

User name:  
**Volodymyr Matiiievskiy**

Check date:  
**31.05.2023 21:05:52 EEST**

Report date:  
**31.05.2023 21:13:59 EEST**

Check ID:  
**1015352559**

Check type:  
**Doc vs Internet**

User ID:  
**100010994**

File name: **РОЗДІЛ 3 — копия (2)**

Page count: **71** Word count: **11285** Character count: **90123** File size: **3.29 MB** File ID: **1015020374**

Text modifications detected (similarity score might be affected)

## 2.8% Matches

Highest match: **1.1%** with Internet source ([https://org2.knuba.edu.ua/pluginfile.php/29517/mod\\_resource/content/4/OGL%20Lay..](https://org2.knuba.edu.ua/pluginfile.php/29517/mod_resource/content/4/OGL%20Lay..))

2.8% Internet sources

43

Page 73

No Library search was conducted

## 0.26% Quotes

Quotes

2

Page 74

Exclusion of references is off

## 0.22% Exclusions

Some exclusions were automatic (exclusion filters: matched word count less than **9 words** and **0%**)

0.22% Internet exclusions

126

Page 75

No Library exclusions

## Modifind

Text modifications detected. Find more details in the online report.

Replaced characters

24

Suspicious formatting

18 Pages





## ЗМІСТ

ВСТУП.....	4
<b>РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО МОДЕЛЮВАННЯ І ДИСКРЕТИЗАЦІЇ ПРОСТОРОВИХ ОБ'ЄКТІВ.....</b>	<b>10</b>
1.1. Аналіз відомих способів подання геометричних тіл.....	10
1.2. Опис топології області.....	17
1.3. Аналіз методів побудови сіток.....	19
1.4. Огляд існуючих програм для дискретизації.....	25
1.4. Висновки до розділу.....	30
<b>РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРЕПРОЦЕСОРА.....</b>	<b>31</b>
2.1. Логічне представлення статичної моделі структури програмної розробки.....	31
2.2. Логічне представлення моделі поведінки програмної розробки.....	36
2.3. Фізичне представлення моделі програмної розробки.....	42
2.4. Архітектура програмного забезпечення препроцесора.....	43
2.5. Висновки до розділу.....	44
<b>РОЗДІЛ 3. РОЗРОБКА ПРЕПРОЦЕСОРА ДЛЯ СКІНЧЕННО-ЕЛЕМЕНТНОГО МОДЕЛЮВАННЯ.....</b>	<b>45</b>
3.1. Обґрунтування вибору середовища розробки препроцесора.....	45
3.2. Розробка інтерфейсу.....	52
3.3. Побудова геометричних об'єктів в препроцесорі.....	59
3.4. Генерація сітки в препроцесорі.....	63
3.5. Формати файлів в препроцесорі.....	66
Висновки до розділу.....	67
<b>ВИСНОВКИ.....</b>	<b>68</b>
<b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....</b>	<b>70</b>
<b>ДОДАТОК А.....</b>	<b>75</b>

## ВСТУП

Дослідники та інженери стикаються з багатьма завданнями, які не можуть бути аналітично вирішені або вимагають великих витрат на експериментальне втілення. Єдиним швидким способом аналізу інженерних проблем є використання комп'ютерного математичного моделювання.

Прогрес у розробці чисельних методів значно розширив коло завдань, які можуть бути проаналізовані. Результати, отримані на основі цих методів, використовуються практично в усіх галузях науки і техніки. У аналізі конструкцій найбільше застосування має метод скінчених елементів (МСЕ).

Поява методу скінчених елементів пов'язана з неможливістю аналітичного розв'язання більшості задач механіки деформованого твердого тіла і механіки руйнування. При розв'язанні задач МСЕ досліджувальний об'єкт представляється як сукупність дискретних підобластей - скінчених елементів (СЕ). Усередині кожного скінченого елемента шукана неперервна величина наближається сукупністю кусково-неперервних функцій. Шукана величина в межах СЕ визначається за допомогою значень цієї величини в кінцевому числі точок досліджуваного об'єкта - вузлових точок або вузлів, які, як правило, є характерними точками СЕ. При розв'язанні задачі шукана величина описується аналітичними залежностями, які моделюють властивості матеріалу (неперервність, ізотропність і т. д.) і закони механіки (закон рівноваги сил, варіаційні принципи). Це дозволяє отримати однозначний розв'язання задачі за заданими початковими і граничними умовами.

Розрахунок конструкцій за допомогою методу скінчених елементів можна розбити на три взаємопов'язаних послідовних процеси:

1. Підготовка початкових даних: це включає скінченно-елементну дискретизацію об'єкта, який підлягає розрахунку, визначення його топології та кінематичних і силових граничних умов, а також фізико-механічних характеристик матеріалу.

2. Чисельний розрахунок скінченно-елементної моделі: цей процес включає обчислення коефіцієнтів матриці жорсткості скінчених елементів, формування глобальної системи розв'язуваних рівнянь і її розв'язання.
3. Обробка результатів розв'язання: на цьому етапі обчислюються параметри напружено-деформованого та температурного стану конструкції, а їх результати представляються у вигляді таблиць, графіків або двовимірних/тривимірних зображень.

Ці процеси чисельної реалізації, як це зазвичай відбувається в автоматизованих розрахунках, виконуються трьома підсистемами: препроцесором, процесором і постпроцесором.

Препроцесор є однією з головних складових будь-якого програмного комплексу чисельного аналізу. Він автоматизує процес побудови геометричної моделі об'єкта дослідження та його дискретизації на скінчені елементи. Якість препроцесора має велике значення для загальної якості програмного комплексу в цілому.

Однією з головних вимог до препроцесора є його здатність ефективно та точно моделювати геометричну форму досліджуваного об'єкта. Він повинен мати можливість створювати, редагувати та маніпулювати геометричними об'єктами, такими як точки, лінії, поверхні та об'єми. Крім того, препроцесор повинен забезпечувати можливість встановлення граничних умов, визначення фізико-механічних властивостей матеріалу та встановлення параметрів розрахункової моделі.

Препроцесор для моделювання конструкцій повинен включати наступні функції:

1. Геометричне моделювання: Препроцесор повинен мати можливість створювати або імпортувати геометричну модель конструкції. Це може включати створення геометричних форм, визначення розмірів, розташування вузлів та з'єднань.

2. Матеріальні властивості: Препроцесор повинен дозволяти встановлювати матеріальні властивості еластомерних матеріалів, такі як модуль пружності, міцність, деформаційні характеристики та інші. Це дозволяє враховувати особливості матеріалу при моделюванні поведінки конструкції.
3. Дискретизація: Препроцесор повинен забезпечувати можливість розбиття геометричної моделі на скінчені елементи. Він повинен дозволяти вибирати типи елементів, встановлювати їх параметри та формувати зв'язки між елементами. Це важливо для створення скінченно-елементної мережі, яка відповідає властивостям конструкції.
4. Граничні умови: Препроцесор повинен дозволяти встановлювати граничні умови для конструкції, такі як закріплення, прикладення сил, температурні умови тощо. Це включає встановлення граничних умов для окремих вузлів або груп вузлів.
5. Підготовка вхідних файлів: Препроцесор повинен забезпечувати генерацію вхідних файлів, які містять інформацію про геометрію, матеріальні властивості, дискретизацію, граничні умови та інші параметри. Ці файли потім можуть бути використані для чисельного розрахунку.
6. Перевірка якості моделі: Препроцесор повинен мати функціонал для перевірки якості скінченно-елементної моделі. Це може включати перевірку на наявність необхідних з'єднань між елементами, перевірку граничних умов, переконання у відповідності моделі реальній геометрії та інші перевірки.
7. Візуалізація та відображення результатів: Препроцесор повинен мати можливість відображати геометричну модель конструкції, скінченну елементну мережу та інші атрибути моделі. Він також повинен дозволяти відображати результати чисельного розрахунку, такі як напруження, деформації, температурний стан

тощо. Це допомагає аналізувати та інтерпретувати результати моделювання.

8. Експорт та імпорт даних: Препроцесор повинен мати можливість експортувати скінченно-елементну модель та інші дані у різних форматах, які можуть бути використані іншими програмами для подальшого аналізу або обробки. Він також повинен підтримувати імпорт даних з інших програм для використання в препроцесорі.
9. Зручний інтерфейс користувача: Препроцесор повинен мати інтуїтивно зрозумілий та зручний інтерфейс користувача, який дозволяє легко виконувати всі необхідні функції. Це допомагає користувачеві швидко і ефективно створювати та налаштовувати скінченно-елементну модель.
10. Підтримка різних типів конструкцій: Препроцесор повинен бути здатний моделювати різні типи конструкцій, включаючи тверді тіла, пластини.
11. Обробка мережових вузлів: Препроцесор повинен мати функціонал для обробки мережових вузлів, які використовуються в певних типах конструкцій, наприклад, в електричних мережах або трубопроводах. Це може включати визначення вузлів, розташування, параметри з'єднання та інші атрибути.
12. Параметризація моделі: Препроцесор повинен мати можливість параметризувати модель конструкції, що дозволяє змінювати значення різних параметрів, таких як розміри, матеріальні властивості, граничні умови і т. д. Це дозволяє проводити аналіз моделі при змінних параметрах і оптимізацію конструкції.
13. Імпорт геометрії: Препроцесор повинен мати можливість імпортувати геометрію з різних форматів файлів, таких як CAD-формати (наприклад, STEP, IGES) або формати графічних об'єктів (наприклад, OBJ, STL). Це спрощує процес створення



геометричної моделі конструкції і дозволяє використовувати наявну геометричну інформацію.

14. Автоматична генерація мережі: Препроцесор може мати можливість автоматично генерувати скінченно-елементну мережу на основі заданої геометрії і параметрів. Це дозволяє швидко створювати деталізовану модель з необхідною кількістю елементів.

Однією з ключових проблем, що виникають при використанні методу скінчених елементів, є побудова дискретної моделі механічної системи, яка підлягає дослідженню. Використання препроцесора покращує ефективність та надійність процесу моделювання, дозволяючи швидко вносити зміни в модель, виконувати різноманітні розрахунки та аналізувати результати. Це робить його незамінним інструментом для інженерів та дослідників, зайнятих розробкою та аналізом конструкцій. Тому розробка препроцесора для скінченно-елементного моделювання конструкцій є актуальною темою.

**Об'єкт дослідження** – геометричне моделювання конструкцій та дискретизація геометричних областей на скінчені елементи.

**Предмет дослідження** – побудова моделі конструкції та генерація розрахункових сіток для скінченно – елементного моделювання конструкцій.

**Мета роботи** – аналіз методів побудови комп'ютерних моделей геометричних областей та їх дискретизації на скінченні елементи заданої форми та розробка препроцесора для скінченно -елементного моделювання конструкцій.

**Методи дослідження:** методи обчислювальної математики і комп'ютерної графіки.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- Провести аналіз обчислювальних комплексів на основі MCE;
- Проаналізувати методи побудови геометричних моделей об'єктів і поширені формати їх опису;

- Проаналізувати основні поширені методи й алгоритми дискретизації плоских та просторових областей;
- Проведення моделювання та аналізу програмного забезпечення препроцесора з метою побудови геометрії та автоматизації створення дискретної (скінченно-елементної) моделі конструкцій;
- Розробити і реалізувати препроцесор для побудови геометрії та автоматизації створення дискретної (скінченно-елементної) моделі конструкцій.

Практичною цінністю роботи є розроблений препроцесор для побудови геометрії та автоматизації створення дискретної (скінченно-елементної) моделі конструкцій. У першому розділі було проведено дослідження найпоширеніших програмних комплексів для моделювання та аналізу складних інженерних конструкцій, як вітчизняних, так і зарубіжних. Розглянуто основні методи побудови геометричних моделей об'єктів і поширені формати, що використовуються для їх опису. Також були проаналізовані основні підходи, які застосовуються в сучасних системах автоматизованого проектування для твердотільного моделювання геометричних об'єктів, а також досліджені поширені методи та алгоритми дискретизації плоских та просторових областей. У другому розділі було здійснено аналіз процесу розробки препроцесора з метою побудови геометрії та автоматизації створення дискретної (скінченно-елементної) моделі конструкцій. Також було проведено моделювання та надано детальний опис архітектури розробленого додатку. У третьому розділі виконано обґрунтування вибору середовища розробки додатку та мови програмування. Описано алгоритм побудови геометрії розрахункових областей та генерації сіток у препроцесорі. У препроцесорі було розроблено два модулі для роботи: модуль "Геометрія" для створення простих геометричних сутностей та модуль "Сітка" для створення 1-, 2- та 3-мірних сіток і їх оптимізації. Для реалізації програми була обрана мова програмування C++ у середовищі

Microsoft Visual Studio 2022. При розробці засобів відображення та візуалізації використовується графічний інтерфейс OpenGL.

## **РОЗДІЛ 1.**

### **АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО МОДЕЛЮВАННЯ І ДИСКРЕТИЗАЦІЇ ПРОСТОРОВИХ ОБ'ЄКТІВ**

#### **1.1. Аналіз відомих способів подання геометричних тіл**

Загальна класифікація основних підходів до моделювання геометричних тіл була дана в роботі [21], там же були сформульовані і перераховані нижче основні вимоги до твердотілих моделей:

- Показність - модель повинна бути придатна для опису безлічі фізичних об'єктів досить широкого класу;
- Однозначність - один і тільки один об'єкт повинен відповідати кожному конкретному опису, щоб не виникало питання про те, який власне об'єкт представляється;
- Унікальність - в ідеалі бажано, щоб кожен модельований об'єкт описувався в обраній схемі подання єдиним чином. Ця властивість забезпечує легкість розрізнення двох різних об'єктів, але на практиці є важко досяжною;
- Точність - бажано, щоб подання точно описувало форму об'єкта, без апроксимації;
- Коректність - в ідеалі, схема подання повинна допускати введення тільки тих описів, які задовольняють всім критеріям визначення твердотілих моделей;
- Замкнутість - подання має бути замкнутим щодо допустимих в ньому операцій. Для геометричного моделювання велике значення мають геометричні перетворення та теоретико - множинні операції;
- Компактність - модель повинна мати компактний опис, економне з точки зору даних, необхідних для її повного завдання;
- Ефективність - модель повинна допускати ефективні алгоритми її обробки, що охоплює введення/висновок, обчислення основних відносин і операцій, редагування і модифікацію, візуалізацію, обчислення метричних характеристик і т.д.

