МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інформаційна технологія слайсингу для радіального будівельного 3D принтера»

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

Виконавець роботи: студент групи ІТ.м-01 Палажченко Євген Володимирович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2021 р.

Науковий керівник _____ к.т.н., доц., Ващенко С.М.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

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

Студент _____
(підпис)

Суми-2021

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

ЗАТВЕРДЖУЮ
Зав. кафедри ІТ

_____ В. В. Шендрик
«___» _____ 2021 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Палажченко Євген Володимирович
(прізвище, ім'я, по батькові)

1 Тема проекту *Інформаційна технологія слайсингу для радіального будівельного 3D принтера*

затверджена наказом по університету від «_29_»___10___2021 р. № __0787-IV___

2 Термін здачі студентом закінченого проекту «_10_» ___грудня___ 2021 р.

3 Вхідні дані до проекту Тривимірна модель у форматі .STL, документація принтера, документація розробленого G-code формату, дані про конструкційні обмеження принтера

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) 1) Аналіз предметної області
2) Постановка задачі
3) Проектування інформаційної технології слайсингу для радіального будівельного 3d принтера
4) Практична реалізація інформаційної технології

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Актуальність роботи, наукова новизна, мета та задачі, вхідні дані, вихідні дані, структурно-функціональне моделювання, діаграма послідовності, діаграма класів, діаграма варіантів використання, математичне моделювання процесів системи, налаштування додатку, завантаження моделі, генерація заповнення, нарізання фігури,порівняння з аналогами, оцінка достовірності інформаційної технології, висновки, оприлюднення результатів.

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання 01.10.2021

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Збір вимок замовника	04.10.2021	
2	Уточнення вимог	06.10.2021	
3	Ідентифікація проблеми	08.10.2021	
4	Аналіз методів слайсингу	12.10.2021	
5	Моделювання інформаційної технології	14.10.2021	
6	Моделювання роботи додатку	18.10.2021	
7	Реалізація базових структур, завантаження моделі та збереження	27.10.2021	
8	Реалізація блоку візуалізації	01.11.2021	
9	Реалізація блоку генерації заповнення	09.11.2021	
10	Реалізація алгоритму нарізання моделі	23.11.2021	
11	Реалізація алгоритму генерації Gcode	30.11.2021	

Магістрант _____ Палажченко Є.В.

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

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інформаційна технологія слайсингу для радіального будівельного 3D принтера».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 35 найменувань та 3 додатків. Загальний обсяг роботи − 94 сторінки, у тому числі 66 сторінок основного тексту, 3 сторінки списку використаних джерел, 24 сторінки додатків.

Кваліфікаційну роботу магістра присвячено розробці інформаційної технології слайсингу для перетворення тривимірної моделі в набір команд керування переміщеннями радіального будівельного 3D принтера.

У роботі проведено аналіз використання адитивного виробництва в будівництві споруд та наявних методів перетворення тривимірної моделі в набір команд керування переміщенням принтером.

Виконано моделювання процесів інформаційної технологій, моделювання роботи додатку та математичне моделювання алгоритму перетворення моделі в набір команд керування 3D принтером.

Результатом проведеної роботи є інформаційна технологія реалізована в Windows додатку, яка дозволяє перетворювати дані тривимірної моделі в G-code з врахуванням технології будівництва та конструкції 3D принтера.

Практичне значення роботи полягає можливості автоматизованого створення команд керування 3D принтером з врахуванням його технічних характеристик та обмежень на основі завантаженої 3D моделі.

Ключові слова: 3D модель, адитивне виробництво, 3D принтер, G-code, інформаційна технологія, слайсинг.

ЗМІСТ

ВСТУП

Адитивне виробництво (3D друк) створює фізичні об’єкти з тривимірної моделі методом нанесення матеріалу шар за шаром [1]. Сьогодні технологія 3D-друку все частіше використовується в прототипуванні, сільському господарстві, в охороні здоров'я, автомобільній промисловості та авіаційній промисловості. Зараз 3D-друк знайшов величезну кількість застосувань. За допомогою технології адитивного виробництва створюють прототипи виробів для досліджень, декоративні елементи та навіть частини механізмів в автомобільній та авіаційній промисловості, за допомогою друку металом створюють мости та ювелірні вироби, в медицині 3D друк використовують для друку органів та кісток, а придбати персональний 3D принтер для друку пластиком може кожна людина [2].

Така різноманітність сфер застосування потребує використання різних за своїми характеристиками матеріалів. Звісно, кожен матеріал потребує особливих умов для його використання в 3D друці, а отже конструкції принтерів відрізняються одна від одної, що неодмінно потрібно враховувати при генерації шляху руху принтера.

3D друк вже також використовується в будівельній індустрії. За допомогою адитивного будівництва роботи маніпулятори створюють елементи декору та побуту, фрагменти стін, каркаси для стін та каркаси опорних колон [3].

Компанією Mellivora було розроблено радіальний будівельний 3D принтер, але його використання потребує іншого підходу до формування команд, якими має керуватися принтер під час друку.

Все вище сказане обумовлює актуальність представленої роботи.

Об’єктом дослідження є процес обробки тривимірної моделі для перетворення її в набір керування радіальним 3D принтером.

Предметом дослідження є технологія слайсингу будівельних 3д моделей.

Практичне значення роботи полягає можливості автоматизованого створення команд керування 3D принтером з врахуванням його технічних характеристик та обмежень на основі завантаженої 3D моделі.

Наукова новизна роботи полягає в тому, що було адаптовано існуючі алгоритми слайсингу тривимірних моделей під роботу з радіальним будівельним 3D принтером.

Метою розробка інформаційної технології слайсингу тривимірних моделей під технологічний процес адитивного будівництва радіальним 3D принтером. Результатом роботи є створення інструменту генерації команд керування зведення будівель за допою радіального 3D принтера на базі запропонованої технології.

Задачі дипломного проекту:

— провести дослідження предметної області;
— проаналізувати існуючі методи слайсингу;
— провести проектування інформаційної технології слайсингу та моделювання відповідного програмного забезпечення;
— реалізувати програмний продукт на основі розробленої інформаційної технології.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Ідентифікація проблеми

Технології адитивного виробництва вже використовуються в будівництві. Створення елементів декору або частин будівельних конструкцій здійснюється портальними принтерами (такими що працюють в декартовій системі координат та мають три ступені свободи), або роботами–маніпуляторами (з різною кількістю ступенів свободи), додатково оснащеними механізмом подачі цементної суміші для друку. Надруковані блоки транспортують до місця встановлення і монтують, що потребує додаткових затрат та часу [4-7].

Також, одним із способів 3D друку будівель є створення надрукованих пластикових каркасів, які потім додатково утеплюють, укріплюють та з’єднують один з одним [5].

Адитивне будівництво дає величезні можливості у створенні об’єктів складних геометричних форм [8]. При використанні традиційних методів будівництва складно та дорого створювати стіни округлої форми. А створення скульптур традиційними способами однозначно потребую кваліфікованого скульптора.

Вчені дослідили, що незважаючи на те, що бетонна суміш спочатку має низьку здатність передавати навантаження, процес адитивного будівництва можна контролювати, враховуючи реологічні і механічні властивості свіжого бетону, а також поточних проектних вимог [9].

3D-друкr збірних компонентів є ефективним з точки зору використання надрукованих елементів для сухої збірки, можливості повторного використання, модульності, універсальності, адаптивності та стійкості [10]. Модульні конструкції хоч і мають вищеперераховані переваги, але з їх використанням не можливо забезпечити створення конструкцій будь якої форми, оскільки необхідно стандартизувати кріплення між збірними блоками.

Також група вчених розробила метод друку суцільних конструкцій без використання опорних елементів шляхом нанесення слоїв під кутом, щоб конструкція утримувала сама себе. Вчені розробили алгоритм, за яким конструкцію можуть зводити одразу декілька роботів [11]. Це може прискорити процес будівництва і не потрібно додатково встановлювати опалубку чи інші опорні конструкції, але не для всіх видів конструкцій підходить такий метод будівництва. Якщо в конструкції є пустоти, то вони не завжди можуть бути відтворені без використання підтримок.

Адитивне будівництва – це не тільки процес «друку» будівлі, це комплексний процес, що відноситься до всього способу будівництва, від вихідного матеріалу до кінцевої споруди. Даному процесу друку може відповідати кілька будівельних систем. Те, що будинок буде «надруковано», а не зведено традиційними способами, необхідно враховувати протягом реалізації всього проекту – від створення проекту будівлі архітектором до встановлення крівлі. В свою чергу, процес «друку» також є комплексним процесом. Необхідно підібрати матеріали, підібрати робототехніку, використати програмне забезпечення, що враховувало б особливості підібраних компонентів [12].

Експериментальне дослідження переходу від масштабного 3D друку до повно розмірного друку в будівництві показало, що такий перехід можливий для досліджуваних в роботі моделей при правильному підборі матеріалів та параметрів друку. Хоч в роботі і не досліджено вплив швидкості друку моделі та висоти шару на придатність введення такої будівлі в експлуатацію, але вченим цілком вдалося відтворити реальні споруди в масштабі, що підтверджує можливість використання адитивних технологій для повномасштабного виробництва [13].

Цементна суміш має інші властивості ніж пластики, що використовуються при звичайному 3D друці. Тому ці властивості матеріалу також необхідно враховувати при створенні системи адитивного будівництва.

В експериментальному дослідженні властивостей склеювання шарів будівельних матеріалів для 3D-друку перевірили на скільки добре шари склеюються один з одним та якими факторами це обумовлено. Суміші, що використовувалися в дослідженні, було виготовлено з будівельних відходів шляхом додавання цементу,

волокна та органічних компонентів. Будівельні матеріали для 3D-друку видавлювалися через сопло принтера за допомогою процесу контурного 3D-друку. Було проаналізовано міцність зв’язку між шарами на розтяг, зсув і вигин будівельних матеріалів для 3D-друку. Випробування на міцність зчеплення зразків для 3D-друку показало, що під час випробування на зсув не спостерігається руйнування. Поперечний переріз зразка, викликаний процесом 3D-друку, нерівний і хвилястий, що збільшує зчеплення між шарами. Процес 3D-друку споруд має технологічну паузу для закладання арматури чи інших компонентів. Як результат, матеріал для 3D-друку швидко ущільнюється і сила зв’язку між новим шаром і старим шаром зменшується. Чим більша пауза, ти слабший зв’язок між шарами [14].

У дослідженні свіжих та затверділих цементних матеріалів для 3D-друку було виявлено, що надруковані зразки є міцнішими за сформовані вручну зразки. Це насамперед обумовлено тим, що при друці цементна суміш піддається стисканню для її проходження через систему. Це має позитивний вплав на загальну міцність конструкції [15].

В дослідженні дії добавок та на властивості сумішей для 3d друку будівель зазначається, що додавання фібри (синтетичних або натуральних волокон) значно збільшує загальну міцність конструкції та додатково утримує форму конструкції поки суміш ще волога [16].

Хоча нинішнє впровадження адитивного будівництва залишається повільним, в основному через те, що технологія ще не отримала широкого розвитку, а дослідження, які зараз знаходяться на стадії експериментів і розробок, переважно зосереджені на гарантії придатності для друку, структурної надійності, безпеки та довговічності, можна з упевненістю сказати що 3D будівництво має потенціал для позитивних змін у галузі [17].

Отже, в світі вже створюють споруди шляхом адитивного будівництва, використовуючи при цьому різні технології (друк елементів для подальшого монтажу, будівництво на місці портальними принтерами або роботами маніпуляторами). Використання цементних сумішей для друку та шляхи

регулювання властивостей цих сумішей шляхом додавання різних домішок також досить широко досліджено та описано.

Було розроблено інноваційний 3D принтер, що працює в полярній системі координат та не потребує багато часу чи ідеально рівної поверхні для початку роботи. Розроблений принтер при площі установки 1.2м^2 може покривати площу до 200 м^2. Таким принтером легко надрукувати багатоповерховий будинок, просто переставляючи конструкцію з поверху на поверх. Але тоді постає завдання враховувати всі особливості конструкції принтера при генерації шляху сопла під час процесу друку. Тому необхідно розглянути існуючі алгоритми слайсингу тривимірних моделей для 3D друку.

1.2 Аналіз методів слайсингу

Слайсери працюють з тривимірними моделями в форматі .STL. Цей формат файлів є дуже розповсюдженим, адже його можуть згенерувати будь-які системи автоматизованого проектування (САПР). Файл складається з набору трикутників, використовуючи які можна апроксимувати з певною точністю будь-які поверхні. Поверхня задається набором трикутників, кожен з яких задається трьома вершинами і вектором нормалі, що однозначно задає положення трикутника в просторі. Чим менші за розміром трикутники, тим краще з їх допомогою можна апроксимувати поверхню, відповідно зростає кількість трикутників в моделі [18]. На рис. 1.1 зображено приклад моделі в .STL форматі.

Інформація у STL файлах може бути записана в ASCII форматі або в бінарному форматі. Кожен з цих форматів має свої переваги та недоліки. З ASCII файлом легше працювати, оскільки інформація в файлі зберігається у придатному до читання людині форматі. В свою чергу, та ж інформація в бінарному форматі буде займати значно менше місця на дисковому просторі, але потребує конвертації з бінарного формату та знань про структуру файлу [19].

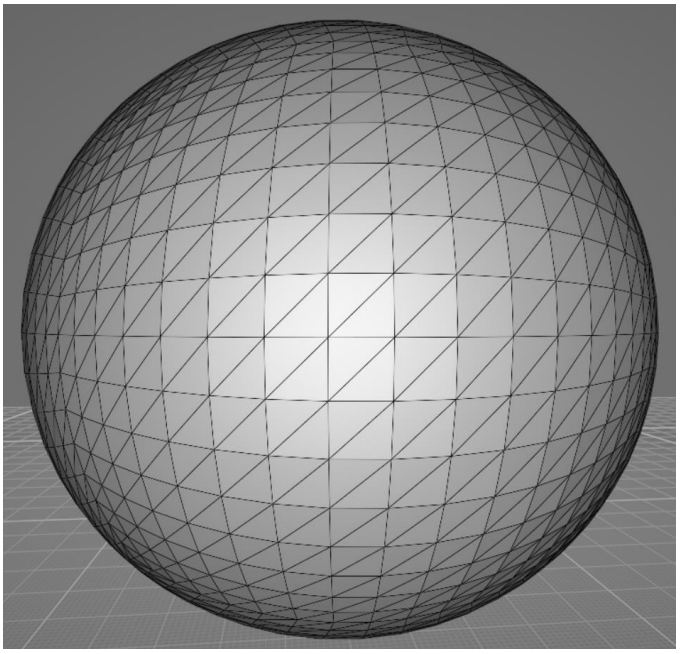

Рисунок 1.1 – Тривимірна модель в форматі .STL

Як видно з рис. 1.1 сферична поверхня досить точно задана декількома тисячами трикутників, які містять достатньо інформації для відтворення зображення фігури на екрані. На рис. 1.2 наведено структуру .STL файлу в форматі ASCII.

```
solid Wall
   facet normal -1.000000e+000 0.000000e+000 0.000000e+000
      outer loop
         vertex 0.000000e+000 0.000000e+000 2.000000e+002
         vertex 0.000000e+000 1.000000e+002 2.000000e+002
         vertex 0.000000e+000 0.000000e+000 0.000000e+000
      endloop
   endfacet
   ...
endsolid
```

Рисунок 1.2 – ASCII представлення .STL файлу

Такий формат запису дає можливість розібратися з форматом файлу не шукаючи при цьому додаткової літератури. Кожен структурний елемент позначається окремими ключовим словом: solid – початок моделі, facet normal – вектор нормалі трикутника (заданий трьома координатами), outer loop – початок циклу перебору вершин трикутника, vertex – вершина трикутника (задана трьома координатами), end

loop – кінець циклу перебору вершин трикутника, endfacet – кінець опису одного трикутника, endsolid – кінець опису моделі [19].

Бінарний формат має іншу структуру файлу. Перші 80 байт відводяться на зберігання назви моделі. Потім вказується кількість трикутників моделі (4 байти). Далі описується масив трикутників по 50 байт на трикутник (12 байт на нормаль, по 12 байт на кожну вершину та 2 байти на зберігання кольору) [19]. Структуру .STL файлу в бінарному форматі наведено на рис. 1.3.

```
UINT8[80]    - Header                 -     80 bytes
UINT32       - Number of triangles    -      4 bytes

foreach triangle                      - 50 bytes:
    REAL32[3] - Normal vector             - 12 bytes
    REAL32[3] - Vertex 1                  - 12 bytes
    REAL32[3] - Vertex 2                  - 12 bytes
    REAL32[3] - Vertex 3                  - 12 bytes
    UINT16    - Attribute byte count      -  2 bytes
end
```

Рисунок 1.3 – Бінарне представлення .STL файлу

Класичний підхід до слайсингу моделі полягає в пошуку ліній перетину між ріжучою площиною та площинами набору трикутників моделі. Площина обраного трикутника може перетинатися з ріжучою площиною одним з 5 способів: ріжуча площина перетинає ребро і проходить через вершину, проходить по ребру, проходить через одну вершину, перетинає два ребра трикутника або трикутник лежить на площині перетину (рис. 1.4). Ріжуча площина підіймається вгору з рівним кроком доки не досягне найвищої точки моделі. Після кожної ітерації отримуємо один або декілька замкнутих контурів, що формуються з набору отриманих ліній перетину. Для збільшення міцності моделі заповнюємо пустоти в середині моделі обраним патерном. Сітка патерну заповнює все поле для друку, а межі обраної моделі обрізають патерн, залишаючи лише частину сітки, що знаходиться всередині моделі.

Наступним кроком є генерація шляху руху екструдера для друку моделі матеріалом для друку [20].

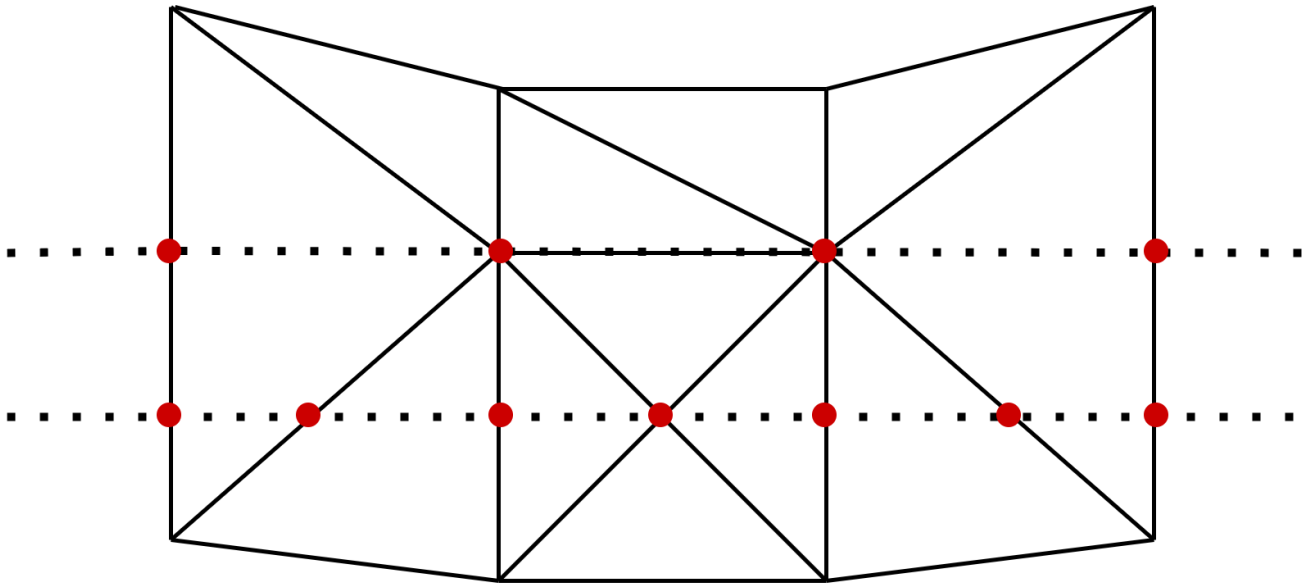

Рисунок 1.4 – Перетин ріжучої площини з трикутниками моделі

На рис. 1.4 пунктиром зображено ріжучі площини, а червоним позначено точки перетину цих площин з трикутними гранями моделі. Пунктирні лінії між червоними точками і є лініями контурів для подальшого їх заповнення.

В дослідженні впливу обраного патерну на механічні властивості надрукованих елементів було виявлено, що обраний патерн та відсоток заповнення впливає на механічні властивості фігури. Те як лінії патерну перетинаються один з одним і забезпечує міцність. Слайсери використовують стандартний набір патернів заповнень [21], які наведені на рис. 1.5.

Спочатку друкуються контури фігури, потім вже заповнюється порожнини між стінками фігури. Патерн, що однотипно повторюється просто обрізається стінками моделі. Таким чином ми отримуємо заповнення лише всередині моделі. Такий спосіб генерації заповнення є рентабельним при друці пластиком і забезпечує можливість використання однотипних патернів на різних моделях. Досить широкий набір патернів дає можливість вибрати потрібний патерн в залежності від задач.

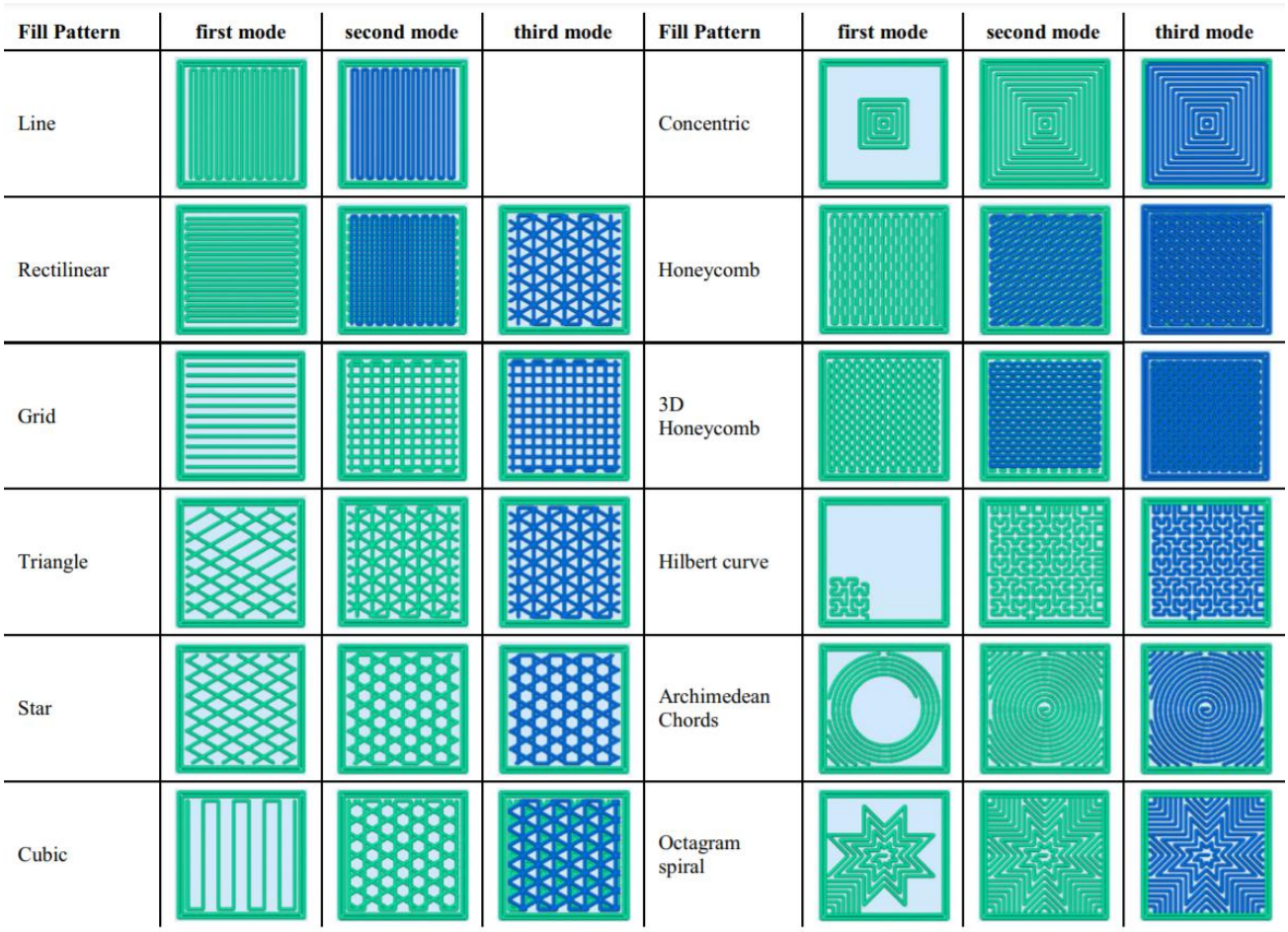

Рисунок 1.5 – Набір стандартних патернів заповнення

Існує також швидший алгоритм нарізання мешу на шари. Суть покращеного алгоритму нарізання полягає в тому, що період нарізання залежить від швидкості зміни моделі. Тобто якщо модель не змінюється в висоту, то використовуємо дані попереднього шару, а якщо змінюється інтенсивно, то нарізаємо модель частіше. Такий підхід дозволяє точніше відтворити фігуру, якщо 3D принтер дозволяю надрукувати шар потрібним розміром. Таким чином отримуємо більшу точність там, де модель інтенсивно змінюється. Але якщо модель змінюється плавно, то накопчується незначне відхилення від реальних розмірів моделі [22].

