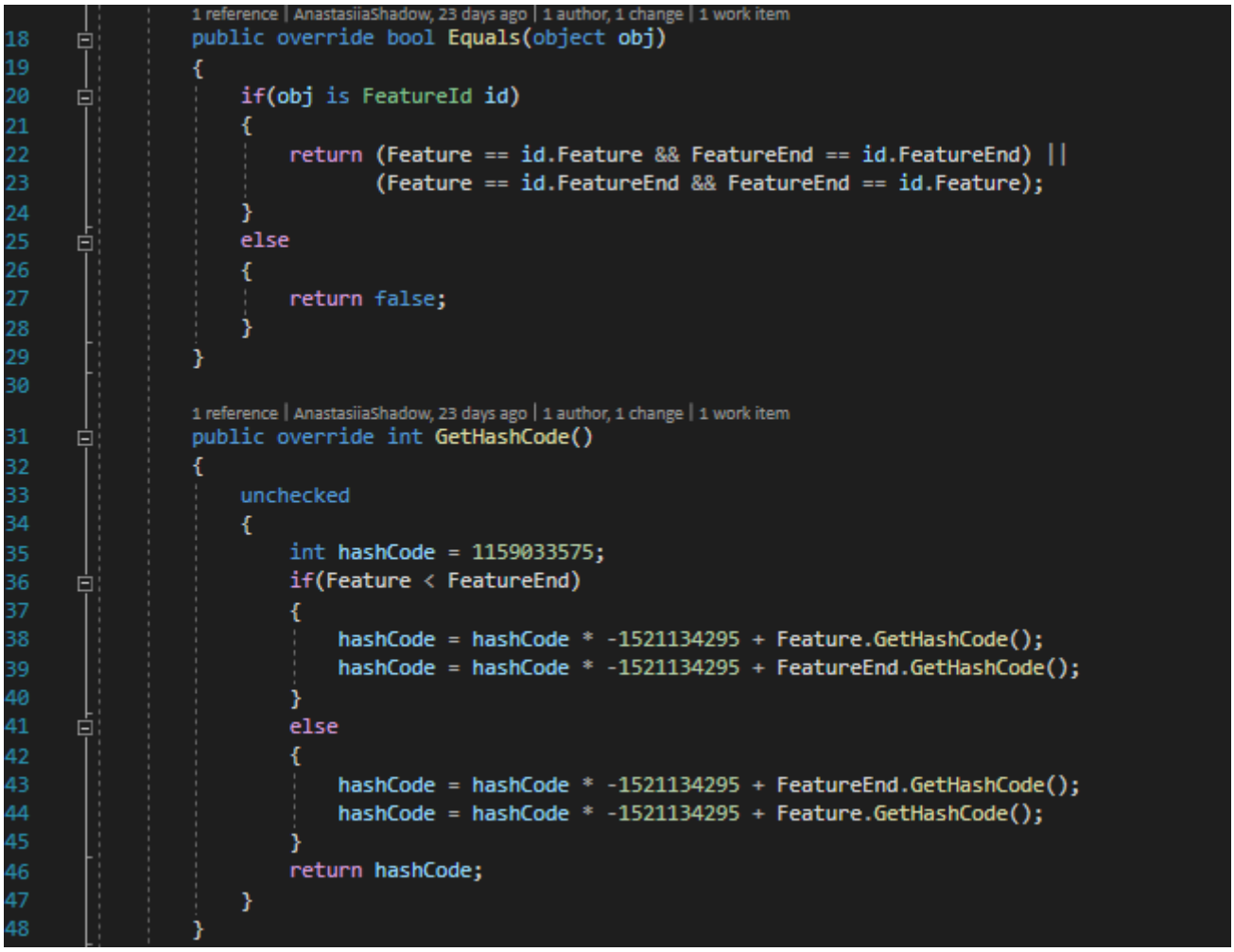
3.2.6 Структури для збереження виділення

Для того, щоб описати виділення було створено три структури: EntityId, FeatureId та PersistentId. Призначення структур та їх властивостей знаходиться в табл. 3.1.

Таблиця 3.1 – Описання структур для збереження виділення та їх властивостей

Назва структури	Її призначення	Властивість	Тип властивості	Опис властивості
EntityId	Визначає в якому шелі та на якому рівні об’єкт має виділення.	Key	ShellKey	Ключ по якому можна унікально ідентифікувати шел.
		FeatureType	FeatureType (enum)	Рівень виділення.
FeatureId	Визначає вершину, ребро чи грань в шелі.	Feature	ulong	Номер елемента (вершини / грані) по порядку за яким він ідентифікується.
		FeatureEnd	ulong	Використовується лише при виділенні ребер для ідентифікації другої вершини ребра.
PersistentId	Визначає усі об’єкти певного типу в одному шелі.	EntityId	EntityId	Ідентифікатор шелу з типом виділення.
		Features	HashSet <FeatureId>	Колекція з виділеними елементами шелу. Має значення null якщо виділення проводиться на рівні шелу.

FeatureId також перевизначає методи GetHashCode та Equals для того, щоб при додаванні нового елементу FeatureId до HashSet ребра зі значеннями вершин (3, 1) та (1, 3) мали однаковий хешкод та не могли повторюватись в колекції. Код методів наведений на рис. 3.18.



```
1 reference | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
public override bool Equals(object obj)
{
    if(obj is FeatureId id)
    {
        return (Feature == id.Feature && FeatureEnd == id.FeatureEnd) ||
               (Feature == id.FeatureEnd && FeatureEnd == id.Feature);
    }
    else
    {
        return false;
    }
}

1 reference | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
public override int GetHashCode()
{
    unchecked
    {
        int hashCode = 1159033575;
        if(Feature < FeatureEnd)
        {
            hashCode = hashCode * -1521134295 + Feature.GetHashCode();
            hashCode = hashCode * -1521134295 + FeatureEnd.GetHashCode();
        }
        else
        {
            hashCode = hashCode * -1521134295 + FeatureEnd.GetHashCode();
            hashCode = hashCode * -1521134295 + Feature.GetHashCode();
        }
        return hashCode;
    }
}
```

Рисунок 3.18 – Методи GetHashCode та Equals для структури FeatureId

3.2.7 Колекція поточного виділення

Для збереження поточного виділення була створена колекція SelectionSet. Вона в собі містить приватне поле типу Dictionary<EntityId, HashSet<FeatureId>>, таким чином для унікального EntityId, можна додавати безліч елементів одного типу без повторень.

SelectionSet реалізує інтерфейс ISet, код якого наведений на рис. 3.19. Призначення методів описані в табл. 3.2.

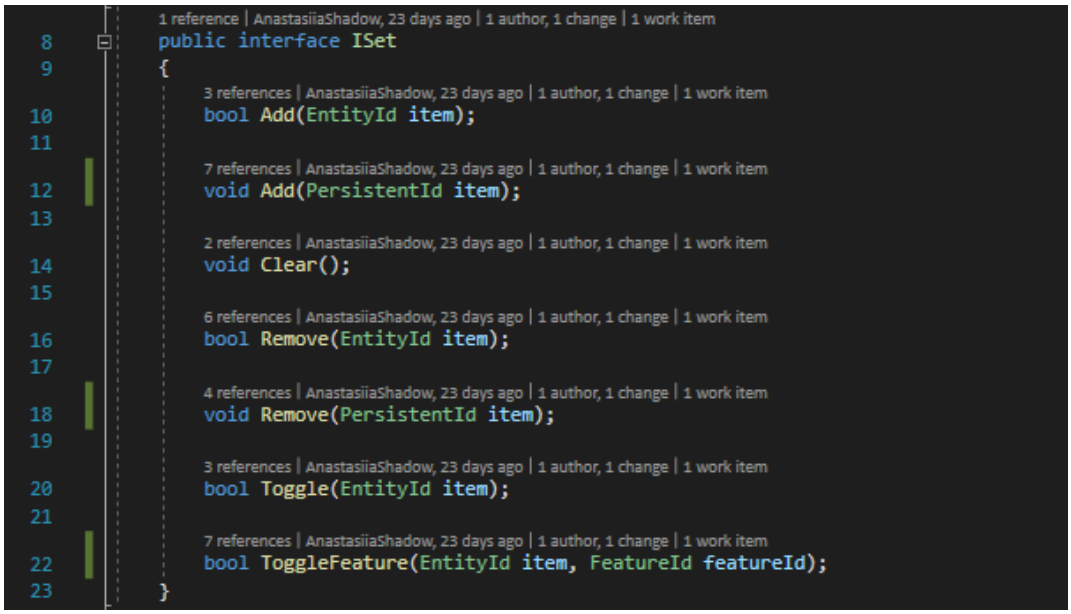


Рисунок 3.19 – Інтерфейс ISet

Таблиця 3.2 – Методи ISet та їх призначення

Сигнатура методу	Призначення методу
bool Add(EntityId item)	Додати до колекції тільки ключ, а значення залишити null. Повертає true якщо об’єкт було додано.
void Add(PersistentId item)	Додати до колекції новий об’єкт.
void Clear()	Видалити всі об’єкти з колекції.
bool Remove(EntityId item)	Видалити з колекції об’єкт за певним ключем. Повертає true якщо об’єкт було видалено.
void Remove(PersistentId item)	Видалити з колекції елементи певного шелу. Якщо в шелі ще існують виділення такого типу, то ключ з елементами залишити в колекції, в іншому разі видалити ключ.
bool Toggle(EntityId item)	Зміна поточного стану шелу на протилежне. Значення true відповідає за результуючий стан виділення шелу.
bool ToggleFeature(EntityId item, FeatureId featureId)	Зміна поточного стану елементу шелу на протилежне. Значення true відповідає за результуючий стан виділення елементу шелу.

Окрім описаних методів, SelectionSet також реалізує:

– властивість Count – кількість об’єктів, враховуючи елементи шелів;

– метод bool Exists(EntityId key) – чи існує ключ в колекції;
– метод List<PersistentId> GetByType(FeatureType featureType) – отримати усі об’єкти певного типу виділення.

3.2.8 Оператор виділення

Під час реалізації було розроблено такі варіанти зміни виділення:

– одиночне натискання лівої кнопки миші – виділення одного об’єкту;
– ліва кнопка + перетягнути – виділення багатьох об’єктів;
– одиночне натискання лівої кнопки миші + Ctrl – зміна поточного стану об'єкту виділення на протилежне;
– ліва кнопка + перетягнути + Ctrl – додати до виділення;
– ліва кнопка + перетягнути + Shift – прибрати з поточного виділення.

Виділення можливе або на рівні шелу, або на рівні окремих елементів, одночасно чи по одинці.

Створення даного оператору схоже на попередні. При натисканні ліву клавішу, запам’ятовується початкове положення миші. Якщо миша здвигнулася, то активується виділення областю та малюється прямокутник виділення. При відпусканні клавіші проводиться знаходження виділених об’єктів, код якого наведено на рис. 3.20. Далі результати, що були отримані, передаються у відповідні методи SelectionManager, який реалізується наступним кроком.

```
146             SelectionResults selectionResults;
147             if(operatorMoved)
148             {
149                 temporaryGeometry.Flush(Search.Type.Geometry, Search.Space.Subsegments);
150                 selectionOptions.SetRelatedLimit(selectedObjectLimit).SetInternalLimit(selectedObjectLimit);
151                 windowKey.GetSelectionControl().SelectByArea(new Rectangle(2, new Point[] { startPoint, inState }),
152                     selectionOptions, out selectionResults);
153             }
154             else
155             {
156                 selectionOptions.SetRelatedLimit(0);
157                 windowKey.GetSelectionControl().SelectByPoint(inState, selectionOptions, out selectionResults);
158             }
```

Рисунок 3.20 – Виділення об’єктів

3.2.9 Створення SelectionManager та методів виділення

Об’єкт класу SelectionManager міститься в ViewerModel та створюється в його конструкторі.

Важливими його полями є об’єкт SelectionSet, тобто поточне виділення, а також властивість Dictionary<FeatureType, bool> Filters, що відповідає за обраний на даний момент рівень виділення.