11

Задовольнити всім цим вимогам одночасно важко, тому різні схеми подання будуються на основі деякого компромісу і мають свої переваги і недоліки.

В даний час можна виділити наступні найбільш поширені методи представлення геометричних тіл [4, 21, 22, 23]:

- параметризовані примітиви,
- граничне уявлення;
- конструктивна геометрія;
- кінематичний метод ("свіппінг" або замітання);
- розкладання на елементи;
- просторове перерахування;
- неявні моделі.

Моделювання за допомогою бібліотеки параметризованих примітивів застосовується зазвичай в різних прикладних областях, де набір використовуваних геометричних об'єктів обмежений і стандартизований. Такий підхід відповідає груповій технології, застосовуваний в автоматизованому проектуванні. Він часто використовується для таких складних і в той же час стандартизованих об'єктів, як болти, зубчасті шестерні і т.п., які втомливо визначати за допомогою булевих комбінацій більш простих об'єктів, але можна охарактеризувати набором високорівневих параметрів (діаметр і кількість зубів для шестерні і так далі).

При використанні граничного опису геометричне тіло задається замкнутою поверхнею, що обмежує це тіло. При цьому для опису форми області використовують як алгебраїчні поверхні 1 -го і 2 -го порядку, так і кусково -аналітичні поверхні. За допомогою граничного опису може бути представлений широкий клас об'єктів. У граничному поданні в явному вигляді міститься інформація про поверхні тіла, дане подання ефективно при візуалізації та чисельному моделюванні. Основним недоліком методу є громіздкість даних, що описують модель і складність обчислення теоретико -множинних операцій.

У методі конструктивної геометрії складний об'єкт формується шляхом виконання теоретико-множинних операцій та операцій геометричних перетворень над простішими об'єктами, званими базовими елементами, або примітивами. Для того, щоб результат застосування теоретико-множинних операцій до твердотілих примітивів сам був твердим тілом, виконується його регуляризація, яка зводиться до замикання внутрішніх точок. Таким чином визначаються регуляризовані теоретико-множинні операції [4].

Модель конструктивної геометрії може бути описана за допомогою дерева побудови [24], в якому нетермінальні вузли представляють оператори, а листя - базові елементи. Метод конструктивної геометрії охоплює широке коло об'єктів, його зручно використовувати при введенні, тому що в ньому інформація про об'єкт представлена в досить структурованій формі, що забезпечує велику наочність і дозволяє уникнути помилок при описі об'єкта. Крім того, подання за допомогою дерева побудови ефективно при редагуванні. Існують алгоритми візуалізації тіл, представлених безпосередньо за допомогою методу конструктивної геометрії, проте часто на етапі отримання зображення здійснюють перехід до граничного опису.

У кінематичному методі [25] двовимірна область представляється як слід рухомої у просторі кривої, а тривимірна область - як слід рухомого двовимірного тіла або перетину. Подальшим розвитком кінематичного методу є, так званий, плазовий метод, в якому об'єкт, що рухається по складній траєкторії, не є жорстким, а деформується відповідно до залежностей тій чи іншій мірі складності. Кінематичний метод зручний при введенні, дозволяє ефективно виконувати ряд обчислювальних операцій, проте він охоплює обмежений клас об'єктів і не передбачає виконання теоретико-множинних операцій.

У методі розкладання на елементи модельований об'єкт представляється як об'єднання деякого набору неперекриваючихся елементів - примітивів. Цей метод близький до методу конструктивної геометрії, проте

у ньому, в порівнянні з останнім, звужене коло перетворень, виконуваних над примітивами в процесі завдання об'єкта.

Дане обмеження значно ускладнює формування опису об'єкта, однак при цьому **зпрошується** виконання операції розбиття, що ефективно для візуалізації та чисельного моделювання.

У методі просторового перерахування об'єкти задаються шляхом перерахування всіх тих позицій простору, які вони займають. При цьому вважають, що простір складено з елементарних осередків, що примикають один до одного, а об'єкт представлений як об'єднання деякої кінцевої безлічі таких осередків. В якості осередків використовують зазвичай замкнуті квадрати (куби) фіксованого розміру зі сторонами, паралельними координатним осям (площинам). Відомі дві **разновидності** методів **просторового перерахування** - воксельні **представлення** і **представлення** за допомогою квадратичних (**вісімкових**) дерев, які **відрізняються способом** описання **сукупності** осередків, які займає модельований об'єкт. У першому із зазначених методів для цих цілей використовуються матричні структури, а в другому - ієрархічні структури - квадратичні і восьмигранні дерева [26]. Розроблено спеціальні дуже ефективні алгоритми для виконання теоретико-множинних операцій і візуалізації воксельних моделей і моделей на основі квадратичних (вісімкових) дерев.

У неявних моделях тіло задається за допомогою деякої процедури, яка дозволяє для кожної точки простору моделювання визначати її приналежність описуваному об'єкту [27, 28]. Ця процедура може бути реалізована різними способами. Наприклад, у вузлах регулярної сітки, що охоплює цілком модельований об'єкт, можна задати предикат приналежності й обчислювати приналежність інших точок за допомогою інтерполяції [29]. Для неявного опису геометричних тіл використовують також функцію відстані, значення якої **в кожній точці дорівнює відстані від цієї точки до** модельованого об'єкта [28]. Така функція може бути також визначена на базі воксельного подання. Нарешті, існує підхід, який в загальному вигляді неявно описує довільний

геометричний об'єкт в просторі  $E_n$  за допомогою функції  $f(x_1, x_2, \dots, x_n)$  координат точок  $(x_1, x_2, \dots, x_n)$  у вигляді нерівності  $f(x_1, x_2, \dots, x_n) \geq 0$ , так що функція  $f()$ , що визначає об'єкт, приймає позитивні значення в точка, що лежать в середині об'єкта, нульове - в граничних точках і негативне - в точках поза об'єктом. Такий підхід отримав назву функціонального подання (F-ger) [2]. Як і в методі конструктивної геометрії, складний F-ger об'єкт, може формуватися з простих за допомогою теоретико -множинних операцій і геометричних перетворень.

Аналітичний вид функції, що описує результат теоретико -множинних операцій, може бути знайдений на основі теорії R- функцій [30]. Застосування R- функцій дозволяє отримувати опис поверхні складного геометричного тіла в неявному вигляді  $f(x, y, z) = 0$ . В даний час розроблені методи параметризації таких поверхонь [31, 32], що забезпечують побудову графічних відображень тіл, представлених за допомогою функціональних моделей. Функціональне уявлення узагальнює різні способи неявного завдання геометричних тіл, в рамках нього реалізуються різноманітні операції, в тому числі замітання, декартовий твір, метаморфозіс та ін.

Аналізуючи описані уявлення з точки зору зазначених вище вимог, що пред'являються до твердотілих моделей, можна відзначити наступне. Серед перерахованих уявлень низьку точність мають методи просторового перерахування, точність граничних уявлень, конструктивних моделей і розкладання на елементи залежить від форми сегментів поверхонь і об'ємних примітивів. При цьому найбільш широке коло об'єктів може бути представлене методами просторового перерахування, граничним поданням, функціональним і конструктивним методами. Методи замітання і розкладання на елементи, а також представлення екземплярами примітивів вельми обмежені в сенсі безлічі представимих тіл. Унікальність уявлення практично гарантують тільки воксельні моделі та вісімкові дерева при додатковому підрозбитті. Серед всіх уявлень, найбільш важко оцінити коректність для граничного подання, де не тільки структури даних, що



представляють вершини, ребра і грані можуть бути суперечливими, але також грані або ребра можуть перетинатися. Складність аналізу модельованого об'єкта на предмет відповідності визначенню твердого тіла характерна і для функціонального подання. Більш легко перевірити коректність для конструктивної геометрії і розкладання на елементи. Найбільш просто оцінити коректність моделей просторового перерахування. Регуляризовані теоретико - множинні операції не визначені для подання екземплярами примітивів, кінематичних моделей і розкладання на елементи. Решта моделей замкнуті щодо цих операцій.

Найбільш компактний опис мають уявлення на базі параметричних примітивів, функціональні, конструктивні та кінематичні моделі. Описи моделей просторового перерахування громіздкі, але мають просту структуру. Найбільш складно і громіздко описуються граничні подання. Ефективність реалізації операцій сильно відрізняються у різних геометричних моделей.

Візуалізація та чисельні розрахунки найбільш просто реалізуються для моделей просторового перерахування і розкладання на елементи.

Полігонізація поверхні, необхідна для відображення 3D об'єктів, найбільш просто будується для граничних моделей і кінематичних моделей. Обчислення кордону є трудомісткою операцією для конструктивної геометрії і особливо для функціонального подання.

Що ж до регуляризованих булевих операцій, то вони природним чином виконуються для конструктивної і функціональної моделей, порівняно легко обчислюються для воксельних моделей і моделей на основі вісімкових дерев і значно більш складно реалізуються для граничних моделей. Питання про приналежність точки модельованого об'єкту найбільш просто вирішується для функціонального уявлення і методу просторового перерахування. Набагато складніше це завдання вирішується для інших моделей.

Зазначена класифікація відображає принципові підходи до моделювання тіл. В рамках кожного з перерахованих типів існує безліч конкретних моделей, що відрізняються способом реалізації. Жоден із

зазначених способів не може вважатися повністю універсальним, ефективність тієї чи іншої моделі залежить від операцій, які необхідно виконувати над об'єктами в процесі моделювання. Тому вибір моделі залежить від прикладної задачі. Перераховані методи опису геометричних тіл доповнюють один одного, тому найбільш ефективним часто виявляється спільне використання декількох уявлень, що припускає їх взаємні перетворення. Проте реалізації моделей різних типів сильно відрізняються по використовуваному математичному апарату, структурам даних і алгоритмам обробки.

При вирішенні багатьох прикладних задач виникає необхідність розбиття модельованих об'єктів і побудови дискретних моделей, які з заданою точністю апроксимують форму вихідного об'єкта. У дискретному поданні важливу роль відіграють засоби опису внутрішньої структури модельованих об'єктів. Для опису дискретних моделей використовуються різні стратифікації, їх огляд подано в роботі [33].

Загалом стратифікація - це опис об'єкта у вигляді об'єднання сукупності непересічних страт, кожна страта є різноманіттям у просторі  $E_n$ , кордон кожної зв'язної компоненти страти має розмірність нижче, ніж розмірність самої страти і в кожній обмеженій множині простору моделювання міститься кінцеве число страт [34]. Окремими випадками стратифікацій є геометричні комплекси [35, 36, 37].

У рамках наведеної вище класифікації дискретні моделі можна віднести до типу розкладання на елементи. Опис просторової структури також важливий при завданні складових кривих і поверхонь, кусочно-аналітичному поданні граничних моделей і при завданні розмірно неоднорідних об'єктів. Для цих цілей також застосовуються різні стратифікації [39, 40]. Зокрема, в роботі [40] було показано, що геометричні комплекси дозволяють єдиним чином представляти граничні моделі, кінематичні моделі та моделі просторового перерахування. Відзначимо також, що в залежності від способу опису примітивів в конструктивному

представленні його можна звести відповідно до граничної моделі, моделі просторового перерахування або функціональної моделі. Таким чином, в якості основної альтернативи у виборі уявлення геометричних тіл можна розглядати модель на основі комплексів, звану також клітинною, і функціональне уявлення. Перша з цих моделей задає явний опис об'єкта, а друга - неявний. Ці моделі принципово відрізняються, однак вони не замінюють один одного, **кожна з них має свої переваги і недоліки.** Так функціональне уявлення забезпечує компактний опис складної, можливо багатовимірної, геометрії, воно придатне для опису об'єктів різної розмірності, включаючи розмірно неоднорідні об'єкти та об'єкти, які не є різноманітні. Однак воно не містить даних про топологічну структуру об'єкта, що викликає проблеми при виконанні багатьох чисельних процедур і операцій. У свою чергу клітинне уявлення, засноване на топологічному розбитті, дає повний опис топологічної структури об'єкта, дозволяє виділяти топологічно однорідні компоненти в складі складних об'єктів. Однак ступінь деталізації клітинного уявлення, що визначається топологічним розбиттям, часто виявляється надлишковою з точки зору опису геометричних властивостей. У додатках, які не використовують повною мірою інформацію про топологічну структуру об'єкта, така надмірна деталізація знижує ефективність роботи з геометричним об'єктом.

Враховуючи вище викладене, можна зробити висновок, що при моделюванні складних неоднорідних об'єктів різні уявлення можуть вдало доповнювати один одного. На практиці для забезпечення можливості спільного використання різних уявлень необхідно забезпечити узгодженість специфікацій різних моделей з урахуванням їх взаємних перетворень.

## 1.2. Опис топології області

Одним з найважливіших елементів чисельного розрахунку напружено-деформованого стану (НДС) тіла, що деформується, є побудова адекватної геометричної моделі досліджуваної області. Як правило, на практиці доводиться мати справу з об'єктами дуже складної конфігурації, що істотно

18

ускладнює побудову таких моделей. В той же час від точності побудованої геометричної моделі багато в чому залежатиме якість отриманого чисельного результату.