Іноді для виробництва важливо гарантувати, що помилка під час виготовлення моделі не є негативною. Тобто надрукована деталь гарантовано є не менше заданого в моделі розміру. Через це деталь може бути більшого розміру ніж модель, але прибрати зайвий матеріал значно простіше ніж «доростити» невистачаючі частини

моделі. Тому британськими вченими Yu Wang та Weishi Li було розроблено алгоритм слайсингу який гарантує не від’ємну похибку в розмірах надрукованої деталі [23].

Існують також методи слайсингу для більш специфічних задач. Наприклад, існує метод сферичного слайсингу, який ідеально підходить для слайсингу сфероподібних об’єктів. Метод передбачає, що нарізання моделі сферичними площинами різного радіусу починаючи від центра моделі. Відтворити згенерований набір інструкцій зможе лише принтер на основі робота-маніпулятора [24].

G-code – це набір команд керування, що застосовується в 3D друці та в ЧПУ станках, роботах маніпуляторах. Прилади отримують звичайний .txt файл з послідовним набором команд. Команди повністю керують діями приладів: їх пересуванням, заміною інструментів, швидкістю обертання фрези або об’ємом видавлювання пластиком [25].

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Мета та задачі дослідження

Аналіз предметної області показав велику різноманітність алгоритмів слайсингу тривимірних моделей. Кожен алгоритм має свої показники ефективності та різні сфери застосування. Жоден зі згаданих методів слайсингу не може бути використано для генерації команд керування радіальним 3D принтером, оскільки радіальний принтер має свої конструкційні обмеження, які необхідно враховувати при нарізанні моделі. До таких обмежень відносяться довжина вильоту стріли (мінімальне та максимальне значення), здатність принтера прокручуватися навколо своєї осі (±359°). Також принтер працює не з декартовими координатами, а з полярними. Тобто команди G-code не є стандартними, а містять унікальні, характерні лише для галузі будівництва команди. Затверджений перелік G-code команд наведено в таблиці 2.1.

Таблиця 2.1 – перелік G-code команд

Назва	Код	Параметри
Переміщення без друку	G0	Z (висота екструдера), R (значення по радіусу), F (значення по Фі) абсолютні координати
Переміщення з друком	G1	R (значення по радіусу), F(значення по Фі) абсолютні координати
Переміщення без друку по траєкторії лінії	G2	R та F переміщуються одночасно
Home	G28	Без параметрів
Зупинка для закладання арматури	M0	Без параметрів

Продовження таблиці 2.1 – перелік G-code команд

Зупинка для закладання арматури фундаменту	M1	Без параметрів
Зупинка перед друком вікна	M2	Без параметрів
Зупинка після друку вікна	M3	Без параметрів
Зупинка перед друком дверей	M4	Без параметрів
Зупинка після друку дверей	M5	Без параметрів
Зупинка після друку фундаменту	M6	Без параметрів
Задає швидкість холостого переміщення	M10	S(швидкість руху екструдера mm/s)
Задає швидкість друку ліній	M11	S(швидкість руху екструдера mm/s)
Прискорення принтера	M12	A (прискорення)
Коефіцієнт видавлювання	M13	E (коефіцієнт видавлювання)
Початковий розхід бетону	M14	Q (в літрах за хвилину)
Початковий здвиг по Z	M15	Z (висота в mm)

Окрім того, будівельні матеріали відрізняються за характеристиками від пластику, що також необхідно врахувати при генерації команд керування таким принтером. А сам процес зведення будівлі значно технологічно відрізняється від процесу друку пластмасових об’єктів, адже потребує закладання арматури всередину стін, укріплення кутів та стін будівлі спеціальним типом заповнення, дотримання

технологічних пауз для гарантії того, що цементна суміш набере достатньо міцності щоб витримати наступні шари.

Заповнення для стінок моделі також не може бути стандартної форми. Адже технологія будівництва потребує утеплення стін, зміцнення стін та наявність технічних каналів в стінах для проведення комунікацій – дротів, труб вентиляції, утеплення та водопроводу. На рис. 2.1 наведено приклад структури стіни, з ілюстрацією елементів утеплення та заповнення.

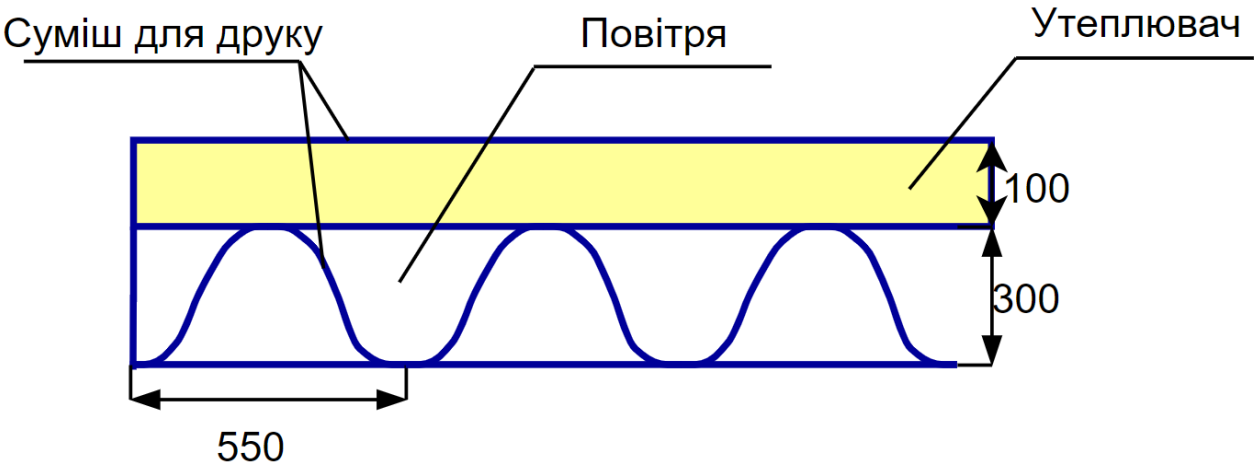

Рисунок 2.1 – Приклад структури стіни

Для забезпечення плавності ходу та зменшення інерційності принтера необхідно щоб кут між лініями заповнення був не більше ніж 120 градусів. Якщо цього правила не дотримуватися, то принтер буде зупинятися на переході між лініями, що призведе до втрати часу. Звісно, кут між лініями заповнення може бути будь-яким, якщо цього буде потребувати конструкція споруди. Система відпрацює всі команди, які будуть у вихідному файлі.

Отже, оскільки принтер має унікальні характеристики, то розроблювана інформаційна технологія має вирішувати всі поставлені задачі з врахуванням конструкційних обмежень пристрою для друку та особливостей технологічного процесу зведення будівель методом адитивного виробництва. Далі детально описано всі задачі, що має вирішувати інформаційна технологія слайсингу для радіального будівельного 3D принтера.

2.1.1. Вхідні та вихідні дані.

Вхідні дані до системи – це файл в форматі .STL, оскільки цей формат є популярним і такий файл можна згенерувати в будь-якій системі автоматизованого проектування. Тобто, необхідно розробити модуль завантаження .STL моделі в текстовому та бінарному форматі, щоб не обмежувати користувача системи. Користувач має обрати файл для завантаження, а додаток має самостійно визначити формат зберігання даних та відповідним чином опрацювати його.

Вихідні дані мають містити набір G-code команд у відповідності до набору команд, зазначених у Таблиці 2.1. Назва команди складається з букви G(команда друку) або M(сервісна команда) та номеру цієї команди. Кожна команда має свій набір параметрів. Причому, кількість параметрів однієї команди може змінюватися. Кожен параметр має свою назву та значення, що вказуються без розділювача. Кожна окрема команда має починатися з нового рядка, а параметри команд мають бути розділені пробілами.

2.1.2 Прив’язка моделі до місцевості.

У всіх слайсерах для друку пластиком відображається поле для друку і розміщувати об’єкти можна тільки в його межах. Тобто, необхідно розмістити на полі як можна більшу кількість об’єктів. А при друці будівель маємо іншу задачу: необхідно прив’язати споруду до реальних точок на місцевості. Наприклад, розмістити модель на вже залитому фундаменті, або зробити її паралельно до іншої будівлі.

Отже необхідно розробити модуль, за допомогою якого можна прив’язати модель до місцевості. Для цього необхідно знати координати двох точок на місцевості відносно початкового положення принтеру. Після встановлення принтера необхідно виїхати в точки, що нас цікавлять та записати їх координати. Потрібно розробити інструмент додавання таких точок до системи та подальшої можливості прив’язати модель до цих точок. Прив’язка моделі має здійснюватися оператором та гарантувати повний контроль положення моделі. Після прив’язки моделі до точок оператор повинен мати можливість змістити модель, або прив’язати модель до інших точок.

Необхідно обмежити можливість пересування об’єкту після виходу з режиму прив’язки моделі до точок на місцевості.

2.1.3. Генерація заповнення.

Дослідження предметної області показало неможливість використання стандартних методів генерації заповнення при підготовці моделі будівлі до виготовлення шляхом нанесення шарів цементної суміші шар за шаром. Потрібно враховувати особливості конструкції кожної споруди, забезпечити наявність утеплювача, та скріплення внутрішньої та зовнішньої стіни за рахунок заповнення.

Отже, необхідно розробити окремий модуль системи, де генерується заповнення та лінії утеплення. Вхідними даними для модулю заповнення є контур шару отриманий в результаті перетину моделі площиною, паралельною площині XY.

Допустимі дії модулю заповнення:

— генерація заповнення для лінії;

— додавання укріплення кута (пусте чи діагональние);

— додавання набору ліній;

— додавання сплайну;

— видалення лінії;

— утеплення;

— точки осідання.

В кожному режимі курсор має «магнітитися» до кінців ліній, якщо відстань до ліній мала.

Генерація заповнення для лінії – дія при якому оператор може додати заповнення для всієї стіни. Генерація здійснюється оператором, шляхом вибору лінії. Параметри заповнення (період та відсоток прилягання) вказуються оператором в налаштування. Під час наведення курсору на лінію вона має підсвічуватися. Після вибору лінії, шляхом натискання на неї, вздовж всієї лінії має з’явитися заповнення потрібної схеми. Для окремих стін може бути згенеровано різна заповнення(різна схема, період та висота ліній). Заповнення має адаптуватися до змін товщини стін без встановлення додаткових параметрів в налаштуваннях автором. Також, модуль

автоматично визначати, де знаходиться внутрішня частина стіни, а де зовнішня. Схеми заповнення порожнин у стінах будівлі наведені на рис. 2.2.

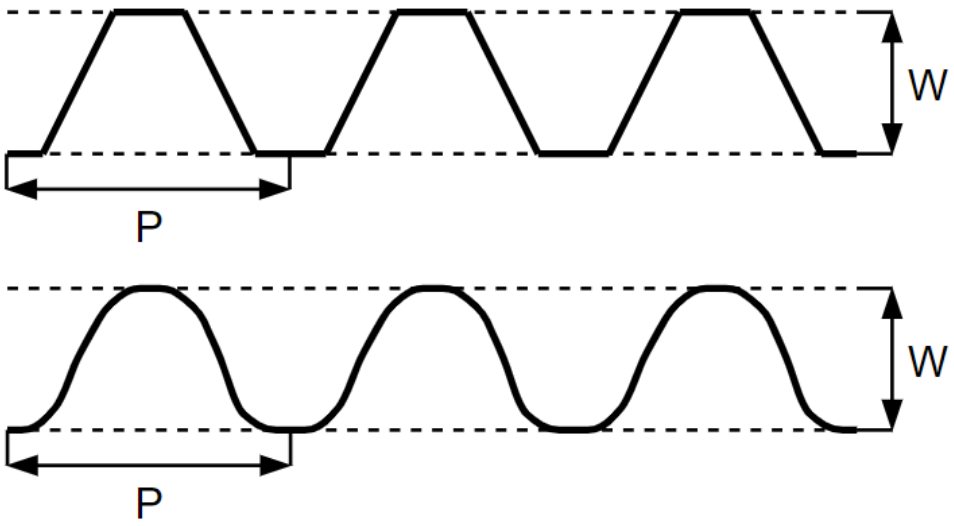

Рисунок 2.2 – Схеми заповнення порожнин у стінах

W – це товщина стіни від краю до краю. P – це період заповнення (відстань через яку повторюється рисунок заповнення). Схема заповнення та період заповнення мають задаватися у вікні налаштувань параметрів друку.

З Рис. 2.2 видно, що для заповнення порожнин у стінах використовуються лише трапецеїдальна та синусоїдальна схеми заповнення. Синусоїдальна схема забезпечує більш плавний перехід від лінії до лінії, а трапецеїдальна збільшує площу дотику ліній заповнення до контурів стіни. В обох випадках порожнини між лініями заповнення можуть слугувати каналами для проведення комунікацій. Яку з них обрати має вирішити оператор, враховуючи особливості проекту конкретної будівлі.

Додавання укріплення кутів в конструкції необхідне для створення колон, які можуть нести додаткове навантаження, що значно зміцнить конструкцію. А додавання перегородок в кутах змістить «точку роси» та допоможе зберегти тепло всередині будівлі [26].

Схему кута обирає оператор враховуючи особливості проекту конкретної будівлі. Кут має додаватися в візуальному режимі. Схема заповнення кутів також

буває двох видів. Схеми заповнення кутів наведено на рис. 2.3. При спробі додавання кута на те ж місце, вже існуючий кут має видалитися.

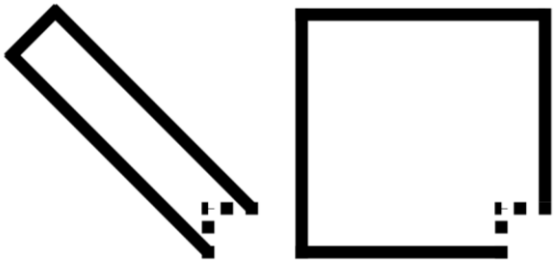

Рисунок 2.3 – Схема кутів

В моделі можуть бути місця, де не потрібно додавати кут чи неможливо додати заповнення вздовж обраної лінії, але конструкції потребує додаткового укріплення. Для додавання цього укріплення необхідно розробити окремий інструмент. Укріплення може бути представленим як у вигляді набору ліній, так і сплайну для забезпечення плавного ходу екструдера. Додавання набору ліній та сплайну має здійснюватися шляхом послідовного вибору точок, натисканням на ліву кнопку миші. Точки сплайну апроксимуються поліномом третього порядку.

Також, дуже важливим є додавання шару утеплення, адже модель має одну стіну і в середині цієї стіни інструментами додатку створити стіну утеплення. Таким чином зовнішній контур стіни буде нерозривним, що забезпечить більшу енергоефективність. Додавання ліній утеплення має здійснюватися шляхом вибору ліній зовнішнього контуру та подальшого їх зсуву всередину стіни.

Встановлення точок осідання(точок зливу цементної суміші) необхідне для створення рівного контуру стіни без видимих напливів цементної суміші. Спочатку принтер виїжджає в точку осідання, відкривається заслінка та вмикається подача цементної суміші для друку і тільки потім починає рух по контуру. Завершення друку шару також здійснюється в точці осідання. Такий підхід гарантує що зовнішня (видима) сторона стіни матиме однорідну текстуру, яка легше піддається обробці та потребує значно менше матеріалів при, наприклад, шпаклюванні.

Забезпечити можливість видалення всіх доданих ліній заповнення в інтерактивному режимі. Після додавання всіх елементів заповнення можна приступити до нарізання моделі на шари.

2.1.4. Нарізання об’єкту.

Нарізання об’єкту на шари є дуже важливим етапом формування G-code команд. Саме під час нарізання формуються контури кожного окремого шару та адаптується заповнення під цей контур. В основному контур шарів не змінюється, оскільки типовий будинок має рівні (у вертикальній площині) стіни, але потрібно додатково обробляти віконні та дверні прорізи, виконуючи обрізання ліній заповнення. На Рис. 2.4 наведено приклад обрізання заповнення при зміні контуру шару.

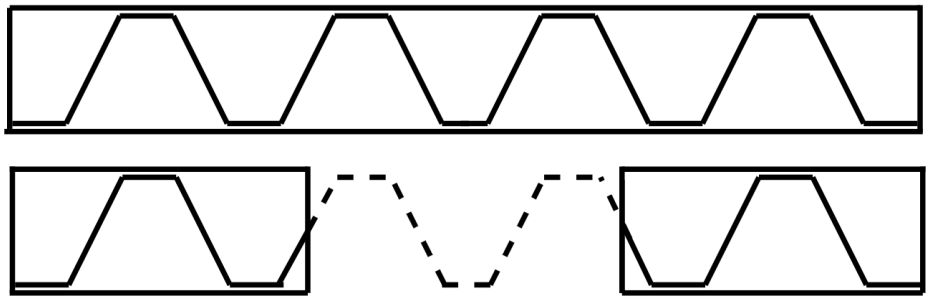

Рисунок 2.4 – Обрізання ліній заповнення

Параметри нарізання моделі мають братися з налаштувань, які вносить користувач. Всі шари мають бути однакової висоти та бути паралельними до основи моделі. В результаті завершення процесу генерації заповнення маємо отримати набір ліній для кожного шару, які має відтворити принтер методом видавлювання суміші для друку.

2.1.5. Генерація G-code.

Модуль генерації G-code має створити набір G-code команд, які може обробляти принтер. Всі команди мають бути створені з врахуванням конструкційних обмежень принтера та з дотриманням технологічного процесу зведення будівель.

В залежності від налаштувань, алгоритм генерації G-code команд має вставляти в код команди технічних пауз для закладання арматури чи встановлення опорних конструкцій для підтримки висячих елементів над віконними та дверними прорізами. Регулювання швидкості переміщення принтера та прискорення принтера також має автоматично інтегруватися в G-code. При генерації шляху обходу ліній шару та ліній заповнення принтер не має зробити оберт більше, ніж на ±360°, адже це призведе до пошкодження або руйнування конструкції. Послідовне виконання всіх команд має гарантувати повне та точне відтворення споруди.

2.1.6. Зовнішній вигляд додатку та керування об’єктами.

Додаток має бути реалізовано у вигляді класичного однодокументного додатку, де ми відкриваємо .STL модель та можемо здійснювати підготовку цієї моделі до друку. Інтерфейс додатку має бути інтуїтивно зрозумілим та зручний у використанні. Користувач повинен мати змогу змінювати кольори робочої області, фону та тривимірних об’єктів. В додатку має бути добре скомбіноване меню з розділенням дій над об’єктом на групи. На рис. 2.5 представлено макет інтерфейсу додатку з візуалізацією пунктів меню.

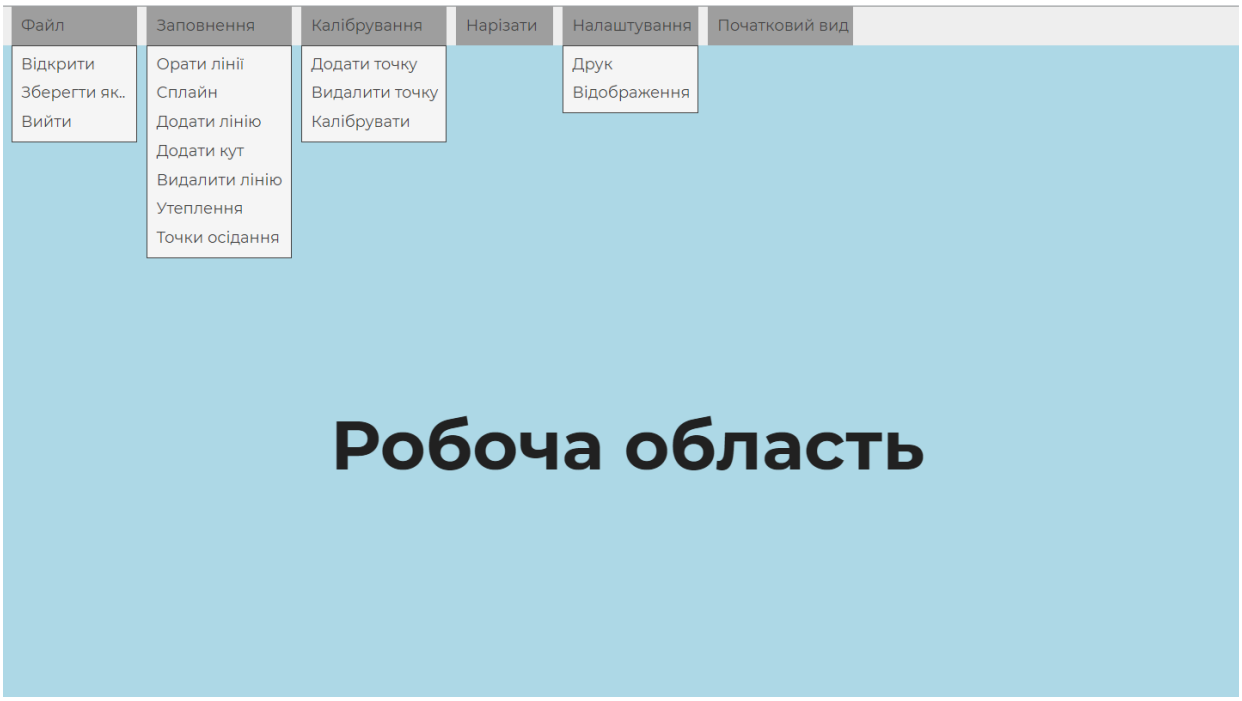

Рисунок 2.5 – Макет інтерфейсу додатку

В робочій області має відображатися поле для друку з сіткою для спрощення роботи оператора. Завантажена тривимірна модель має «приклеюватися» основою до поля для друку. Для огляду сцени та моделі необхідно імітувати ефект пересування камери, а отже потрібно реалізувати операції Zoom, Pan та Orbit.

Zoom, Pan та Orbit – базові способи переміщення камери, які використовуються в кінематографії, ігровій індустрії та системах автоматизованого проектування. Це інтуїтивно зрозумілі та вже знайомі всім користувачам операції [27].

Всі операції реалізувати за допомогою дій миші. Щоб перемістити камеру по орбіті потрібно утримувати праву клавішу миші та переміщаючи мишу навколо малюнка. Pan – утримувати ліву клавішу миші і переміщувати по малюнку. Операції zoom in та zoom out (приближення та віддалення зображення) здійснюється з використанням колеса миші та направлене до центру екрана.

Додавання об’єктів до системи має здійснюватися за допомогою під-пунктів в меню «Файл». Керування доданими об’єктами має здійснюватися за допомогою миші та назначених клавіш. Обрати всі об’єкти можна натисканням на назначену комбінацію клавіш (Таблиця 2.2). Після того, як об’єкт обрано можна переміщувати та обертати його.

Вибір дій над обраним об’єктом має здійснюватися за допомогою вибору пункту меню з бокової панелі або назначених комбінацій клавіш (Таблиця 2.2).

Переміщення обраного об’єкту має здійснюватися шляхом утримування та переміщення лівої клавіші миші.

Таблиця 2.2 – Комбінації клавіш в програмі

Enter	Підтвердження дії
Esc	Вихід з поточного режиму або скасування поточної дії
Ctrl+s	Зберегти G-code
Ctrl+o	Відкрити файл
o	Обрати об’єкт
r	Активація режиму обертання
v	Режим перегляду G-code

2.1.7. Налаштування

В залежності від проекту будівлі та поточної конфігурації принтера для слайсингу моделі можуть застосовуватися різні параметри. Отже, для зручності зміни цих параметрів необхідно розробити діалогове вікно налаштувань. В таблиці 2.3 наведено перелік параметрів доступних налаштувань слайсера.

Таблиця 2.3 – Параметри налаштувань

Параметр	Тип поля	Вимір (значення)
Мінімальний радіус	Число	мм (1–10000000)
Максимальний радіус	Число	мм (1–10000000)
Радіус сопла	Число	мм (1–200)
Максимальний кут повороту принтера	Число	мм (0–10000000)
Висота шару	Число	мм (1–5000)
Час зупинок	Число	мм (1–10000000)
Початковий тиск бетону	Число	Бари (1–10)
Друк фундаменту	Прапорець	(true, false)
Висота фундаменту	Число	мм (0–2000)
Висота «стартових» шарів	Число	мм (0–2000)
Кількість «стартових» шарів	Число	Штуки (0–100)
Швидкість друку фундаменту	Число	м/с (1–1000)
Швидкість друку «стартових» шарів	Число	м/с (1–1000)
Коефіцієнт видавлювання «стартових» шарів	Число	(0.1 – 10)
Період закладання арматури в фундаменті	Число	мм (0–2000)
Армування стін	Прапорець	(true,false)
Період закладання арматури на стінах	Число	мм (0–2000)
Висота закладання першого шару арматури	Число	мм (0–2000)
Утеплення	Прапорець	(true, false)
Товщина утеплення	Число	мм (0–1000)
Заповнення	Прапорець	(true, false)

Продовження таблиці 2.3 – Параметри налаштувань

Швидкість друку заповнення	Число	м/с (1–1000)
Період заповнення	Число	мм (0–1000)
Рівень перекриття заповнення	Число	Відсотки (0 – 100)
Довжина прилягання заповнення	Число	Відсотки (0 – 100)
Форма заповнення	Радіокнопка	(сплайн, трапецеїдальна)
Додавання сервісних команд	Прапорець	(true,false)
Задання швидкості	Прапорець	(true,false)
Зупинка на час T	Прапорець	(true,false)
Старт з середини лінії	Прапорець	(true,false)

Всі наведені в таблиці 2.3 параметри налаштувань необхідно розмістити в діалоговому вікні з розділенням на групи (параметри принтера, параметри друку фундаменту, параметри друку стін, параметри заповнення). Після введення користувачам параметрів вони мають зберігатися у закодованому файлі. При старті програми мають підтягнутися останні введені користувачем параметри. При відсутності значень параметрів в файлі чи пошкодження файлу мають встановлюватися параметри за замовчуванням.