Його методи наведені рис. 3.21.

```
5 references | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
11  public class SelectionManager
12  {
13      Private Fields
19
        36 references | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
20      public Dictionary<FeatureType, bool> Filters { get; set; }
21
        1 reference | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
22      public SelectionManager(Canvas canvas, ViewerModel viewerModel)...
37
38      #region SelectionSet Methods
        1 reference | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
39      public void CreateSet()...
44
        2 references | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
45      public int GetSelectionCount()...
49
50      /// <summary> Remove all features of an entity from selection.
        2 references | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
54      public void RemoveFeatures(EntityId item)...
63      #endregion
64
65      #region Highlight Methods
66      /// <summary> Highlight selected items and add them to the SelectionSet.
        2 references | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
70      public void HighlightSelected(SelectionResultsIterator selectionResults)...
121
122     /// <summary> Unhighlight selected items and remove them from the SelectionSet.
        1 reference | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
126     public void UnhighlightSelected(SelectionResultsIterator selectionResults)...
177
178     /// <summary> Change highlight state of selected item to opposite.
        1 reference | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
182     public void ToggleSelected(SelectionResultsIterator selectionResults)...
235
236     /// <summary> Highlight and add to SelectionSet all items of selected types
        1 reference | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
239     public void SelectAll()...
305
306     /// <summary> Clear SelectionSet and unhighlight all items.
        2 references | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
309     public void ClearAll()...
314     #endregion
315
316     /// <summary> Get if any of the selection filters are checked.
        2 references | AnastasiiaShadow, 23 days ago | 1 author, 1 change | 1 work item
320     public bool IsSelectionPossible()...
324 }
```

Рисунок 3.21 – Складові класу SelectionManager

Реалізація методів HighlightSelected, UnhighlightSelected та ToggleSelected полягає в тому, що за допомогою ітератору, який передається як параметр, відбуваться обхід усіх об’єктів, що були знайдені при останній операції миші. Якщо рівень виділення обраний FeatureType.Shell, то проводиться його додавання чи виділення з колекції. Якщо ж обраний інший рівень, то ми спочатку показуються які ж з елементів саме були виділені та проводиться вже їх обробка.

Результати роботи оператору та методів наведені на рис. 3.22 – 3.24

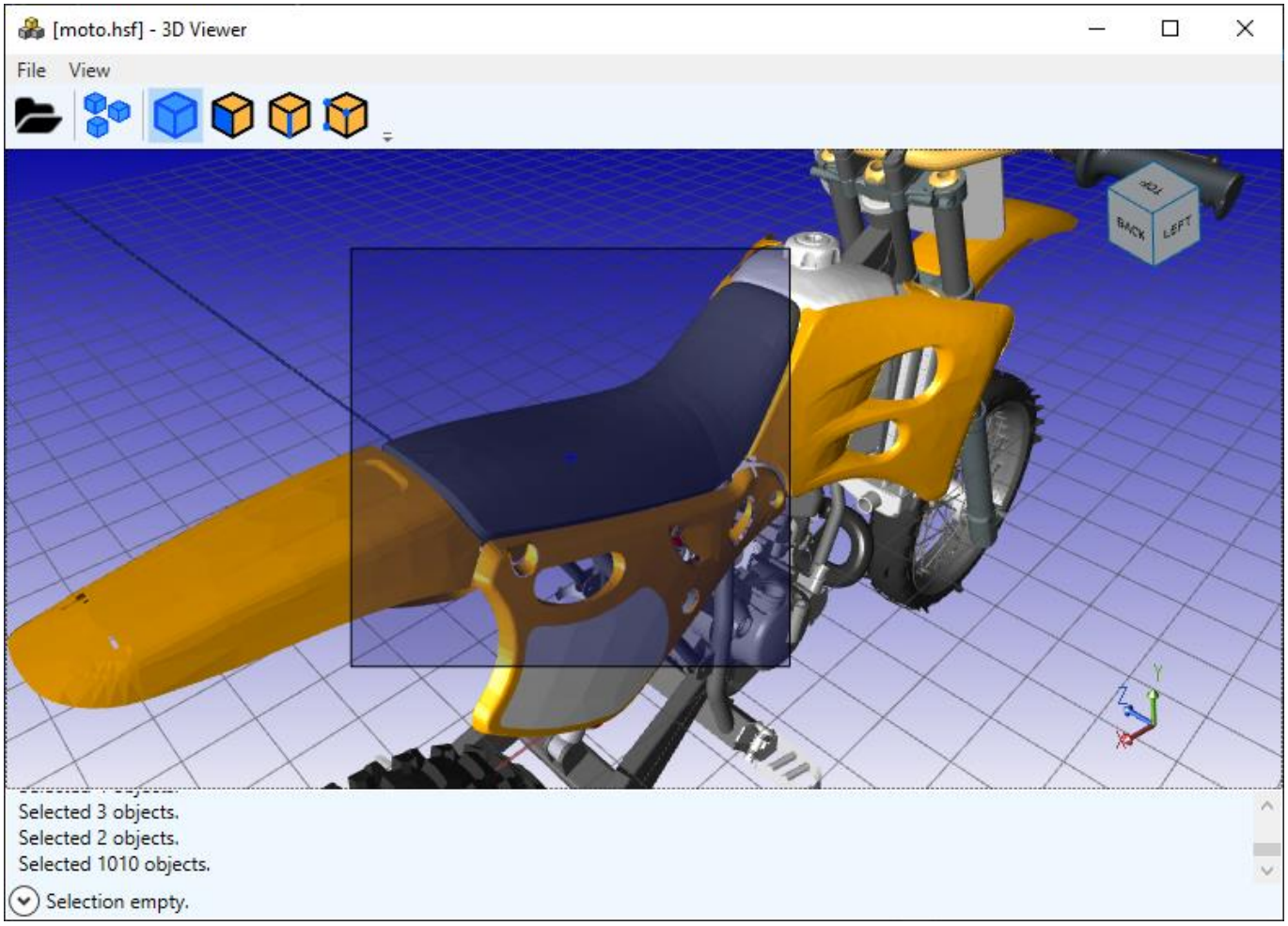


Рисунок 3.22 – Процес виділення областю

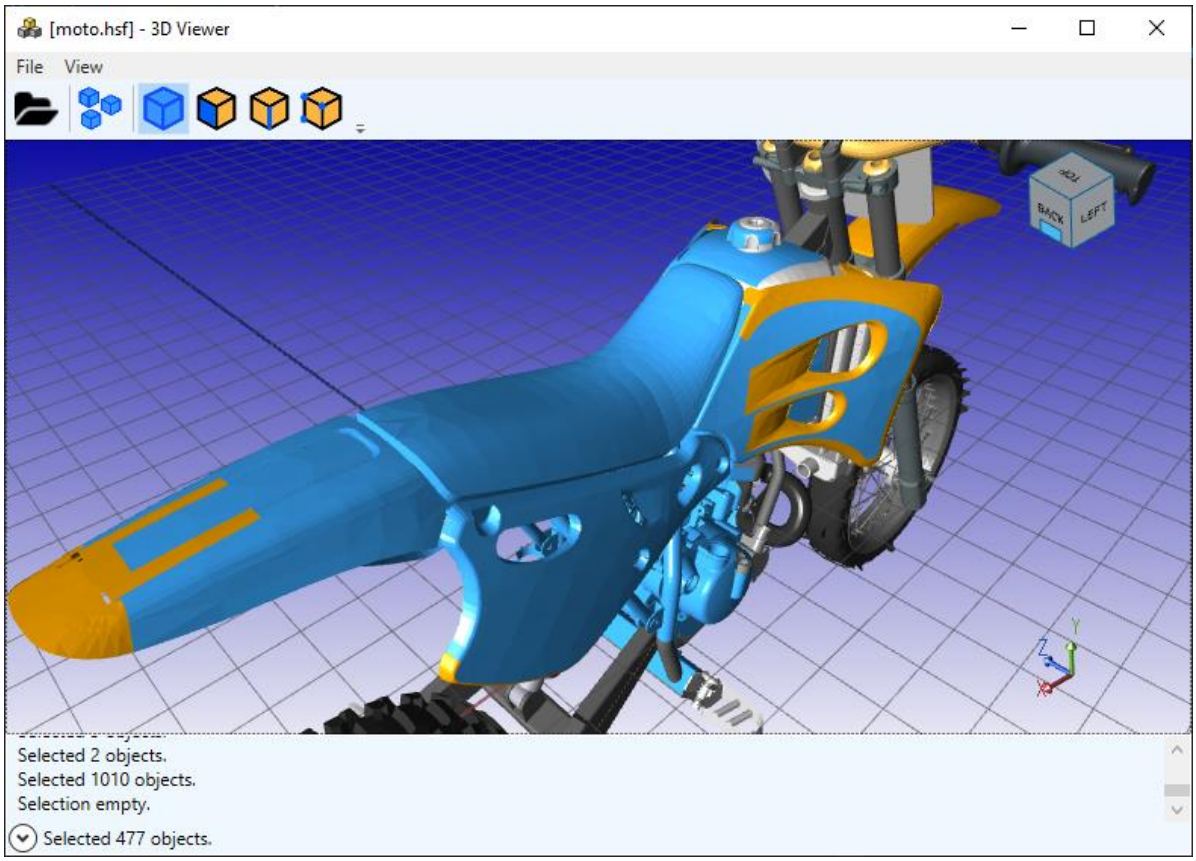


Рисунок 3.23 – Результат виділення областю

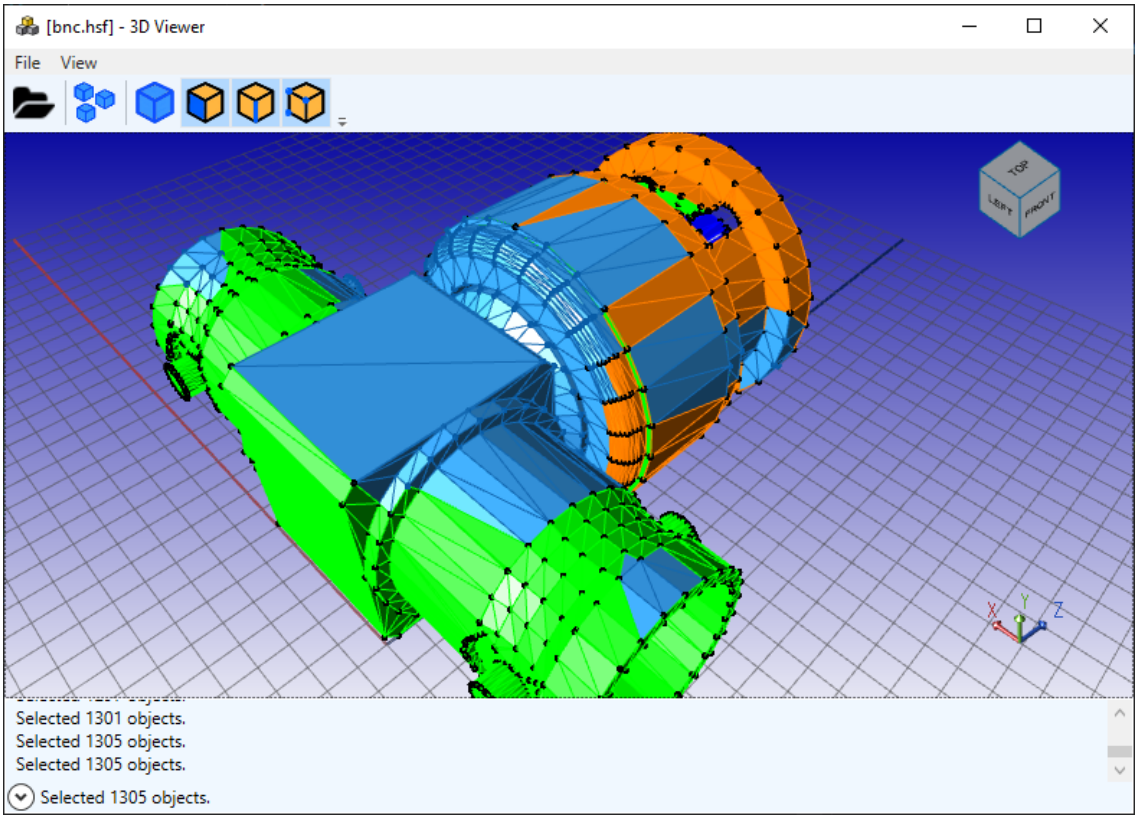


Рисунок 3.24 – Виділення елементів

3.2.10 Створення операцій редагування

Для кожної операції редагування створюється окремий UserControl, що має власну ViewModel, яка містить модель, що відповідає за здійснення редагування об’єктів.

Далі наведено приклад роботи однієї з операцій SetTexture.