Нині існують різні способи опису геометрії модельованої області. Одним з найчастіше використовуваних підходів являється використання спеціалізованих CAD -систем, що дозволяють побудувати необхідну топологічну модель, як деяку сукупність базових геометричних примітивів. Такий підхід застосовується, наприклад, в системах ANSYS, COSMOS і COSAR [12]. У препроцесорах цих систем є бібліотеки таких графічних примітивів, як точка, лінія, сплайн, ламана, коло, сфера, конус, куб та ін., над якими визначені ейлереві операції їх об'єднання, перетини і віднімання, що дозволяють задати практично довільну область. Альтернативним є підхід, що полягає в параметричному описі геометрії модельованої області за допомогою деякої мови опису топології області. Наприклад, система геометричного моделювання NETGEN [12] використовує для опису топології області мову CSG (ConstructiveSolidGeometry), що дозволяє описувати невеликі і середні плоскі і тривимірні області. У CSG в текстовому форматі ASCII можна описувати довільну просторову область, як логічну комбінацію наступних базових геометричних примітивів:

- площина;
- циліндр;
- сфера;
- еліптичний циліндр;
- еліпсоїд;
- конус;
- паралелепіпед;
- многогранник.

Геометрія об'єкту визначається як деяка сукупність ейлеревих операцій (об'єднання, перетин і доповнення) над вищеописаними примітивами.

Іншим поширеним способом опису топології тривимірних об'єктів є так званий формат стерео літографії (STL - формат). Цей формат застосовується в автоматизованих системах проектування для опису тривимірних моделей і є для них найбільш часто-використовуваним стандартним форматом.

Інформація про об'єкт в STL – файлі включає список трикутних граней, які описують поверхню його твердотілої моделі із заданою точністю, і може бути представлена у вигляді текстового (ASCII) або бінарного файлу. Текстове представлення STL - файлу повинне починатися ключовим словом SOLID і закінчуватися ENDSOLID. Між цими програмними дужками наводиться опис трикутників. Опис кожного трикутника включає завдання одиничного вектору нормалі, спрямованого від його поверхні, після чого слідує список тривимірних координат усіх вершин. Усі координати представлені в ортогональній декартовій системі координат і записані у вигляді дійсних чисел.

На рис. 1.1 наведений приклад опису одного трикутника в STL -форматі:



Рис.1.1. Опис трикутника в STL -форматі.

При правильному описі тріангульованої поверхні усі сусідні трикутники повинні мати по дві загальні вершини.

### 1.3. Аналіз методів побудови сіток

Процес побудови сіток може розглядатися як перетворення геометричних моделей. Методи дискретизації істотно залежать від способу подання об'єкту, що розбивається, і від обмежень, накладених прикладним завданням на дискретну модель.

Огляд алгоритмів побудови сіток можна знайти зокрема в роботах [40, 41]. Серед різних способів генерації сіток можна виділити наступні групи найбільш поширених методів:

- методи тетраедризації (тріангуляції);
- методи побудови неструктурованих гексагональних (чотирикутних) сіток;
- кінематичні методи або методи об'єднання перетинів;
- методи накладення неузгоджених з формою області сіток;
- методи відображень геометрично нерегулярних областей в області правильної форми;
- методи оптимізації.

Аналіз різних методів тріангуляції і тетраедризації дається в роботах [42, 43]. Серед методів тріангуляції можна виділити дві основні групи. У методах першої групи вихідною інформацією для генерації сіток є набір вузлів, які потім з'єднуються, утворюючи трикутні елементи або тетраедри. Найбільш часто при цьому використовується метод Делоне, в якому вузли **з'єднуються** таким чином, що всередину кола (кулі), описаного навколо формованого трикутника (тетраедра), не потрапляють ніякі інші вузли сітки, відмінні від вершин даного елемента.

Існують різні способи початкового вибору вузлів. Одні автори пропонують спочатку розміщувати вузли на межі області, а потім генерувати внутрішні вузли випадковим чином, залишаючи тільки ті з них, які задовольняють заданій щільності сітки. У ряді робіт спочатку вводиться набір горизонтальних прямих, що покриває всю область рішення. Потім вузли рівномірно розподіляються на відрізках цих прямих, що лежать всередині області. У тривимірному випадку для автоматичного завдання вузлів пропонується також використовувати представлення області с допомогою вісімкових дерев.

До другої групи методів триангуляції відносяться так звані методи декомпозиції. У методах декомпозиції можна виділити рекурсивні та ітераційні методи.

У рекурсивних методах вихідна область спочатку розбивається на підобласті, які потім діляться в відповідності з тим же самим алгоритмом розбиття. Процес продовжується до тих пір, поки розміри отриманих підобластей не відповідатимуть необхідної щільності сітки. Цей метод дозволяє будувати сітки в областях, які є багатокутниками, які в загальному випадку можуть бути і криволінійними. Вихідна груба сітка виходить шляхом побудови діагоналей, що з'єднують несуміжні вершини багатокутника. Елементи грубої сітки потім подрібнюються, для цього вводяться додаткові вузли, що розміщуються на кордонах елементів, ці вузли з'єднуються один з одним відрізками, що лежать всередині елементів. Даний метод допускає узагальнення й на тривимірний випадок.

В ітераційних методах декомпозиції розбиття здійснюється таким чином, що на кожному кроці будується один або кілька елементів результуючої сітки. Процес продовжується до тих пір, поки елементи сітки не заповнять всю розрахункову область. Методи цієї групи дозволяють будувати, як правило, трикутні і тетраедральні сітки. В ітераційних методах існує можливість управління розмірами і формою елементів безпосередньо в процесі побудови сітки, що є їх безперечною перевагою порівняно з рекурсивними методами.

Ітераційні методи універсальні і, як правило, застосовуються для областей досить довільної форми. Саме тому ітераційні методи в основному і використовуються в автоматичних програмних комплексах. Недоліком цього класу методів є ресурсомісткість, істотно більш повільна швидкість роботи (у порівнянні з прямими методами) і менша надійність.

Ітераційні методи через свою універсальність отримали найбільший розвиток. Розроблено кілька різних підходів, які можна розділити на три

підкласти: методи граничної корекції, методи на основі критерію Делоне і методи вичерпання.

Методи граничної корекції є найшвидшими з ітераційних методів, але, на жаль, мають ряд невикорінних недоліків. Побудова сіток в цих методах здійснюється в два етапи. На першому етапі проводиться триангуляція якоїсь простої "супер- області", повністю включає в себе задану область. Як правило, ця супер- область являє собою паралелепіпед (через простоту триангуляції), триангуляція якого здійснюється на основі одного з численних шаблонів. На другому етапі всі вузли отриманої сітки, що лежать поблизу кордону заданій області, проектується на поверхню кордону; а вузли, що лежать поза заданій області - видаляються. Щоб компенсувати неминучі геометричні спотворення елементів сітки поблизу кордонів, часто додатково проводять ще один етап - етап оптимізації сітки, що в підсумку дозволяє отримати досить хороші результати.

Очевидно, що даний метод не можна застосовувати для дискретизації областей із заданою триангуляцією кордонів. Це істотне обмеження, а також інші складності знижують популярність методу, зводячи нанівець його основну перевагу - високу швидкість роботи.

Сутність методів вичерпання полягає в послідовному "вирізання" із заданої області фрагментів тетраедріческой форми доти, поки вся область не опиниться "вичерпана". В англомовній літературі цей метод отримав назву "advancing front", що також добре відображає ідею методу. Вихідними даними на кожній ітерації є "фронт", тобто триангуляція кордону ще не "вичерпана" частина області. Кожен трикутник цієї триангуляції є основою області тетраедра, який вилучається, причому на кожній ітерації може вилучатися або один тетраедр, або відразу цілий шар тетраедрів. Після вилучення тетраедра "фронт" оновлюється, після чого відбувається перехід до наступної ітерації.

Методи вичерпання універсальні і можуть бути використані для областей довільної форми та конфігурації (навіть для незв'язних областей), що пояснює їх популярність. Зокрема, саме ці методи використовуються в



програмному комплексі ANSYS. Разом з тим слід відзначити їх високу ресурсомісткість і низьку швидкість роботи.

Методи на основі критерію Делоне часто називають просто методами Делоне, хоча це не зовсім коректно, оскільки сам Б.Н. Делоне ніяких методів не розробляв, а лише запропонував простий і ефективний критерій, що використовується при установці зв'язків між вузлами. Відповідно, ідеєю цього класу методів є розміщення в заданій області вузлів і подальша розстановка між ними зв'язків згідно з критерієм Делоне (чи іншому схожому критерію).

Неструктуровані гексагональні сітки будуються зазвичай на основі симпліціальних сіток шляхом об'єднання сусідніх елементів [40].

Кінематичний метод дозволяє будувати сітки в **областях**, заданих методом замітання [45]. При дискретизації області спочатку розбивається рухома крива (двовимірний розтин), а потім відповідні один одному точки (відрізки) на сусідніх за часом кривих (перетинах) з'єднуються один з одним, утворюючи елементи двовимірної (тривимірної) сітки. Кінематичний метод є окремим випадком загального методу об'єднання перерізів. У методі об'єднання перерізів передбачається відома довільна послідовність отриманих яким-небудь способом перерізів об'єкта, який розбивається. Тоді, якщо на перетинах вдається побудувати ізоморфні сітки, то з'єднуючи відповідні один одному елементи перерізів, можна отримати чотирикутну або призматичну сітку, яка є дискретизацією розглянутого об'єкта. Даний спосіб є досить ефективним, однак він застосовується до обмеженого кола об'єктів. У деяких випадках, коли сітки на сусідніх перерізу не є ізоморфними, для дискретизації простору між такими перерізами використовуються методи триангуляції.

При використанні неузгоджених сіток вихідна область покривається деякою регулярною сіткою. Сіткові елементи всередині області залишаються без змін, а елементи, що лежать на кордоні і поза області ігноруються. У результаті виходить сітка зі ступінчастим або ламаним кордоном, для

згладжування якої можуть застосовуватися різні підходи. Зокрема, проєціювання точок ступінчастого кордону на кордон області, або зсув прикордонних вузлів уздовж ліній сітки до перетину з кордоном.

Ефективний підхід до побудови неузгоджених сіток пов'язаний з використанням представлення області з допомогою вісімкових дерев. Цей метод дозволяє в автоматичному режимі здійснювати дискретизацію складних тривимірних об'єктів [44].

Для генерації сіток активно застосовуються також різні відображення [45, 46], що дозволяють перетворювати сітки, поставлені на областях правильної форми, сітки, що покривають області складної форми. В даний час найбільше поширення одержали конформні відображення і трансфінитні перетворення. Методи, засновані на використанні різних відображень, що забезпечують побудову, як правило, якісних розрахункових сіток. Однак застосування подібних алгоритмів пов'язана з істотними обмеженнями на форму вихідної області і на спосіб подання цієї області, що призводить до необхідності індивідуального підходу до вирішення кожної нової задачі.

Методи відображень дозволяють будувати блочно-структуровані сітки в галузях, визначених методом розкладання на елементи, що припускає завдання галузі у вигляді об'єднання макроблоків певної форми. Для автоматичного розбиття складних областей на макроблоки застосовується також перетворення, яке дозволяє виділити кістяк об'єкта, що складається з безлічі точок, кожна з яких є рівно віддаленою від найближчих до неї точок межі об'єкта [30].

Методи оптимізації сіток використовуються для покращення та адаптації до умов поставленої задачі вже сформованих яким-небудь способом сіток [31, 32, 33]. У загальному вигляді суть цих методів можна сформулювати наступним чином:

$$\text{необхідно знайти } E = \min \left( \max_i E_i \right), i=1,2,\dots,N,$$

де  $E_i$  - якась міра помилки, що дозволяє кількісно оцінити якість кінцево-різницевої або скінчено-елементної апроксимації -  $E_i$ -помилка на  $i$ -му

25

елементі,  $N$  - загальна кількість елементів. Міра  $E$  може бути обрана самими різними способами, вона може залежати тільки від геометричних характеристик елементів або ж враховувати характер прикладної задачі. При оптимізації сіток або варіюють тільки координатами вузлів, залишаючи топологію без зміни, або допускають зміну топології, в цьому випадку можливе розбиття або навпаки укрупнення деяких елементів, зміна конфігурації зв'язків між вузлами або введення додаткових вузлів на кордонах і всередині елементів.

Кожен з перерахованих методів побудови сіток розрахований на певну модель подання розрахункової області [34]. Так при триангуляції, заснованої на методах декомпозиції, використовується граничний опис області. При побудові неузгоджених сіток застосовується завдання області методом просторового перерахування або граничним методом. Кінематичний метод придатний для розбиття тільки тих областей, опис яких також будується на основі кінематичного методу. Методи відображень припускають завдання розрахункових областей або методом розкладання на елементи, або граничним методом.

Слід зазначити, що застосування тих або інших способів побудови сіток істотний вплив мають обмеження, що накладаються методами, використовуваними для вирішення прикладних завдань.

Таким чином, вибір типу розрахункових сіток і алгоритмів їх побудови з одного боку залежить від прикладної задачі і методів її чисельного рішення, а з іншого боку визначається способом завдання геометричної моделі розрахункової області.

#### 1.4. Огляд існуючих програм для дискретизації

Сітка робить досить істотний вплив на точність рішення проєкційно - сітковими методами (а в деяких випадках і на збіжність методів). Якість сітки з точки зору точності рішення визначається трьома обставинами: розмірами елементів, їх формою і тим, наскільки добре сітка апроксимує вихідну

область. Таким чином, можна сформулювати такі вимоги до будь-якої системи автоматичної триангуляції (CAT).

1. Максимально точна апроксимація кордонів області та її внутрішніх обмежень: лінії з'єднань поверхонь повинні бути апроксимовані ланцюжками ребер сітки, а самі поверхні - безліччю плоских трикутних граней.
2. Форма елементів повинна бути по можливості близька до форми правильного симплекса.
3. CAT повинна дозволяти контролювати розміри елементів сітки, щоб можна було забезпечити згущення сітки в потрібних областях.

Важливою також є можливість перевірки якості та коректності побудованої сітки.

Нижче наводиться огляд різних програмних пакетів, призначених для дискретизації складних просторових областей (вибірка обмежена програмами з відкритим кодом, повністю безкоштовними програмами та програмами, безкоштовними для академічного використання).

Пакет Geompack розроблений доктором Баррі Джо, відомим своїми роботами в області дискретизації методами на основі критерію Делоне. Перші версії пакету з'явилися більше 20 років тому і мали - вид програмної бібліотеки на мові FORTRAN77 (старі версії пакету поширюються з відкритим кодом). Тепер Geompack є потужною консольною програмою для виконання різних операцій з сітками - побудови, згущення, оптимізації, перебудови і т.п. В основі алгоритму використовується метод корекції спільно з методом на основі критерію Делоне.