Передбачити окреме діалогове вікно для встановлення кольору фону, кольору об’єкта та поля.

2.2 Методи дослідження

Перед початком розробки інформаційної технології необхідно проводити дослідження. Дослідження предметної області дозволяє зрозуміти всі аспекти поставленої задачі з різних сторін і дає уявлення про можливі причини виникнення проблеми та шляхи їх вирішення.

Існує декілька загальнонаукових методів дослідження – емпіричні, теоретичні, функціональний, системний та конкретно-соціальний. Серед емпіричних методів дослідження виділяють експеримент, спостереження та опис, а серед теоретичних – дедукція, індукція, абстрагування, пояснення, аналіз, узагальнення, опис та інші. Щоб назвати метод дослідження науковим, він повинен ґрунтуватися на зборі спостережуваних, емпіричних і вимірних доказів відповідно до конкретних принципів міркування. Науковий метод складається зі збору даних шляхом спостереження та експериментування, а також формулювання та перевірки гіпотез. Серед інших аспектів, спільних для різних сфер дослідження, є переконання, що процес має бути об'єктивним, щоб зменшити необ'єктивні інтерпретації результатів[28].

В даній роботі під час вивчення предметної області використано аналіз та узагальнення, що дало можливість оцінити наявні способи друку будівель за допомогою адитивного виробництва та наявні методи слайсингу тривимірних моделей.

Для визначення ефективності розробленої інформаційної технології буде проведено експериментальний слайсинг та описано результати в порівнянні з іншими слайсерами. Проведена робота дасть можливість якісно оцінити розроблену інформаційну технологію.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ СЛАЙСИНГУ ДЛЯ РАДІАЛЬНОГО БУДІВЕЛЬНОГО 3D ПРИНТЕРА

3.1 Модель інформаційної технології

IDEF0 – це графічний опис системи чи предмета, розробленого для певної мети та з вибраної точки зору. Мова IDEF0 стандартизована, тому моделі є чітко визначеними, добре структурованими, легкими для розуміння, простими в модифікації та використанні та можуть бути розширені на будь-яку глибину деталей. Моделі IDEF0 є гнучкими, масштабованими та адаптованими до різних ситуацій і умов [29].

На рис. 3.1 зображено контекстну діаграму процесу слайсингу тривимірної моделі споруди для її перетворення в набір G-code команд з точки зору користувача системи.

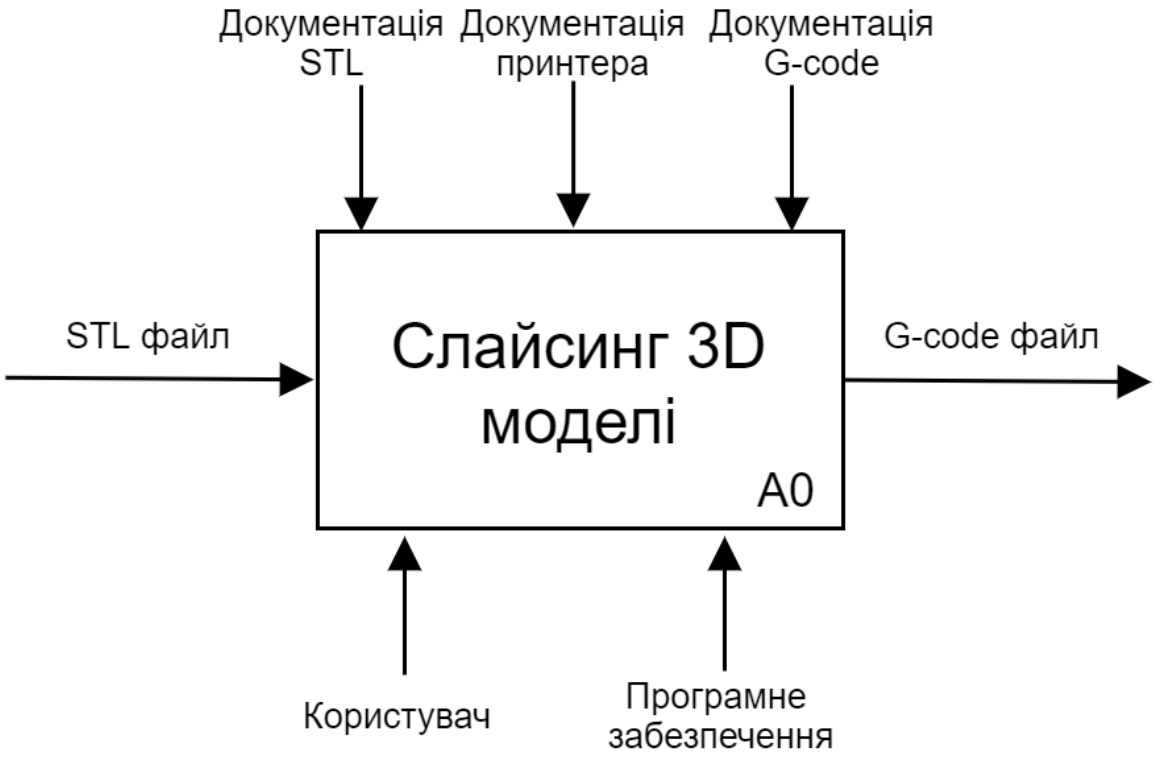

Рисунок 3.1 – Контекстна діаграма у нотації IDEF0

Стандартний процес генерації G-code команд з тривимірної моделі відтворюваного об’єкту складається з трьох кроків: нарізання моделі, створення внутрішнього заповнення, генерація траєкторії руху екструдера [30]. Для розроблюваної системи цей процес є дещо складнішим, адже необхідно враховувати особливості адитивного будівництва споруд реального розміру.

На рис. 3.2 наведено декомпозицію контекстного представлення для отримання більш детального уявлення про процеси, що відбуваються всередині системи.

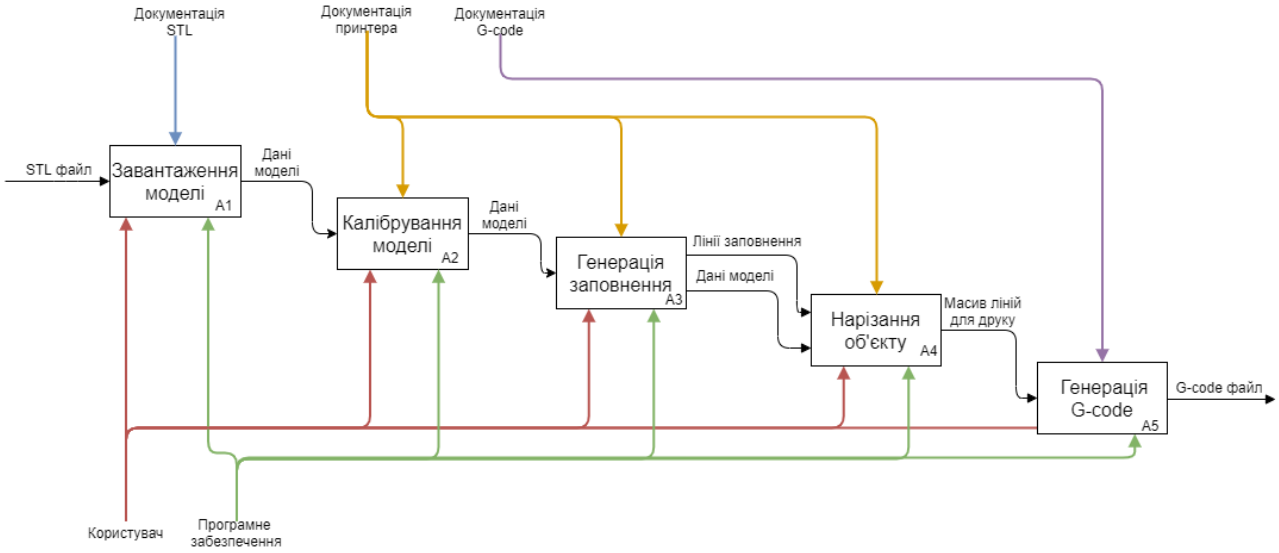

Рисунок 3.2 – Діаграма декомпозиції

П’ять процесів, що представлені на діаграмі є послідовними процесами, які має здійснити користувач для перетворення вхідної тривимірної моделі в форматі STL в набір G-code команд для керування 3D принтером. Зверху в блоки входять стрілки, що позначають інструменти перетворення даних, а знизу – механізми перетворення даних. Для початку роботи необхідно завантажити STL модель. Потім «відкалібрувати» модель, тобто прив’язати віртуальну модель до фізичного оточення для забезпечення правильного позиціонування моделі в просторі. Після чого користувач генерує заповнення та нарізає завантажений об’єкт. Генерація G-code команд є заключним процесом.

Для користувача процес генерації заповнення також є комплексним процесом, який можна розбити на під процеси. Вхідними даними до цього процесу є дані моделі,

а вихідними – дані моделі та набір ліній заповнення. Спочатку користувач має додати лінії утеплення, потім додати лінії заповнення і укріплення кутів.

На рис. 3.3 продемонстровано декомпозицію процесу генерації заповнення для стін будівлі.

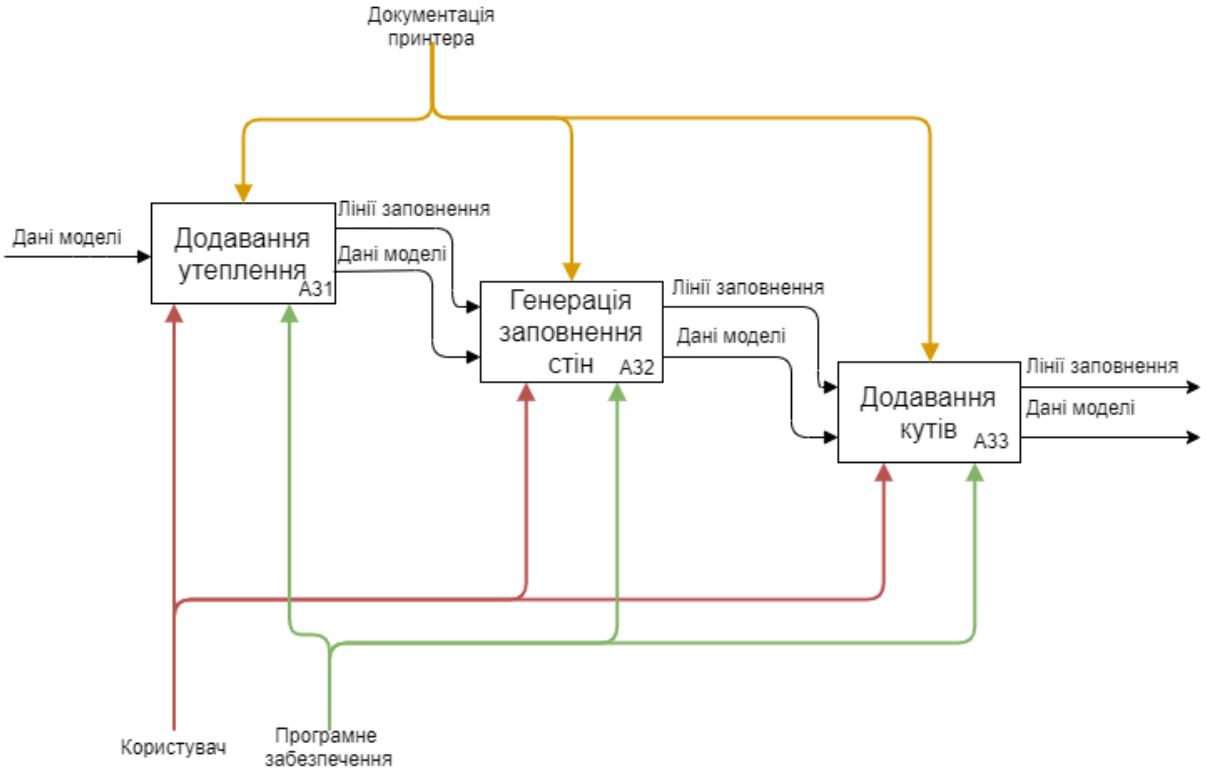

Рисунок 3.3 – Діаграма декомпозиції процесу генерації заповнення

Як видно з рис. 3.3, дані моделі входять і виходять з кожного процесу. При чому дані моделі залишаються не змінними. При чому набір ліній заповнення постійно доповнюється.

Генерація утеплення здійснюється першим кроком, оскільки подальша генерація заповнення стін та додавання кутів залежить від наявності утеплення. В залежності від проекту споруди деякі етапи генерації заповнення можуть бути пропущені. Наприклад, у якоїсь технічної споруди може не бути утеплення або конструкція може не потребувати зміцнення кутів.

3.2 Моделювання роботи додатку

UML (Unified Modeling Language) – це відкритий стандарт для аналізу та проектування програмного забезпечення в графічному вигляді. Одним з найважливіших компонентів UML є діаграми класів, які моделюють структуру класів та взаємозв’язки між цими класами. Діаграми класів UML дозволяють моделювати в декларативний спосіб статичну структуру домену програми з точки зору понять і відносин між ними [31].

Діаграми варіантів використання є одним з ключових етапів розробки на основі UML. Use case діаграми фіксують вимоги користувачів і демонструють можливі варіанти використання системи акторами [32].

В розроблюваній інформаційній технології є тільки один актор (користувач), який взаємодіє з системою. Діаграму варіантів використання розроблюваної технології наведено на рис. 3.4.

Нижче наведено опис варіантів використання (ВВ) майбутньої системи, яка буде реалізовувати інформаційну технологію:

— ВВ завантаження моделі – перша дія, яку має здійснити користувач для подальшої роботи з додатком;

— ВВ перегляд STL моделі – стає доступний відразу після завантаження моделі в систему.

— ВВ калібрування моделі – процес «прив’язки» STL моделі до місцевості для подальшої роботи з моделлю.

— ВВ генерація заповнення – весь комплекс дій формування ліній заповнення порожнин між стінами.

— ВВ перегляд ліній заповнення – стає доступний після формування ліній заповнення та дає можливість переглянути отриманий шаблон заповнення.

— ВВ нарізання об’єкту – дає можливість користувачу запустити процес нарізання об’єкту.

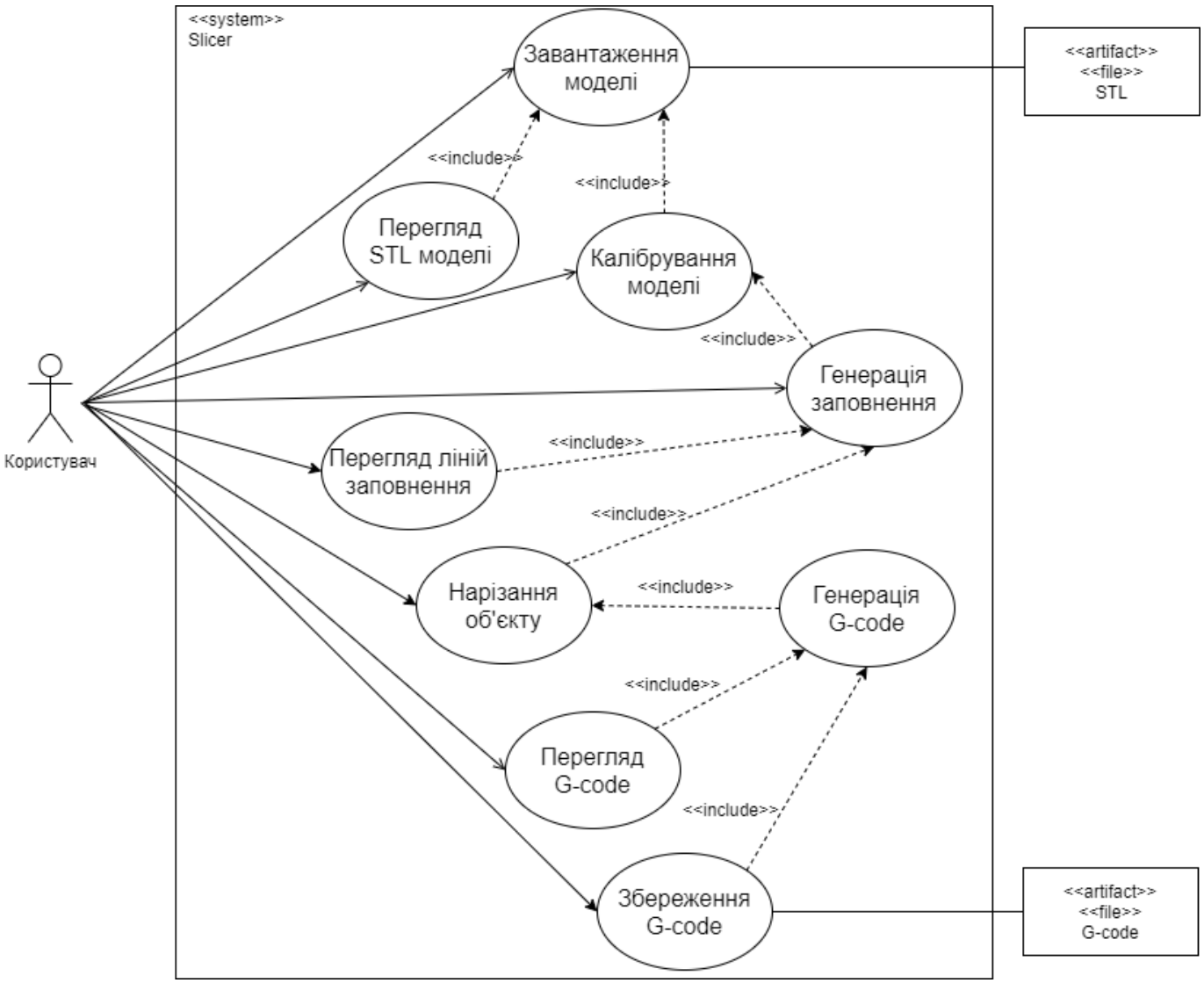

Рисунок 3.4 – Варіанти використання

— ВВ генерація G-code – процес формування G-code команд, що враховує сформоване заповнення та результат нарізання моделі на шари.

— ВВ перегляд G-code – стає доступним після генерації G-code команд та дає можливість візуально побачити траєкторію переміщення сопла принтера.

— ВВ Збереження G-code – дає можливість зберегти сформований набір G-code команд в файл.

Для представлення структури проекту та взаємозв’язків між класами проекту було створено діаграму класів. Також, діаграма класів демонструє структуру кожного з наведених в ній класів, тобто описує всі поля та методи класу з зазначенням типу доступу до кожного елементу. На рис. 3.4 представлено діаграму класів проекту.

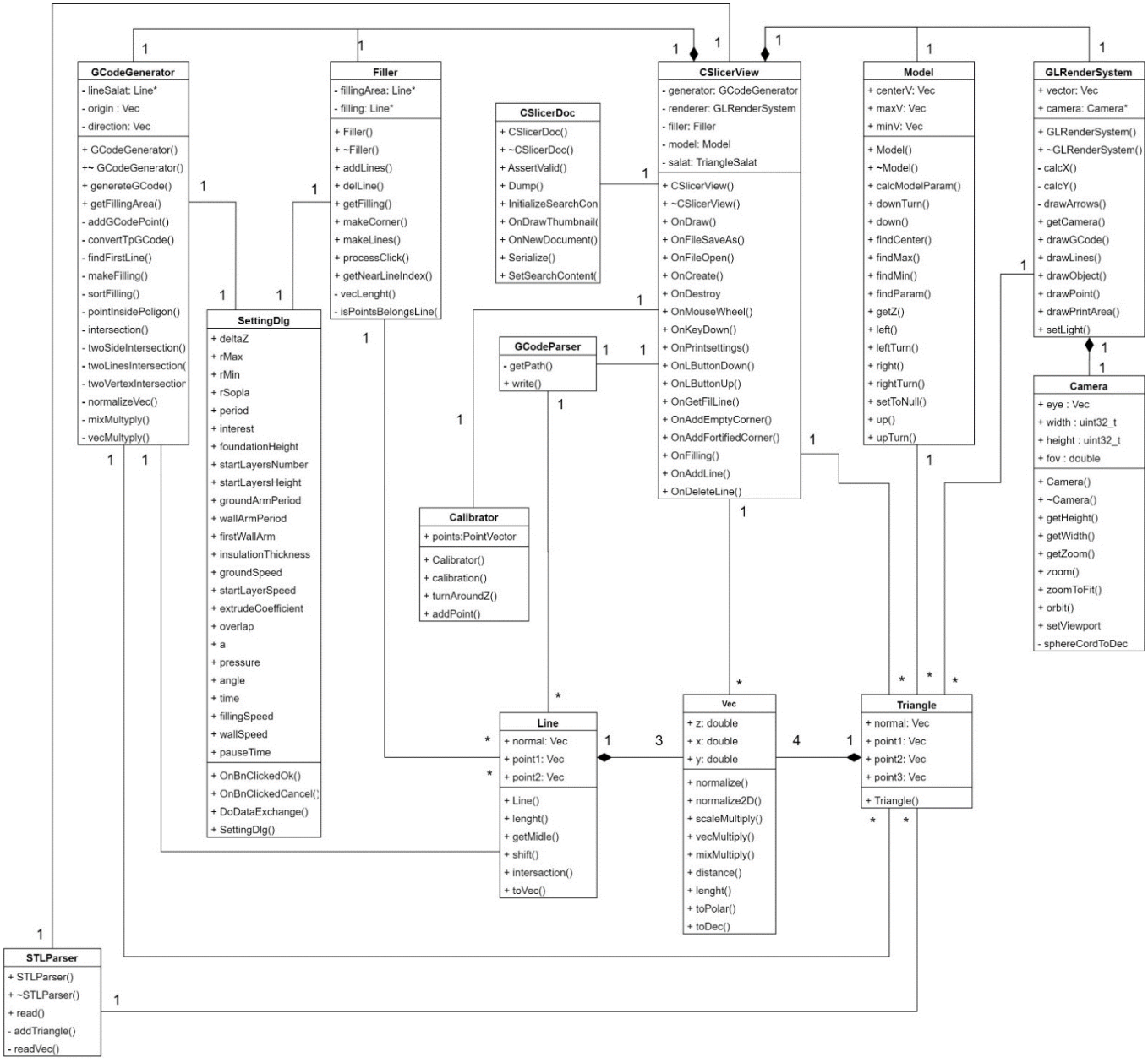

Рисунок 3.4 – Діаграма класів

Як видно з діаграми класів, клас CSlicerView є головним класом, який керує основною логікою проекту. Саме цей клас містить методи, які обробляють всі дії користувача та викликають методи інших класів.

Базовими структурами програми є класи Vec, Line та Triangle. Дані зчитуються з .STL файлу в масив трикутників, що спрощує їх подальшу обробку. Парсинг .STL файлу здійснюються за допомогою методів класу STLParser. Потім дані моделі потрапляють в клас CSlicerView та записуються в клас Model. Клас Model забезпечує

функціонал для переміщення моделі по полю та обертання. Тепер модель готова для подальшої обробки.

Керування налаштуваннями системи здійснюється за допомогою класу SettingDlg. Це клас, до якого прив’язане діалогове вікно. При зміні параметрів налаштувань в діалоговому вікні та підтвердження цих змін, натисканням на кнопку «ОК», всі параметри перезаписуються в файл.

Клас GLRenderSystem відповідає за відображення всіх об’єктів на екрані (поля для друку, тривимірної моделі, ліній заповнення та G-code команд). Клас Camera використовується класом GLRenderSystem для реалізації імітування операцій переміщення камери (Pan, Orbit та Zoom).

Всі процеси генерації заповнення реалізовані в класі Filler. Для генерації заповнення використовується параметри налаштувань з класу SettingDlg. CSlicerView обробляє дії користувача та визиває відповідні методи класу Filler, які і обробляють ці дії.

Клас GCodeGenerator нарізає тривимірну модель та генерує заповнення для кожного шару. Після нарізання моделі та додавання заповнення до шару алгоритм генерує оптимальний по переміщенням шлях з врахуванням конструкційних обмежень принтера. GCodeGenerator також використовує клас налаштувань SettingDlg для отримання параметрів нарізання та значень обмежень конструкції принтера.

Згенерований G-code записується в файл за допомогою класу GCodeParser. Після збереження G-code вже не можна редагувати.

Діаграма послідовності зображує взаємодію між об’єктами системи в послідовному порядку, тобто порядку, в якому ці взаємодії відбуваються. Діаграми послідовності описують, як і в якому порядку функціонують об’єкти в системі. Ці діаграми широко використовуються для документування та розуміння вимог до нових та існуючих систем [33].

Далі за допомогою діаграм послідовності описано взаємодію користувача з системою та проілюстровано яким чином обробляються запити користувача всередині додатку. Кожен запит користувача обробляється керуючим класом, а вже

керуючий клас передає викликає відповідні методи обробки команд в інші компоненти системи (класи). У відповіді на кожну дію користувач отримує від системи певну інформацію, що також продемонстровано на діаграмі.

На рис. 3.5 змодельований процес завантаження тривимірної моделі в систему.