View для операції загалом складається з двох елементів ToggleButton та Popup. При натисканні на ToggleButton відкривається вікно, куди заносяться параметри операції, і яке також має кнопку Run для її запуску. Вигляд команди наведений на рисунку 3.25.

```
0 references | AnastasiiaShadow, 10 hours ago | 1 author, 1 change
public RelayCommand SetTextureCommand
{
    get
    {
        if(setTextureCommand == null)
        {
            setTextureCommand = new RelayCommand(o => SetTexture(), o => CanExecuteSetTexture());
        }
        return setTextureCommand;
    }
}
#endregion

#region Methods
1 reference | AnastasiiaShadow, 10 hours ago | 1 author, 1 change
private void SetTexture()
{
    PortfolioKey portfolioKey = setTexture.DefinePortfolio(TextureFilePath, SelectedMapping, out string textureName);
    RGBAColor rgbaColor = new RGBAColor(new RGBA32Color(DiffuseColor.Value.R, DiffuseColor.Value.G,
        DiffuseColor.Value.B, DiffuseColor.Value.A));

    if(IsShellLevelActive)
    {
        setTexture.SetShellTexture(portfolioKey, rgbaColor, textureName);
    }

    if(IsFaceLevelActive)
    {
        setTexture.SetFaceTexture(portfolioKey, rgbaColor, textureName);
    }

    IsPopupOpen = false;
}
```

Рисунок 3.25 – Команда SetTexture

Для можливості використання текстури, вона визначається у загальному портфоліо для моделі за допомогою методу DefinePortfolio (рис. 3.26).

```
84      public PortfolioKey DefinePortfolio(string textureFilePath, Material.Texture.Parameterization parameterization, out string textureName)
85      {
86          string textureFileFormat = System.IO.Path.GetExtension(textureFilePath).TrimStart('.');
87          if(!imageFormats.TryGetValue(textureFileFormat, out Image.Format format))
88          {
89              throw new IOException("Not supported image format.", IOResult.UnsupportedFormat);
90          }
91
92          // Importing the image and defining it in the GeometrySegment portfolio.
93          Image.ImportOptionsKit importOptions = new Image.ImportOptionsKit();
94          importOptions.SetFormat(format);
95          ImageKit imageKit = Image.File.Import(textureFilePath, importOptions);
96          PortfolioKey portfolioKey = viewer.ViewPort.TexturePortfolio;
97          ImageDefinition imageDefinition = portfolioKey.DefineImage(textureFilePath, imageKit);
98
99          // Defining a texture from the image.
100         TextureOptionsKit textureOptionsKit = new TextureOptionsKit();
101         textureOptionsKit.SetParameterizationSource(parameterization);
102         textureName = textureFilePath + parameterization;
103         portfolioKey.DefineTexture(textureName, imageDefinition, textureOptionsKit);
104         return portfolioKey;
105     }
```

Рисунок 3.26 – Визначення текстури в портфоліо

Потім, в залежності від того, на якому рівні проводиться редагування, визиваються методи SetShellTexture та SetFaceTexture.

При SetShellTexture створюється MaterialMappingKit, в якій установлюються параметри введені користувачем, та застосовується для всіх шелів в SelectionSet.

При SetFaceTexture створюється новий матеріал в MaterialPaletteDefinition і застосовується для граней.

Таким же чином створена решта операцій.

3.3 Використання додатку

При запуску додатку користувач бачить усі головні елементи інтерфейсу, проте панель перегляду 3D-моделей є пустою, а більшість кнопок на панелі інструментів є неактивними (рис. 3.27).

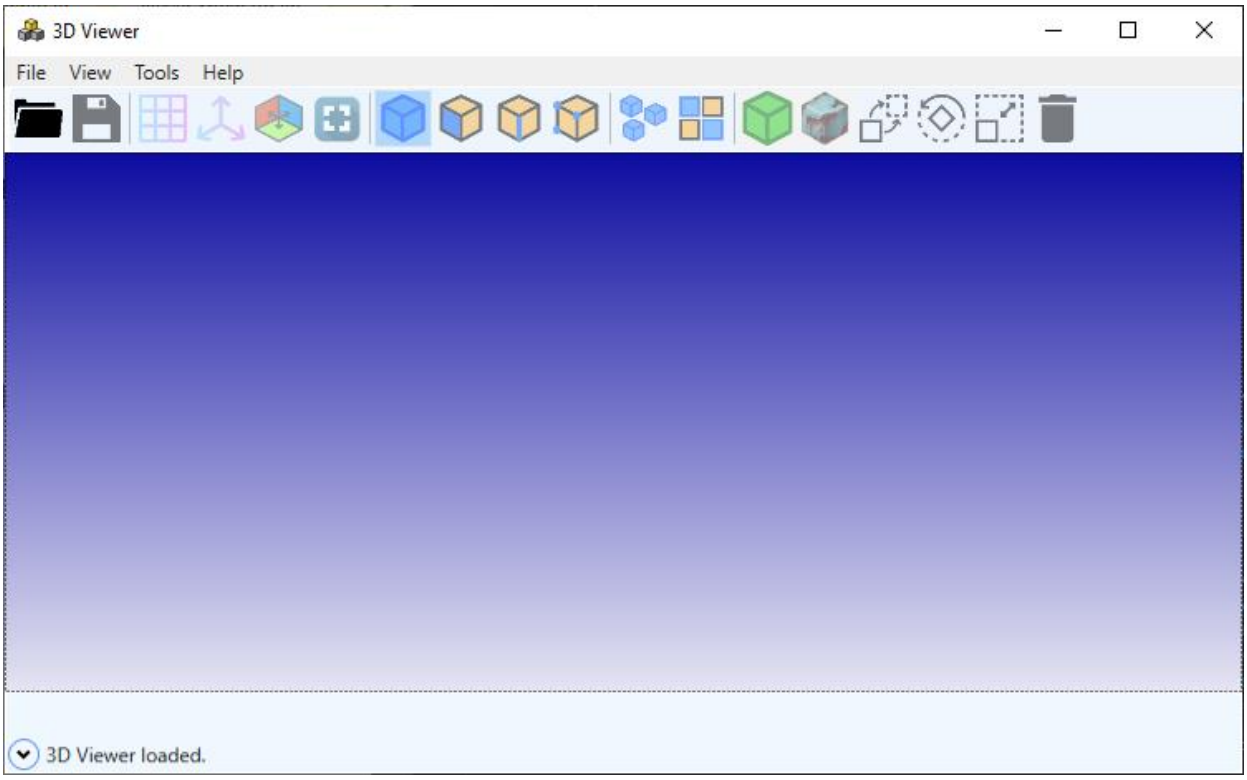


Рисунок 3.27 – Початковий вид програми

Після відкриття файлу відображаються модель та допоміжні елементи панелі перегляду. Також усі кнопки на панелі інструментів стають активними (рис. 3.28).

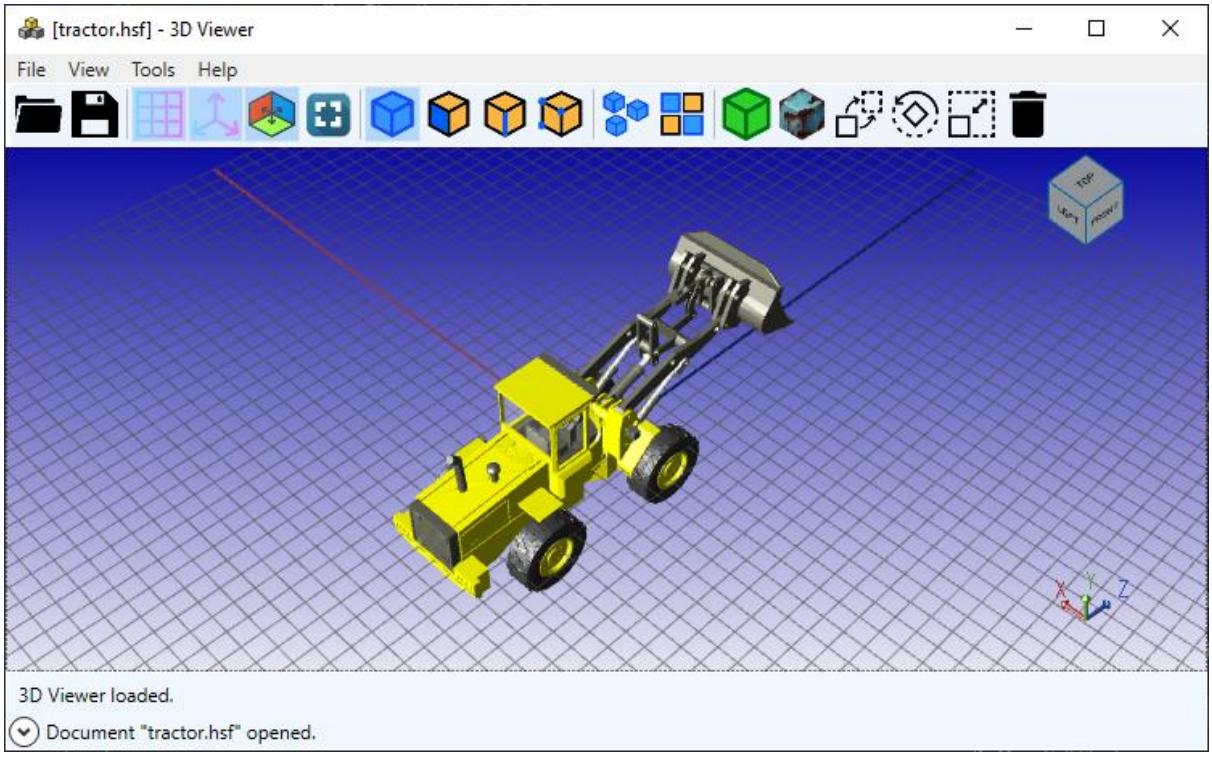


Рисунок 3.28 – Відкрита модель

Є можливість змінити видимість сітки, осей та навігаційного кубу за допомогою кнопок на панелі інструментів (рис. 3.29).

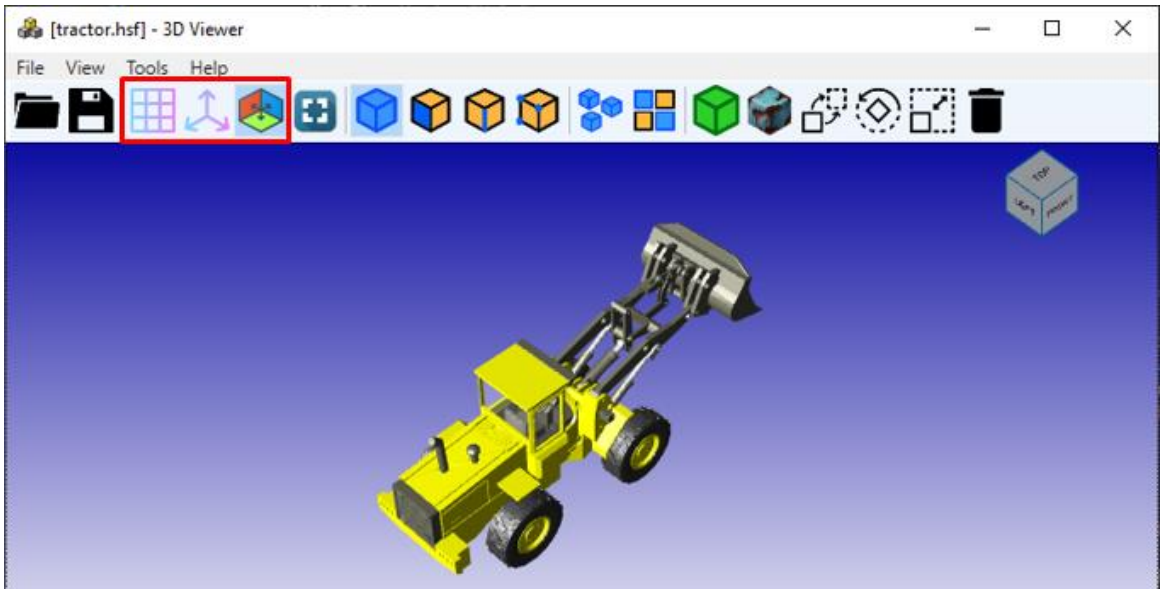


Рисунок 3.29 – Зміна видимості сітки та трійки осей

Далі наведено приклад роботи виділення та операцій. На рис. 3.30 проводиться виділення об’єктів на рівні шелу.