Пакет Geompack безкоштовний для академічного використання. У комплекті з пакетом поставляється докладна документація англійською мовою і безліч прикладів. Автор здійснює обмежену підтримку користувачів по електронній пошті. Візуальний інтерфейс і можливості по візуалізації відсутні. Імпорт та експорт даних здійснюється через текстові та бінарні

файли спеціальних (нестандартних) форматів. Пакет може бути використаний під операційними системами сімейств Unix і Windows.

До недоліків пакета слід віднести неможливість прямого контролю над розмірами елементів.

Домашня сторінка пакета в Інтернеті: <http://members.shaw.ca/bjoe/>

TetGen - програма, розроблена доктором Хан Сі з інституту Вейерштраса (WIAS). В основі алгоритму лежить метод на основі критерію Делоне. Для обліку внутрішніх обмежень використовується метод перебудови. TetGen володіє візуальним інтерфейсом (може запускатися і як консольна програма), дозволяє будувати та оптимізувати сітки, оцінювати їх якість, а також виробляти локальне згущення.

Пакет TetGen безкоштовний для академічного використання. У комплекті з пакетом поставляється докладна документація англійською мовою і безліч прикладів. Імпорт та експорт даних здійснюються через текстові файли різних (у тому числі і стандартних) форматів. Розміри елементів контролюються глобальним чином.

Пакет може бути використаний під будь-якими операційними системами, для яких існує компілятор C ++.

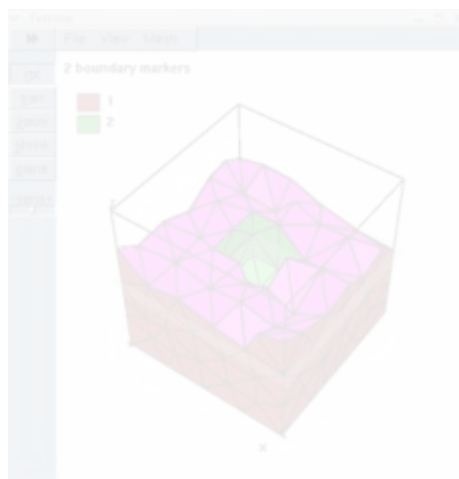


Рис. 1.1. Вікно програми TetGen

Домашня сторінка пакета в Інтернеті: <http://tetgen.berlios.de/>

Програмний комплекс "Mefisto", розроблений колективом французьких авторів під керівництвом Алана Перрона, призначений для вирішення різних завдань математичної фізики, і включає в себе модуль дискретизації, постпроцесингу, вирішувача і візуалізації. Модуль для дискретизації областей (Mefisto - maillages ) можна використовувати і окремо.

Програмний комплекс Mefisto призначений для роботи під управлінням операційних систем сімейства Unix. Поширюється разом з вихідними кодами (написаний на мовах FORTRAN77 і C) і демонстраційними прикладами. Документація французькою мовою.

Для триангуляції поверхні і внутрішній області використовується комбінований метод на основі критерію Делоне та алгоритму Quad - tree. Управління розмірами елементів здійснюється за допомогою спеціальних функцій (тобто згущення сітки можна отримати в процесі побудови). Область для триангуляції можна задавати за допомогою текстових файлів спеціального формату або візуального інтерфейсу. Експорт сітки проводиться в текстові файли спеціального формату.

Домашня сторінка програмного комплексу в Інтернеті: <http://www.ann.jussieu.fr/~perronnet/mefistoa.gene.html>

NetGen - один з найпопулярніших безкоштовних пакетів для дискретизації складних областей. Поширюється з відкритим кодом, супроводжується документацією англійською мовою, безліччю прикладів, а також має власний Інтернет -форум. Може бути використаний під будь-якими операційними системами, для яких існує компілятор C++. Володіє візуальним інтерфейсом, але може бути запущений і як консольна програма. Дозволяє згущувати та оптимізувати сітки.

NetGen призначений для генерації і відображення 3D- сіток і складається з декількох бібліотек. Основна бібліотека nglib реалізує безпосередньо генерацію сіток. Є два способи завдання простору, яке має бути заповнене тетраедальними елементами:

29

Завдання простору за допомогою операторів конструктивної блокової геометрії (Constructive Solid Geometr, CSG). У цій технології шуканий простір описується сукупністю примітивних об'єктів, над якими задаються такі операції як об'єднання, перетин, різниця.

Опис поверхні, що є кордоном шуканого простору, у файлі формату STL. У цьому форматі інформація про поверхні представляється у вигляді координат трикутних граней поверхні і їх нормалей.

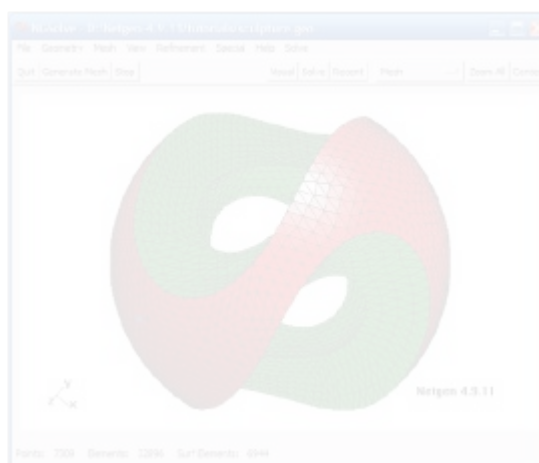


Рис. 1.2. Вікно програми NetGen

У пакеті використовується алгоритм на основі комбінованого методу вичерпання і методу на основі критерію Делоне. Імпорт та експорт даних може бути здійснений через найрізноманітніші формати, в тому числі і через стандартні. Зокрема, NetGen "знає" багато форматів CAD -систем.

Ймовірно, єдиний недолік пакета - неможливість прямого контролю над розмірами елементів.

Автор програми - Joachim Schöberl. Сторінка в Інтернеті: <http://www.hpfem.jku.at/netgen/>.

T3D - програма для побудови якісних сіток методом вичерпування. Складні області дискретизуються методом узгодження по кордону. Сама область задається як сукупність поверхонь кордону і обмежень. Дані

імпортуються та експортуються через текстові файли спеціального формату. Крім того, пакет вміє працювати безпосередньо з системами CAD.

Існують версії пакета для ОС Windows і Linux. Програма володіє консольним інтерфейсом. Для ОС сімейства Unix передбачена можливість візуалізації сіток за допомогою сторонньої програми (Elixir). Також можливий запис сіток в графічних форматах VRML 2.0 і VTK.

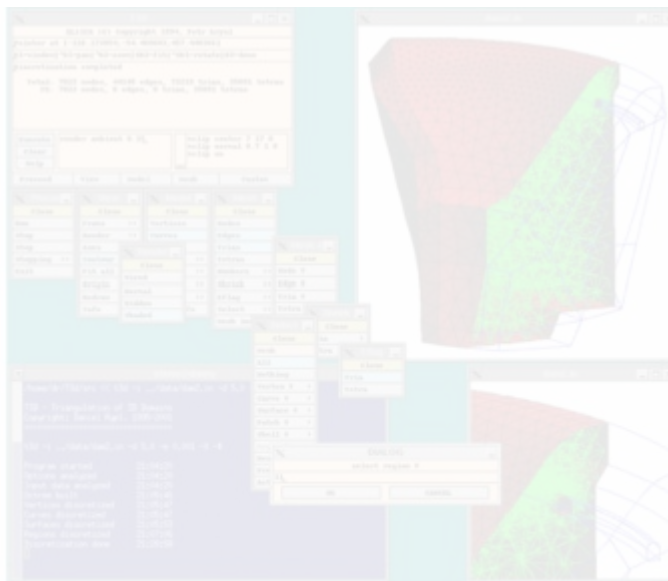


Рис. 1.3. Вікно програми T3D

Пакет безкоштовний для академічного використання, але автором підтримується дуже обмежено. Автор пакета - Daniel Ryppl. Домашня сторінка проекту: <http://ksm.fsv.cvut.cz/Mlr/t3d.html>

#### **1.4. Висновки до розділу**

В результаті огляду літературних джерел проаналізовано способи подання геометричних тіл та методи побудови сіток. Розглянуто та проаналізовано критерії якості побудованої сітки. Зроблено огляд сучасних програм для дискретизації просторових конструкцій.



## РОЗДІЛ 2.

### МОДЕЛЮВАННЯ ТА АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРЕПРОЦЕСОРА

#### 2.1. Логічне представлення статичної моделі структури програмної розробки

Повний проєкт програмної системи являє собою сукупність моделей логічного і фізичного уявлень, які повинні бути узгоджені між собою. У мові UML для статичного представлення моделей систем використовуються діаграми класів. Вони визначають склад класів об'єктів і їх зв'язків.

На діаграмі класів (Рис.2.1.) прямокутники, поділені на три частини – це класи, лінії – зв'язки між ними. Ім'я класу записано в верхній частині прямокутника. В другій і третій частині – відповідно список атрибутів і операцій, що мають специфікатори доступу.

Специфікація класів препроцесора.

**cPoint**- клас вершин елементів в  $n$ -мірному просторі, який містить координати точки, її індивідуальний номер в списку вершин усіх точок.

**cElement**- клас елементів в просторі  $R_n$ . Цей клас зберігає в собі список номерів вершин, їх кількість, індивідуальний номер елементу, координати його центру мас, об'єм, номери сусідніх елементів, що мають з ним загальну грань. Для моделювання завдань, що мають об'єкти з різними фізичними властивостями використовується змінна, що містить номер підобласті, якій належить елемент. Крім того клас елементу містить методи для динамічного обчислення граней.

**cFacet**- клас граней елементу, який містить список номерів вершин, що утворюють грань; їх кількість, площу, нормаль, орієнтовану з елементу хазяїна грані, номер елементу сусіда хазяїна грані, і номер елементу хазяїна грані. Грань обчислюється динамічно у міру використання для економії пам'яті.

**cMesh-** клас для зберігання і доступу до сітки в тілі програми. Цей клас містить список точок і елементів сітки. Іноді такі класи містять список граней. Проте, як показує практика, зберігання списків граней призводить до значної перевитрати пам'яті. Цей клас містить у тому числі методи для: читання і запису сіток в різних сіткових форматах, методи для реалізації згущення і розрідження сіток, які використовуються для адаптивних сіткових методів, методи по перебудові сіток.



Рис. 2.1. Діаграма класів препроцесора

Рис.2.2. Діаграма класів нових класів

Квадрати з вкладками вгорі є модулями. Показано «публічні» методи, атрибути та функції, які викликаються за класами.

Обчислювальна сітка міститься в об'єкті `chi_mesh::MeshContinuum`, який можна отримати за допомогою:

```
auto grid_ptr = cur_handler->GetGrid();
```

## Детальніше про структури даних Mesh

**chi mesh::MeshHandler**

Меши та операції з сітками обробляються `chi_mesh::MeshHandler`. Обробники сітки завантажуються в глобальну змінну `chi_meshhandler_stack`, а поточний обробник відстежується інший глобальної змінної, `chi_current_mesh_handler`. Якщо виконується будь-яка операція з сіткою, вона повинна поміщати новостворені об'єкти у стек в обробнику. На рисунку 2.3 показано узагальнену архітектуру.



Рис. 2.3. Огляд ієрархії сітки

Кожен обробник сітки обладнано однією сіткою для поверхні та однією сіткою для об'єму, обидва спочатку невизначені. Етап сітки поверхні можна розглядати як етап попередньої обробки.

Види поверхневих сіток:

**chi\_mesh::SurfaceMesherPredefined.** Пройти через препроцесор. Не виконує жодного зчеплення.

**chi\_mesh::SurfaceMesherDelaunay.** Змінює поверхневу сітку за допомогою триангуляції Делоне.

**chi\_mesh::SurfaceMesherTriangle.** Змінює сітку поверхні за допомогою Triangle 1.6.

Так само існують різні типи об'ємних сіток:

**chi\_mesh::VolumeMesherLinemesh1D.** Перетворює лінійні сітки на сляби.

**chi\_mesh::VolumeMesherPredefined2D.** Перетворює попередньо визначені сітки поверхонь у двовимірні трикутники, чотирикутники або багатокутники.

**chi\_mesh::VolumeMesherExtruder.** Екстрадує попередньо визначені поверхневі сітки в тривимірні трикутні призми, шестигранники або багатогранники.

**chi\_mesh::VolumeMesherPredefined3D.** Перетворює завантажені 3D-сітки на 3D-тетраедри, шестигранники або багатогранники.

**chi\_mesh::VolumeMesherPredefinedUnpartitioned.** Перетворює легку нерозділену сітку на правильну розділену сітку з повними деталями. Цей мешер забезпечує велику гнучкість для читання сіток із зовнішніх джерел.