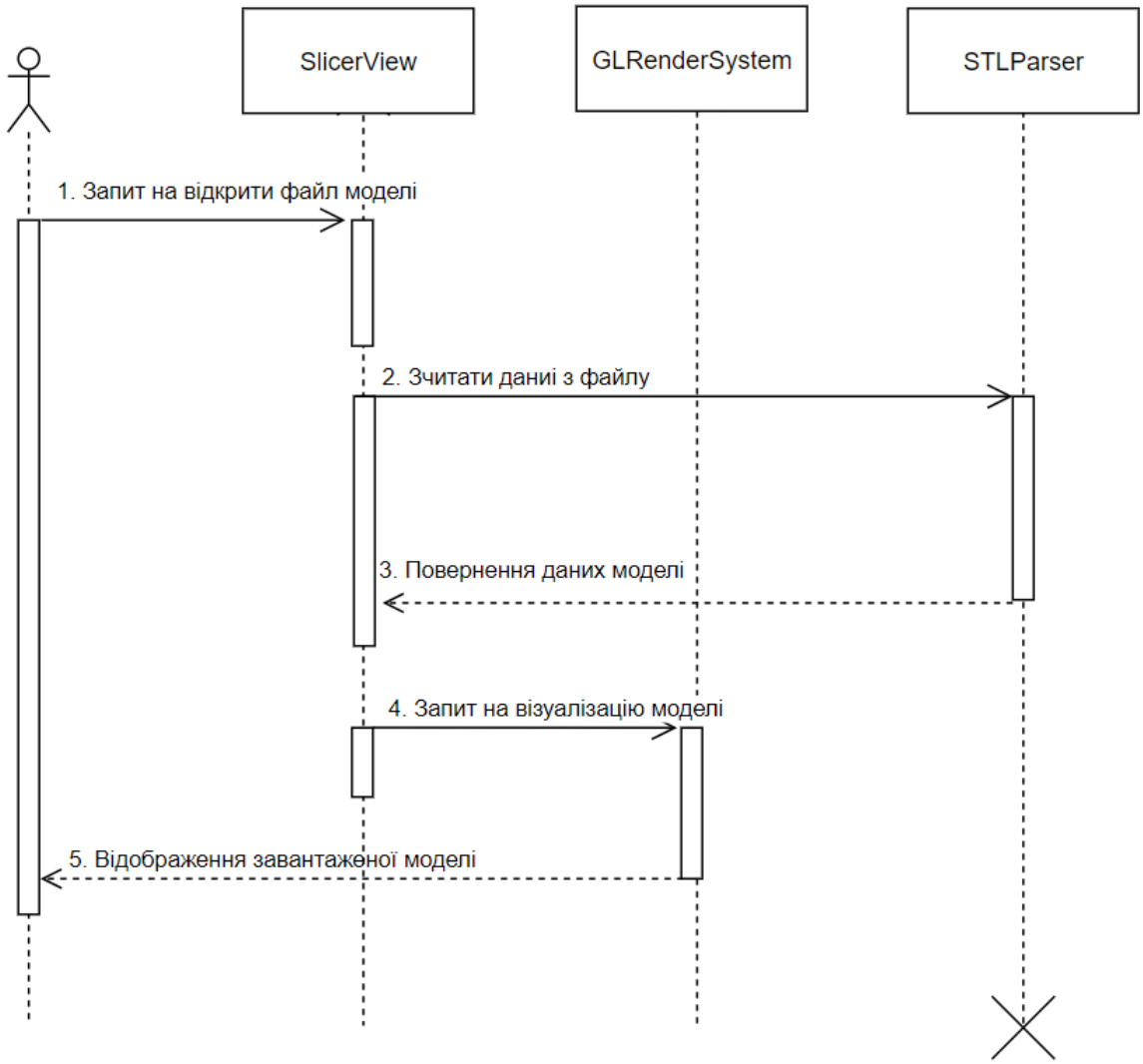

Рисунок 3.5 – Діаграма послідовності процесу завантаження моделі

Користувач надсилає запит керуючому класу (SlicerView) на відкриття файлу моделі. Керуючий клас надсилає повідомлення класу STLParser, який повертає дані моделі керуючому класу. Потім керуючий клас надсилає запит до класу GLRenderSystem на візуалізацію моделі.

Після завантаження моделі можна перейти до процесу калібрування моделі для її «прив’язки до місцевості». На рис. 3.6 наведено діаграму послідовності процесу калібрування моделі.

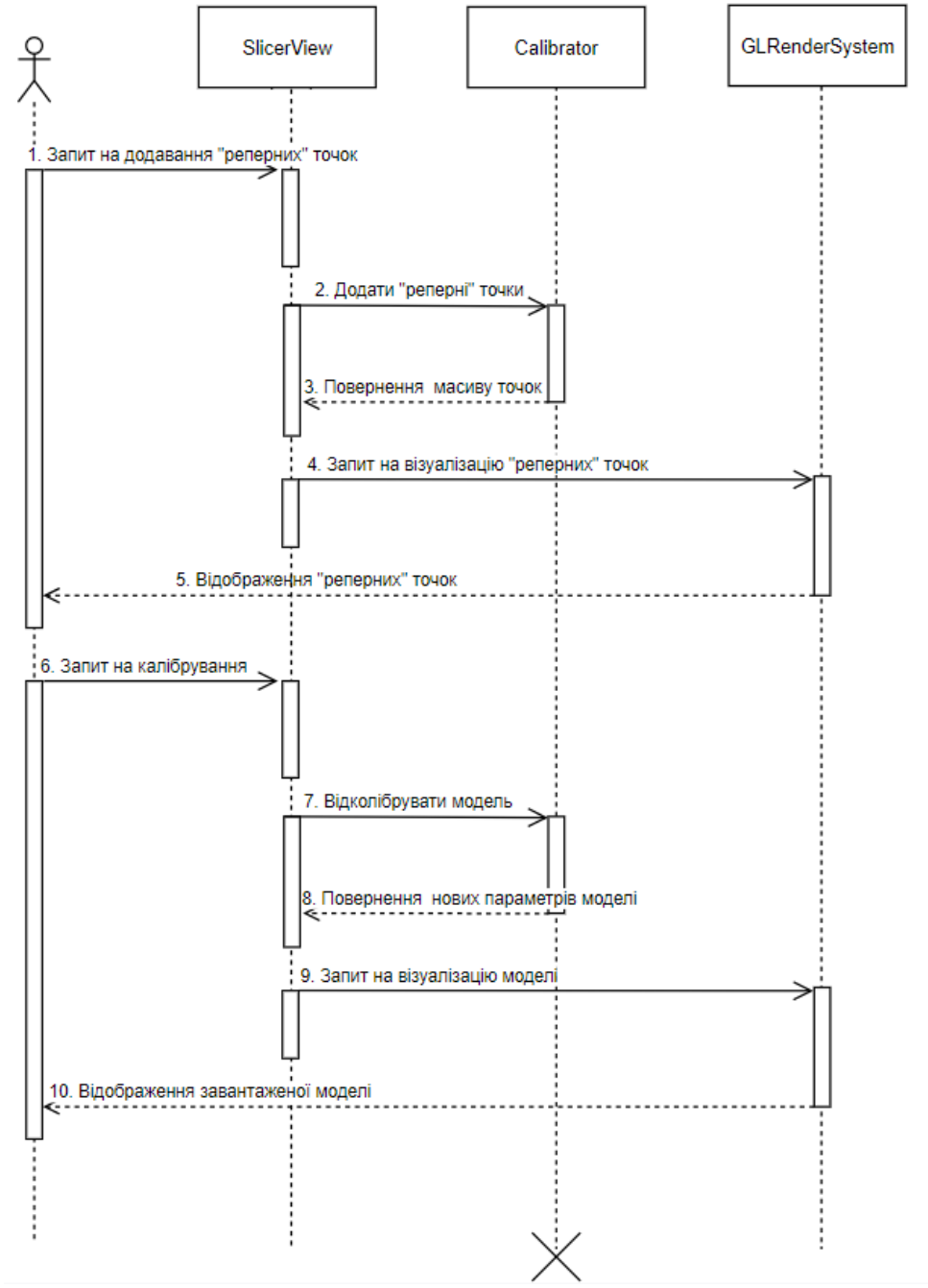

Рисунок 3.6 – Діаграма послідовності процесу калібрування моделі

Користувач надсилає запит керуючому класу на додавання реперних точок. SlicerView надсилає запит до класу Calibrator, який повертає масив доданих точок керуючому класу. Потім точки відображаються на екрані класом GLRenderSystem.

Тапер необхідно додати лінії заповнення для подальшої обробки моделі. На рис. 3.7 наведено діаграму послідовності процесу генерації заповнення.

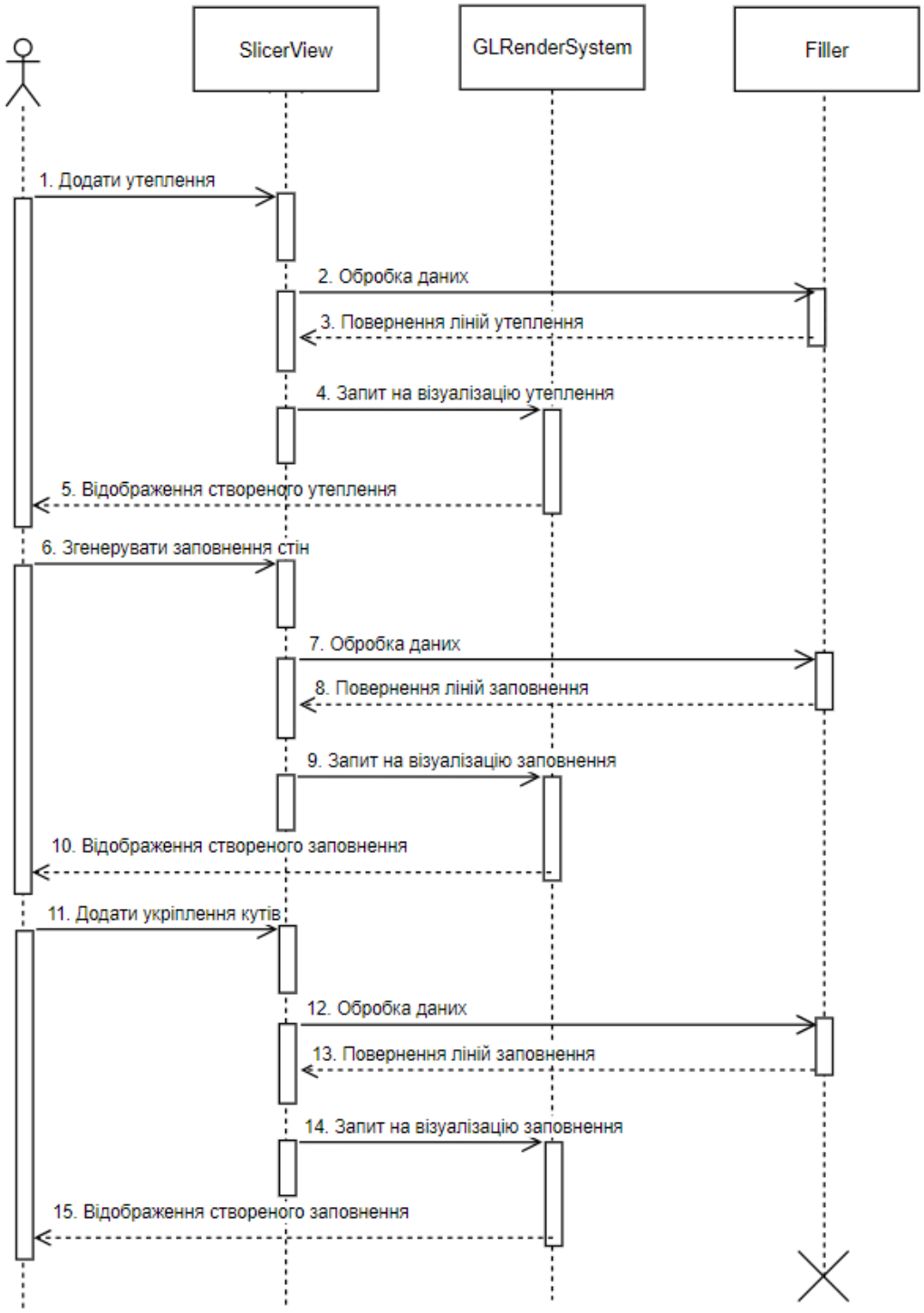

Рисунок 3.7 – Діаграма послідовності процесу генерації заповнення

Спочатку користувач надсилає запит керуючому класу на додавання утеплення, потім на генерацію заповнення стін і на додавання укріплень кутів. Керуючий клас

передає повідомлення до класу Filler, який генерує відповідний набір ліній і повертає їх у вигляді масиву. Після додавання кожного типу заповнення користувач може бачити результат своїх дій, адже всі лінії відображаються на екрані за допомогою класу GLRenderSystem.

Наступним кроком є генерація набору G-code команд. На рис. 3.8 представлено процес генерації G-code команд.

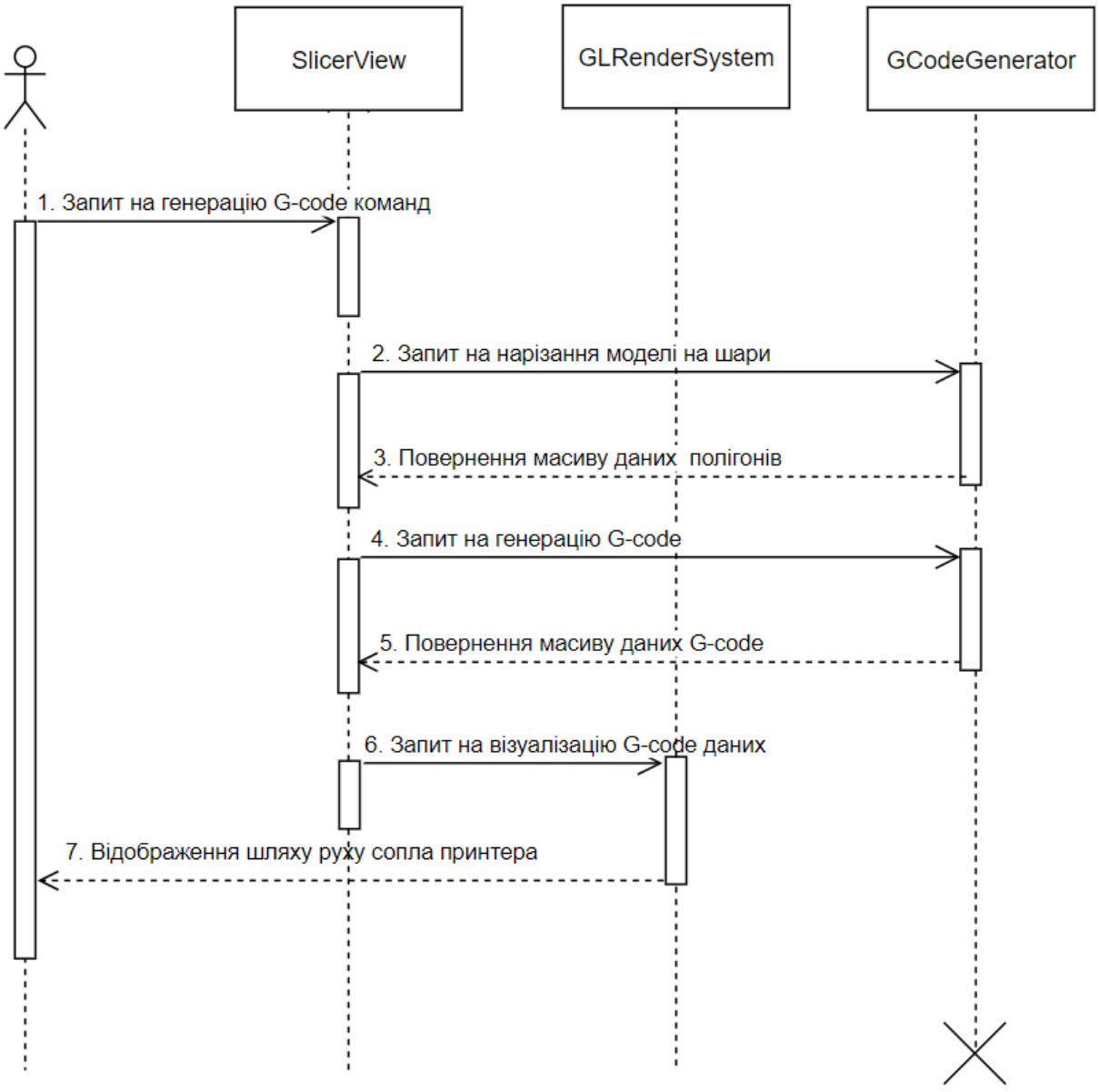

Рисунок 3.8 – Діаграма послідовності процесу генерації G-code команд

Користувач надсилає запит на генерацію G-code команд керуючому класу. Клас SlicerView надсилає запити класу GCodeGenerator на нарізання моделі та на генерацію шляху. Клас GLRenderSystem відображає шлях переміщення сопла принтера.

Тапер G-code команди можна зберегти в файл. На рис. 3.9 зображено процес збереження даних у файл.

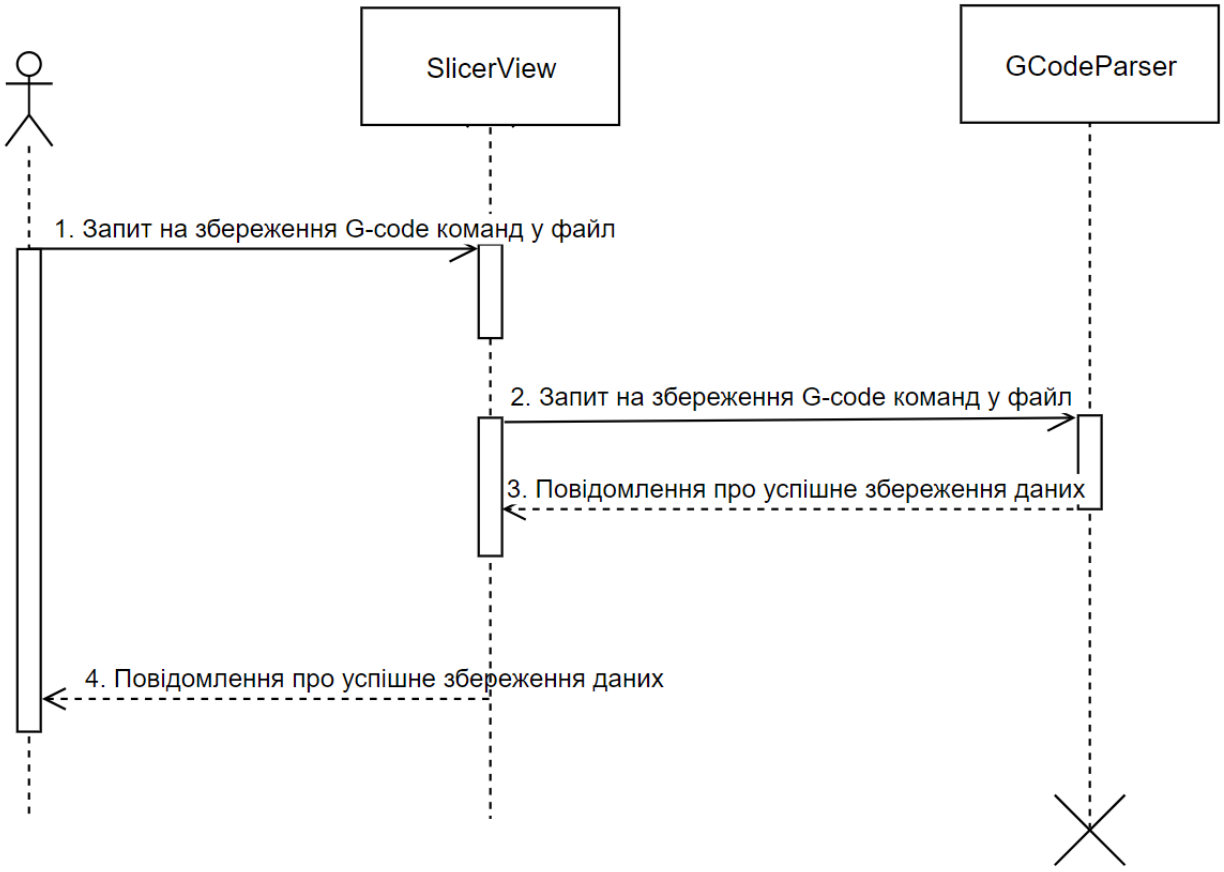

Рисунок 3.9 – Діаграма послідовності процесу генерації G-code команд

Користувач надсилає запит керуючому класу на збереження G-code команд. GCodeParser зберігає дані в файл та надсилає користувачу повідомлення про успішне збереження даних.

Далі користувач може переглядати сформований набір G-code команд (без можливості зміни команд), або почати процес нарізання моделі з початку, встановивши модель в новому місці чи задавши нові параметри друку.

3.3. Математична модель

Оскільки вся робота пов’язана з геометричними розрахунками, то в роботі використовується велика кількість формул. Основою програми є геометричні примітиви такі як точка (3.1), лінія (3.2) та трикутник (3.3).

$$\mathrm{Point}\{x, y, z\} \tag{3.1}$$

де x – проекція точки на вісь x, y – проекція точки на вісь y, – проекція точки на вісь z.

Лінія складається з двох точок та вектора нормалі. Набір ліній утворює замкнуті полігони, які є контуром фігури.

$$\mathrm{Line}\begin{Bmatrix} \mathrm{point1}(x, y, z) \\ \mathrm{point2}(x, y, z) \\ \mathrm{normal}(x, y, z) \end{Bmatrix} \tag{3.2}$$

де point1 та point2 – початкова та кінцева точки лінії, normal – вектор нормалі.

Трикутник складається з трьох точок та вектора нормалі. Це дозволяє однозначно ідентифікувати положення трикутника в просторі та спрощує подальші розрахунки. Масив трикутників зберігає всі дані моделі у такому вигляді, як вони зберігалися в .STL файлі.

$$\mathrm{Triangle}\begin{Bmatrix} \mathrm{normal}\ (x, y, z) \\ \mathrm{point1}(x, y, z) \\ \mathrm{point2}(x, y, z) \\ \mathrm{point3}(x, y, z) \end{Bmatrix} \tag{3.3}$$

де point1, point2 та point3 – точки трикутника, normal – вектор нормалі.

Під час генерації заповнення порожнин у стінах необхідно шукати точку перетину двох ліній (3.4) для автоматичної генерації заповнення вздовж обраної лінії. В даному випадку достатньо розглянути випадок на площині, а не в тривимірному просторі, адже заповнення генерується для одного шару і тільки потім буде оброблятися для перенесення на інші шари.

$$P = \begin{cases} A_1x+B_1y+C_1=0; \\ A_2x+B_2y+C_2=0; \end{cases} \tag{3.4}$$

де $A_1x+B_1y+C_1=0$, $A_2x+B_2y+C_2=0$ – рівняння ліній, що перетинаються, P – точка перетину цих ліній. Перед тим як шукати точку перетину двох прямих на площині необхідно з'ясувати чи існує точка перетину двох прямих. Дві прямі на площині мають точку перетину, якщо вони не паралельні. Дві прямі є паралельними, якщо скалярний добуток двох векторів побудованих на цих прямих дорівнює добутку довжин цих векторів. Далі наведена формула розрахунку скалярного добутку двох векторів (3.5).

$$\bar{a} \times \bar{b} = |\bar{a}| \times \left|\bar{b}\right| \times \cos\alpha \tag{3.5}$$

де $\bar{a}$ та $\bar{b}$ – вектори, α – кут між цими векторами.

Якщо відомо, що лінії перетинаються та відомо точку перетину цих ліній, то останнім кроком залишається з'ясувати чи належить знайдена точка лінії. Для цього необхідно підставити координати знайденої точки в рівняння лінії (3.6).

$$(x - x_1) \times (y_2 - y_1) - (x_2 - x_1) \times (y - y_1) = 0 \tag{3.6}$$

де x_1 та y_1 – координати точки початку лінії, x_2 та y_2 – координати точки кінця лінії, x та y – координати шуканої точки.

Ще однією важливою частиною генерації заповнення є створення сплайну для покращення якості друку заповнення. В роботі використано інтерполяційний многочлен Лагранжа (3.7) для знаходження проміжних значень функції.

$$\mathrm{L(x)} = \sum_{\mathrm{j=0}}^{\mathrm{n}} \mathrm{y_j l_j(x)} \tag{3.7}$$

де L(x) – шукане значення функції, $l_j(x)$ – базисний поліном, що визначається за формулою (3.8).

$$l_{\mathrm{j}}(\mathrm{x}) = \prod_{\mathrm{i=0}, j \neq i}^{\mathrm{n}} \frac{\mathrm{x - x_i}}{\mathrm{x_j - x_i}} = \frac{\mathrm{x - x_0}}{\mathrm{x_j - x_0}} \ldots \frac{\mathrm{x - x_{j-1}}}{\mathrm{x_j - x_{j-1}}} \frac{\mathrm{x - x_{j+1}}}{\mathrm{x_j - x_{j+1}}} \ldots \frac{\mathrm{x - x_n}}{\mathrm{x_j - x_n}} \tag{3.8}$$

де $\mathrm{x_i}$ – відповідне значення аргументу у відомих точках.

Поліном Лагранжа дає можливість створити плавні переходи між лініями, що пришвидшить процес друку оскільки не потрібно гальмувати на кінцях відрізків.