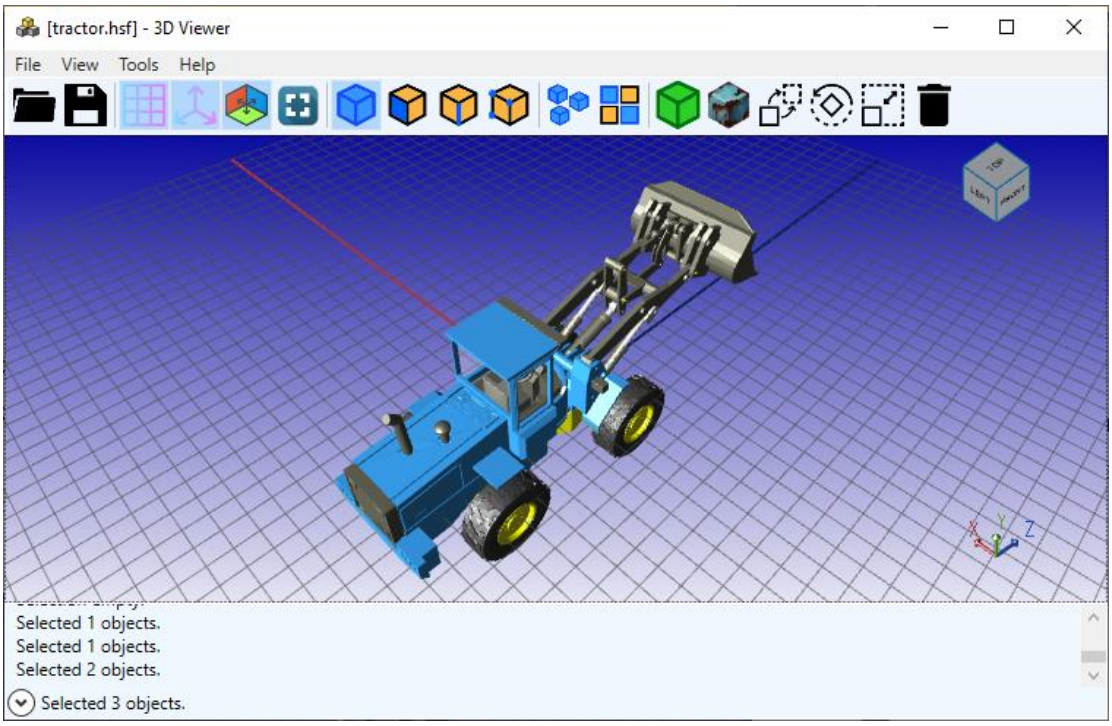


Рисунок 3.30 – Виділення об’єктів на рівні шелу

Для накладення на виділені об’єкти текстури, натискається кнопку SetTexture, де потрібно ввости необхідні параметри (рис. 3.31) та запустити операцію, результат якої наведено на рис. 3.32.

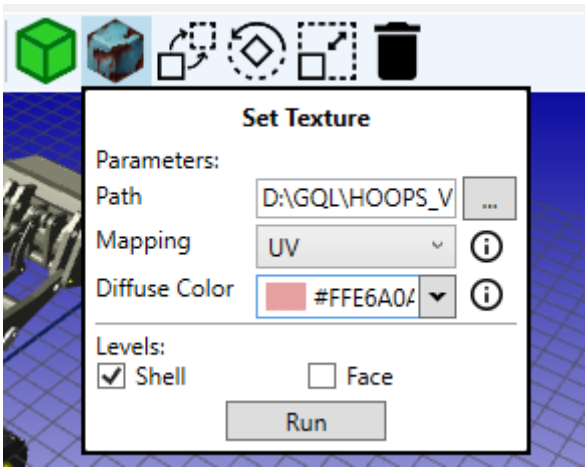


Рисунок 3.31 – Введення параметрів SetTexture

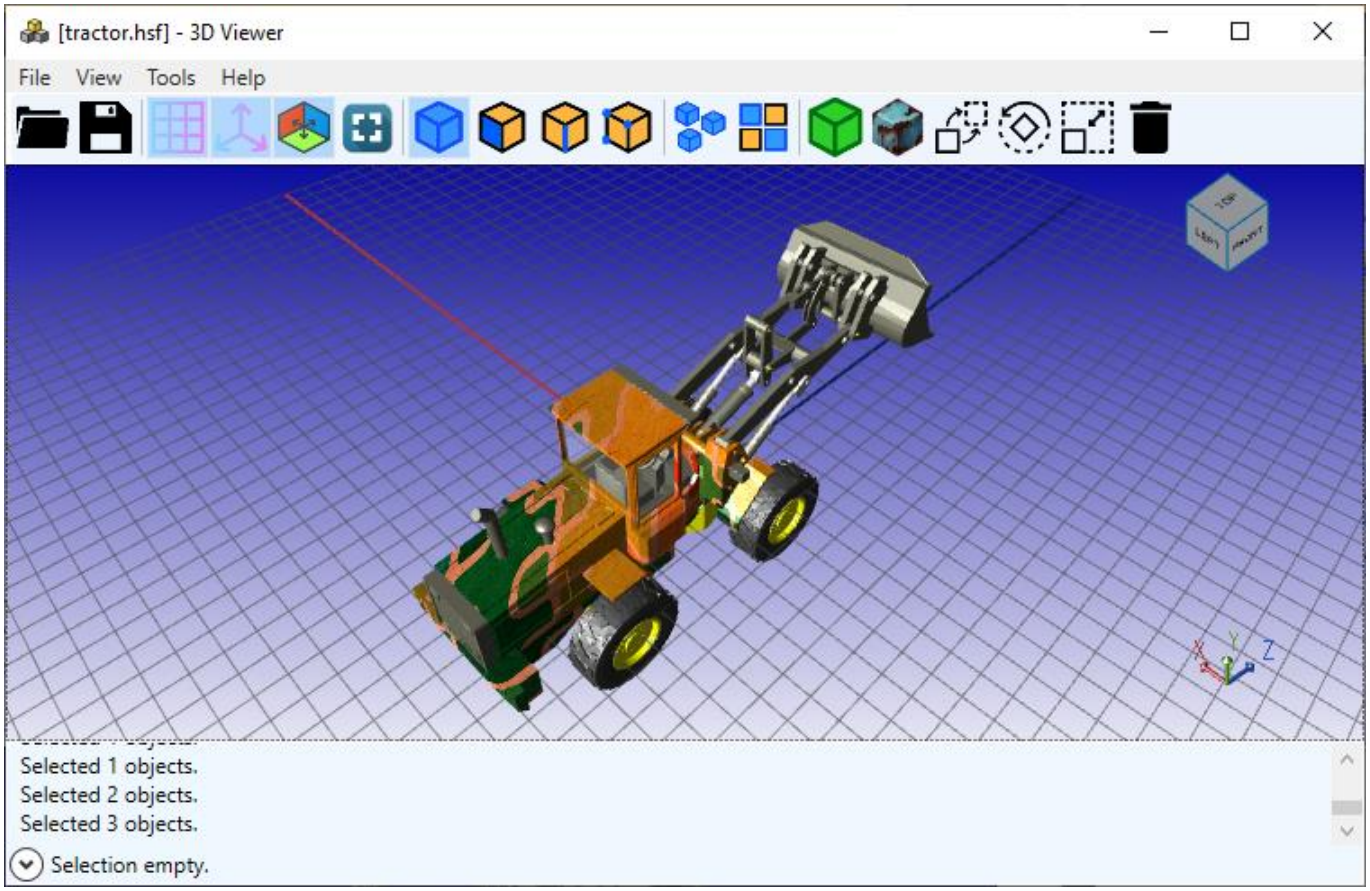


Рисунок 3.32 – Результат SetTexture

На рисунку 3.33 наведений приклад виділення об’єктів на рівні ребер.

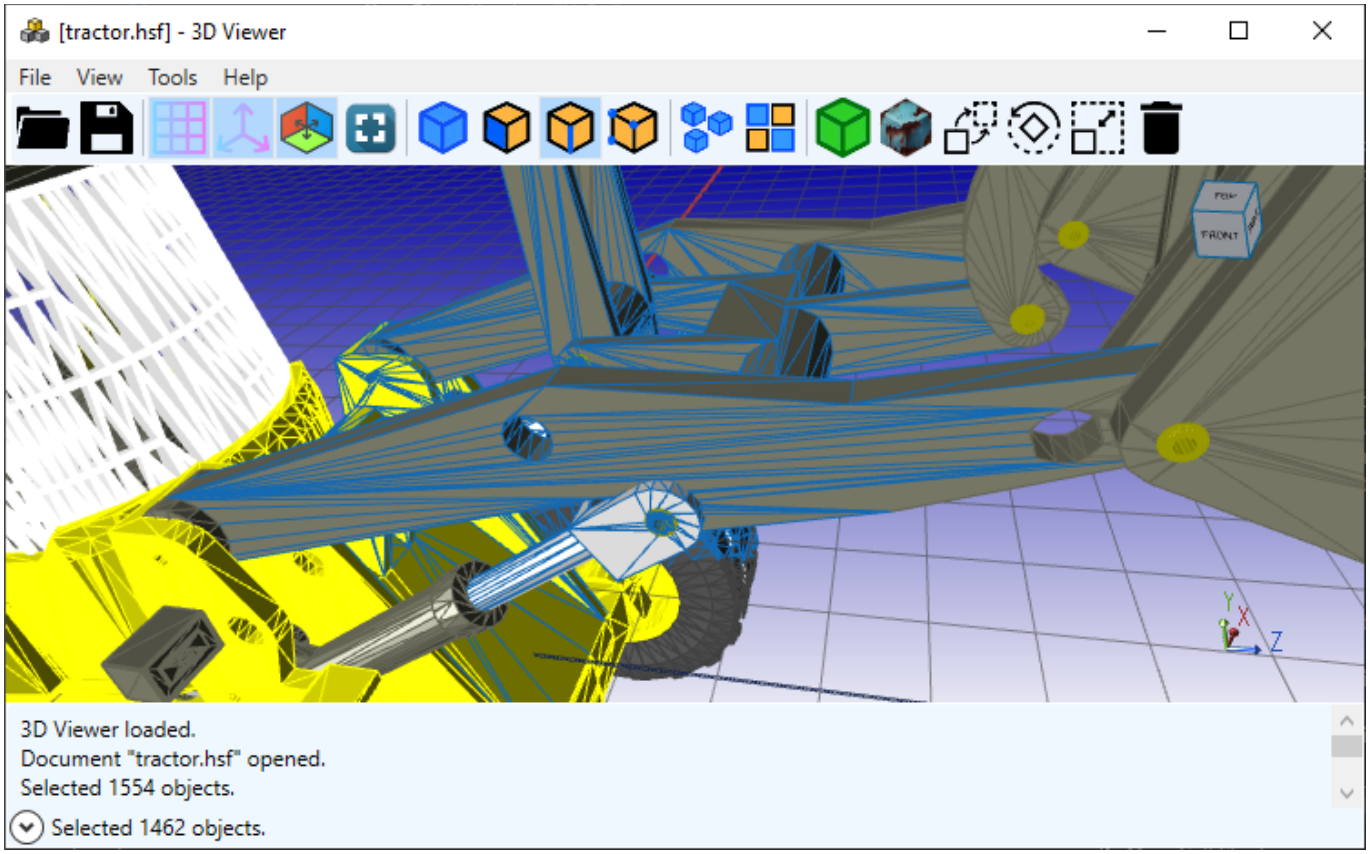


Рисунок 3.33 – Виділення об’єктів на рівні ребер

Для зміни їх кольору, натискається кнопка SetColor, вводяться необхідні параметри (рис. 3.34) та запускається операція, результат якої наведено на рис. 3.35.

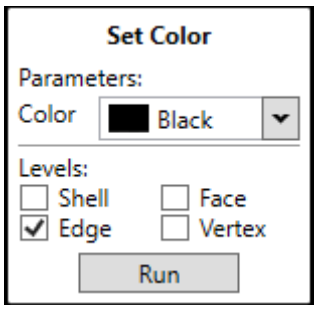


Рисунок 3.34 – Введення параметрів SetColor

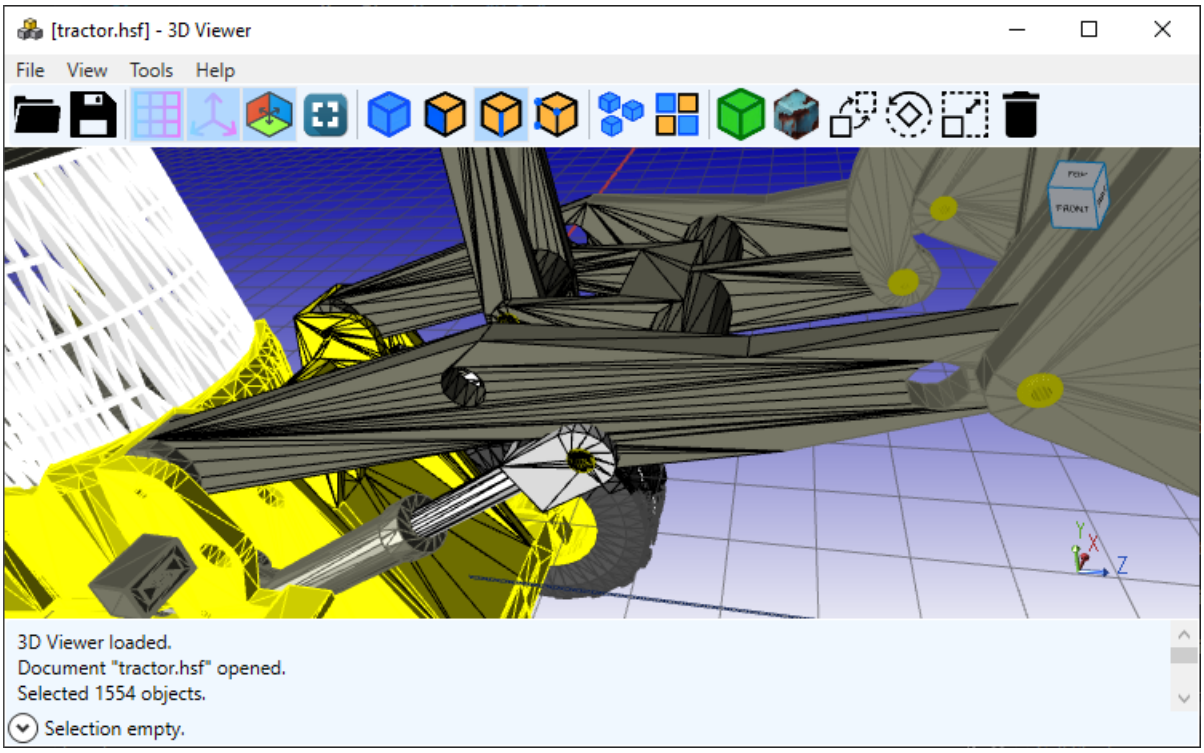


Рисунок 3.35 – Результат SetColor

Останнім прикладом операції є операція видалення, спочатку обираються грані, які необхідно видалити (рис. 3.36), обирається рівень видалення та запускається операція (рис. 3.37), результат якої на рис. 3.38.