Поверхневі сітки та об'ємні сітки призначаються обробнику як:

```
cur_handler->surface_mesher = new chi_mesh::SurfaceMesherPredefined;  
cur_handler->volume_mesher = new chi_mesh::VolumeMesherPredefined3D;
```

Щоб виконати поверхневі сітки, просто виконайте:

```
cur_handler->surface_mesher->Execute();  
cur_handler->volume_mesher->Execute();
```

Комірки в Chi-Tech є основними будівельними блоками для наукових обчислень на основі сітки. Деякі типи сіток показані на рисунку 2.4 та визначаються переліком, який вони містять, як визначено `chi_mesh::CellType`. Наразі підтримуються такі типи комірок:

- `chi_mesh::CellType::SLAB`
- `chi_mesh::CellType::TRIANGLE`
- `chi_mesh::CellType::QUADRILATERAL`
- `chi_mesh::CellType::POLYGON`
- `chi_mesh::CellType::TETRAHEDRON`
- `chi_mesh::CellType::HEXAHEDRON`
- `chi_mesh::CellType::POLYHEDRON`

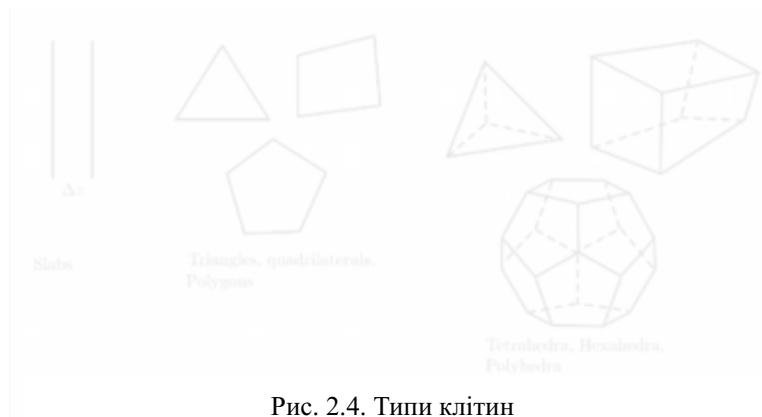


Рис. 2.4. Типи клітин

Клітини живуть у члені `local_cells` сітки, яка має тип об'єкта `chi_mesh::MeshContinuum::LocalCells` під егідою `native_cells` або `external_cells`. Гарантовано, що локальні клітини є повністю визначеними, тоді як нелокальні клітини, швидше за все, будуть клітинами-привидами. Найшвидший спосіб отримати доступ до комірки за допомогою її локального ідентифікатора.

```
auto cell = grid->local_cells[cell_local_id];
```

Об'єкт `local_cells` також сприяє ітератору.

```
for (auto cell = grid->local_cells.begin();
```

```
cell != grid->local_cells.end();
```

```
++cell)
```

```
{//do stuff}
```

а також ітератор на основі діапазону

```
for (auto& cell : grid->local_cells)
```

```
{ //do stuff}
```

Крім того, більш дорогий спосіб отримати доступ до комірки за допомогою її глобального індексу через елемент клітинки сітки. Цей член є допоміжним об'єктом, який шукатиме в `native_cells` і `external_cells` клітинку з пов'язаним глобальним ідентифікатором

```
auto cell = grid->cells[cell_global_id];
```

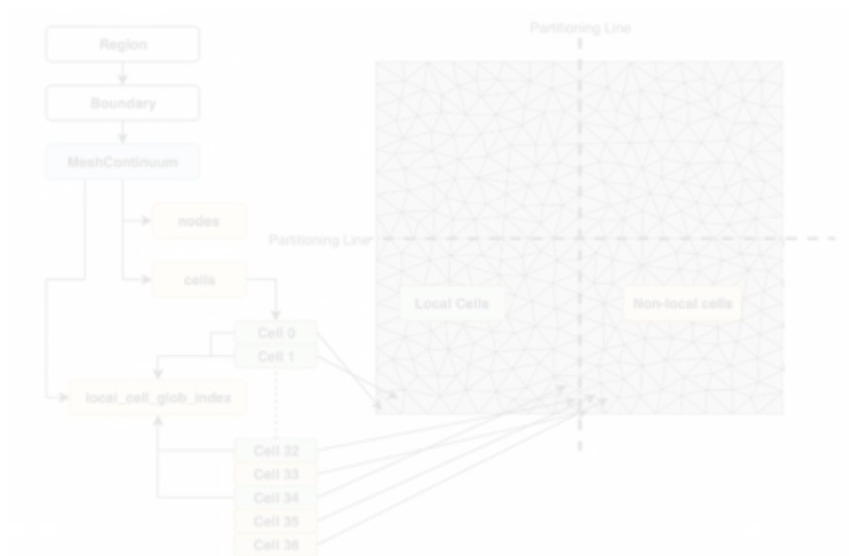


Рис. 2.5. Логіка відображення клітинок

## 2.2. Логічне представлення моделі поведінки програмної розробки

Поведінка програмної розробки визначається множиною об'єктів, що обмінюються повідомленнями.

На UML діаграмі варіантів використання (Use Case Diagram) (Рис.2.6.) показано взаємодію між варіантами використання і діючими особами. Вона відображає вимоги до системи з точки зору користувача. В нашому випадку варіанти використання - це функції, виконувані програмним комплексом, а дійові особи - це користувачі створюваного програмного забезпечення. Такі діаграми показують, які діючі особи ініціюють варіанти використання. З них також видно, коли дійова особа отримує інформацію від варіанту використання.

Специфікація Use Case Diagram «Дискретизація фігури» (Рис.2.6.)

Коротке описання:

Use Case Diagram дозволяє користувачу дискретизувати фігуру на 1D, 2D, 3D - скінченні елементи.

Користувачі системи:

1. Оператор- задає команди для побудови моделі.

2. Програмний комплекс-виконує команди оператора.

Варіанти використання:

1. Запуск програми.
2. Встановити масштаб.
3. Вибір моделі.
4. Задати вид моделі.
5. Задати тип дискретизації.
6. Розбиття фігури на скінчені елементи.
7. 1D- розбиття.
8. 2D- розбиття.
9. 3D- розбиття.
10. Дискретизована фігура.
11. Оптимізація сітки скінчених елементів.
12. Збереження сітки скінчених елементів.

Користувач системи «Оператор»

Коротке описання :

Даний користувач системи описує дії оператора для побудови моделі.

Основні події:

1. Оператор запускає програму.
2. Програма завантажується.
3. Оператор вибирає фігуру.
4. Програмний комплекс завантажує обрану фігуру.
5. Оператор задає вид моделі.
6. Оператор задає масштаб
7. Оператор задає тип дискретизації.
8. Програмний комплекс автоматизує послідовність дій оператора.
9. Програмний комплекс виконує поставлені задачі оператора.

Користувач системи «Програмний комплекс»

Коротке описання:



Даний користувач системи описує дії програмного комплексу щодо побудови моделі.

Основні події:

1. Отримав запит оператора, завантажується для роботи.
2. Оператор задає тип дискретизації.
3. Програмний комплекс розбиває фігури на скінчені елементи.
4. Оператор прагне покращити якість сітки.
5. Програма оптимізує сітку скінчених елементів.
6. Оператор зберігає побудовану модель.
7. Програмний комплекс зберігає сітку скінчених елементів.

Варіант використання «Запуск програми»

Коротке описання:

Даний варіант використання описує запуск програми для дискретизації.

Основні події:

1. Оператор відкриває програму .
2. Програмний комплекс завантажується.

Варіант використання «Вибір фігури»

Коротке описання:

Даний варіант використання описує вибір оператором фігури.

1. Оператор вибирає потрібну фігуру.
2. Програма відображає вибрану фігуру.

Варіант використання «Встановити масштаб»

Коротке описання:

Даний варіант використання описує завдання оператором масштабу фігури.

1. Оператор обирає потрібний масштаб.
2. Програма відображає фігуру.

Варіант використання «Задати тип дискретизації»

Коротке описання:

Даний варіант використання описує вибір типу дискретизації для розбиття фігури.

1. Оператор задає тип дискретизації (1D,2D,3D)

2. Програмний комплекс дискретизує фігуру на скінчені елементи.

Варіант використання «1D- розбиття» , «2D-розбиття», «3D-розбиття»

Коротке описання:

Дані варіанти використання описують розбиття фігури для 1D,2D,3D-ПЛОЩИНИ.

1. Оператор задає команду для розбиття фігур, а саме для 1D-ПЛОЩИНИ.

2. Програмний комплекс виконує розбиття 1D,2D,3D- розбиття.

Варіант використання «Дискретизована фігура»

Коротке описання:

Варіант використання описує автоматичне виконання дискретизації на 1D, 2D чи на 3D скінчені елементи.

1. Програма отримує запит на розбиття фігури.

2. Програма дискретизує фігуру.

Варіант використання «Оптимізація сітки скінчених елементів»

Коротке описання:

Варіант використання описує покращення топологічної якості сітки.

1. Програма отримує запит від оператора на покращення якості сітки.

2. Програма оптимізує сітку.

Варіант використання «Збереження сітки скінчених елементів»

Коротке описання:

Варіант використання описує процес збереження побудованої моделі.

1. Програмний комплекс отримує запит на збереження моделі.

2. Програма зберігає модель по заданому шляху.

Рис.2.6. Use Case Diagram «Дискретизація фігури»

Діаграма послідовності дій (Рис.2.7.) відображає взаємодію об'єктів, впорядковану за часом. На ній показані об'єкти і класи, використовувані в сценарії, і послідовність повідомлень, якими обмінюються об'єкти, для виконання сценарію. Діаграми послідовності дій зазвичай відповідають реалізаціям прецедентів в логічному представленні системи.

Діаграма відображає послідовність повідомлень між об'єктами.

Рис2.7. Діаграма послідовності дій

Кооперативна діаграма відображає потік подій через сценарій варіанта використання. Діаграма (Рис.2.8.) загострює увагу на зв'язках між об'єктами.

Рис.2.8.Діаграма кооперації

### 2.3. Фізичне представлення моделі програмної розробки

Діаграми станів визначають всі можливі стани, в яких може перебувати конкретний об'єкт, а також процес зміни станів об'єкта в результаті настання деяких подій.

На діаграмі є два спеціальних стану - початкове (start) і кінцеве (stop). Початковий стан виділено чорною точкою, він відповідає стану об'єкта, коли він тільки що був створений. Кінцевий стан позначається чорною точкою в білому кружку, він відповідає стану об'єкта перед його знищенням. На діаграмі станів може бути один і тільки один початковий стан.



Рис.2.9. Діаграма стану (Statechart diagrams)

Діаграма діяльності (Activity diagrams) деталізує особливості алгоритмічної реалізації виконання програмним комплексом операцій.

Рис.2.10. Діаграма діяльності (Activity diagrams)

## 2.4. Архітектура програмного забезпечення препроцесора

Архітектурою програмного забезпечення є структура системи, яка представляє набір компонентів, що виконують певну функцію або набір функцій. Основне призначення архітектури - організація компонентів з метою забезпечення певної функціональності.

Розглядається архітектура програмного забезпечення препроцесора. Розроблена схема (Рис.2.11.) з основними функціональними компонентами, які входять до складу додатку автоматичної генерації скінчено-елементного моделювання.

Архітектура програмного комплексу складається з:

- підсистеми скінчено-елементної дискретизації;
- підсистеми моделювання програмного забезпечення.

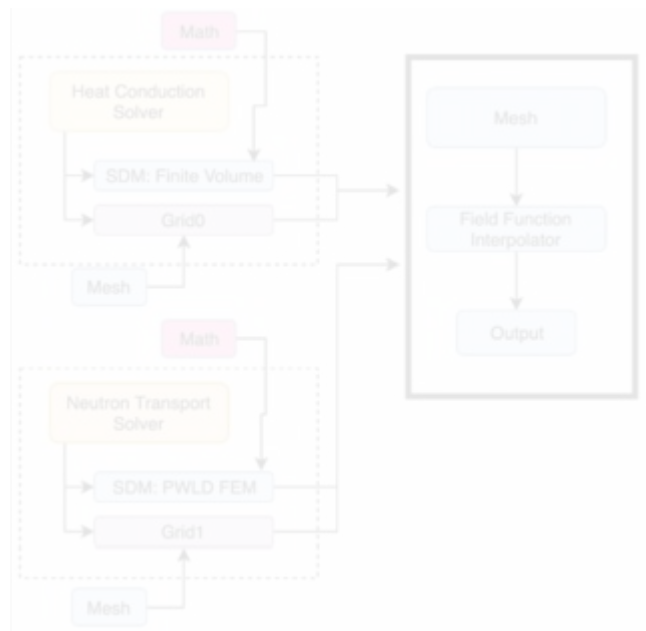


Рис.2.11. Схема компонентів програмного комплексу

## 2.5. Висновки до розділу

В цьому розділі представлено статичну модель та модель поведінки програмної розробки. А також виконано фізичне представлення моделі.

Описано архітектуру програмного забезпечення додатка для автоматизації побудови дискретної (скінченно-елементної) моделі конструкцій.

### РОЗДІЛ 3.

#### РОЗРОБКА ПРЕПРОЦЕСОРА ДЛЯ СКІНЧЕННО-ЕЛЕМЕНТНОГО МОДЕЛЮВАННЯ

##### 3.1. Обґрунтування вибору середовища розробки препроцесора

Microsoft Visual C++ (MSVC) — інтегроване середовище розробки додатків на мові C++, що розроблене фірмою Microsoft і поставляється як частина комплексу Microsoft Visual Studio 2022.

Visual C++ 2022 надає потужне і гнучке середовище розробки, що дозволяє створювати додатки для Microsoft Windows і додатки, засновані на Microsoft .NET. Це середовище можна використовувати як інтегроване середовище розробки, так і в якості окремих засобів. Visual C++ складається з наступних компонентів:

- **Компілятор C++:** Visual C++ постачається з компілятором, який перетворює код на мові C++ у машинний код, який може бути виконаний операційною системою.
- **Інтегроване середовище розробки (IDE):** Visual C++ має потужне інтегроване середовище розробки (IDE) під назвою Visual Studio. Це середовище надає розширені можливості для розробки, налагодження, тестування та керування проектами на мові C++. Воно має інтерфейс користувача з багатьма функціями, такими як редактор коду, відладчик, система контролю версій, інструменти тестування та багато іншого.
- **Бібліотеки Windows API:** Visual C++ надає доступ до багатьох бібліотек Windows API, які дозволяють розробникам взаємодіяти з операційною системою Windows. Ці бібліотеки надають функції для керування вікнами, обробки подій, мережевого взаємодії, роботи з файлами та багато іншого.
- **Бібліотеки стандарту C++:** Visual C++ включає бібліотеки стандарту C++, такі як STL (Standard Template Library), які надають реалізації різних алгоритмів, контейнерів та інших

47

ЛНУ



корисних компонентів. Ці бібліотеки полегшують розробку програм на мові C++ і забезпечують переносимість коду між різними платформами.