Наступним кроком є нарізання тривимірної моделі на шари. Оскільки трикутник представлено трьома точками та вектором нормалі, то можемо отримати рівняння площини. Рівняння площини можна отримати декількома способами: через три точки, через одну точку перпендикулярно вектору або на двох векторах [34]. В роботі використано формулу для знаходження рівняння площини через точку перпендикулярно вектору (3.9).

$$A(x - x_0) + B(y - y_0) + C(z - z_0) = 0 \tag{3.9}$$

де A, B, C – координати вектора нормалі, x_0, y_0, z_0 – координати будь якої з трьох точок трикутника. Ріжуча площина також задається однією точкою та вектором нормалі. Оскільки ріжуча площина має бути паралельною площині поля, то вектор

нормалі такої площини має наступні вигляд $\bar{n} = \{0,0,1\}$. Точку можна обрати довільно, а координата z буде змінюватися з підняттям площини вгору.

Спочатку необхідно з'ясувати чи перетинаються ріжуча площина та площина трикутника. Площини не перетинаються тільки якщо вони паралельні. Для того, щоб з'ясувати чи паралельні площини необхідно і достатньо перевірити чи є паралельними вектори нормалі цих площин [34]. Два вектори паралельні, якщо їх векторний добуток дорівнює нуль-вектору (3.10).

$$\bar{a} \times \bar{b} = \bar{0} \tag{3.10}$$

де $\bar{a}$ та $\bar{b}$ – вектори в тривимірному просторі, $\bar{0}$ – нульовий вектор.

Розрахунок векторного добутку двох тривимірних векторів здійснюється через детермінант (3.11)

$$\bar{a} \times \bar{b} = \begin{matrix} \bar{\imath} & \bar{\jmath} & \bar{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{matrix} = (a_2 b_3 - a_3 b_2)\bar{\imath} + (a_3 b_1 - a_1 b_3)\bar{\jmath} + (a_1 b_2 - a_2 b_1)\bar{k} \tag{3.11}$$

де a_1, a_2, a_3 – координати вектора $\bar{a}$, b_1, b_2, b_3 – координати вектора $\bar{b}$.

Оскільки площина побудована на трикутнику обмежена крайніми точками цього трикутника, то необхідно переконатися, що всі три точки трикутника не знаходяться нижче чи вище ріжучої площини. Для цього необхідно підставити координати кожної точки трикутника в рівняння площини (3.9). Якщо результат розрахунку дорівнює нулю – точка лежить на площині, якщо результат менше нуля – точка знаходиться нижче площини, якщо результат більше нуля – точка лежить вище ріжучої площини. Ріжуча площина розрізає трикутник якщо хоча б дві точки лежать по різні сторони трикутника або дві точки лежить на площині трикутника.

Далі потрібно знайти лінію, яка є результатом перетину двох площин. Для цього необхідно розв'язати систему рівнянь (3.12), результатом якої і буде рівняння

шуканої лінії. Це і є базовий алгоритм нарізання моделі на шари, який взято за основу інформаційної технології.

$$\begin{cases} A_1x+B_1y+C_1z+D_1=0; \\ A_2x+B_2y+C_2z+D_2=0; \end{cases} = \begin{cases} C_1z+D_1=0; \\ A_2x+B_2y+C_2z+D_2=0; \end{cases} \quad (3.12)$$

де рівняння даної системи – рівняння площин, що перетинаються. A, B і C не можуть одночасно дорівнювати нулю. Рівняння ріжучої площини можна спростити, оскільки ця площина гарантовано паралельна площини XY.

На рис. 3.10 представлено можливі варіанти перетину ріжучої площини з площиною трикутника.

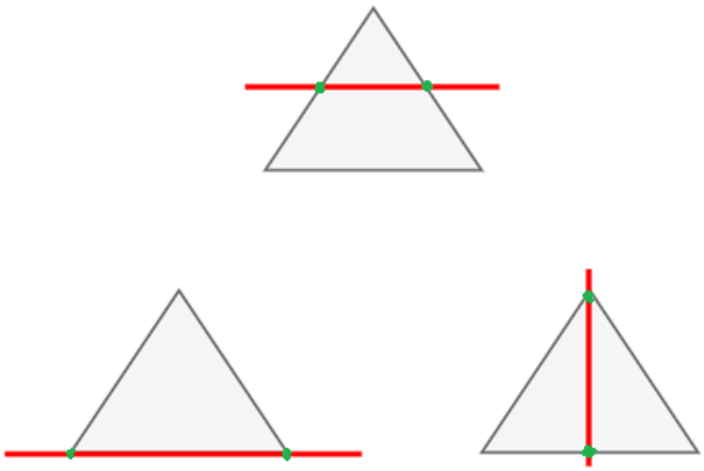

Рисунок 3.10 – Варіанти перетину ріжучої площини з трикутником

В результаті отримуємо набір ліній, що утворює або декілька замкнутих контурів. Тепер можна переходити до обрізання заповнення для конкретного шару. Для того, щоб визначити чи знаходиться лінія всередині замкнутого полігону використано метод суми кутів. Метод дозволяє визначити чи знаходиться точка всередині замкнутого контуру з будь-якою кількістю кутів [35].

Необхідно розрахувати суму кутів за формулою (3.13).

$$\sum_{i=1}^{n} \varphi_i \tag{3.13}$$

де φ_i – кут (в радіанах і зі знаком) між променями PV_{i-1} та PV_i (3.14).

$$\varphi_i = \arccos\left(\frac{PV_{i-1} \cdot PV_i}{|PV_{i-1}| \cdot |PV_i|}\right) \operatorname{sign}\left(\det\begin{pmatrix} PV_{i-1} \\ PV_i \end{pmatrix}\right) \tag{3.14}$$

де P – точка, місцезнаходження якої, по відношенню до полігона, намагаємося визначити, V – вершина замкнутого набору ліній.

Точка лежить зовні багатокутника, якщо сума кутів дорівнює нулю (або є достатньо близькою до цього значення).

Не можна використовувати адаптивний алгоритм нарізання моделі [22], оскільки виробничий процес друку будівель потребує однакової висоти шару. Обрізання заповнення для кожного шару є ресурс ємким процесом, а будівля не сильно змінюється в горизонтальній площині. Тому можна аналізувати чи змінився зріз шару і якщо не змінився, то використовувати заповнення з попереднього шару.

Найпростішим способом аналізу зміни зрізу шару є аналіз зміни площі шару. Споруди можуть бути абсолютно різної форми та з різною кількістю полігонів, тому площа таких полігонів розраховується з формулою площі Гауса (3.15). Вона працює для багатокутників в декартовій системі координат. Інша назва цієї формули – формула шнурування або алгоритм шнурування, оскільки від'ємні та додатні множники розташовані хрест-навхрест (рис. 3.11)

$$S = \frac{1}{2}\left|\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_n y_1\right| \tag{3.15}$$

де S – площа багатокутника, x та y – координати вершин багатокутника, n – кількість вершин багатокутника.

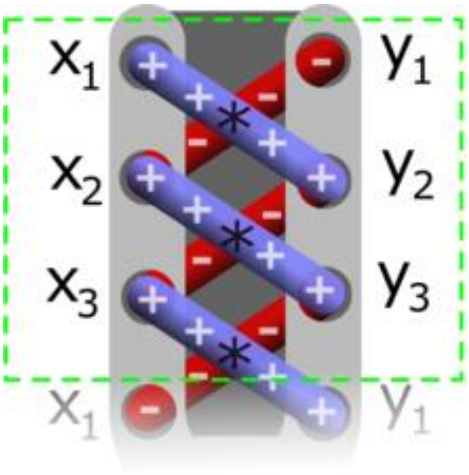

Рисунок 3.11 – Візуалізація методу шнурування

Тепер, коли маємо набір ліній, по якому необхідно пройти залишається перевести координати з точок з декартових в полярні. Полярний радіус можна розрахувати зі значень декартових координат з використаніням формули (3.16).

$$\rho = \sqrt{x^2 + y^2} \tag{3.16}$$

де ρ – полярний радіус, x та y – координати точки в декартовій системі.

Для обчислення полярного кута в інтервалі $[0, 2\pi)$ можна використати систему рівнянь (3.17).

$$\varphi = \begin{cases} arctg\left(\frac{y}{x}\right) & x > 0, \quad y \geq 0 \\ arctg\left(\frac{y}{x}\right) + 2\pi & x > 0, \quad y < 0 \\ arctg\left(\frac{y}{x}\right) + \pi & x < 0 \\ \frac{\pi}{2} & x = 0, \quad y > 0 \\ \frac{3\pi}{2} & x = 0, \quad y < 0 \end{cases} \tag{3.17}$$

де φ – полярний кут, x та y – координати точки в декартовій системі.

Для обчислення полярного кута в інтервалі $[-\pi, \pi)$ можна використати систему рівнянь (3.18).

$$\varphi = \begin{cases} arctg\left(\frac{y}{x}\right) & x > 0, \\ arctg\left(\frac{y}{x}\right) + \pi & x < 0, \quad y \geq 0 \\ arctg\left(\frac{y}{x}\right) - \pi & x < 0 \quad y < 0 \\ \frac{\pi}{2} & x = 0, \quad y > 0 \\ -\frac{\pi}{2} & x = 0, \quad y < 0 \end{cases} \tag{3.18}$$

де φ – полярний кут, x та y – координати точки в декартовій системі.

Залишилося записати сформований шлях в полярній системі координат в файл і можна друкувати будівлю.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

4.1. Приклад роботи з програмою

Перед початком роботи з програмою необхідно підготувати модель. Часто буває так, що у замовника не вистачає кваліфікації чи відповідних експертів, які б підготували модель для друку. Тому замовник надає креслення споруди, а створювати тривимірну модель та готувати її до друку має оператор. На рис. 4.1 наведено креслення споруди кафетерію розміром 6Х8 метрів.

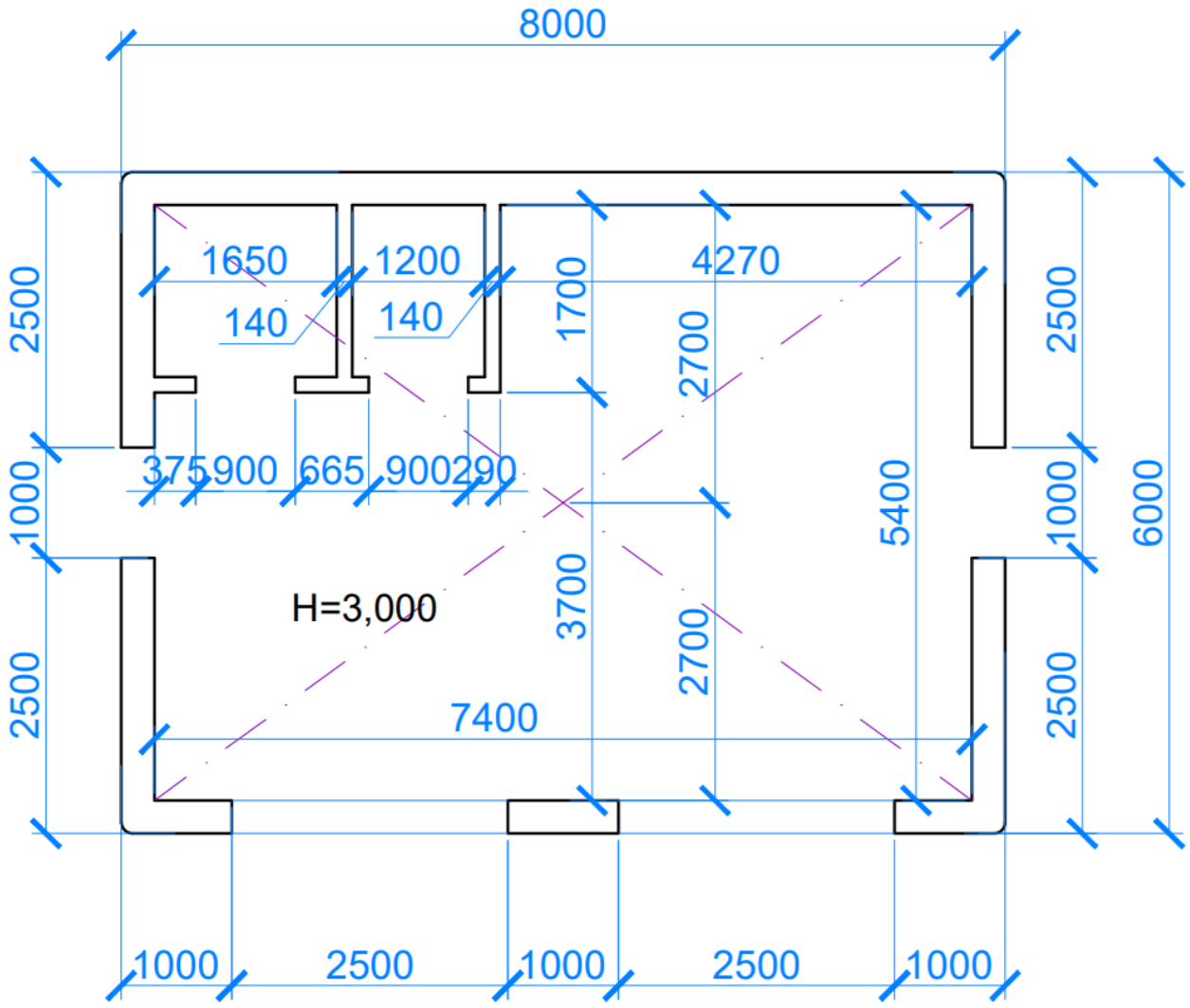

Рисунок 4.1 – Креслення споруди кафетерію

Для створення тривимірної моделі можна використати будь-яку систему автоматизованого проектування. Далі описано порядок дій при створенні тривимірної моделі в SolidWorks. Відкриваємо SolidWorks та створюємо новий документ типу «Part» (рис. 4.2).

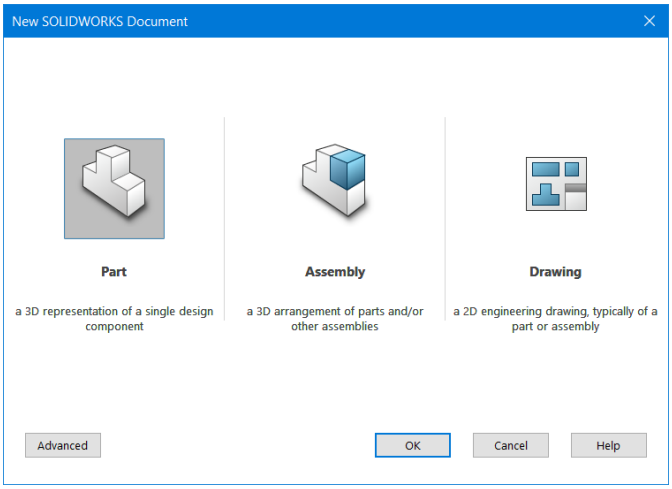

Рисунок 4.2 – Створення нового документу SolidWorks

Далі створюємо «Sketch» і за допомогою інструментів «Line», «Corner Rectangle» та «Smart Dimension» створюємо ескіз споруди з встановленням розмірів (рис. 4.3).

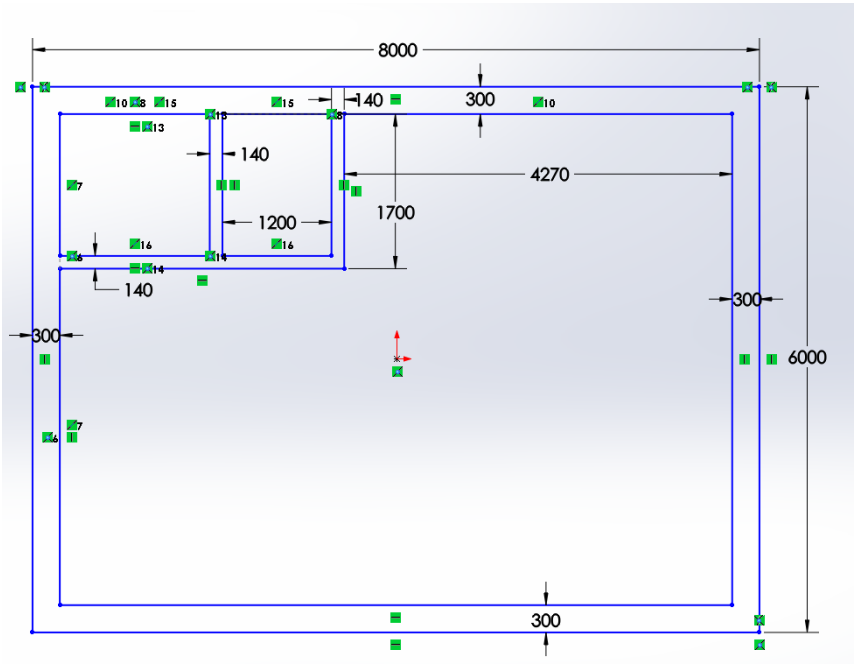

Рисунок 4.3 – Створення нового документу

Далі за допомогою інструменту «Extruded Boss/Base» видавлюємо ескіз на висоту 3200 міліметрів (рис. 4.4).

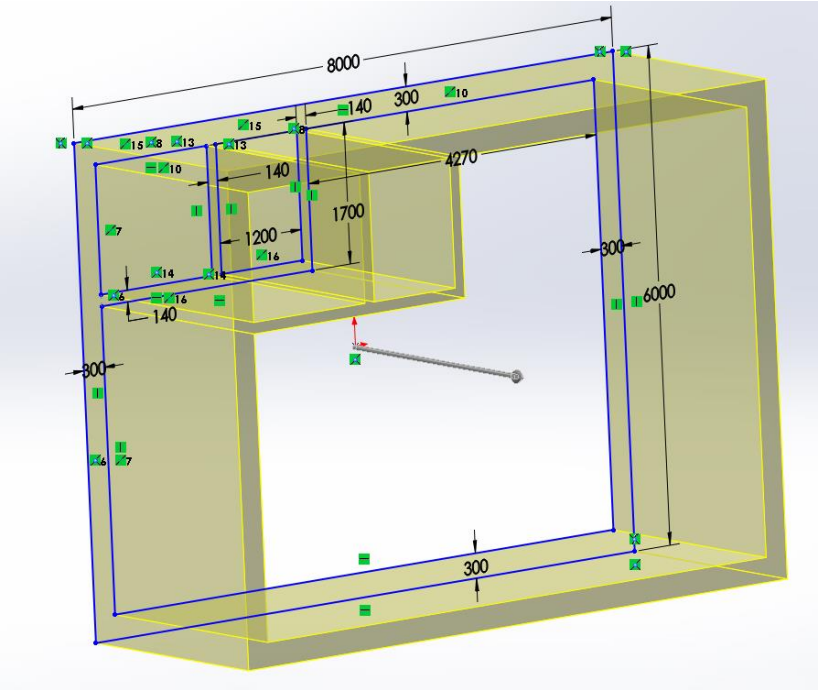

Рисунок 4.4 – Видавлювання ескізу

Для того, щоб додати отвори для встановлення дверей та вікон, потрібно на гранях моделі створити ескіз та за допомогою інструменту «Extruded Cut» зробити отвори в моделі. Результат додавання дверей та вікон наведено на рис. 4.5.

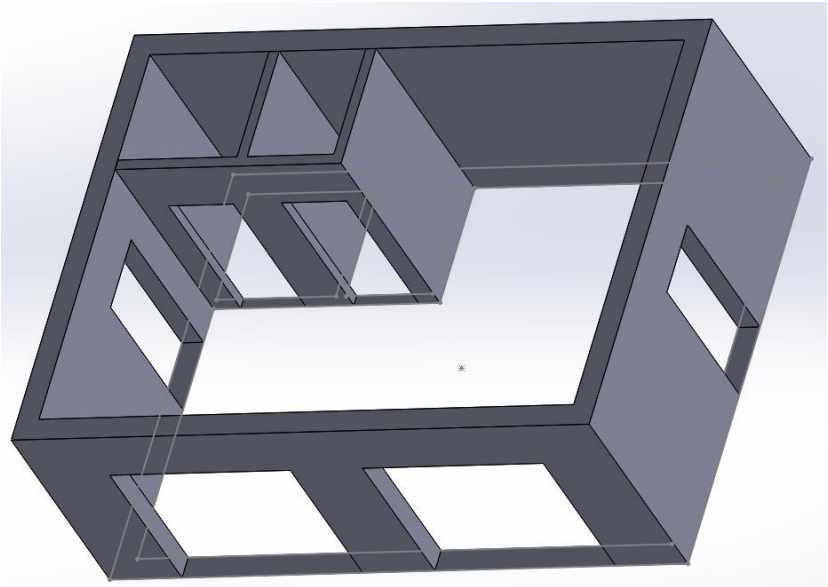

Рисунок 4.5 – Додавання дверей та вікон до моделі

Тепер необхідно зберегти створену тривимірну модель в файл формату .STL. Для цього переходимо в пункт меню «File=>Save As...» та обираємо формат STL для збереження. В діалоговому вікні налаштувань параметрів збереження моделі (рис. 4.6) можна виставити точність збереження моделі. Чим вища точність, тим більша кількість трикутників буде додана в файл на заокруглених поверхнях. Також можна обрати формат збереження даних (Binary або ASCII) та встановити одиниці вимірювання.

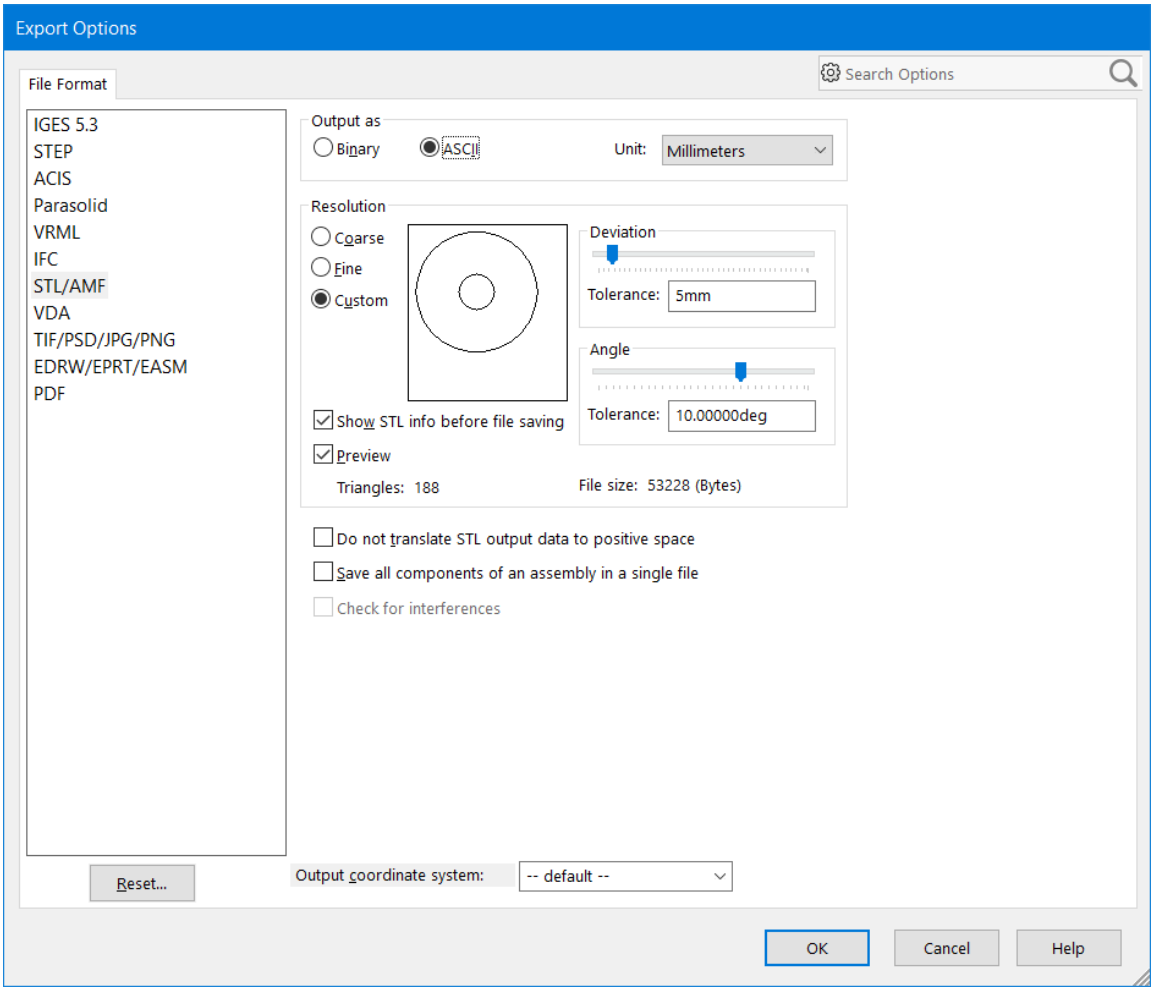

Рисунок 4.6 – Вікно налаштувань параметрів збереження моделі

Після підтвердження параметрів збереження моделі на екрані можна побачити прив'ю споруди та діалогове вікно з відомостями про збережений файл (кількість трикутників в моделі, розмір файлу в байтах та шлях збереження файлу). На рис. 4.5 прив'ю збереження моделі в форматі STL.