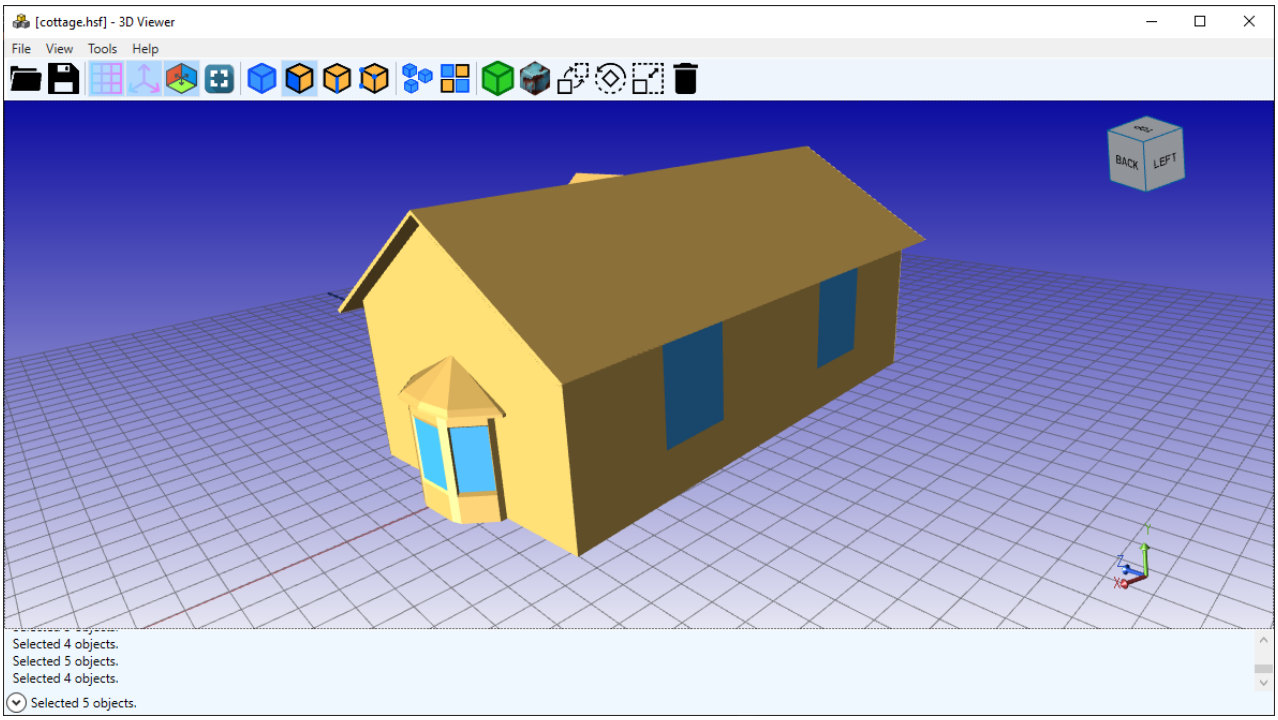


Рисунок 3.36 – Виділення об’єктів на рівні граней

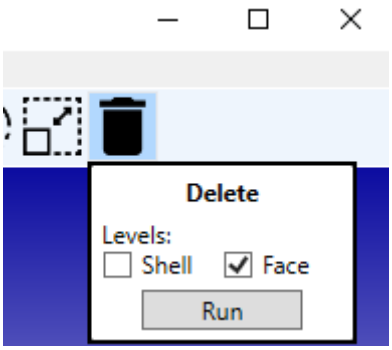


Рисунок 3.37 – Обрання рівня роботи Delete

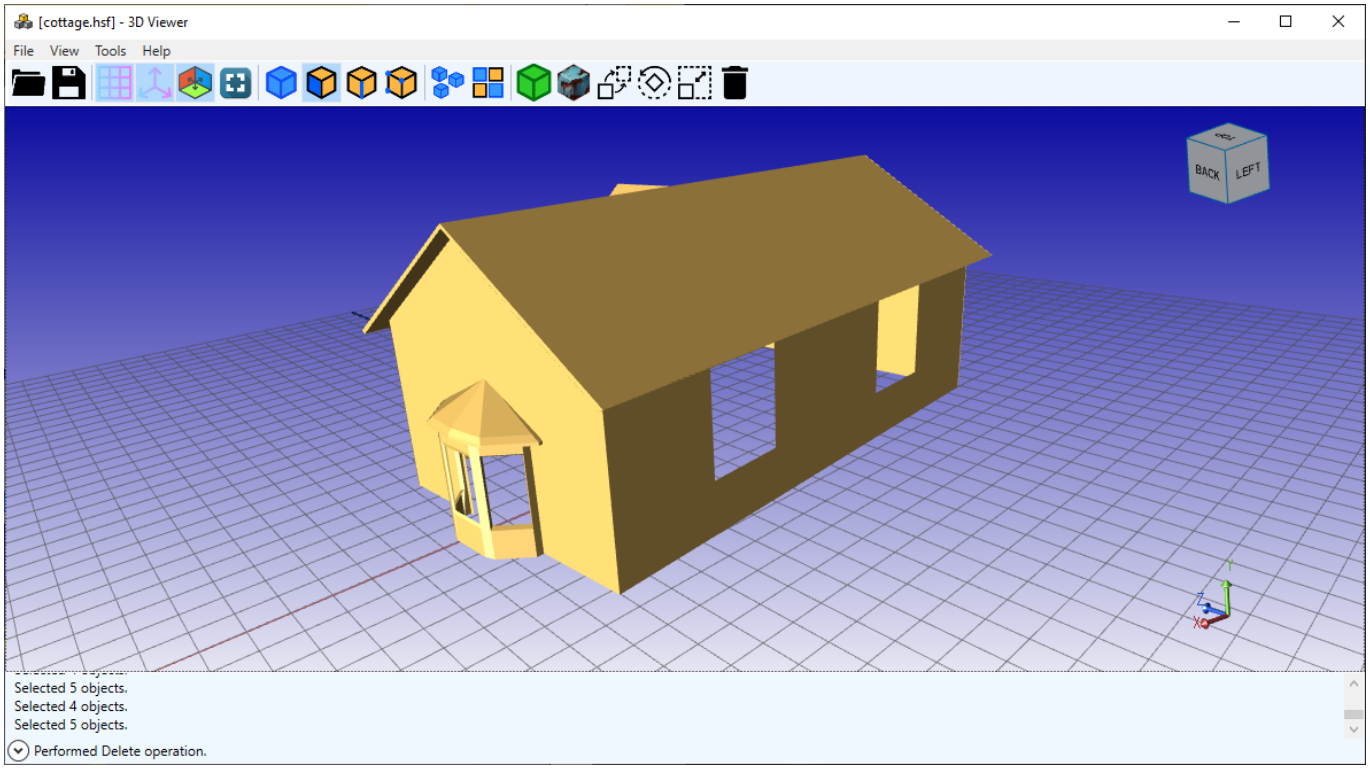


Рисунок 3.38 – Результат Delete

Для збереження змін у файл після операції Delete можна використати дві функції: «Save» та «Save As». При натисканні на Save, зміни застосуються безпосередньо до відкритого файлу. Якщо ж в пункті меню «File» обрати «Save As», то відкриється вікно (рис. 3.39), в якому можна обрати розташування, формат та ім’я нового файлу. Файл після збереження наведений на рис. 3.40.