- **Інструменти для налагодження та профілювання:** Visual C++ надає різноманітні інструменти для налагодження та профілювання програм. Це включає точковий налагоджувач (debugger) для виявлення та виправлення помилок, аналізатор пам'яті для виявлення витоків пам'яті.
- **Пакети розробки (SDK):** Visual C++ має різноманітні пакети розробки (SDK), які дозволяють розробникам створювати програми для конкретних платформ або функціональностей. Наприклад, Windows SDK надає набір інструментів та бібліотек для розробки програм під операційну систему Windows.
- **Інструменти для розробки графічного інтерфейсу:** Visual C++ має набір інструментів для розробки графічного інтерфейсу користувача. Включаючи дизайнер форм, який дозволяє створювати і налаштовувати вікна, кнопки, поля введення та інші елементи інтерфейсу.

Ці компоненти разом створюють потужне середовище розробки для програм на мові C++, дозволяючи розробникам створювати різні типи програм з використанням широкого набору інструментів та бібліотек.

Мова C++, що є найпопулярнішою у світі мовою рівня системи, і Visual C++ разом надають розробникові висококласний засіб світового рівня для побудови програмного забезпечення.



Рис. 3.1. Microsoft Visual Studio 2022

Програма реалізована на мові програмування C++ в **середовищі Visual C++ 2022**. Вибір середовища програмування обумовлений відносною простотою реалізації графічного інтерфейсу користувача (GraphicUserInterface – GUI).

При створенні засобів відображення і візуалізації використаний графічний інтерфейс OpenGL.

**OpenGL** має основну мету - відображення статичних та динамічних сцен з двовимірними та тривимірними об'єктами. У цьому контексті, об'єкти можуть бути представлені як набір вершин (для геометричних фігур) або пікселів (для растрових зображень). Починаючи з вихідних даних, OpenGL виконує перетворення, щоб перетворити примітиви та зображення в піксельне представлення. Кожен сформований піксель асоціюється з необхідними даними для його відображення та подальшої обробки, а результат перетворення розміщується в буфері кадру. Реалізація OpenGL для Windows включає понад 400 функцій, з яких 368 є базовими функціями основної бібліотеки `opengl32.dll`, а ще 52 функції входять до складу бібліотеки утиліт `glu32.dll`.

Базові функції OpenGL забезпечують можливість побудови зображень графічних примітивів, таких як точки, лінії, багатокутники та растрові зображення. Вони також включають функції для перетворення координат, обмеження області видимості, керування кольором, освітленням, текстурою та туманом.

Функції бібліотеки утиліт OpenGL є розширенням базового набору функцій і служать для створення зображень складніших об'єктів, таких як сфери, диски, конічні циліндри. Вони також дозволяють керувати текстурою та виконувати перетворення координат, проводити тріангуляцію багатокутників та побудову кривих та поверхонь на нерегулярній сітці контрольних точок з використанням форм Без'є та раціональних B-сплайнів.

Всі базові функції OpenGL можна розділити на п'ять категорій:

- Функції опису примітивів визначають нижній рівень ієрархії об'єктів, які можуть бути відображені графічною системою. У OpenGL такими примітивами є точки, лінії, багатокутники і т.д.
- Функції опису джерел світла використовуються для визначення положення та параметрів джерел світла, які знаходяться у тривимірній сцені.
- Функції завдання атрибутів дозволяють програмісту визначати зовнішній вигляд відображуваних об'єктів. Ці атрибути включають колір, характеристики матеріалу, текстури, параметри освітлення.
- Функції візуалізації дозволяють задавати положення спостерігача у віртуальному просторі та параметри об'єктива камери. Це дозволяє системі правильно побудувати зображення і відсікти об'єкти, які не потрапляють в поле зору.

Набір функцій геометричних перетворень дозволяє програмісту здійснювати різноманітні трансформації об'єктів, такі як повороти, зсуви, масштабування.

OpenGL складається з набору бібліотек, і основні функції знаходяться в основній бібліотеці, яку позначають аббревіатурою GL. Крім основної бібліотеки, OpenGL включає кілька додаткових бібліотек. Перша з них - бібліотека утиліт GL (GLU - GL Utility). Усі функції цієї бібліотеки визначаються за допомогою базових функцій GL. Бібліотека GLU містить реалізацію складніших функцій, таких як набір популярних геометричних примітивів (куб, куля, циліндр, диск), функції побудови сплайнів та додаткові операції над матрицями.

OpenGL сам по собі не включає спеціальних команд для роботи з вікнами або отримання інформації від користувача. Тому були створені спеціальні бібліотеки, які забезпечують такі функції, які часто використовуються при взаємодії з користувачем та для відображення інформації за допомогою віконної підсистеми. Однією з найпопулярніших таких бібліотек є GLUT (GL Utility Toolkit). Формально GLUT не є частиною OpenGL, але практично включається в багато дистрибутивів OpenGL і має реалізації для різних платформ. GLUT надає мінімальний набір функцій, необхідних для створення OpenGL-додатків. Існує також менш популярна бібліотека GLX, яка має аналогічні функції.

Крім того, функції, специфічні для конкретної віконної підсистеми, зазвичай входять до її прикладного програмного інтерфейсу (API). Наприклад, функції, специфічні для виконання OpenGL, можуть бути включені в Win32 API для системи Windows або в X Window для системи X Window.

На схемі 3.2, яка представлена нижче, зображена організація системи бібліотек у версії, що працює під управлінням системи Windows.

(Тут не вдається відобразити схему, але можна навести опис зображеного на схемі.)

На схемі видно, що основна бібліотека GL містить базові функції OpenGL. Бібліотека GLU є додатковою і включає більш складні функції. Бібліотека GLUT, хоча не є частиною OpenGL, забезпечує мінімальний набір

функцій для створення OpenGL-застосунків. Інші функції, пов'язані з віконною підсистемою, можуть бути включені в Win32 API.

Отже, OpenGL складається з основної бібліотеки GL, додаткової бібліотеки GLU і, за необхідності, використовується разом з бібліотекою GLUT або іншими віконними підсистемами для роботи з вікнами та отримання інформації від користувача.



Рис. 3.2. Організація бібліотеки OpenGL

OpenGL використовує модель клієнт-сервер для реалізації своїх функцій. У цій моделі прикладна програма виступає в ролі клієнта, що генерує команди, а сервер OpenGL інтерпретує та виконує ці команди. Сервер може бути розташований на тому ж комп'ютері, що й клієнт (наприклад, у вигляді динамічної бібліотеки - DLL), або на іншому комп'ютері, використовуючи спеціальний протокол передачі даних між машинами.

OpenGL обробляє та формує зображення графічних примітивів в буфері кадру з урахуванням вибраних режимів. Примітив може бути точкою, відрізком, багатокутником тощо. Кожен режим може бути змінений незалежно від інших. Визначення примітивів, вибір режимів та інші операції виконуються за допомогою команд, які викликаються у вигляді функцій прикладної бібліотеки.

Примітиви визначаються набором однієї чи декількох вершин (vertex). Кожна вершина визначає точку, кінець відрізка або кут багатокутника. Кожній вершині приписуються певні дані, такі як координати, колір, нормалі,

текстурні координати і т.д., що називаються атрибутами. У більшості випадків кожна вершина обробляється незалежно від інших.

Архітектура OpenGL реалізує конвеєрну схему обробки графічних даних, яка складається з кількох послідовних етапів обробки. На рисунку 3.3 показана ця схема.



Рис. 3.3. Схема функціонування конвеєра OpenGL

Команди OpenGL завжди обробляються в порядку їх надходження, але можуть відбуватися затримки, перед тим як проявиться ефект від їх виконання. Зазвичай OpenGL реалізує прямий інтерфейс, що означає, що визначення об'єкта призводить до його візуалізації в буфері кадру. Для розробників OpenGL - це набір команд, які керують використанням графічного апарату.

Якщо графічний апарат складається тільки з адресованого буфера кадру, то функції OpenGL повністю реалізуються за допомогою ресурсів центрального процесора. Зазвичай графічний апарат надає різні рівні прискорення, починаючи від апаратної реалізації виводу ліній та багатокутників до високопродуктивних графічних процесорів з підтримкою різних операцій над геометричними даними.

OpenGL є прошарком між апаратним рівнем та рівнем користувача, що дозволяє забезпечити єдиний інтерфейс на різних платформах, використовуючи можливості апаратної підтримки.

### 3.2. Розробка інтерфейсу

Дуже важливу роль в ефективності роботи програми грає правильно розроблений інтерфейс. Саме від цього буде залежати виконання декількох з основних вимог - зручність і простота в освоєнні. Складний, перевантажений інтерфейс може зменшити ефективність роботи з препроцесором. Головне правило, від якого варто відштовхуватися - інтерфейс не повинен заважати роботі користувача й не повинен відволікати його увагу від роботи, одночасно не втрачаючи при цьому функціональності.

Виходячи з цього, було вирішено використовувати типові для windows-програм візуальні компоненти. Це дасть можливість більшості користувачів швидко розібратися й включитися в роботу. Інтерфейс прямо залежить від функцій, що виконуються програмою.

Розроблений програмний комплекс виконує створення простих геометричних сутностей та дискретизацію на скінчені елементи досліджуваної області в конструкціях для елементів чисельного розрахунку напружено-деформованого стану деформівного тіла.

Має змогу:

- відкривати файли: Gmsh geometry .geo - файл геометрії GMSH; STEP і IGES - відомі формати файлів геометрії, підтримувані багатьма CAD –пакетами;
- будує точки, відрізки, сплайни, B-сплайн, дугу кола, еліптичну дугу, плоску фігуру, обтягуючу поверхню, об'ємне тіло;
- візуалізує нумерацію точок;
- візуалізує точки, відрізки, сплайни, B-сплайни, дуги кола, еліптичні дуги, плоскі фігури, обтягуючі поверхні, об'ємні тіла;
- автоматично виконувати дискретизацію на 1D, 2D чи на 3D скінчені елементи;

54

- візуалізує дискретизацію на 1D, 2D чи на 3D скінчені елементи;
- візуалізує нумерацію скінчених елементів;
- автоматизує рекомбінацію сітки скінчених елементів;
- виконувати оптимізацію сітки скінчених елементів.

Програмний комплекс зберігає в файл геометрію з дискретизацією 1D, 2D чи 3D скінчені елементи.

Програмний комплекс має досить зручний інтерфейс. В ньому передбачена можливість виконувати дискретизацію на 1D, 2D чи 3D скінчені елементи для розбиття криволінійних, просторових поверхонь і поверхонь, що обмежують об'ємні тіла та об'ємні тіла;

Програма має змогу виконувати оптимізацію скінчених елементів та забезпечує збереження в файл сітку скінчених елементів конструкцій різних форм.

Інтерфейс програми представлений в вигляді поєднання двох вікон (Рис.3.4.). Одне з них являється Графічним Вікном, де створюється модель і редагування моделі. А інше містить в собі основне Меню Програми та набір команд для роботи з фігурами.



Рис. 3.4. Интерфейс препроцессора





В нижній частині Графічного Вікна знаходиться Рядок Стану (Рис.3.5).

Рис.3.5. Рядок Стану

Він містить команди для керування зображенням і інформацію про дію активної команди. (Табл.3.1.)

Таблица 3.1.

## Команди Рядка Стану

Зображення піктограми	Комбінація клавіш	Дія
	Alt+x, Alt+Shift+x	Вид моделі відносно осі X.
	Alt+y, Alt+Shift+y	Вид моделі відносно осі Y.
	Alt+z, Alt+Shift+z	Вид моделі відносно осі Z.
	Shift Alt	Поворот моделі на 90 за часовою стрілкою та проти часової. Синхронізація повороту.

1:1	Alt	Встановлення масштабу. Синхронізація масштабу.
	Alt+o, Alt+Shift+o	Перемикання режиму проекції.
S	Esc	Перемикач миші Вкл/Викл.

Вікно Меню Програми містить в собі основне меню програми та набір команд для роботи з фігурами. У верхній ділянці розташований рядок падаючих меню. В нього входять File, Tools. (Рис.3.6).

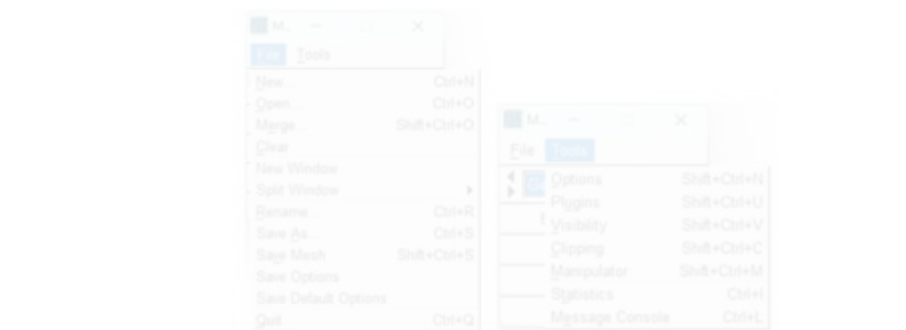


Рис.3.6. Меню File, Tools

### Опис змісту діалогу меню File

#### Відкриття фігури

1. Натискаємо на меню **File**.
2. З випадаючого меню обираємо **Open** або користуємося комбінацією клавіш **Ctrl+O**.
3. В списку знаходимо папку, в якій зберігаються фігури.
4. Подвійним натисканням відкриваємо папку.
5. Подвійним натисканням обираємо фігуру

#### Об'єднання декількох елементів фігур

1. Натискаємо на меню File.
2. З випадаючого меню обираємо **Merge, Shift+Ctrl+O**.
3. Вибираємо спочатку одну фігуру, а потім таким чином обираємо ще одну .

#### Відкриття нового вікна

1. Натискаємо на меню File.

2. З випадаючого меню обираємо New Window.

#### *Розділення вікна*

1. Натискаємо на вікно File.
2. З випадаючого вікна обираємо Split Window => Horizontally для горизонтального розділення, Split Window =>Vertically для вертикального розділення.

#### *Збереження проекту*

1. Натискаємо на меню **File**.
2. З випадаючого меню обираємо **Save As** або **Ctrl+S**.
3. Вибираємо папку, в яку виконується зберігання .
4. В поле **Ім'я файлу** задаємо ім'я моделі.
5. Натиснути кнопку Сохранить.