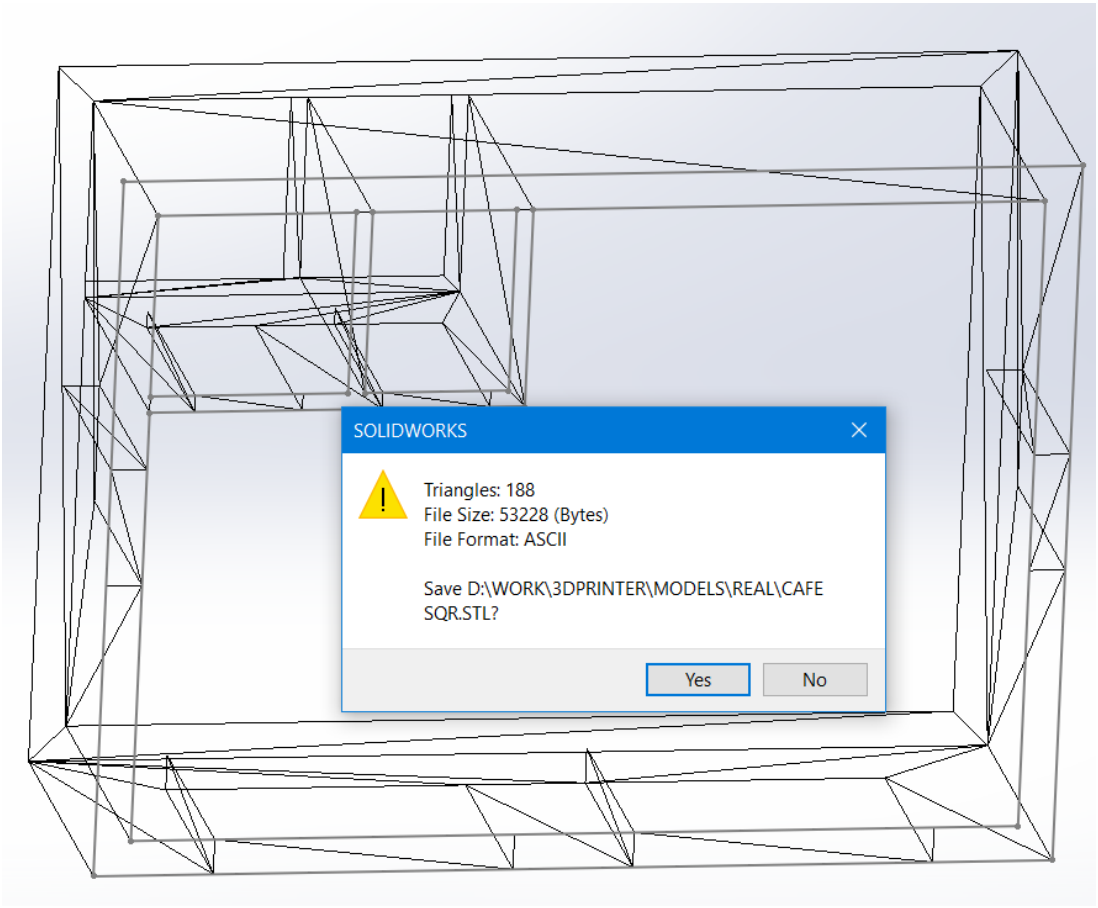

Рисунок 4.7 – Попередній перегляд збереження моделі в форматі STL

Модель збережена у файлі формату .STL, а отже можна починати процес слайсингу моделі за допомогою додатка Slicer.exe (рис. 4.8).

Рисунок 4.8 – Іконка програми Slicer.exe

Після запуска програми перед користувачем з’являється вікно програми з полем для розміщення моделі та координатними осями (рис. 4.9).

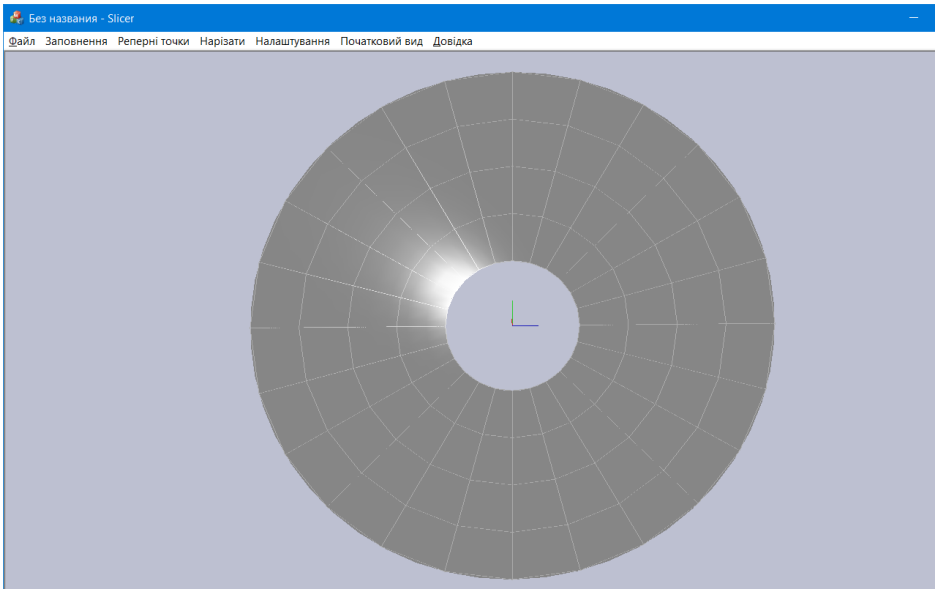

Рисунок 4.9 – Початкове вікно програми

Для того, щоб завантажити модель до програми необхідно обрати пункт меню «Файл=>Відкрити» або скористатися комбінацією клавіш «Ctrl+o». Далі необхідно обрати потрібний файл та натиснути кнопку «Open». Модель з’явиться на полі і буде доступною для подальшої роботи. Модель підсвічується червоним кольором, якщо перетинає границі поля для друку (рис. 4.10).

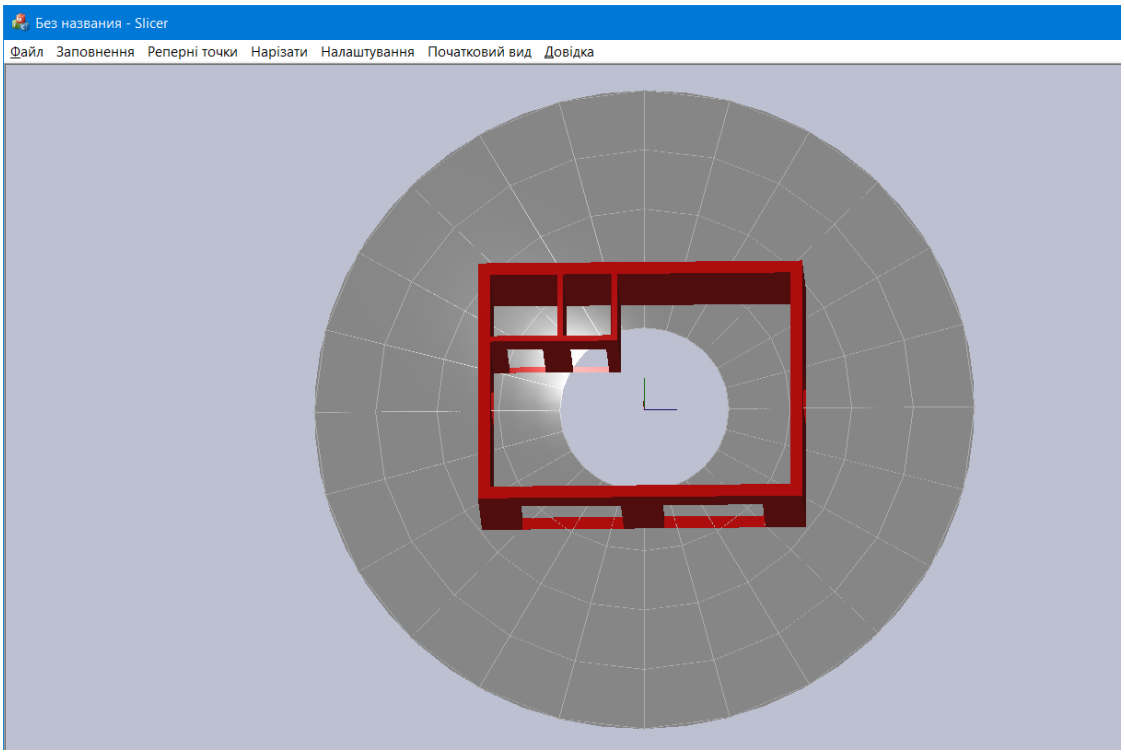

Рисунок 4.10 – Неправильне розміщення моделі на полі

Для переходу в режим переміщення моделі необхідно натиснути на клавішу «o» та пересувати модель утримуючи ліву кнопку миші. Для виходу з режиму переміщення моделі можна скористатися кнопкою «o» або «Esc» Правильно розміщена модель підсвічуються синім кольором (рис. 4.11).

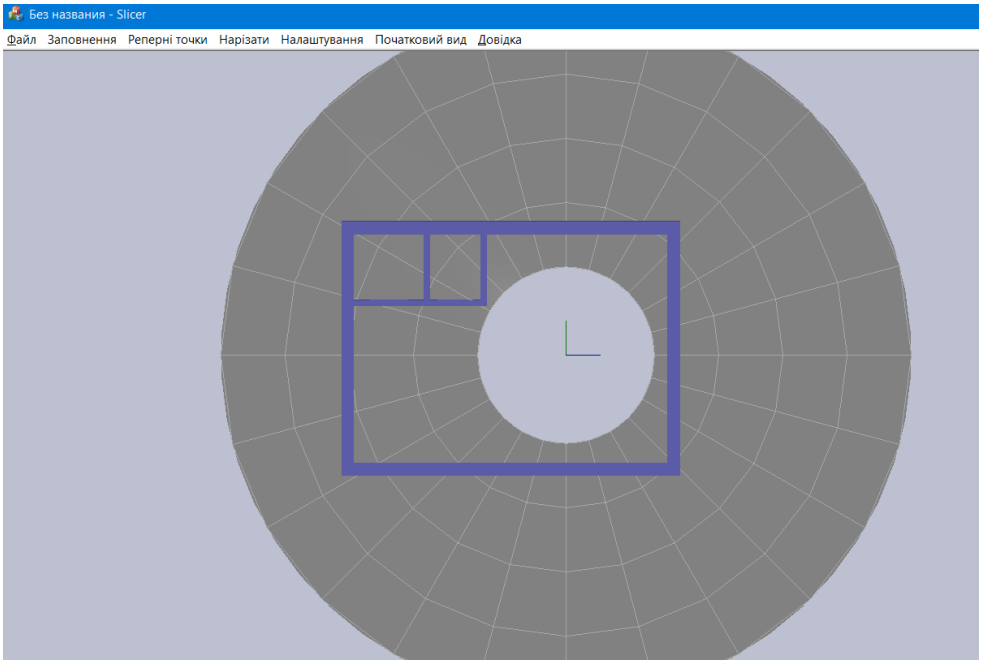

Рисунок 4.11 – Правильне позиціонування моделі

Для точного розміщення моделі та її «прив'язки» до місцевості необхідно перейти в режим калібрування моделі. Для цього обираємо пункт меню «Реперні точки=>Калібрування». Далі додаємо реперні точки на поле за допомогою діалогового вікна «Реперні точки=>Додати точку», що наведено на рис. 4.12.

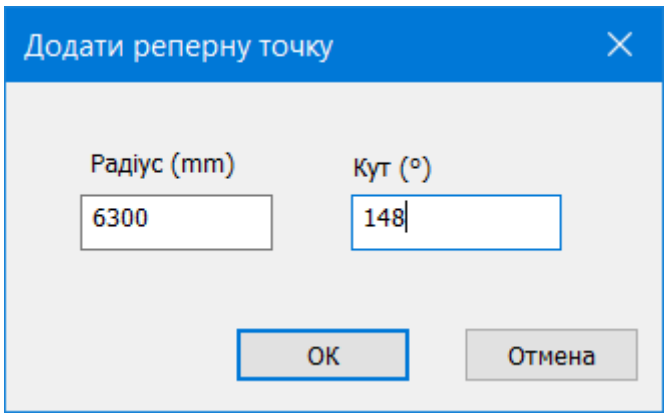

Рисунок 4.12 – Додавання реперної точки

Після додавання точки підсвічуються червоним та доступні для вибору. Прив’язка моделі здійснюється вибором чотирьох точок: спочатку потрібно обрати дві реперні, потім – дві точки на моделі та натиснути «Enter». На рис. 4.13 наведено зображення положення моделі до калібрування, а на рис. 4.14 – після.

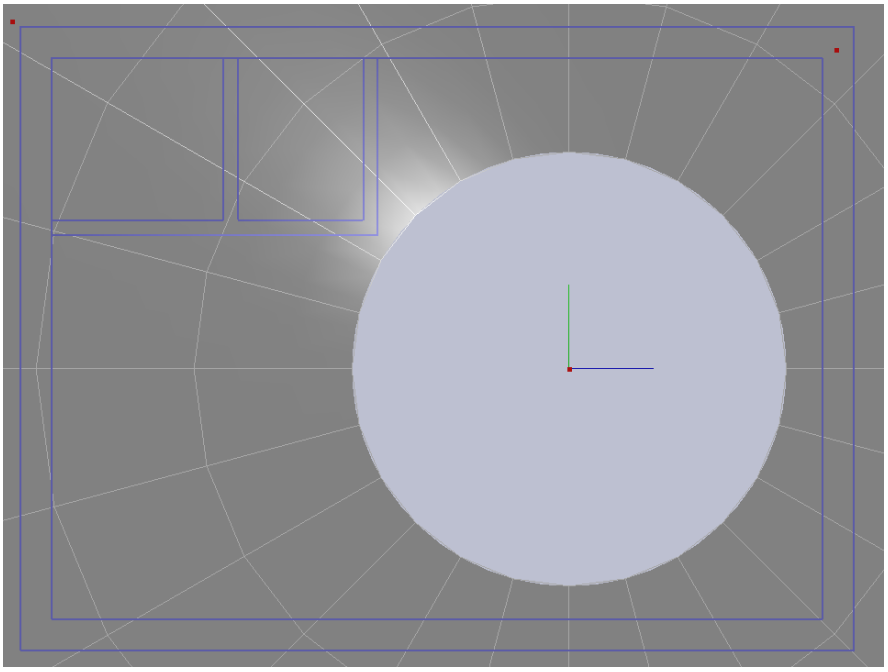

Рисунок 4.13 – Положення моделі до калібрування

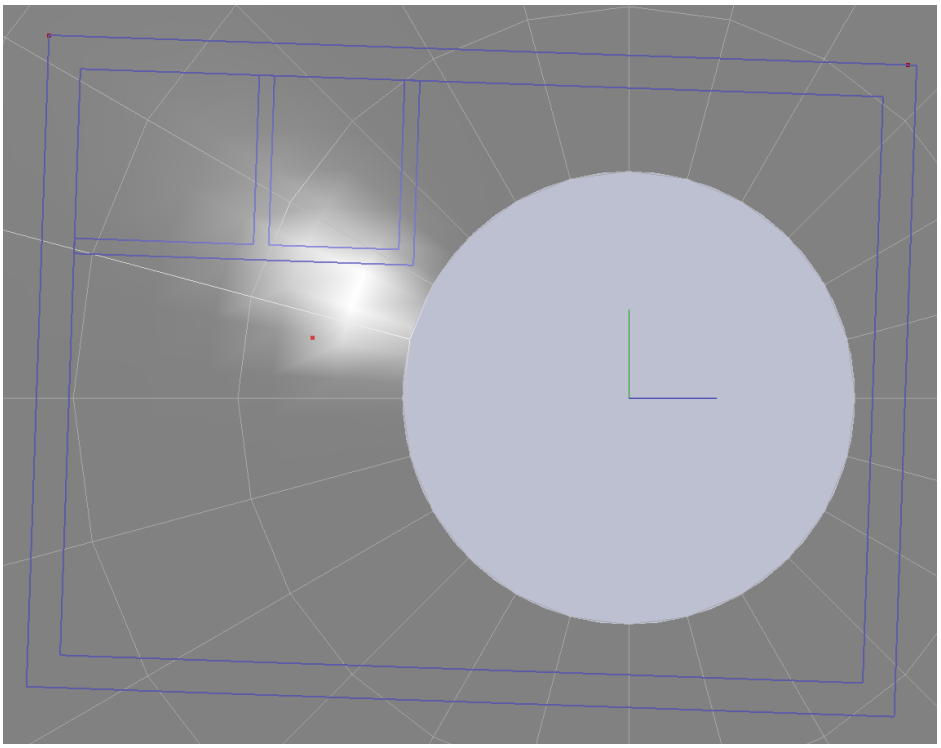

Рисунок 4.14 – Положення моделі до калібрування

Перед початком генерації заповнення необхідно встановити параметри налаштувань для конкретного проекту. На рис. 4.15 зображено вікно налаштувань з конфігурацією параметрів під проект кафетерію.

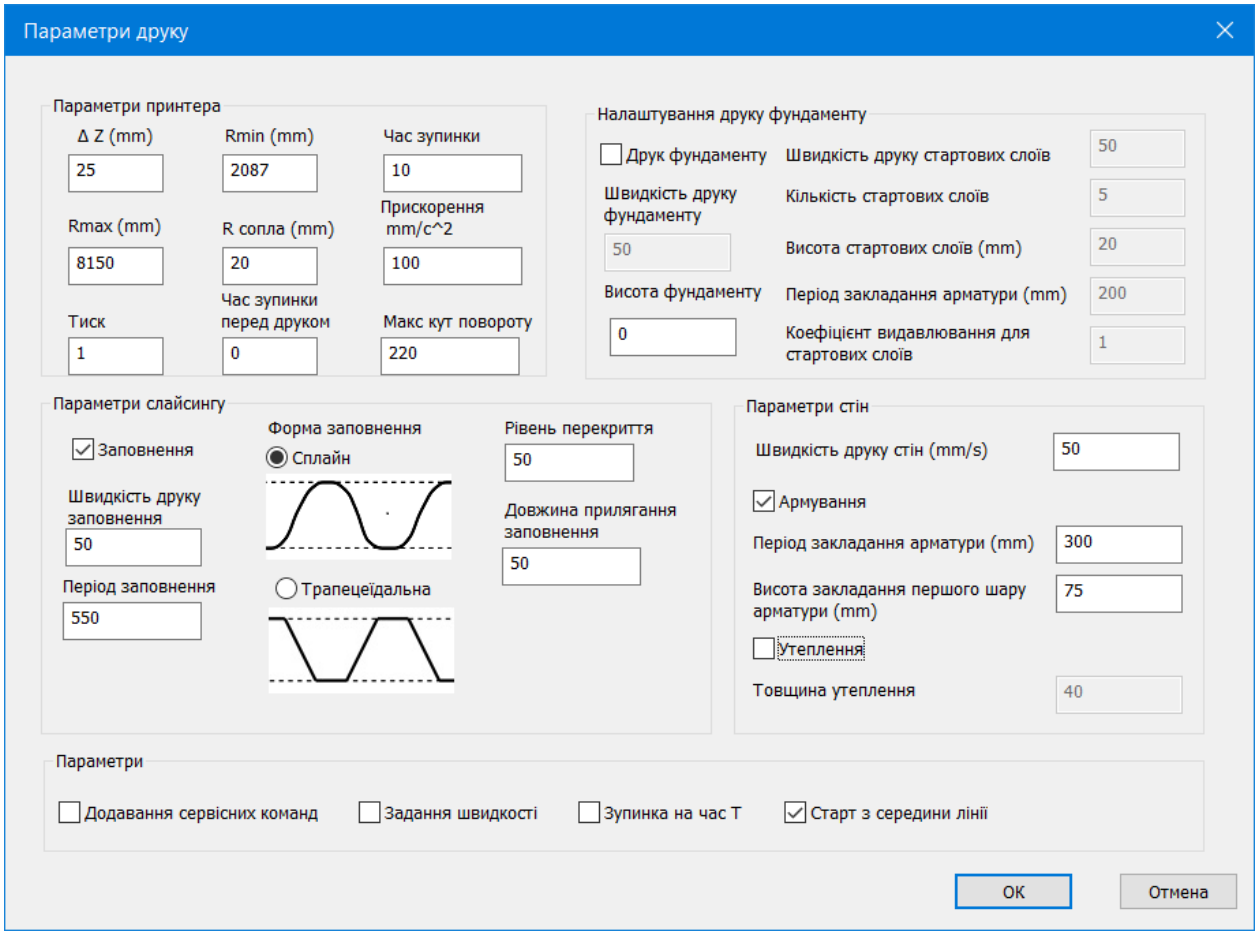

Рисунок 4.15 – Вікно налаштувань проекту

Як видно з рис. 4.15 формою заповнення обрано сплайн, а період заповнення дорівнює 550 міліметрам. Також, можна помітити, що прапорець утеплення вимкнутий, оскільки проект не передбачає наявність утеплювача. В параметрах стін увімкнене армування та встановлено період закладання арматури й висоту закладання першого шару арматури. В параметрах принтера вказано конструкційні параметри принтера (максимальний та мінімальний радіус вильоту стріли, радіус сопла принтера та максимальний кут повороту). Також, продемонстровано параметри друку (висота шару, час зупинки перед друком, прискорення принтера та початковий тиск в системі подачі суміші для друку).

Для переходу до режиму генерації заповнення необхідно обрати пункт меню «Заповнення=>Режим заповнення». Хоча, даний проект не передбачає наявність утеплення на рис. 4.16 продемонстровано додавання ліній утеплення «Заповнення=>Утеплення».

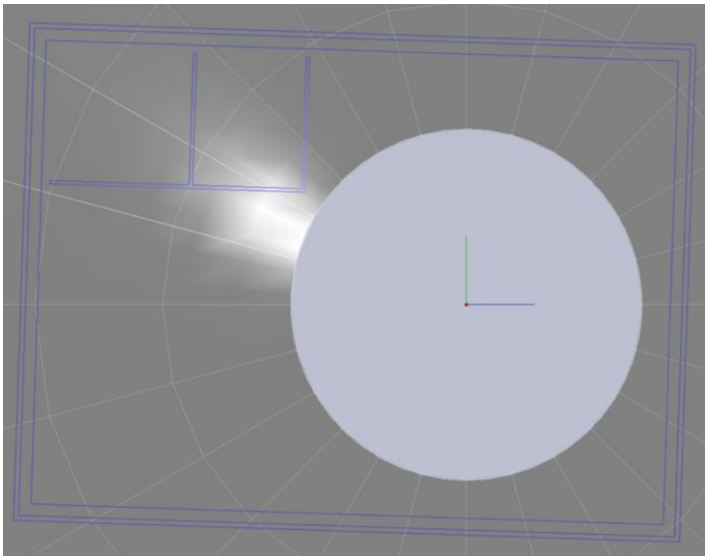

Рисунок 4.16 – Додавання утеплення

Як видно з рис. 4.16, з’явився додатковий контур утеплення. Тепер можна додавати лінії укріплення кутів та лінії заповнення стін. На рис. 4.17 продемонстровано заповнення порожнин стін та укріплення кутів сплайном.

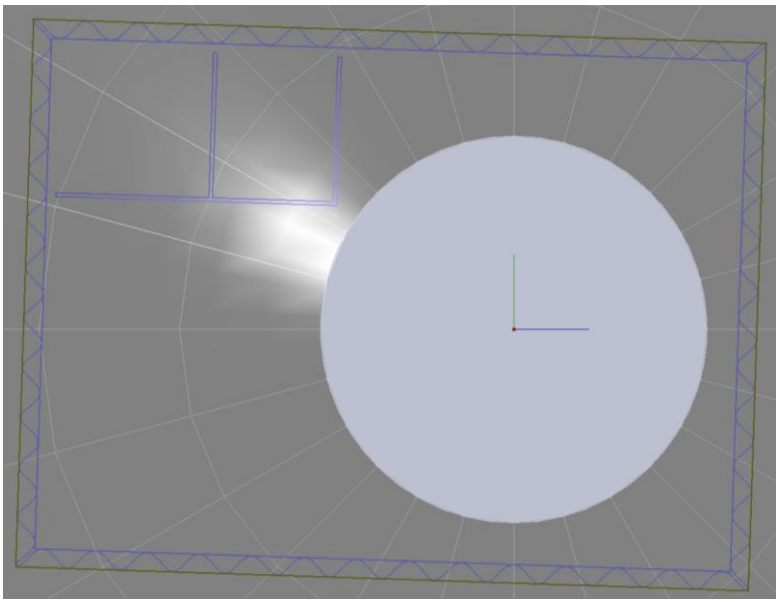

Рисунок 4.17 – Заповнення сплайном

Для генерації трапецеїдального заповнення необхідно встановити відповідні налаштування та обрати пункт меню «Заповнення=>Обрати лінії». Для додавання укріплень кутів окремим елементом необхідно обрати пункт меню «Заповнення=> Додати кут». На рис. 4.18 продемонстровано трапецеїдальну схему заповнення та обидва види укріплень кутів.