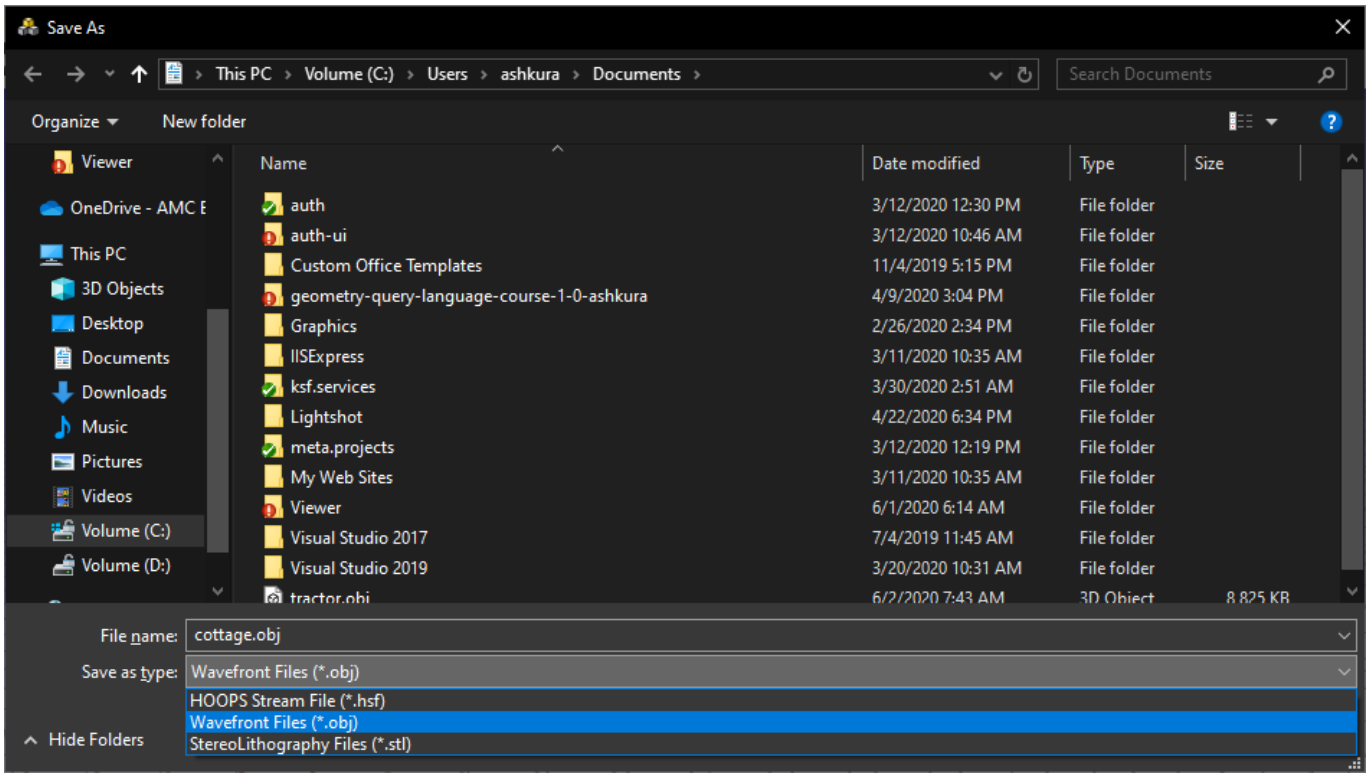


Рисунок 3.38 – Вибір параметрів збереження

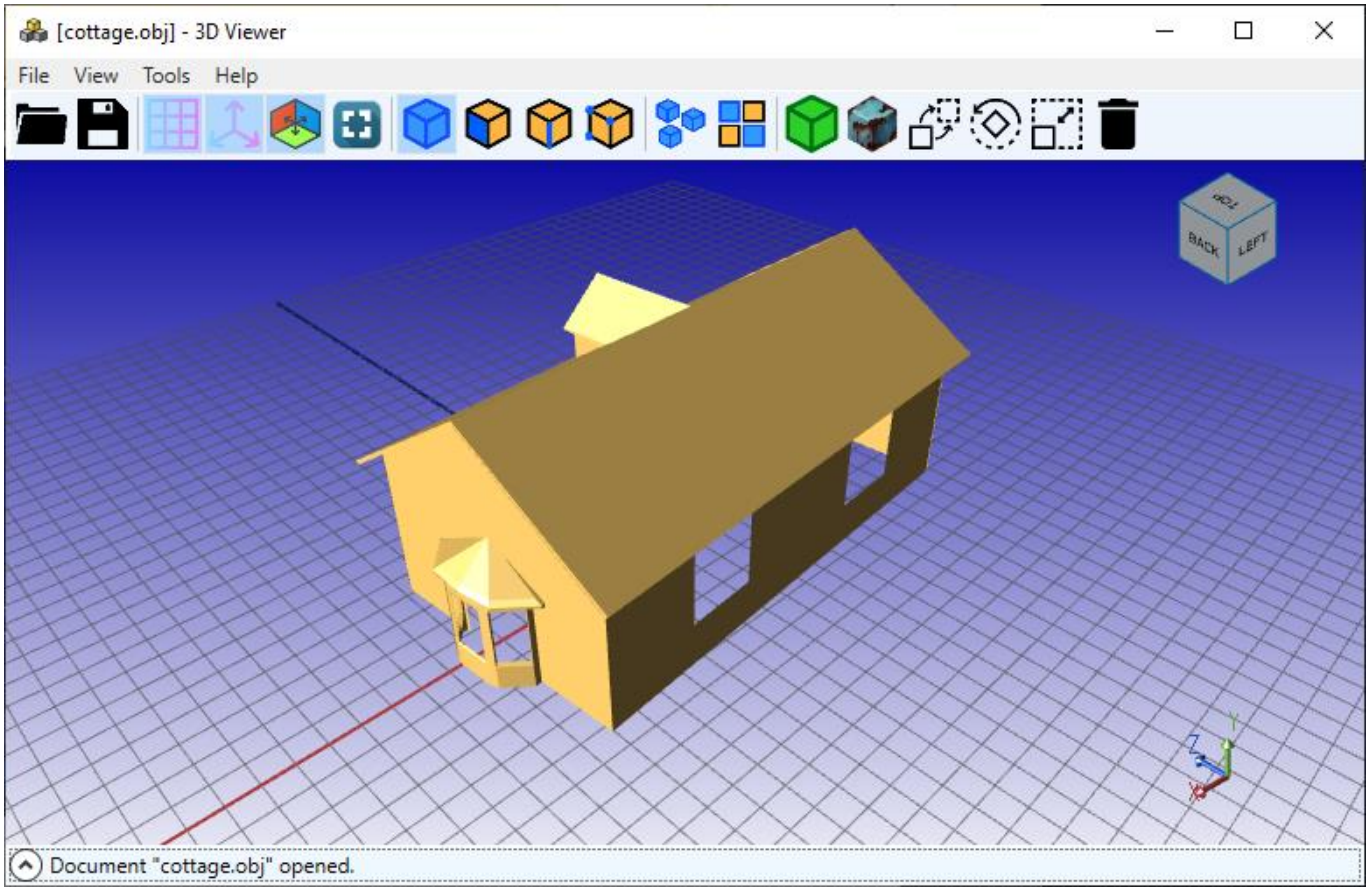


Рисунок 3.38 – Файл після збереження

ВИСНОВКИ

Під час виконання дипломної роботи було розглянута предметна область в цілому та проведений її аналіз. Було описано існуючі дослідження та публікації, а також програмні продукти, для вирішення проблем в даній області За результатми аналізу була визначена актуальність даної розробки та створений перелік вимог до створюваної інформаційної системи.

Також сформовано технічне завдання, яке містить детальні вимоги до розробленого додатку та виконано планування робіт.

На етапі проектування були побудовані контекстна діаграма та діаграма декомпозиції структурно–функціональної моделі основного процесу перегляду та редагування 3D-моделі, який забезпечується програмним додатком, що розроблявся. Для опису можливостей програми була створена діаграма варіантів використання.

На етапі реалізації була створена програма для перегляду та редагування 3D-моделей. Створена програма може бути використана для показу та редагування 3D-об’єктів. Також отримані моделі можна використовувати для 3D-печаті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шкура А.В. Програмний модуль перегляду та редагування графічних 3D-моделей. / Шкура А.В., Ващенко С.М. // Інформатика, математика, автоматика: матеріали та програма науково-технічної конференції, м. Суми, 2020 р. – Суми : СумДУ, 2020. – 113 с.
2. Petty J. What is 3D Modeling & What's It Used For? [Електронний ресурс] / Josh Petty – Режим доступу до ресурсу: https://conceptartempire.com/what-is-3d-modeling/.
3. Sun Z. 3D printing in medicine: current applications and future directions. Quant Imaging Med Surg. 2018
4. Bringing Cosmic Objects Down to Earth: An Overview of 3D Modelling and Printing in Astronomy and Astronomy Communication [Електронний ресурс] / M.Watzke, J. DePasquale, A. Jubett, P. Edmonds // CAPjournal. – 2017. – Режим доступу до ресурсу: https://pdfs.semanticscholar.org/0452/d975e224b32a9c6fafc8f62bf3f220b8763d.pdf.
5. Research and Development of 3DModeling [Електронний ресурс] / L.Xi-Dao, X. Yu-Xiang, Y. Long, W. Ling-Da // IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.1. – 2008. – Режим доступу до ресурсу: http://paper.ijcsns.org/07_book/200801/20080108.pdf.
6. A. Fusiello, U. Castellani, V. Murino. A complete system for online 3D modelling from acoustic images. Signal Processing: Image Communication, 2005.
7. Toshihiro Asai, Naokazu Yokoya, Masayuki Kanbara. 3D Modeling of Outdoor Environments by Integrating Omnidirectional Range and Color Images. Proceedings of the Fifth International Conference on 3D Digital Imaging and Modeling.
8. M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz. The digital Michelangelo Project: 3D scanning of large statues. In Siggraph 2000.

9. P. Mueller, T. Vereenooghe, L. Van Gool. Photo-realistic and detailed 3D modeling: the Antonine nymphaeum at Sagalassos (Turkey). Computer Applications and Quantitative Methods in Archaeology (CAA): Beyond the artifact - Digital interpretation of the past.
10. Lindblad M. Blender Overview: Free 3D Modeling and VFX Software [Електронний ресурс] / Mason Lindblad. – 2018. – Режим доступу до ресурсу: https://filtergrade.com/blender-overview-free-3d-modeling-vfx-software/.
11. AutoCAD and its Uses: What is AutoCAD Used for? [Електронний ресурс] – Режим доступу до ресурсу: https://tutorial45.com/what-is-autocad-used-for/.
12. 3 reasons to use the MVVM pattern [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.mrlacey.com/2017/04/3-reasons-to-use-mvvm-pattern.html.
13. MVVM architecture, ViewModel and LiveData (Part 1) [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1.
14. Alle M. Singleton Design Pattern In C# [Електронний ресурс] / Mahesh Alle. – 2020. – Режим доступу до ресурсу: https://www.c-sharpcorner.com/UploadFile/8911c4/singleton-design-pattern-in-C-Sharp/.
15. View hierarchy [Електронний ресурс] – Режим доступу до ресурсу: https://docs.techsoft3d.com/hps/latest/build/prog_guide/0301_core.html.
16. Work Breakdown Structure (WBS) [Електронний ресурс] – Режим доступу до ресурсу: https://www.workbreakdownstructure.com/.
17. Project Management Institute. 2004. A guide to the project management body of knowledge (PMBOK guide). Newtown Square, Pa: Project Management Institute.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

1 Призначення й мета створення інформаційної системи

1.1 Призначення інформаційної системи

Інформаційна система повинна бути створена у вигляді простого редактору 3D-моделей.

1.2 Мета створення ІС

Надання можливості зручного перегляду 3D-моделей та можливості її легкого та швидкого редагування.

1.3 Цільова аудиторія

Цільовою аудиторією програми є викладачі та студенти, що вивчають комп’ютерну графіку.

2 Вимоги до інформаційної системи

2.1 Вимоги до структури й функціонування

Розроблювана програма повинна бути stand-alone додатком, клієнтська частина якого є WPF підсистема з набором елементів керування, що дозволяють відтворювати 3D-модель та редагувати її.

2.2 Вимоги до інформаційної та програмної сумісності

Продукт повинен мати інсталяційний пакет для установки на ОС Windows на платформі x64. Для коректної роботи програми необхідний .NET Framework 4.6.1 або вище. Інтерфейс продукту повинен бути оформлений англійською мовою.

2.3 Вимоги до користувацького інтерфейсу програми

Основну частину програми повинна складати область перегляду моделі. При наявності відкритого файлу окрім самих об’єктів має бути також відображені осі симетрії (triad axis), навігаційний куб та сітка.

Панель меню повинна складатися з таких пунктів:

- File (меню для роботи з файлом);
- View (налаштування виду та відображення елементів);
- Tools (Інструменти для редагування моделі);
- About (інформація про програму).

Під панеллю меню повинна знаходитись панель інструментів, яка повинна відображати основні можливості та операції.

В нижній частині програми має знаходитись лог-панель, яку можна тимчасово приховати (extendable).

При відкритому документі рядок заголовку повинен містити назву файлу без шляху.

Макет вікна програми наведений на рис. А.1.

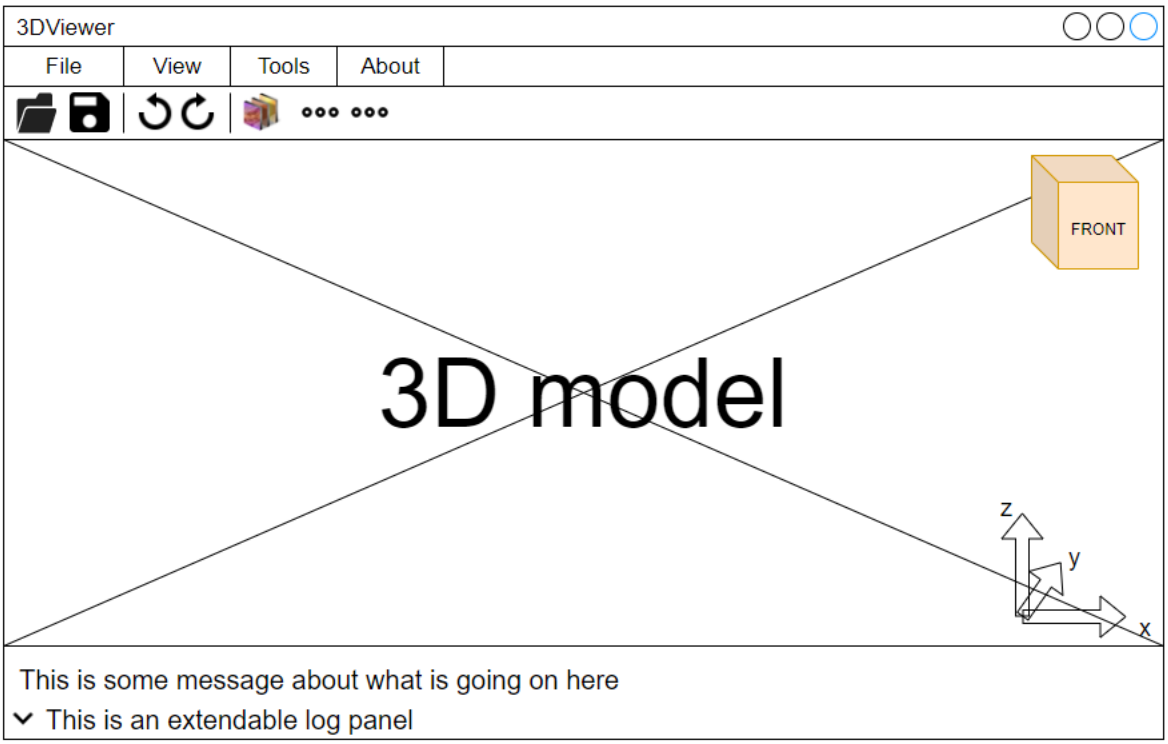


Рисунок А.1 – Макет вікна програми

2.4 Функціональні вимоги

Додаток має забезпечувати виконання таких функцій:

- Відкриття та збереження файлів форматів STL, OBJ та HSF;
- Відображення 3D-моделі;
- Відображення лог-панелі для виведення повідомлень та помилок;
- Виділення об’єктів на рівнях об’єкту, граней, ребер та вершин;
- Можливість зміни видимості осей симетрії, навігаційного кубу та сітки;
- Зміна камери операціями миші (zoom, pan, orbit);
- Редагування об’єктів:
  - Зміна кольору (параметр – RGB колір);
  - Накладання матеріалу (параметри – файл з текстурою, тип мепінгу, альфа канал, дифузний колір);
  - Переміщення, обертання та масштабування об’єктів (параметри – значення за осями x, y, z);
  - Видалення виділених об’єктів (без параметрів).