#### *Закриття проекту*

1. Натискаємо на меню File.
2. З випадаючого меню обираємо Quit.

### **Опис змісту діалогу меню Tools**

#### *Опції налаштування відображення*

1. Натискаємо на меню **Tools**.
2. З випадаючого меню обираємо Options.
3. З'являється вікно Options General (Рис. 3.7).

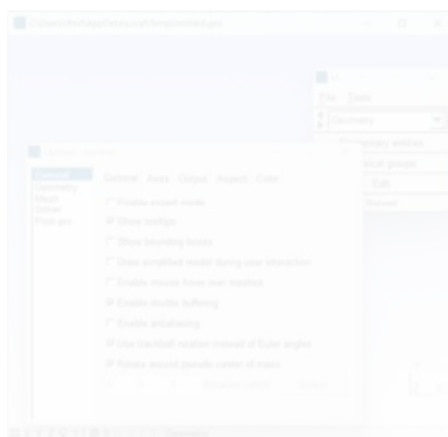


Рис. 3.7. Вікно Options General

4. При виборі Mesh на вкладці та Visibility можливо обирати відображення на геометричній фігурі точок, ліній, поверхонь та об'ємів та нумерацію точок, ліній, поверхонь та об'ємів (рис.3.8).

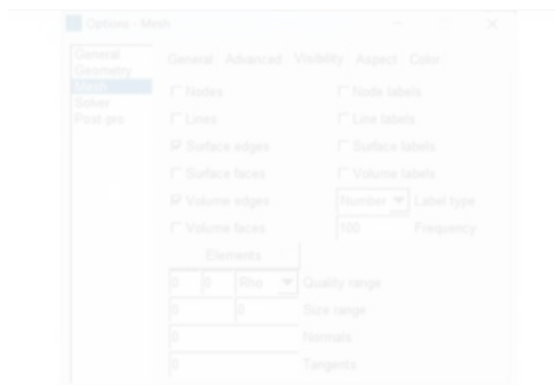


Рис. 3.8. Вікно Options Mesh

### Онції Statistics

1. Натискаємо на меню **Tools**.
2. З випадаючого меню обираємо *Statistics*.
3. З'являється вікно *Statistics* (Рис. 3.9).

Показує інформацію про кількість елементів геометрії, сітки, а також деякі дані про якість сітки.



Рис. 3.9. Вікно Statistics

### Онції Clipping

1. Натискаємо на меню **Tools**.
2. З випадаючого меню обираємо *Clipping*.
3. З'являється вікно *Clipping*.

Перетин - напевно, ще більш важливий інструмент перегляду сітки, ніж Visibility. Адже як інакше заглянути всередину моделі, якщо не зробити розтин? Gmsh дозволяє робити розтин геометрії і сітки. Інтерфейс наступний:

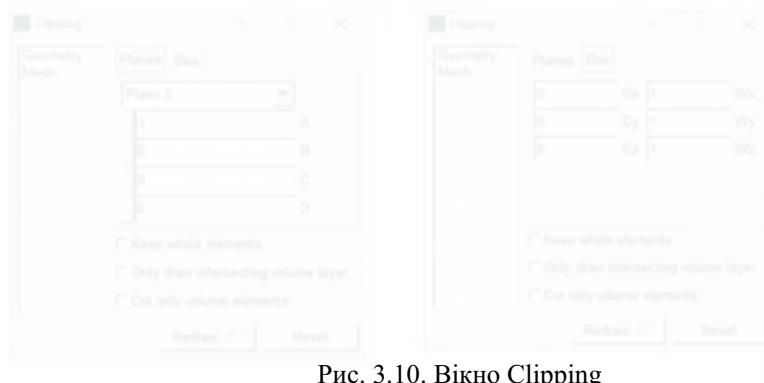


Рис. 3.10. Вікно Clipping

І для геометрії і для сітки можна зробити перетин або площинами (Planes) або параллелепипедом (Box). Для цього потрібно вибрати відповідну вкладку.

**Planes.** Можна задати 6 січних площин коефіцієнтами A, B, C, D з рівняння площини  $A \cdot x + B \cdot y + C \cdot z + D = 0$ . Можливо також інтерактивний "рух" площин за допомогою миші. Для цього порушайте мишею з затиснутою лівою кнопкою в полі введення числа для коефіцієнта. Ви побачите, як змінюється площину перетину і, відповідно, саме перетин сітки.

**Box.** Cx, Cy, Cz - координати центру перетину паралелепіпеда. Wx, Wy, Wz - довжини відповідних сторін, які симетричні центру.

Якщо ви поставите галочку навпроти Keep whole elements (зберегти цілі елементи), то отримаєте ще більш цікаву картину, дивлячись на яку вже можна говорити про якість тетраедрів (звичайно, чисто візуально), про згущення сітки і т.д.

### 3.3. Побудова геометричних об'єктів в препроцесорі

Розглянемо роботу модулю геометрія на прикладі побудови прямокутника розмірами 5 на 3 одиниці та побудову об'ємної фігури, яку отримусмо витягуванням нашого прямокутника.

У Вікні Меню вибираємо: Geometry => Elementary Entities => Add => New => Point. Виникне вікно "Contextual Geometry Definitions" з активною закладкою "Point" (рис.3.11).



Рис. 3.11. Введення координат точок

Вводимо в "текстбокси"  $X=0$ ,  $Y=0$ ,  $Z=0$  - значення координат першої точки і тиснемо Add. Також, є можливість вводити точки покажчиком миші безпосередньо в Графічному Вікні (для цього переміщаємо покажчик і тиснемо "e" на клавіатурі).

В процесі введення точок вони відображаються в Графічному Вікні. Звернемо увагу на "текстбокс" з назвою "Characteristic length" ("Характеристична довжина"), в якій вже вказано за умовчанням значення 0.1. Ця величина пов'язана з розміром SE сітки що примикає до точки, яка вводиться. Закриваємо вікно "Contextual Geometry Definitions".

Відобразимо номери введених точок. Tools => Options => Geometry на закладці Visibility ставимо галочку "Point Numbers". Результат показаний на рис. 3.12.

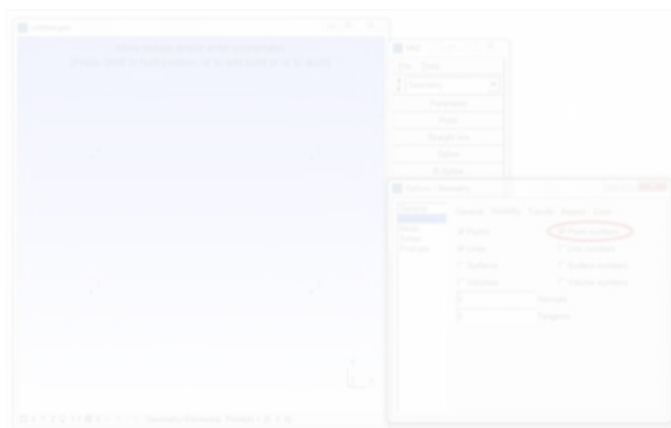


Рис. 3.12. Відображення номерів точок

Вводимо лінії (відрізки ліній). Для цього у Вікні Меню обираємо: Geometry => Elementary Entities => Add => New => Straight Line. І здійснюємо введення, послідовно кликаючи покажчиком миші по точках в Графічному вікні (рис. 3.13).



Рис. 3.13. Відображення ліній

Залишилося побудувати прямокутник (плоску фігуру) обмежений введеними лініями. Вибираємо:

Geometry =>ElementaryEntities =>Add =>New =>PlaneSurface

У заголовку Графічного вікна з'являється напис Select Boundary (Обери межу). Клікаємо по будь-якій лінії, що обмежує замкнутий контур і спостерігаємо "почервоніння". У заголовку Графічного вікна з'являється напис Select hole boundaries (Вибери межі отвору або порожнини усередині прямокутника). Оскільки ніяких отворів у нас спочатку не планувалося тиснемо "е" на клавіатурі для закінчення виділення.

У Графічному Вікні усередині прямокутної області позначилися пунктирні лінії, що утворюють хрест. Це - поточне позначення поверхні в програмі. Повертаємося в Geometry => Edit. Відкривається програма "Блокнот" і перед нами предстають команди, записані в процесі виконання дій в інтерактивному режимі.



Рис. 3.14. Твердотіла модель прямокутника

Якщо ви хочете подивитися і підправити вміст файлу в процесі роботи, то обираєте Edit у вікні Меню (рис. 3.15). Geo файл із записаними в ньому командами відкриється в текстовому редакторі. Після правки і збереження, вам треба виконати Reload, тобто перезавантажити модель.



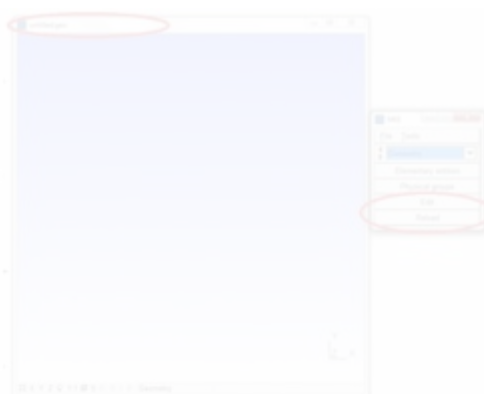


Рис. 3.15. Правка моделі

Побудуємо 3D - тіло витягуванням нашого прямокутника. Geometry => Elementary Entities => Extrude => Rotate => Surface.

Виникне вікно "Contextual Geometry Definitions" з активною вкладкою "Translation". Вводимо в нього значення, як показано на рис. 3.16, виділяємо покажчиком миші нашу єдину поверхню і натискаємо клавішу "е" на клавіатурі.



Рис. 3.16. Вікно "Contextual Geometry Definitions"

В результаті в гео -файл додається вираз:

```
Extrude {{1,0,0}, {b/2,10*h, 0}, Pi/2} {
    Surface{6};
}
```

А створене обертанням об'ємне тіло матиме вигляд як на рис. 3.17.

64

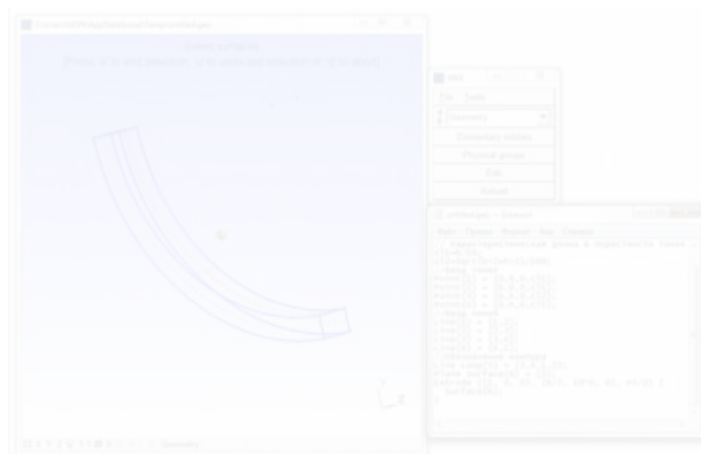


Рис. 3.17. Отримання об'ємного тіла обертанням

### 3.4. Генерація сітки в препроцесорі

Розглянемо дискретизацію плоских та тривимірних об'єктів на прикладі геометричних тіл описаних в пункті 3.3.

Виправимо і закоментуємо команди в geo – файлі прямокутниками таким чином:

```
//Початкові дані:
//Розміри
b=5;
h=3;
// Характеристична довжина в околиці точок
cl1=b/10;
cl2=Sqrt(b^2+h^2)/100;
//Ввід точок
Point(1) = {0,0,0,cl1};
Point(2) = {b,0,0,cl1};
Point(3) = {b,h,0,cl2};
Point(4) = {0,h,0,cl1};
//Ввод ліній
Line(1) = {1,2};
Line(2) = {2,3};
Line(3) = {3,4};
Line(4) = {4,1};
//Позначення контура
Line Loop(5) = {3,4,1,2};
//Поверхня прямокутника
Plane Surface(6) = {5};
```

65

Вибираємо в "Комбобоксі" Вікна Меню "Mesh" ("мэш" - тобто сітка) і натискаємо на кнопку "2D". Результат показаний на рис. 3.18. Вийшло те, що називають "тріангуляція", причому обертає на себе увагу місцеве згущування сітки в районі точки №3 внаслідок прийняття для неї відповідного значення характеристичної довжини.



Рис. 3.18. Розбиття прямокутника на СЕ

Отже, у нас вийшли трьохвузлові елементи першого порядку. А якщо ми натиснемо на кнопку "Second Order" на екрані Графічного Вікна будуть вже шести-вузлові елементи (трикутні, з проміжними вузлами), як відомо - на порядок точніші з розрахункової точки зору, а отже що не вимагають великого подрібнення для отримання заданої точності. На практиці сітка згущується в зонах концентрації напруги, там де напруга змінюється різко на невеликому відрізку. Там, де градієнт напруги не високий, можна обійтися грубішою сіткою, а дрібнити усе однаково - непродуктивно з точки зору точності і економії машинних ресурсів.

Якщо ми зайдемо на вкладку: Mesh => Define => Recombine, виділимо поверхню, після закінчення виділення натиснемо "е" і розіб'ємо заново, то отримаємо результат, показаний на рис. 3.19, а в список команд додасться рядок:

Recombine Surface {6};



Рис. 3.19. Рекомбінована сітка

Отже, ми скористалися для розбиття на СЕ кнопкою "2D". "2D" - не обов'язково означає плоску сітку, нею треба користуватися і для розбиття криволінійних, просторових поверхонь і поверхонь, що обмежують об'ємні тіла.

Для розбиття об'ємних тіл на об'ємні елементи треба користуватися кнопкою "3D".

Для розбиття на кінцеві елементи встановимо нові значення параметра сітки, тобто перепризначаємо характеристичну довжину (при занадто великому числі елементів програма може вилетіти через нестачу пам'яті в Windows).