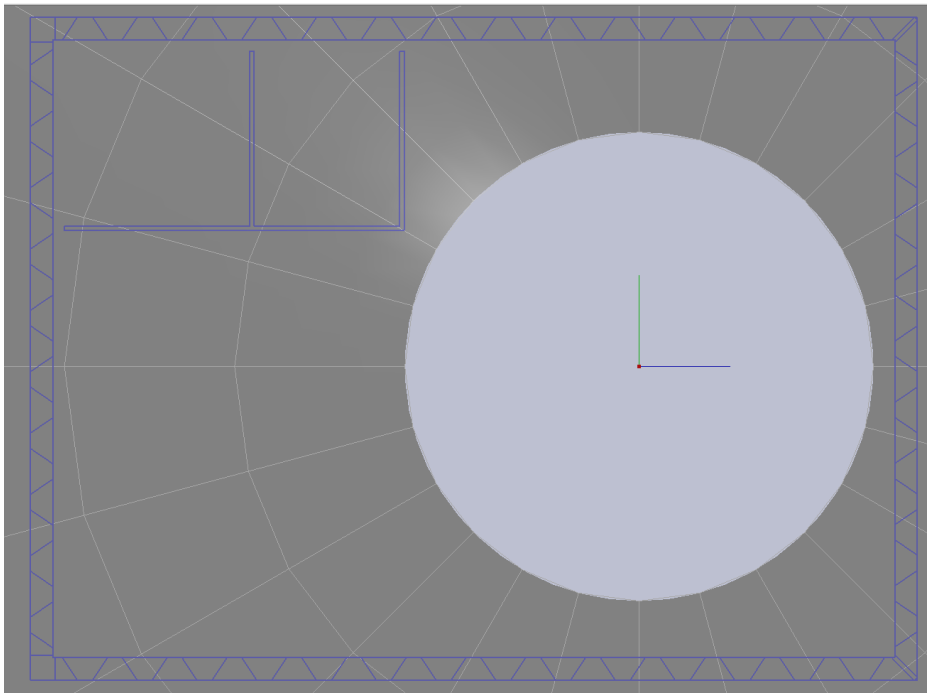

Рисунок 4.18 – Трапецеїдальна схема заповнення та укріплення кутів

Тепер можна переходити до генерації G-code команд, обравши пункт меню «Нарізати». Після спішної генерації G-code команд користувачу надсилається повідомлення про це, що зображено на рис. 4.19.

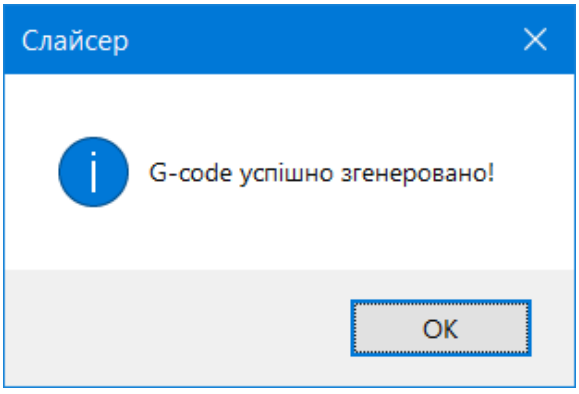

Рисунок 4.19 – Повідомлення про успішну генерацію G-code

Для переходу в режим перегляду G-code команд потрібно скористатися клавішою «v». Сині лінії – це лінії друку, а червоні лінії – це лінії холостого переміщення. Вид згори на фігуру наведено на рис. 4.20.

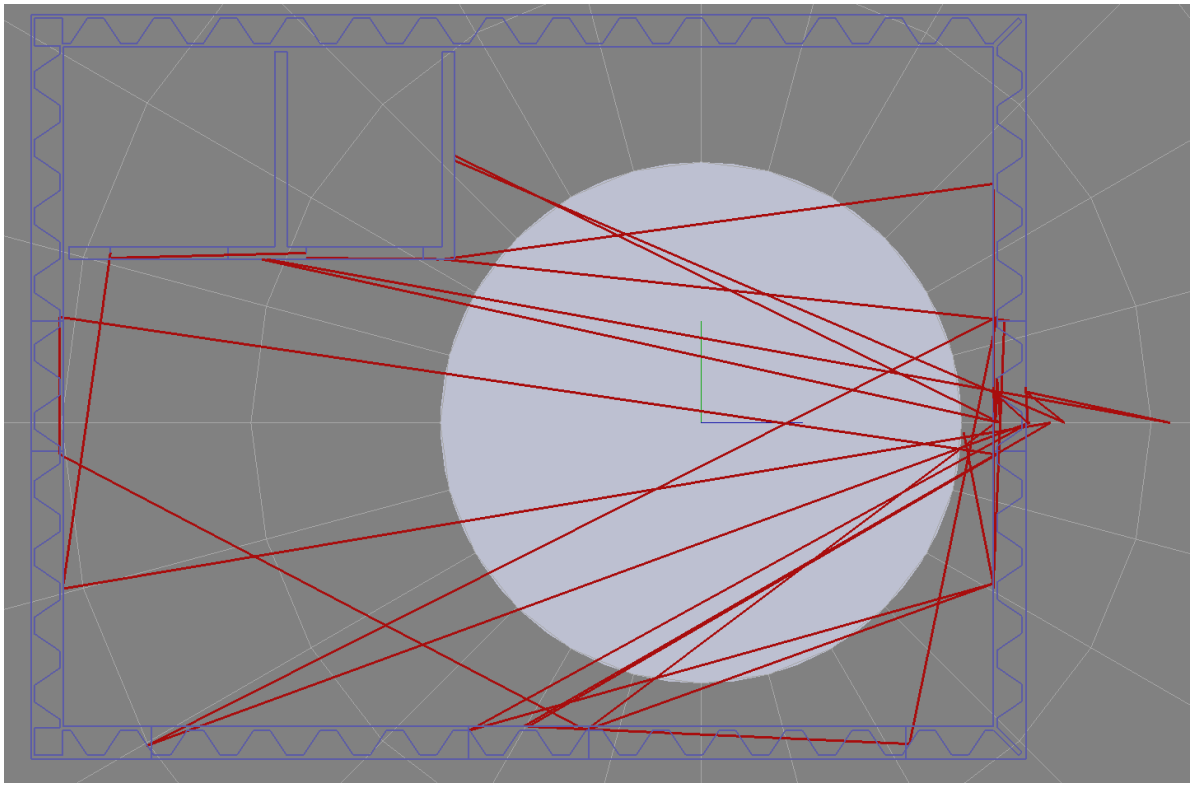

Рисунок 4.20 – Режим перегляду G-code команд

Також за допомогою повзунка можна регулювати кількість ліній для перегляду. Приклад наведено на рисунках 4.21 – 4.22.

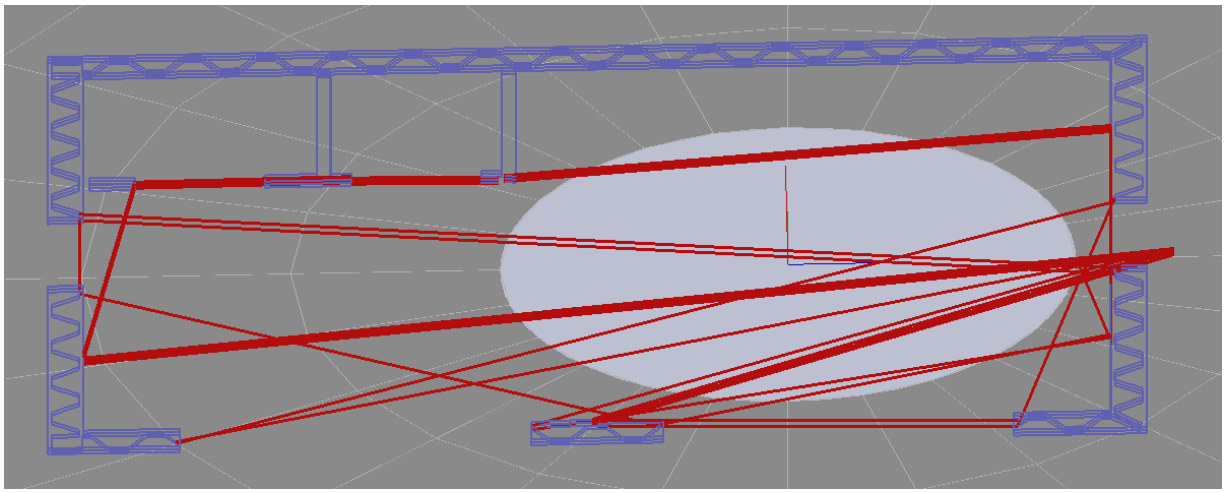

Рисунок 4.21 – Регулювання кількості відображуваних команд

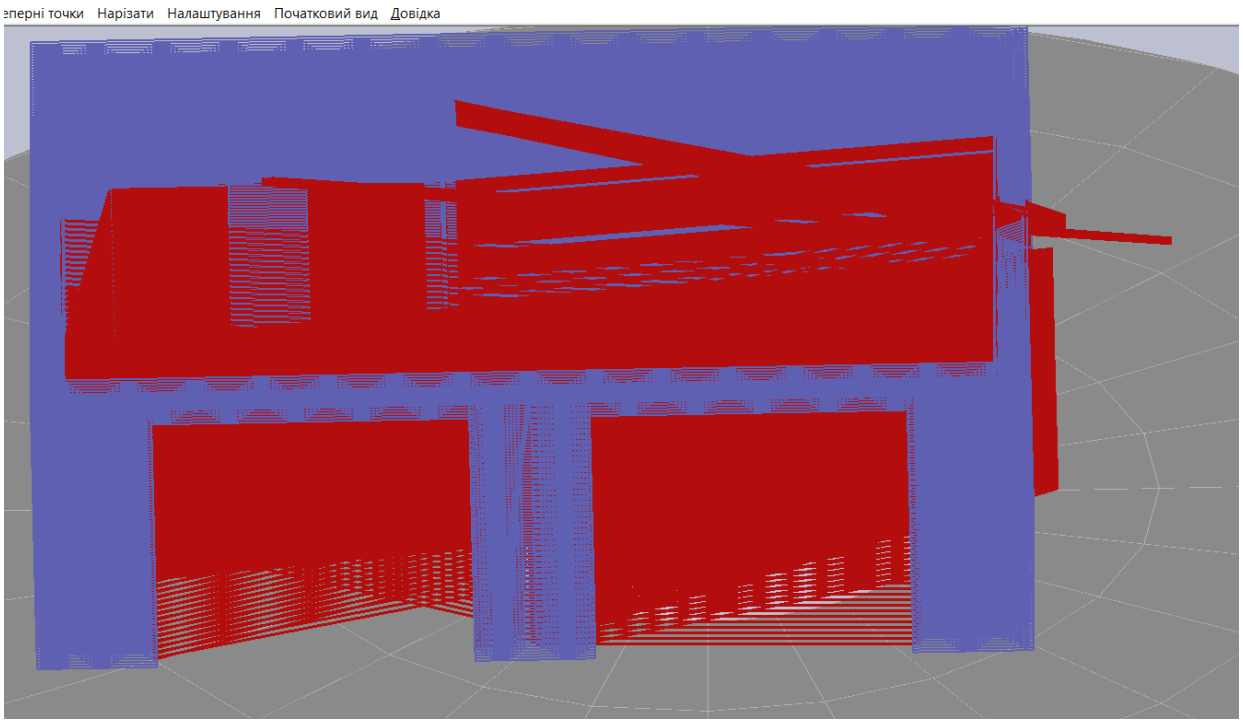

Рисунок 4.22 – Регулювання кількості відображуваних команд

Тепер дані можна зберегти у файл, скориставшись пунктом меню «Файл=>Зберегти як…». Після обрання файлу та завершення збереження даних в файл система відправить користувачу повідомлення про успішне збереження даних G-code (рис. 4.23).

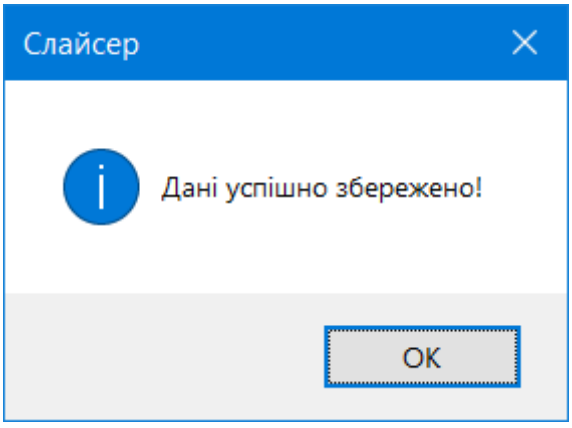

Рисунок 4.23 – Повідомлення про успішне збереження даних

На рис. 4.24 продемонстровано фрагмент результатів збереження G-code команд у файл.

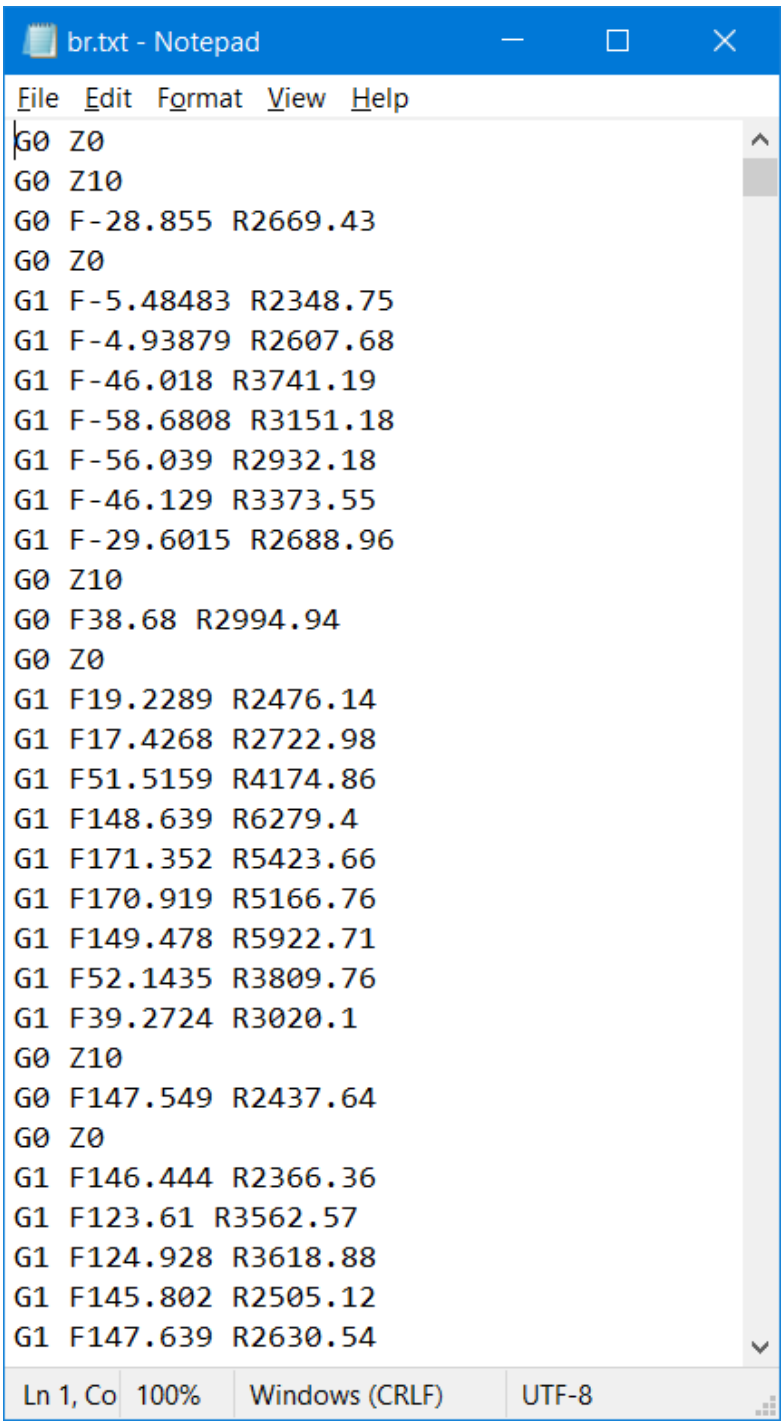

Рисунок 4.23 – Повідомлення про успішне збереження даних

Спочатку принтер виїжджає в потрібну висоту, починає холосте переміщення до точки початку друку. Потім починається друк полігон за полігоном. Для холостого переміщення принтер піднімається на 10 міліметрів над шаром, щоб не чіпляти соплом попередній шар. Фрагменти коду програмного продукту наведено в Додатку Б, а приклад G-code команд наведено в додатку В.

4.2 Оцінка застосування інформаційної технології

Для порівняння швидкодії роботи розробленого додатку з іншими слайсерами, які використовуються для друку пластиком, було розроблено дві моделі (модель будинку для розробленого слайсеру та зменшена модель будинку для інших слайсерів). У таблиці 4.1 наведено час виконання слайсингу для різних програмних продуктів, з зазначенням параметрів слайсингу.

Таблиця 4.1 – Порівняння швидкодії роботи додатку

Слайсер	Час	Параметри слайсингу
Slicer	53 с.	Висота шару: 2мм Заповнення: так Кількість шарів: 1500 Розмір моделі: 12400x11100x3000
Cura	123 с.	Висота шару: 0.032мм Заповнення: так Кількість шарів: 1500 Розмір моделі: 200x179x48
IdeaMaker	164 с.	
KISSlicer	200 с.	
Repetier-Host	211 с.	
OctoPrint	217 с.	
Simplify3D	140 с.	
AstroPrint	157 с.	
3DPrinterOS	181 с.	

Параметри слайсингу моделей були підібрані так, щоб забезпечити однакову кількість шарів для всіх програмних продуктів. Як видно з таблиці, розроблений програмний продукт виконав слайсинг найшвидше. На мою думку, що такий результат було досягнуто через оптимізацію алгоритму обрізання заповнення. Розроблений програмний продукт може використати заповнення з попереднього шару, а інші додатки кожного разу генерують нове заповнення.

Інформаційну технологію було випробувано в польових умовах. В селищі міського типу Недригайлів було надруковано кафетерій, креслення якого представлено на рис. 4.1. На рис. 4.24 зображено процес друку будівлі технологією адитивного виробництва. Набір G-code команд було згенеровано в розробленому додатку.

Рисунок 4.24 – Процес друку будівлі

ВИСНОВКИ

Адитивне виробництво знаходить застосування в різних сферах. Не виключенням є і будівництво. З метою спрощення початку будівництва та зменшення витрат на встановлення опорних конструкції для принтера було створено радіальний принтер для друку будівель.

Постає задача генерації G-code команд для такого типу принтера. Аналіз предметної області показав, що через особливості конструкції принтера та особливості технологічного процесу є неможливим використання слайсерів для друку пластиком.

В ході виконання роботи було проаналізовано алгоритми нарізання моделі та створення шляху руху сопла принтера. Аналіз алгоритмів слайсингу показав, що базовий алгоритм нарізання тривимірної моделі є досить ефективним та може бути використаний, як основа для створення інформаційної технології. Але стандартні патерни заповнення порожнин у стінах не можуть бути використані в адитивному будівництві, тому необхідно додати свої патерни заповнення до розроблюваної інформаційної технології.

На етапі проектування інформаційної технології слайсингу для радіального будівельного 3d принтера було розроблено моделі інформаційної технологій, створено діаграму варіантів використання, проведено моделювання роботи додатку та математичне моделювання процесів інформаційної технології.

Розроблена інформаційна технологія була реалізована в програмному додатку та показала хороші показники швидкодії. Розробленим додатком було створено набір команд для друку кафетерію в селищі міського типу Недригайлів.

Результати виконання кваліфікаційної роботи магістра доповідались на двох міжнародних наукових конференціях: «IMA 2021» та «Модернізація та наукові дослідження: парадигма інноваційного розвитку суспільства і технологій». Також робота посіла перше місце в конкурсі наукових робіт за напрямом «Інформаційні системи та технології».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. P. Holzmann, J. Robert, A. Aqeel Breitenecker, Soomro, & J. S. Erich, “User entrepreneur business models in 3D printing,” Journal of Manufacturing Technology Management, Vol. 28, No. 1, pp. 75-94, 2017.

2. N. Shahrubudina, T. C. Leea, R. Ramlana. An Overview on 3D Printing Technology: Technological, Materials, and Applications 2019. 11 с. (Elsevier).

3. Tay Y.W.D., Panda, B., Paul, S. C., та ін. 3D printing trends in building and construction industry: a review. Singapore: 2017.

4. B. Furet, P. Poullain, S. Garnier. Furet B. 3D printing for construction based on a complex wall of polymer-foam and concrete, 2019. 14 с. (Additive manufacturing).

5. D. Han, H. Yin, M. Qu та ін. Technical Analysis and Comparison of Formwork-Making Methods for Customized Prefabricated Buildings: 3D Printing and Conventional Methods. 2020. 13 с.

6. Y. Wei Daniel Tay, B. Panda, S. Chandra Paul та ін. 3D printing trends in building and construction industry: a review, 2017. 17 с. (Virtual and Physical Prototyping).

7. T. D. Ngo, A. Kashani, G. Imbalzano та ін. Additive manufacturing (3D printing): A review of materials, methods, applications and challenges, 2018. 80 с. (Composites).

8. A. Paolini, S. Kollmannsberger, E. Rank. Additive manufacturing in construction: A review on processes, applications, and digital planning methods. 2019. 45 с. (Additive Manufacturing).

9. M. Hoffmann, S. Skibicki, P. Pankratow та ін. Automation in the Construction of a 3D-Printed Concrete Wall with the Use of a Lintel Gripper. 2020. 15 с. (Materials).

10. S. Volpe, V. Sangiorgio, A. Petrella та ін. Building Envelope Prefabricated with 3D Printing Technology. 2021. 13 с.

11. J. McPherson, W. Zhou. A chunk-based slicer for cooperative 3D printing, 2018. 12 с. (Rapid Prototyping Journal).

12. R. Duballet, O. Baverel, J. Dirrenberger. Classification of building systems for concrete 3D printing, 2017. 12 с.

13. Z. Zuo, J. Gong, Y. Huang та ін. Experimental research on transition from scale 3D printing to full-size printing in construction. [2019. 11 с. (Construction and Building Materials).

14. G. Jie, B. Jie, J. Hongxue. Experimental study on layer bonding property of 3D printing building materials. 2019. 7 с.

15. C. P.Suvash, D. T. Yi Wei, P. Biranchi, J. T. Panda. Fresh and hardened properties of 3D printable cementitious materials for building and construction, 2018. 9 с. (Elsevier).

16. D. Filarska, S. Uncik, T. Cabanova. Specification of the properties and effects of additives and admixtures on a mixture suitable for 3d printing of buildings. 2020. 7 с.

17. S. Pessoa, A.S. Guimaraes. The 3D printing challenge in buildings. 2020. 7 с.

18. M. Szilvasi-Nagy, Gy. Matyasi. Analysis of STL Files. 2003. 16 с. (Elsevier).

19. Y. Zhang, X. Shi. Research on the Three-dimensional Displaying of STL ASCII and Binary File. 2014. 4 с. (Journal of Computational Information Systems).

20. Hu, J. Study on STL-based slicing process for 3D printing: Solid Freeform Fabrication 2017: Proceedings of the 28th Annual International Solid Freeform Fabrication Symposium - An Additive Manufacturing Conference, SFF 2017, 20.

21. B. Akhoundi, A.H. Behravesh. Effect of Filling Pattern on the Tensile and Flexural Mechanical Properties of FDM 3D Printed Products. 2018. 15 с.

22. R. Minetto, N. Volpato, J. Stolfi та ін. An optimal algorithm for 3D triangle mesh slicing. 2017. 32 с. (Computer-Aided Design).

23. Y. Wang, W. Li. A slicing algorithm to guarantee non-negative error of additive manufactured parts. 2018. 10 с. (The International Journal of Advanced Manufacturing Technology).

24. I. Enes Yigit, I. Lazoglu. Spherical slicing method and its application on robotic additive manufacturing. 2020. 8 с. (Progress in Additive Manufacturing).

25. F.Wasserfall, N. Hendrich, D. Ahlers, J. Zhang. Topology-aware routing of 3D-printed circuits. 2020. 12 с. (Elsevier).

26. H.Yin, M. Qu, H. Zhang, Y. Lim. 3D Printing and Buildings: A Technology Review and Future Outlook. 2019. 19 с.

27. G. Fitzmaurice, J. Matejka, I. Mordatch та ін. Safe 3D navigation. Proceedings of the Symposium on Interactive 3D Graphics and Games, 2008.

28. Glazunov N. M. Foundations of scientific research. 2012. 168 с.

29. G.Waissi, M. Demir, J. Humble, B. Lev. Automation of strategy using IDEF0 - A proof of concept, 2015. 8 с.

30. Hu, J. Study on STL-based slicing process for 3D printing: Solid Freeform Fabrication 2017: Proceedings of the 28th Annual International Solid Freeform Fabrication Symposium - An Additive Manufacturing Conference, SFF 2017, 20.

31. D. Berardi, D. Calvanese, G. De Giacomo. Reasoning on UML class diagrams, 2005. 49 с.

32. X.Dolques, M. Huchard, C. Nebut, P. Reitz. Fixing Generalization Defects in UML use Case Diagrams. 2010. 14 с.

33. M.Seidl, M. Scholz, C. Huemer, G. Kappel. UML @ Classroom. An Introduction to Object-Oriented Modeling, 2015. 215 с. (Springer).

34. Г. Ч. Курінний, О. М. Невмержицька, О. О. Шугайло. Пряма та площина у тривимірному дійсному просторі. Харків, 2013. 28 с.

35. K. Hormann, A. Agathos. The Point in Polygon Problem for Arbitrary Polygons. 2001. 15 с.

Додаток А

Планування робіт

Деталізація мети проекту методом SMART.

Мета проекту: адаптація існуючих алгоритмів слайсингу для створення інструменту генерації команд керування зведення будівель за допою радіального 3D принтера.

Деталізація мети проекту методом SMART наведена в таблиці А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Адаптувати існуючі алгоритми слайсингу для створення інструменту генерації команд керування зведення будівель за допою радіального 3D принтера.
Measurable (вимірювана)	Головним показником успішності виконання даного проекту швидкодія алгоритму та мінімальний розбір накопичуваної похибки під час проведення розрахунків.
Achievable (досяжна)	Ціль є досяжною оскільки технології, за допомогою яких буде розроблятись додаток є у відкритому доступі, а мета була повністю погоджена із замовником.
Relevant (реалістична)	Розробка даного додатку є цілком реалістичною, оскільки розробники мають усі необхідні програмні та технічні засоби, а також достатній рівень кваліфікації.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Всі строки виконання даного додатку вказано в календарному плані.