При виборі операції редагування, яка потребує параметрів, повинно бути показана панель для вводу параметрів.

2.5 Вимоги до інформаційного забезпечення

Реалізація додатку відбувається за допомогою таких технологій:

- C#;
- .NET Framework 4.6.1;
- WPF, MVVM;
- .NET Hoops Visualize SDK;
- NuGet package Extended.Wpf.Toolkit;
- WiX Toolset v3.

2.6 Вимоги до лінгвістичного забезпечення

Програма повинна бути створена англійською мовою.

3 Склад і зміст робіт зі створення сайту

Опис робіт, які необхідно виконати для отримання робочої програми наведено в табл. А.1.

Таблиця А.1 – Етапи розробки програми

№	Назва роботи	Зміст роботи	Строк розробки (в годинах)
1	Середовище перегляду	Реалізація можливості перегляду моделей.	200

Продовження табл. А.1

1.1	Створення проекту	Створення WPF додатку за макетом.	7
1.2	Рендер моделі	Створення контролеру для відображення моделі та його налагодження.	4
1.3	Завантаження	Завантаження STL, OBJ або HSF файлів.	4
1.4	Збереження	Збереження моделі до файлу (Зберегти / Зберегти як).	8
1.5	Види за замовчування	Список типових видів перегляду моделі (спереду, зліва, зверху і т.д.).	3
1.6	Лог-панель	Створення панелі для відображення повідомлень та помилок.	2
1.7	Камера	Забезпечення можливості роботи з виглядом за допомогою операцій панорамування (pan), збільшення (zoom) та обертання (rotate) камери.	64
1.8	Виділення	Реалізація можливості виділення, а також його зміни за допомогою миші на рівні об’єкту, грані, ребра та вершини.	84
1.9	Виділити все	Створення операції Select All як у вигляді кнопки на панелі інструментів, так і гарячої клавіші.	13
1.10	Invert	Операція інверсії поточного виділення.	11
2	Редагування	Реалізація операцій редагування моделі.	128
2.1	SetColor	Операція для зміни кольору об’єктів.	6
2.2	ApplyMaterial	Нанесення на поверхні текстури.	18
2.3	Translate	Переміщення об’єктів у просторі.	18

Продовження табл. А.1

2.4	Rotate	Обертання об’єкту щодо його центру.	18
2.5	Scale	Масштабування об’єкту щодо його центру.	18
2.6	Delete	Видалення об’єкту.	26
3	Create Installer	Створення інсталятору.	24
		Загалом	357

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Деталізація мети методом SMART. Мета проекту: побудувати додаток для перегляду та редагування 3D-об’єктів у вигляді полігонної сітки, де користувач з легкістю зможе відкрити модель, використовуючи стандартні операції мишкою, змінювати положення камери, виділяти об’єкти в панелі перегляду, та за допомогою кнопок на панелі інструментів редагувати модель.

Результат деталізації мети проекту методом SMART наведено в табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна)	Розробити додаток для перегляду та редагування 3D-моделей.
Measurable (вимірювана)	Оскільки проект не має комерційної цілі, то його результатом є позитивна оцінка замовника.
Achievable (досяжна)	Реалізація проекту здійснюється з використанням мови програмування C#, а для рендеру, взаємодії та редагування 3D-моделей використовується HOOPS Visualize.
Relevant (реалістична)	Для реалізації продукту наявні всі необхідні програмні та технічні засоби. Розробник має достатньо досвіду роботи з використовуваними технологіями для успішного виконання поставлених задач.
Time-framed (обмежена у часі)	Додаток розроблюється з обмеженням у часі звісно до календарного плану.

Планування змісту структури робіт проекту. Важливим етапом створення проекту є планування змісту структури робіт. Правильне планування має вирішальне значення для встановлення чітких очікувань та задач для кожного учасника команди, щоб завершити проект вчасно та в рамках встановленого бюджету. Основним інструментом для його виконання є Work Breakdown Structure (структура декомпозиції робіт).

WBS допомагає розкласти великий проект на більш дрібні керовані секції, дозволяючи кожному учаснику команди зосередитись на конкретних завданнях та вимірюваних етапах. Чітко визначені роботи легко призначити. Вони дозволяють точно оцінити необхідний час та ресурси на виконання кожної з них та відстежити їх завершення.

Побудуємо WBS для даного проекту, в якій ми зазначимо елементарні роботи необхідні для його успішного завершення. Структура декомпозиції наведена на рис. Б.1.

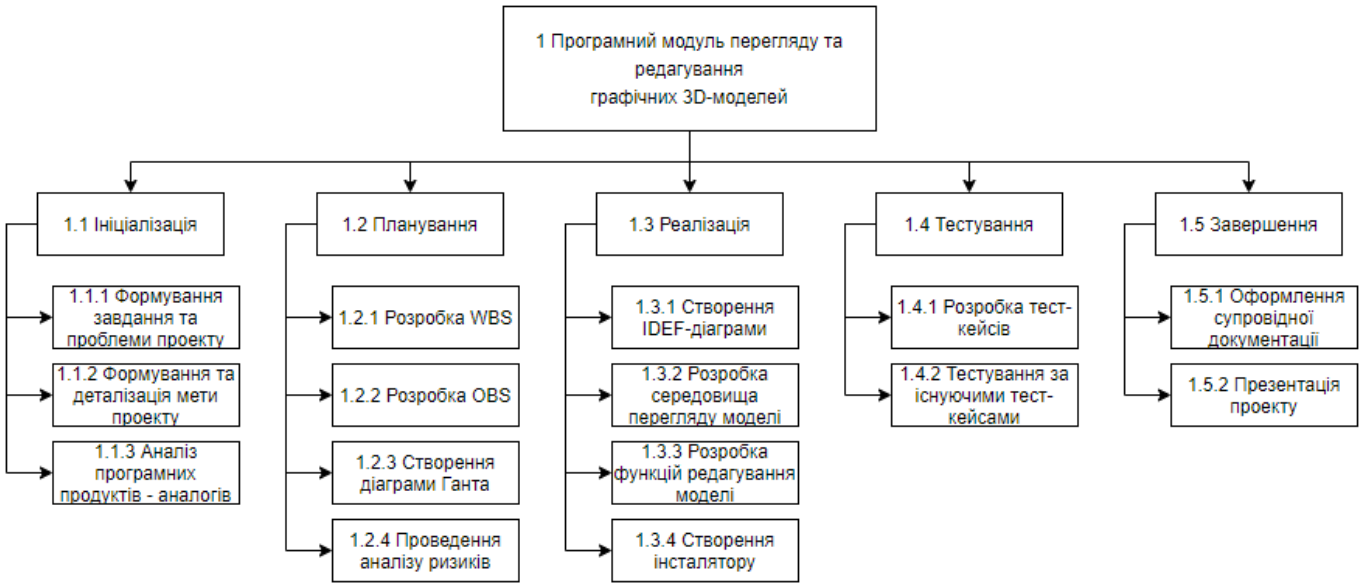


Рисунок Б.1 – Структура декомпозиції робіт

Планування структури організації, для впровадження готового проекту (OBS). Після побудови WBS можна перейти до створення OBS, що представляє собою графічне відображення усіх учасників проекту, які задіяні у його реалізації.

Організаційна структура проекту вказує які організаційні підрозділи виконують які роботи. Вона виконується на основі уже створеної WBS. Список учасників проекту та їх ролі наведено в табл. Б.2. Діаграма OBS вказана на рис. Б.2.

Таблиця Б.2 − Учасники проекту

Роль	Ім’я	Проектна роль
Виконавець	Шкура А.В.	Відповідає за виконання термінів. Виконує збір та аналіз даних. Виконує розробку основного функціоналу проекту.
Тестувальник	Представник компанії	Відповідає за тестування функціоналу та дизайну додатку.
Консультант проекту	Ващенко С.М.	Формує завдання на розробку проекту.

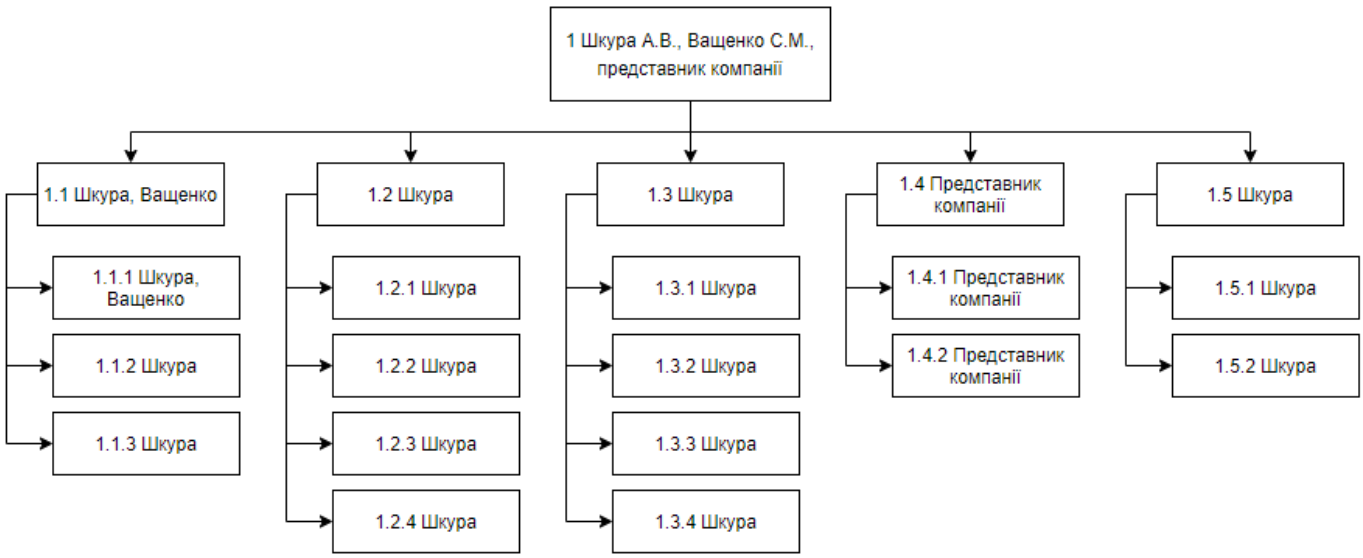


Рисунок Б.2 – Організаційна структура проекту

Діаграма Ганта. Для успішного виконання проекту в зазначений період часу створимо календарний план, який допоможе графічно представити початок та закінчення кожної роботи, а також взаємозв’язки між ними. Тривалість робіт наведена в годинах.

Діаграма Ганта наведена на рисунках Б.3 – Б.4.