Заходимо на Mesh => Define => Characteristic length, що вводиться значення  $b/6$  замість 0,1, виділяємо точки в Графічному вікні і після закінчення виділення (усі точки почервоніють) даємо "е", як пропонує нам підказка в заголовку Графічного вікна (для виділення точок не поодиночці, а за допомогою ласо користуємося мишею з натиснутою і утримуваною клавішею "Ctrl"). В результаті в geo -файл додається вираз:

Characteristic Length {26,28,3,2,4,1,5,14,6,10} =  $b/6$ ;

Заходимо на вкладку Mesh, і натискаємо кнопку 3D. Розбите на СЕ тіло показане на рис. 3.20.

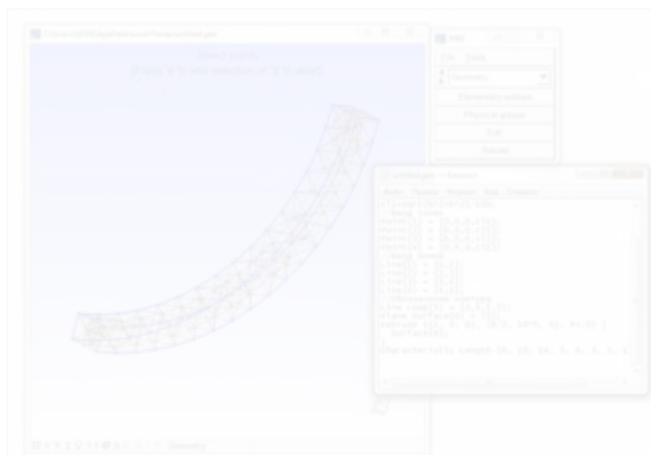


Рис. 3.20. Об'ємне тіло, розбите на скінченні елементи

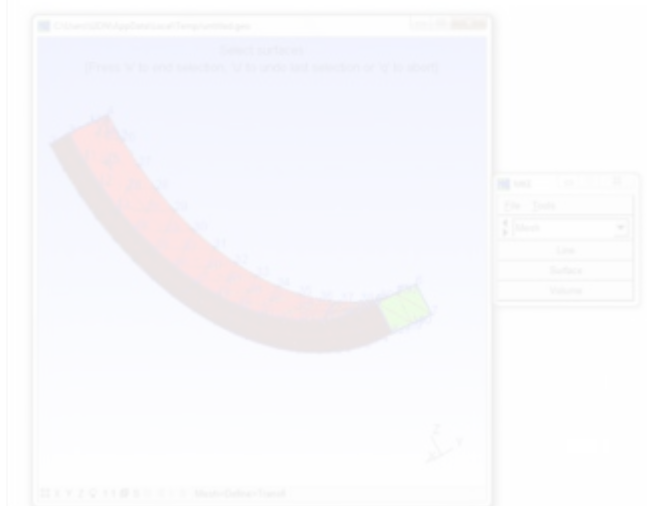


Рис. 3.21. Об'ємне тіло, розбите на скінченні елементи

### 3.5. Формати файлів в препроцесорі

Препроцесор нічого сам не обчислює, тому він має бути пов'язаний з іншими програмами за допомогою створюваних і прочитуваних ним файлів.

Зробити це можна через випадне меню "File" Вікна Меню, вибравши "Open" або "Save As" (рис. 3.22).



Рис. 3.22. Випадаюче меню "File": "Open" та "Save As"

Розглянемо деякі формати файлів по їх розширеннях:

- Gmsh geometry .geo - файл геометрії GMSH.
- STEP і IGES - відомі формати файлів геометрія, підтримувана багатьма CAD -пакетами. Імпорт STEP і IGES, розроблений на основі відкритої бібліотеки OpenCascade дозволяє завантажувати в програму моделі, побудовані в сторонніх CAD -ах (зворотне вивантаження моделі з .geo в ці формати доки не передбачена), в т.ч. у відкритому пакеті Salome.
- STL surface mesh - поширений формат для передачі поверхонь через трикутні сітки.
- .msh файл - файл сіткипрепроцесора.

#### Висновки до розділу

В цьому розділі було розроблено загальний алгоритм роботи програми й інтерфейс на основі передбачуваної функціональності, були обрані програмні засоби для реалізації поставленого завдання. Програма реалізована на мові програмування C++ в середовищі Visual C++ 2022. При створенні засобів відображення і візуалізації використаний графічний інтерфейс OpenGL.

У препроцесорі реалізовано два модулі для роботи: геометрія, сітка. У модулі геометрія доступно створення простих геометричних сутностей, в модулі сітка доступно створення 1- 2 - і 3-х мірних сіток, і їх оптимізація.

## ВИСНОВКИ

У дипломній роботі було розглянуто розробку препроцесора для скінчено-елементного моделювання конструкцій.

Препроцесор є однією з найважливіших складових частин будь-якого програмного комплексу для чисельних розрахунків. Від якості реалізації препроцесора та побудованих за його допомогою геометричних та дискретних моделей проєктованих конструкцій залежить і якість всього програмного комплексу в цілому.

Для досягнення поставленої мети були вирішені наступні завдання:

- Проведено аналіз найпоширеніших в наш час вітчизняних й зарубіжних програмних комплексів моделювання та аналізу напружено-деформованого стану складних інженерних конструкцій і споруд на основі МСЕ. Відзначено, що сучасні програмні комплекси автоматизації проєктування можна умовно розділити на три підсистеми: препроцесор, процесор і постпроцесор. Препроцесор відповідає за підготовку вихідних даних, яка включає такі етапи роботи, як опис геометричної моделі проєктованого об'єкта, та його дискретизацію на заданий тип скінчених елементів. Процесор виконує всі необхідні для конкретного типу задачі розрахунки: формує матриці мас, жорсткостей і демпфірування; будує систему лінійних алгебраїчних рівнянь; враховує початкові й граничні умови; розв'язує систему рівнянь і виводить результати розрахунку. Постпроцесор автоматизує процес аналізу результатів та генерацію документації.
- Розглянуто основні методи побудови геометричних моделей об'єктів і поширені формати їх опису. Проаналізовано основні підходи, що застосовуються в сучасних САПР для твердотільного моделювання геометричних об'єктів, а також основні поширені методи й алгоритми дискретизації плоских та

70

просторових областей. Найпоширеніші алгоритми дискретизації можна розбити на дві частини: побудова первинної дискретизації (найбільш складний етап), та її оптимізація. У препроцесорі для скінченно- елементного моделювання конструкцій для первинної дискретизації області на скінченні елементи прийнято застосувати модифікований алгоритм Ватсона-Лавсона. Оптимізацію отриманої скінченно-елементної сітки робити шляхом застосування алгоритму Рапперта в плоскому випадку та Шевчука – в тривимірному.

- Розроблено препроцесор для скінченно - елементного моделювання конструкцій. У препроцесорі реалізовано два модулі для роботи: геометрія, сітка. У модулі геометрія доступно створення простих геометричних сутностей, в модулі сітка доступно створення 1- 2 - і 3-х мірних сіток, і їх оптимізація.



## Matches

Internet sources

43

1	<a href="https://org2.knuba.edu.ua/pluginfile.php/29517/mod_resource/content/4/OGL%20Layout.pdf">https://org2.knuba.edu.ua/pluginfile.php/29517/mod_resource/content/4/OGL%20Layout.pdf</a>	4 Sources	1.1%
2	<a href="http://org2.knuba.edu.ua/pluginfile.php/70268/mod_resource/content/1/GMKG%20Layout%20final.docx">http://org2.knuba.edu.ua/pluginfile.php/70268/mod_resource/content/1/GMKG%20Layout%20final.docx</a>	3 Sources	1.07%
3	<a href="https://r.donnu.edu.ua/bitstream/123456789/1491/1/%d0%97%d0%b1%d1%96%d1%80%d0%bd%d0%b8%d0%ba_%d1%">https://r.donnu.edu.ua/bitstream/123456789/1491/1/%d0%97%d0%b1%d1%96%d1%80%d0%bd%d0%b8%d0%ba_%d1%</a>	2 Sources	0.35%
4	<a href="http://referatu.com.ua/referats/7569/166440">http://referatu.com.ua/referats/7569/166440</a>		0.28%
5	<a href="http://er.nau.edu.ua/handle/NAU/22405">http://er.nau.edu.ua/handle/NAU/22405</a>	5 Sources	0.2%
6	<a href="http://ir.nusta.edu.ua/jspui/bitstream/doc/4136/1/2958_IR.pdf">http://ir.nusta.edu.ua/jspui/bitstream/doc/4136/1/2958_IR.pdf</a>		0.2%
7	<a href="https://openarchive.nure.ua/bitstream/document/14971/1/2020_M_Pi_Serbin_VV.pdf">https://openarchive.nure.ua/bitstream/document/14971/1/2020_M_Pi_Serbin_VV.pdf</a>		0.17%
8	<a href="https://007time.ru/uml-diagramma-komponentov-opisanie-modelirovanie-na-uml-diagrammy.html">https://007time.ru/uml-diagramma-komponentov-opisanie-modelirovanie-na-uml-diagrammy.html</a>	3 Sources	0.18%
9	<a href="https://ela.kpi.ua/bitstream/123456789/40357/1/Harnytskyi_magistr.pdf">https://ela.kpi.ua/bitstream/123456789/40357/1/Harnytskyi_magistr.pdf</a>		0.16%
10	<a href="http://diplomba.ru/work/27816">http://diplomba.ru/work/27816</a>		0.14%
11	<a href="https://ronl.org/referaty/transport/305561">https://ronl.org/referaty/transport/305561</a>		0.12%
12	<a href="https://onelab.info/pipermail/gmsh/2019/012802.html">https://onelab.info/pipermail/gmsh/2019/012802.html</a>	2 Sources	0.11%
13	<a href="http://eir.zntu.edu.ua/bitstream/123456789/4473/1/MR_Yermilov.pdf">http://eir.zntu.edu.ua/bitstream/123456789/4473/1/MR_Yermilov.pdf</a>		0.1%
14	<a href="http://znp-vo.nuou.org.ua/article/download/193150/193751">http://znp-vo.nuou.org.ua/article/download/193150/193751</a>		0.09%
15	<a href="http://elar.tsatu.edu.ua/bitstream/123456789/16395/1/Akuchev.pdf">http://elar.tsatu.edu.ua/bitstream/123456789/16395/1/Akuchev.pdf</a>		0.08%
16	<a href="http://matan.kpi.ua/public/files/drozd-teoriya-polya.pdf">http://matan.kpi.ua/public/files/drozd-teoriya-polya.pdf</a>		0.08%
17	<a href="http://sn-ecoman.cfuv.ru/wp-content/uploads/2017/04/uch_25_1econ.pdf">http://sn-ecoman.cfuv.ru/wp-content/uploads/2017/04/uch_25_1econ.pdf</a>	2 Sources	0.08%
18	<a href="http://refsmarket.org.ua/moreinfo.php?diplomID=20634">http://refsmarket.org.ua/moreinfo.php?diplomID=20634</a>	7 Sources	0.08%
19	<a href="https://www.stud24.ru/pedagogy/rozrobki-vprav-dlya-rozvitku-fzichnih/512143-2216484-page1.html">https://www.stud24.ru/pedagogy/rozrobki-vprav-dlya-rozvitku-fzichnih/512143-2216484-page1.html</a>	4 Sources	0.08%
20	<a href="https://ua-referat.com/%D0%A8%D0%BB%D1%8F%D1%85%D0%B8_%D0%B2%D0%B4%D0%BE%D1%81%D0%BA%D0%BE%D...">https://ua-referat.com/%D0%A8%D0%BB%D1%8F%D1%85%D0%B8_%D0%B2%D0%B4%D0%BE%D1%81%D0%BA%D0%BE%D...</a>		0.08%

## Quotes

Quotes

2

1 Use Case Diagram «Дискретизація фігури» Діаграма послідовності дій (Рис.2.7.) відображає взаємодію об'єктів, впорядковану за часом.

2 3.4. Інтерфейс препроцесора В нижній частині Графічного Вікна знаходиться Рядок Стану (Рис.3.5).

Internet exclusions	126
---------------------	-----

https://epdf.pub/-301f99b4672b37a8cfc60fe59a48969e12616.html		0.07%
https://ela.kpi.ua/bitstream/123456789/51720/3/Yershyn_magistr.pdf		0.07%
http://www.dut.edu.ua/uploads/p_421_17777559.pdf	2 Sources	0.07%
https://fmab.khadi.kharkov.ua/fileadmin/F-FUB/%D0%95%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D1%96%D0%BA%D0%B	5 Sources	0.07%
https://el-conf.com.ua/wp-content/uploads/2021/01/%D0%94%D0%BD%D1%96%D0%BF%D1%80%D0%BE_%D1%81%D0%B	29 Sources	0.07%
http://molodyvcheny.in.ua/files/conf/other/56march2021/56march2021.pdf	3 Sources	0.07%
http://znc.com.ua/ukr/news/2012/TPMK_12_zbirnik.pdf		0.07%
http://znp-vo.nuou.org.ua/issue/download/11563/5998		0.07%
https://ktepcknute.kyiv.ua/wp-content/uploads/2019/07/Zbirnyk_materialiv_24.04.2019.pdf	27 Sources	0.07%
http://ep3.nuwm.edu.ua/19768/1/Vt9017.zax.pdf	27 Sources	0.07%
https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/26055/1/%D0%9A%D1%80%D0%B8%D0%B2%D0%B5%D0%BD%D0%BA%D0%B	15 Sources	0.07%
https://ir.nmu.org.ua/bitstream/handle/123456789/151653/%210.01%20%D0%A1%D0%B0%D0%BB%D1%8C%D1%87%D0%B	3 Sources	0.07%
https://ela.kpi.ua/bitstream/123456789/26909/6/Masechko_magistr.docx	8 Sources	0.07%
https://ela.kpi.ua/handle/123456789/49710	2 Sources	0.07%
http://eprints.kname.edu.ua/551/1/%D0%9F%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5%D0%A2%D0%90%D0%A3_%D0...		0.07%