	Режим задачи	Название задачи	Длительнс	Начало	Окончани	Предш
1		**Ініціалізація**	**14 ч**	**Пн 23.03.20**	**Вт 24.03.20**	
2		Формування завдання та проблеми проекту	4 ч	Пн 23.03.20	Пн 23.03.20	
3		Формування та деталізація мети проекту	4 ч	Пн 23.03.20	Пн 23.03.20	2
4		Аналіз програмних продуктів - аналогів	6 ч	Вт 24.03.20	Вт 24.03.20	3
5		**Планування**	**15 ч**	**Вт 24.03.20**	**Чт 26.03.20**	**4**
6		Розробка WBS	4 ч	Вт 24.03.20	Ср 25.03.20	4
7		Розробка OBS	2 ч	Ср 25.03.20	Ср 25.03.20	6
8		Створення діаграми Ганта	5 ч	Ср 25.03.20	Чт 26.03.20	7
9		Проведення аналізу ризиків	4 ч	Чт 26.03.20	Чт 26.03.20	8
10		**Реалізація**	**369 ч**	**Чт 26.03.20**	**Пт 29.05.20**	**9**
11		Створення IDEF-діаграми	12 ч	Чт 26.03.20	Пн 30.03.20	9
12		Розробка середовища перегляду моделі	205 ч	Пн 30.03.20	Пн 04.05.20	11
13		Розробка функцій редагування моделі	128 ч	Пн 04.05.20	Вт 26.05.20	12
14		Створення інсталятору	24 ч	Вт 26.05.20	Пт 29.05.20	13
15		**Тестування**	**14 ч**	**Пт 29.05.20**	**Вт 02.06.20**	**14**
16		Розробка тест-кейсів	6 ч	Пт 29.05.20	Пн 01.06.20	14
17		Тестування за існуючими тест-кейсами	8 ч	Пн 01.06.20	Вт 02.06.20	16
18		**Завершення**	**10 ч**	**Вт 02.06.20**	**Ср 03.06.20**	**17**
19		Оформлення супровідної документації	8 ч	Вт 02.06.20	Ср 03.06.20	17
20		Презентація проекту	2 ч	Ср 03.06.20	Ср 03.06.20	19

Рисунок Б.3 – Роботи для побудови діаграми

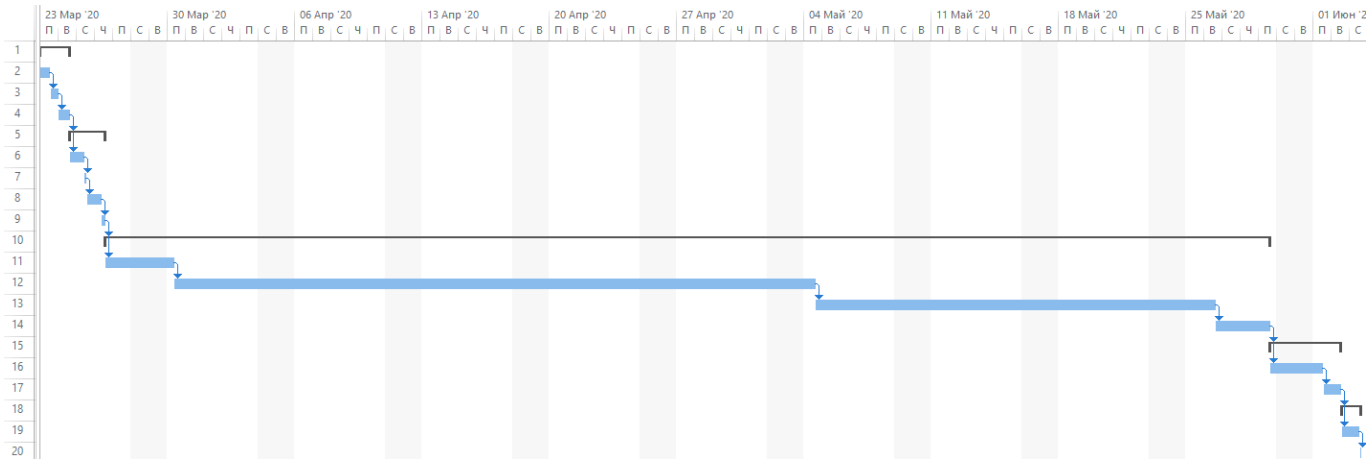


Рисунок Б.4 – Графічне представлення діаграми Ганта

Аналіз ризиків. Важливим етапом при роботі над проектом є визначення та аналіз ризиків, що можуть вплинути на якість та час розробки продукту. Тому виконаємо оцінку ризиків проекту. Для цього наведемо таблицю з виділеними ризиками (табл. Б.3) та проведемо їх класифікацію за ймовірністю виникнення та величиною втрат.

Таблиця Б.3 – Класифікація ризиків

№	Назва ризику	Ймовірність виникнення	Величина витрат
R1	Некоректно ТЗ	Середня	Висока
R2	Недотримання графіку	Середня	Середня
R3	Некоректна робота ПЗ	Висока	Висока
R4	Хвороба розробника	Слабка	Середня
R5	Неякісне тестування	Середня	Слабка
R6	Зміна вимог	Слабка	Середня

На основі створеної класифікації побудуємо матрицю ризиків (табл. Б.4), де зеленим кольором позначено прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

Таблиця Б.4 – Матриця ризиків

Ймовірність виникнення	Величина витрат		
	1 Слабка	2 Середня	3 Висока
3 Висока			R3
2 Середня	R5	R2	R1
1 Слабка		R4, R6	

Тепер на основі отриманих значень прокласифікуємо їх за рівнем ризику та занесемо значення в табл. Б.5.

Таблиця Б.5 – Класифікація за рівнем

№	Назва	Ризики, що входять
1	Прийнятні	R4, R5, R6
2	Виправдані	R2
3	Недопустимі	R1, R3

В табл. Б.6 введемо плани по запобіганню та реагуванню виникнення ризиків.

Таблиця Б.6 – Повна оцінка ризиків проекту

№	Опис ризику	План запобігання	План реакції
R1	Некоректне ТЗ	Замовник повинен скласти детальне ТЗ, чітко дотримуючись мети та задач проекту, а також вказати усі вимоги та терміни. Потрібно обговорити його з розробником.	Уважно проаналізувати, що були зазначено або виконано неправильно, та зробити зміни в ТЗ.
R2	Недотримання графіку	Створення чіткого реалістичного графіку із затвердженими термінами. Робота над проектом за створеним графіком.	Обговорення можливих рішень по зміні термінів реалізації.
R3	Некоректна робота ПЗ	Встановлення лише ліцензійного ПЗ з перевірених джерел. Створення резервних копій. Робота з Git-системами.	Переустановка програми, що некоректно працює, або відкат програм до останньої версії.
R4	Хвороба розробника	По можливості залишатися вдома на період карантину та уникати великого скупчення людей.	Лікуватись/звернутися до лікаря.
R5	Неякісне тестування	Створення повністю покриваючих програму тестів.	Повторне тестування.
R6	Зміна вимог	Створення чітко обговорених та осмислених вимог до початку розробки.	Коригування плану та термінів, спробувати зменшити вплив на час виконання.

ДОДАТОК В

ПРОГРАМНИЙ КОД

Current project is the property of the AMC Bridge company.

Метод Import класу Document

```
public static Document Import(string filePath, SegmentKey geometrySegment, out CameraKit
defaultCamera)
{
   defaultCamera = null;
   Enum.TryParse(Path.GetExtension(filePath).TrimStart('.'), true, out Format fileExtension);
   IONotifier importNotifier;
   if(fileExtension == Format.OBJ)
   {
      OBJ.ImportOptionsKit objOptionsKit = new OBJ.ImportOptionsKit();
      objOptionsKit.SetSegment(geometrySegment);
      importNotifier = OBJ.File.Import(filePath, objOptionsKit);
   }
   else if(fileExtension == Format.STL)
   {
      STL.ImportOptionsKit stlOptionsKit = new STL.ImportOptionsKit();
      stlOptionsKit.SetSegment(geometrySegment);
      importNotifier = STL.File.Import(filePath, stlOptionsKit);
   }
   else
   {
      HPS.Stream.ImportOptionsKit hsfOptionsKit = new HPS.Stream.ImportOptionsKit();
      hsfOptionsKit.SetSegment(geometrySegment);
      HPS.Stream.ImportNotifier hpsImportNotifier = HPS.Stream.File.Import(filePath, hsfOptionsKit);
      HPS.Stream.ImportResultsKit importResults = hpsImportNotifier.GetResults();
```

```
        importResults.ShowDefaultCamera(out defaultCamera);
        importNotifier = hpsImportNotifier;
    }

    importNotifier.Wait();
    if(importNotifier.Status() == IOResult.Failure || importNotifier.Status() ==
IOResult.UnableToOpenFile)
    {
        throw new HPS.IOException("Can't open the file.", IOResult.Canceled);
    }

    return new Document(filePath);
}
```

Метод RelativeOrbit класу CustomOrbitOperator

```
private bool RelativeOrbit(WindowPoint inLocation, KeyPath eventPath)
{
    viewKey.ShowCamera(out CameraKit camera);
    camera.ShowTarget(out Point target);
    eventPath.ConvertCoordinate(Coordinate.Space.World, centerOfRotation, Coordinate.Space.Camera,
out Point convertedCenterOfRotation);
    eventPath.ConvertCoordinate(Coordinate.Space.World, target, Coordinate.Space.Camera, out Point
convertedCameraTarget);

    Vector dollyDelta = convertedCenterOfRotation - convertedCameraTarget;
    WindowPoint newPoint = inLocation;
    OperatorUtility.ScreenToSphereMousePoint(newPoint, out Vector newSpherePosition);
    Vector rotationAxis = startSpherePosition.Cross(newSpherePosition);
    Vector mouseMoveVector = newPoint - startPoint;
    float distance = (float)mouseMoveVector.Length() * ORBIT_INTENSITY_MAGNIFIER;
    float tolerance = 0.0000001f;
    bool isRotationChanged = Math.Abs(rotationAxis.x) > tolerance || Math.Abs(rotationAxis.y) >
tolerance || Math.Abs(rotationAxis.z) > tolerance;
```

```
    if(isRotationChanged)
    {
        if(viewKey.GetDrawingAttributeControl().ShowWorldHandedness(out Drawing.Handedness
worldHandedness)
            && worldHandedness == Drawing.Handedness.Right)
        {
            rotationAxis.y *= -1;
            rotationAxis.z *= -1;
        }
        rotationAxis.Normalize();

        float angle = OperatorUtility.CalculateAngleOrbitOnPlane(rotationAxis,
OperatorUtility.ProjectedPlane.Plane_YZ);
        float panPhi = rotationAxis.x < 0 ? -angle * distance : angle * distance;
        angle = OperatorUtility.CalculateAngleOrbitOnPlane(rotationAxis,
OperatorUtility.ProjectedPlane.Plane_XZ);
        float panTheta = rotationAxis.y < 0 ? angle * distance : -angle * distance;
        angle = OperatorUtility.CalculateAngleOrbitOnPlane(rotationAxis,
OperatorUtility.ProjectedPlane.Plane_XY);
        float rollTheta = rotationAxis.z < 0 ? angle * distance : -angle * distance;

        camera.Dolly(dollyDelta.x, dollyDelta.y, dollyDelta.z);
        camera.Orbit(panTheta, panPhi);
        camera.Roll(rollTheta);
        camera.Dolly(-dollyDelta.x, -dollyDelta.y, -dollyDelta.z);
        viewKey.SetCamera(camera);
        GetAttachedView().Update();
    }

    startSpherePosition = newSpherePosition;
    startPoint = newPoint;
    return true;
}
```

Методи UpdateZoomLimit та ComputeNewField класу CustomZoomOperator

```
private void UpdateZoomLimit()
{
   view.GetAttachedModel().GetSegmentKey().ShowBounding(out BoundingKit modelBounding);
   modelBounding.ShowVolume(out SimpleSphere sphere, out SimpleCuboid boundingCuboid);
   float boundingDiagonalLength = (float)(boundingCuboid.max - boundingCuboid.min).Length(),
      minZoomLimitCoefficient = 0.001f,
      maxZoomLimitCoefficient = 200;
   minZoomLimit = boundingDiagonalLength * minZoomLimitCoefficient;
   maxZoomLimit = boundingDiagonalLength * maxZoomLimitCoefficient;
}
private CameraField ComputeNewField(Point newTarget, CameraKit oldCamera)
{
   CameraField field;
   oldCamera.ShowTarget(out Point target);
   oldCamera.ShowPosition(out Point position);
   oldCamera.ShowField(out field.Width, out field.Height);

   Vector oldDirection = target - position,
      newDirection = new Vector(newTarget) - new Vector(position);
   double oldLength = oldDirection.Length(),
      newLength = newDirection.Length();
   float ratio = (float)(newLength / oldLength);
   field *= ratio;
   return field;
}
```

Метод DefinePortfolio класу SetTexture

```
public PortfolioKey DefinePortfolio(string textureFilePath, Material.Texture.Parameterization
parameterization, out string textureName)
{
   string textureFileFormat = System.IO.Path.GetExtension(textureFilePath).TrimStart('.');
```

```
    if(!imageFormats.TryGetValue(textureFileFormat, out Image.Format format))
    {
        throw new IOException("Not supported image format.", IOResult.UnsupportedFormat);
    }

    // Importing the image and defining it in the GeometrySegment portfolio.
    Image.ImportOptionsKit importOptions = new Image.ImportOptionsKit();
    importOptions.SetFormat(format);
    ImageKit imageKit = Image.File.Import(textureFilePath, importOptions);
    PortfolioKey portfolioKey = viewer.ViewPort.TexturePortfolio;
    ImageDefinition imageDefinition = portfolioKey.DefineImage(textureFilePath, imageKit);

    // Defining a texture from the image.
    TextureOptionsKit textureOptionsKit = new TextureOptionsKit();
    textureOptionsKit.SetParameterizationSource(parameterization);
    textureName = textureFilePath + parameterization;
    portfolioKey.DefineTexture(textureName, imageDefinition, textureOptionsKit);
    return portfolioKey;
}
```