Планування змісту структури робіт. Для планування змісту структури робіт в графічному вигляді використовують WBS діаграми. На верхньому першому рівні WBS фіксується продукт проекту. Наступний II рівень відповідає діям або основним заходам для досягнення продукту проекту. Потім триває розбивка цих дій доти, поки не відбувається виконання дій елементарних робіт.

Побудуємо структуру WBS, у якій детально опишемо роботи, які потрібно виконати на кожному етапі створення проекту. Декомпозована діаграма WBS зображена на рис. А.1.

Планування структури організації, для впровадження готового проекту. Після побудови WBS розробимо організаційну структуру виконавців. Визначається

за переліком пакетів робіт нижнього рівня кожної гілки WBS-структури. Відповідальні – це ті співробітники, які безпосередньо організовують і відповідають у виконавця за виконання елементарної роботи, зазначеної у WBS. Список виконавців, що функціонують в проекті знаходиться в таблиці А.2.

Таблиця А.2 − Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Палажченко Є.В.	Виконує розробку основного функціоналу проекту та інтерфейс користувача
Проектувальник	Палажченко Є.В.	Проектує процеси інформаційної технології та взаємозв’язки між компонентами всередині додатку.
Консультант проекту	Ващенко С.М.	Формує завдання на розробку проекту.
Менеджер проекту	Ващенко С.М.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками.

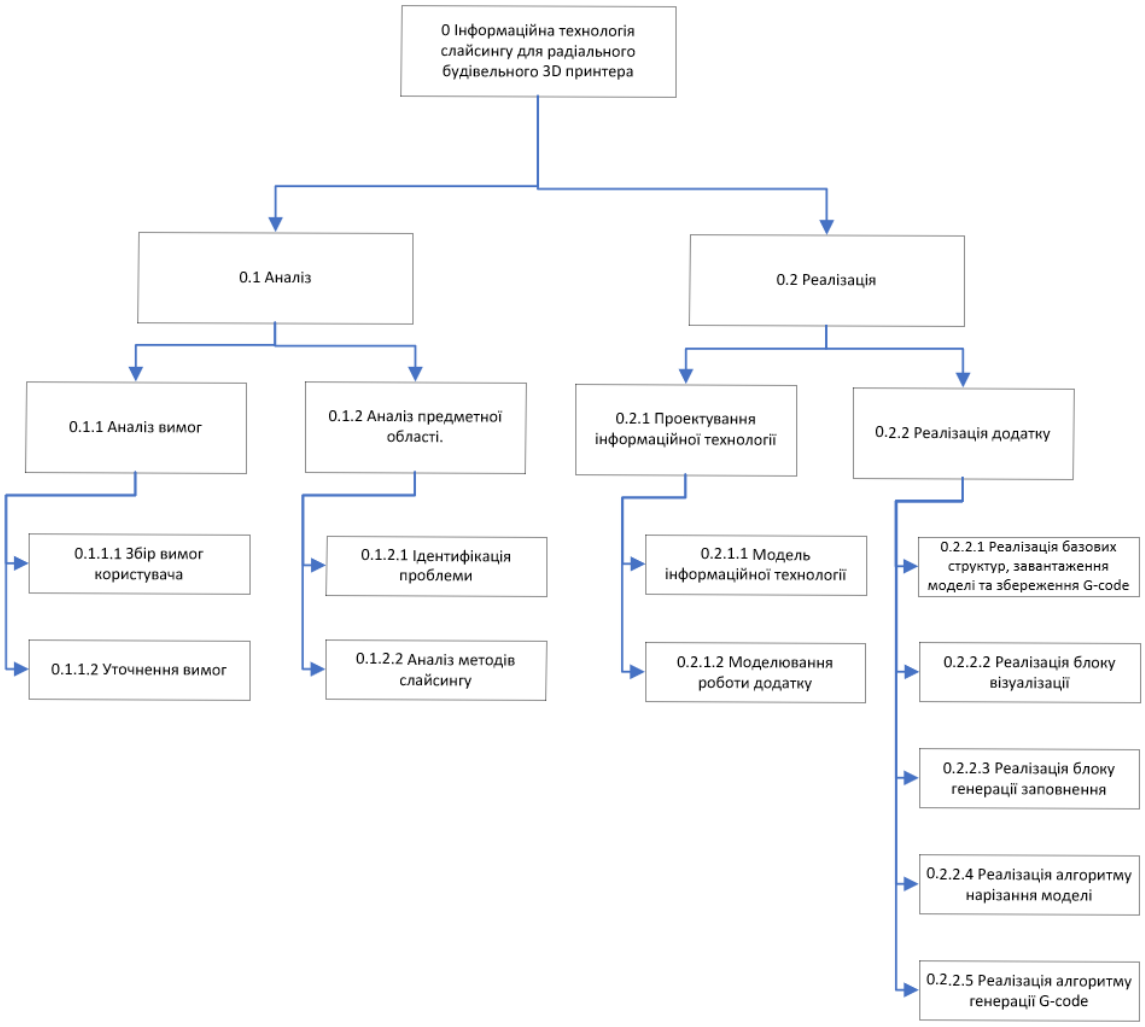

Рисунок А.1 − WBS. Структура робіт проекту

Діаграма Ганта. Для того щоб мати реальне уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, на підставі часткової мережевої моделі, а також, проекту в цілому з урахуванням вихідних та святкових днів, будують календарний графік робіт (рисунки А.2 – рисунок А.3). Він є реальним розподілом робіт з пакету за календарними датами, тобто своєрідним розкладом виконання робіт. Графік Ганта є достатньо зручним для користування. Його будують так. На горизонталі фіксують календар у тих одиницях часу, які обрані для проекту (години, дні). Ліворуч на вертикалі розташовують найменування всіх робіт. На полі, що утворилось, поставляють у вигляді прямокутників роботи, довжина яких по горизонталі відповідає їхній тривалості. Між роботами лініями вказують логічні зв’язки.

ДИАГРАММА ГАНТА

	Режим задачи	Task Name	Длит	Начало	Окончание	Названия ресурсов
1		**Аналіз**	**8 дней**	**Пт 01.10.21**	**Вт 12.10.21**	
2		Збір вимог користувача	2 дней	Пт 01.10.21	Пн 04.10.21	Палажченко Є.В.
3		Уточнення вимог	2 дней	Вт 05.10.21	Ср 06.10.21	Палажченко Є.В.
4		Ідентифікація проблеми	2 дней	Чт 07.10.21	Пт 08.10.21	Палажченко Є.В.
5		Аналіз методів слайсингу	2 дней	Пн 11.10.21	Вт 12.10.21	Палажченко Є.В.
6		**Реалізація додатку**	**35 дней**	**Ср 13.10.21**	**Вт 30.11.21**	
7		Моделювання інформаційної технології	2 дней	Ср 13.10.21	Чт 14.10.21	Палажченко Є.В.
8		Моделювання роботи додат	2 дней	Пт 15.10.21	Пн 18.10.21	Палажченко Є.В.
9		Реалізація базових структур, завантаження моделі та збереження	7 дней	Вт 19.10.21	Ср 27.10.21	Палажченко Є.В.
10		0.2.2.2 Реалізація блоку візуалізації	3 дней	Чт 28.10.21	Пн 01.11.21	Палажченко Є.В.
11		Реалізація блоку генерації заповнення	6 дней	Вт 02.11.21	Вт 09.11.21	Палажченко Є.В.
12		Реалізація алгоритму нарізання моделі	10 дней	Ср 10.11.21	Вт 23.11.21	Палажченко Є.В.
13		Реалізація алгоритму генерації G-code	5 дней	Ср 24.11.21	Вт 30.11.21	Палажченко Є.В.

Рисунок А.2 − Діаграма Ганта

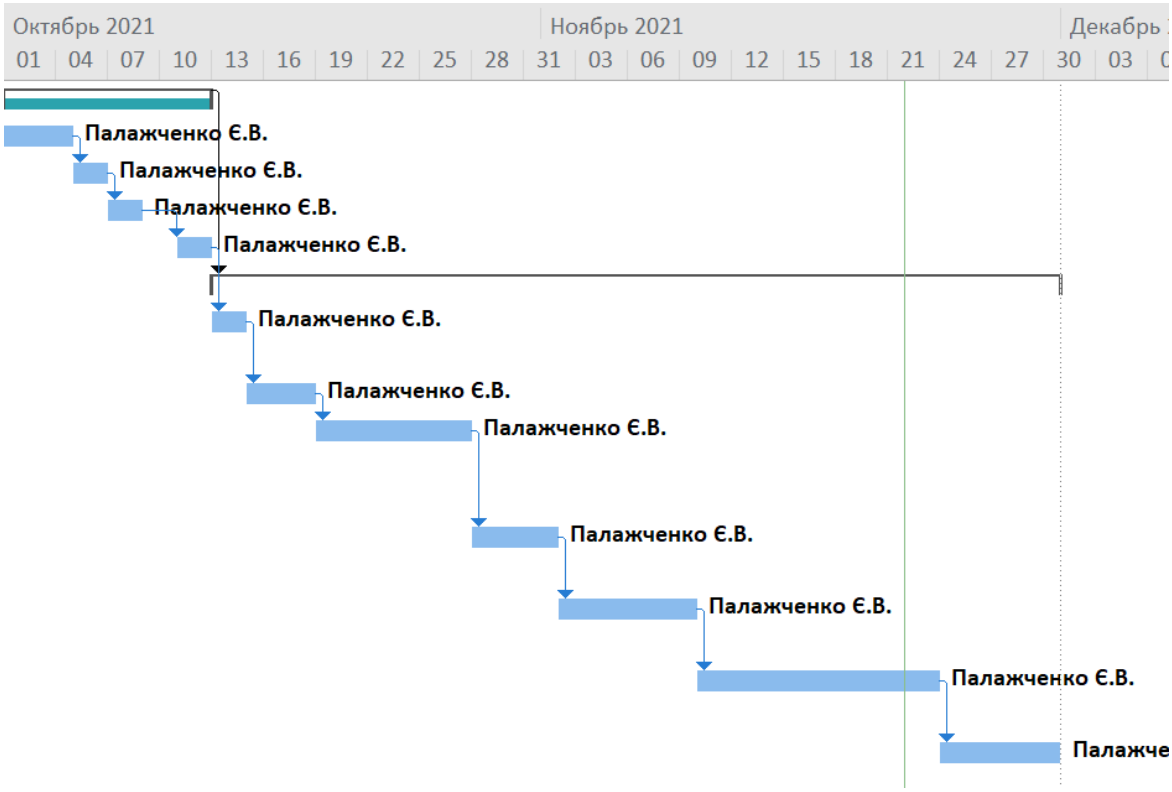

Рисунок А.3 – Діаграма Ганта

Аналіз ризиків. Виконаємо якісну і кількісну оцінку ризиків роботи. При якісній оцінці визначимо ризики, що потребують швидкого реагування. Така оцінка визначить ступінь важливості ризику і дозволить вибрати спосіб реагування. Кількісна оцінка ризиків буде виконана для більш повної ідентифікації ризиків та ступеня їхнього впливу на виконання проекту. Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків. У таблиці А.5 знаходиться класифікація ризиків за показниками ймовірності виникнення ризику та величині втрат.

Далі виконаємо планування реагування на ризики — це розробка методів і технологій зниження негативного впливу ризиків на проект. Визначимо ефективність розробки реагування на проект, визначимо чи будуть наслідки впливу ризику на проект позитивними або негативним. Оцінюємо ризики за показниками, що знаходяться в таблиці А.3. На основі оцінки будуємо матрицю ймовірності виникнення ризиків та впливу ризику, що наведено на рис. А.4.

Таблиця А.3 − Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	Ймовірність виникнення	Вплив ризику
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

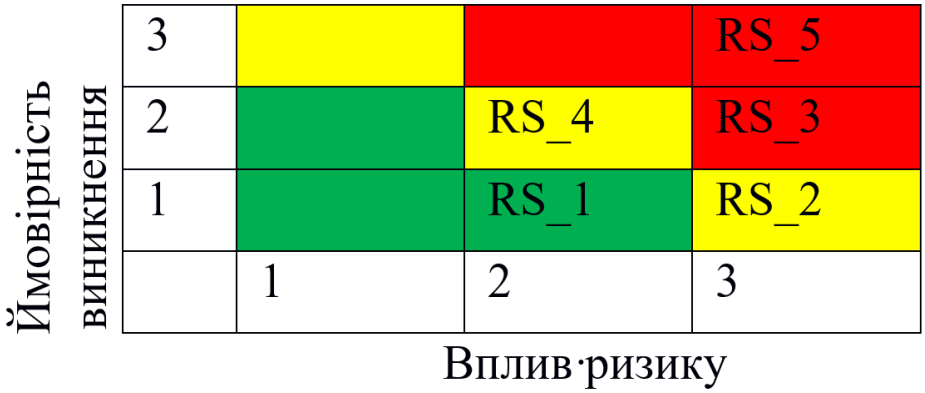

Рисунок А.4 – Матриця ймовірності виникнення ризиків та впливу ризику

– зелений колір – прийнятні ризики;
– жовтий колір – виправданні ризики;
– червоний колір – недопустимі ризики.

На підставі отриманого значення індексу ризики класифікують: за рівнем ризику, що знаходиться в таблиці А.4.

Таблиця А.4 − Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	1≤R≤2	1
2	Виправдані	3≤R≤4	2,4
3	Недопустимі	6≤R≤9	3,5

Таблиця А.5 − Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Непорозуміння між розробником та замовником	Низька	Середній	2	Проведення зустрічей для підвищення взаєморозуміння	Попередження	З'ясувати причини непорозуміння та обговорити варіанти вирішення.
RS_2	Відкритий	Вихід на ринок продукту конкурентів до виходу нашого	Низька	Високий	3	Провести попереднє дослідження альтернативних продуктів.	Прийняття	Обрати унікальну стратегію розробки
RS_3	Відкритий	Погано прописане завдання на розробку	Середня	Високий	7	1. Ясно і однозначно обговорити із замовником усі види вимог.	Попередження	Скласти глосарій для запобігання розбіжностей у розумінні слів та термінів.
RS_4	Відкритий	Низька кваліфікація розробників проекту	Середня	Середній	4	Підвищити кваліфікацію персоналу. Переглянути онлайн-ресурси для підвищення рівня знань.	Пом'якшення	Врахувати час на підготовку працівників. Видати літературу, переглянути онлайн-уроки.
RS_5	Відкритий	Неоптимальний розподіл часу	Висока	Високий	9	Провести аналіз актуальності найважливіших процесів та робіт та чітко дотримуватися календарного плану.	Пом'якшення	Знайти способи оптимізації роботи із вже існуючою розстановкою. Обговорити варіанти внесення правок замовником.

Додаток Б

Базові структури

```
inline bool is_equal(double x, double y, double eps=EPS) {
return std::fabs(x - y) < eps;
}

struct Vec {
double x, y, z;

Vec(double x = 0, double y = 0, double z = 0): x(x), y(y), z(z) {};
Vec(const Vec& a) { x = a.x; y = a.y; z = a.z; };
Vec(const Vec& point1, const Vec& point2) {
     x = point2.x - point1.x;
     y = point2.y - point1.y;
     z = point2.z - point1.z;
}

inline Vec operator+(Vec a) {
     a.x += x;
     a.y += y;
     a.z += z;
     return a;
}

inline Vec operator-(const Vec a) const {
     Vec b(a.x - x, a.y - y, a.z - z);
     return b;
}

inline void operator-=(Vec a) {
     x-=a.x;
     y -= a.y;
     z -= a.z;
}
inline Vec operator*(double b) {
     Vec a(x*b,y*b,z*b);
     return a;
}

inline Vec operator-(double b) {
     Vec a(x - b, y - b, z - b);
     return a;
}
inline Vec operator/(double b) {
```

```
        Vec a(x / b, y / b, z / b);
        return a;
    }
    inline bool operator==(Vec b) const {
        return
            is_equal(x, b.x, std::pow(10.0, -4)) &&
            is_equal(y, b.y, std::pow(10.0, -4)) &&
            is_equal(z, b.z, std::pow(10.0, -4));
    }

    inline bool operator!=(Vec b) const {
        return !(b==*this);
    }

    inline void operator/=(double b) {
        x /= b;
        y /= b;
        z /= b;
    }
    };

    struct Triangle
    {
    Vec point1;
    Vec point2;
    Vec point3;
    Vec normal;
    Triangle(Vec p1, Vec p2, Vec p3):point1(p1), point2(p2), point3(p3)
{}
    Triangle() {}
    };

    struct GCodePoint {
    GCodePoint(Vec polarPoint, double e=0):polarPoint(polarPoint),
e(e){};
    GCodePoint() : e(0) { polarPoint = Vec(); };
    Vec polarPoint;
    double e;
    };

    using PointVector = std::vector<Vec>;

    using GCode = std::vector<GCodePoint>;

    using TriangleSalat = std::vector<Triangle>;

    struct Line
    {
```

```
     Line(const Vec& point1, const Vec& point2, const Vec&
normal):point1(point1), point2(point2),normal(normal) {};
     Line(const Vec& point1, const Vec& point2) :point1(point1),
point2(point2) {
          normal = Vec();
     };
     Line() { point1 = Vec(); point2 = Vec(); normal = Vec(); };
     inline bool operator==(Line b) {
          return point1==b.point1 && point2==b.point2 && normal==
b.normal;
     }
     Vec point1;
     Vec point2;
     Vec normal;
     };

     using LineSalat = std::vector<Line>;
```

STLParser.h

```
     TriangleSalat STLParser::read(const std::string& filename) {
     TriangleSalat Salat;

     std::ifstream fin(filename);
     bool result = fin.is_open();
     while (!fin.eof())
     {
          std::string temp;
          fin >> temp;
          if (temp == "facet")
               addTriangle(fin,Salat);
     }
     fin.close();

     return Salat;
     }

     void STLParser::addTriangle(std::ifstream& fin, TriangleSalat&
Salat) {
     Triangle triangle;
     readVec(fin, triangle.normal);
     std::string temp;
     fin >> temp>>temp;
     readVec(fin, triangle.point1);
     readVec(fin, triangle.point2);
     readVec(fin, triangle.point3);
     Salat.push_back(triangle);
     }
```

```
void STLParser::readVec(std::ifstream& fin, Vec& vec) {
std::string temp;
fin >> temp;
fin >> vec.x;
fin >> vec.y;
fin >> vec.z;
}
```

GLRenderSystem.cpp

```
#include "pch.h"
#include "GLRenderSystem.h"
GLRenderSystem::GLRenderSystem(Vec& vector):vector(vector) {
PrintSetSaver::uploadPrintSet(printSet);
ViewSetSaver::uploadViewSet(viewSet);
}

GLRenderSystem::~GLRenderSystem() {

}

void GLRenderSystem::updateViewSet() {
ViewSetSaver::uploadViewSet(viewSet);
}

void GLRenderSystem::updatePrintSet() {
PrintSetSaver::uploadPrintSet(printSet);
}

void GLRenderSystem::setLight() {
glEnable(GL_DEPTH_TEST);
//Set up for 3D
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

//Light0 params
GLfloat ambientColour[] = { 0.2f, 0.2f, 0.2f, 1.0f };
GLfloat lightColour0[] = { 0.5f, 0.5f, 0.5f, 1.0f };
GLfloat lightPos0[] = { -50.0f, 50.0f, -100.0f, 0.0f };
GLfloat lightSpecular0[] = { 1.0, 1.0, 1.0, 1.0 };

//Bind the params to the light
glLightModelfv(GL_LIGHT_MODEL_LOCAL_VIEWER, ambientColour);
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColour0);
glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);
glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular0);
//Enable light
glEnable(GL_LIGHTING);
```

```
glEnable(GL_LIGHT0);
glLoadIdentity();

GLfloat mat_ambient[] = { 0.2f, 0.2f, 0.2f, 1.0f };
GLfloat mat_diffuse[] = { 0.2f, 0.2f, 0.2f, 1.0f };
GLfloat mat_Specular[] = { 0.8f, 0.8f, 0.8f, 1.0f };
GLfloat mat_emmissive[] = { 0.05f, 0.05f, 0.05f, 1.0f };
float shininess;
shininess = 80.0f;

glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, &shininess);
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_Specular);
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, mat_emmissive);

glEnable(GL_COLOR_MATERIAL);
glEnable(GL_NORMALIZE);

glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
}

void GLRenderSystem::clearScreen() {
glClearColor(0.745, 0.7568, 0.8235, 1.0);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
}

void GLRenderSystem::drawArrows() {
glColor3f(0, 0, 1);
glBegin(GL_LINE_STRIP);
glVertex3f(0, 0, 0);
glVertex3f(.05, 0, 0);
glEnd();
glColor3f(0, 1, 0);
glBegin(GL_LINE_STRIP);
glVertex3f(0, 0, 0);
glVertex3f(0, .05, 0);
glEnd();
glColor3f(1, 0, 0);
glBegin(GL_LINE_STRIP);
glVertex3f(0, 0, 0);
glVertex3f(0, 0, .05);
glEnd();
}

inline GLfloat GLRenderSystem::calcX(GLfloat r, GLfloat f){
return  r * cos(f);
}
```

```
inline GLfloat GLRenderSystem::calcY(GLfloat r, GLfloat f){
return r * sin(f);
}

void GLRenderSystem::drawPrintArea() {
glLineWidth(1);
drawArrows();

COLORREF color(viewSet.getPoleColor());
glColor3f(
     GetRValue(color) / (GLfloat)255,
     GetGValue(color) / (GLfloat)255,
     GetBValue(color) / (GLfloat)255);

GLdouble rMax = printSet.getRMax();
GLdouble rMin = printSet.getRMin();
rMin = rMin / rMax / 2;
rMax = rMax / rMax / 2;

glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
GLUquadric* quad = gluNewQuadric();
gluDisk(quad,rMin ,rMax,50,10);

int segment = 24;
GLfloat dF/*0-360*/ = ((double)2) / (double)segment * PI;
GLint stepCount = viewSet.getStep()+1;

glColor3f(1, 1, 1);
for (int i = 0; i < segment; i++){
     glBegin(GL_LINE_STRIP);
     glVertex3f(calcX(rMin,dF*i), calcY(rMin,dF*i), 0.0001);
     glVertex3f(calcX(rMax,dF*i), calcY(rMax,dF*i), 0.0001);
     glEnd();
     if (stepCount != 1)
     for (GLfloat j = rMin; j < rMax; j+=(rMax-rMin)/ stepCount) {
          glBegin(GL_LINE_STRIP);
          glVertex3f(calcX(j,dF*i), calcY(j,dF*i), 0.0001);
          glVertex3f(calcX(j,dF*(i+1)), calcY(j,dF*(i+1)), 0.0001);
          glEnd();
     }
}
}

     void GLRenderSystem::drawSlider(double progres, uint32_t height,
uint32_t width) {

     double w = width * 0.9 / width;
     double h = height / 2.f / height * 0.9;
```

```
     Vec p1(w, h);
     Vec p2(w,-h);
     drawLine(p1, p2, RGB(255, 255, 255));

     Triangle tr1(Vec(0.01,0.01,0), Vec(0.01, -0.01, 0), Vec(-0.01, -
0.01, 0));
     Triangle tr2(Vec(0.01,0.01,0), Vec(-0.01, -0.01, 0), Vec(-0.01,
0.01, 0));

     double p = h * progres/100.f;

     glBegin(GL_TRIANGLES);
     glNormal3f(0, 0, 1),
          glVertex4f(tr1.point1.x + w, tr1.point1.y + p, tr1.point1.z,
1);
          glVertex4f(tr1.point2.x + w, tr1.point2.y + p, tr1.point2.z,
1);
          glVertex4f(tr1.point3.x + w, tr1.point3.y + p, tr1.point3.z,
1);
     glEnd();

     glBegin(GL_TRIANGLES);
     glNormal3f(0, 0, 1),
          glVertex4f(tr2.point1.x + w, tr2.point1.y + p, tr2.point1.z,
1);
          glVertex4f(tr2.point2.x + w, tr2.point2.y + p, tr2.point2.z,
1);
          glVertex4f(tr2.point3.x + w, tr2.point3.y + p, tr2.point3.z,
1);
     glEnd();

     }

     void GLRenderSystem::drawObject(TriangleSalat& Salat, double dX,
double dY, bool selected, bool intersect) {
     glLineWidth(1);
     GLfloat zoom = printSet.getRMax()*2;

     COLORREF color(viewSet.getObjColor());
     if (selected)
          glColor4f(1, 1, 0, 1);
     else

     glColor4f(
          GetRValue(color) / (GLfloat)255,
          GetGValue(color) / (GLfloat)255,
          GetBValue(color) / (GLfloat)255,1);
```

```
     if (intersect)
          glColor4f(1, 0, 0, 1);

     glPushMatrix();
     //glRotated(beta, 0, 0, 1);
     for (const Triangle& trian : Salat){
          glBegin(GL_TRIANGLES);
               glNormal3f(trian.normal.x, trian.normal.y,
trian.normal.z),
               glVertex4f((trian.point1.x+vector.x+dX) / zoom,
(trian.point1.y+vector.y+dY)/ zoom, (trian.point1.z+vector.z) / zoom,1);
               glVertex4f((trian.point2.x+vector.x+dX) / zoom,
(trian.point2.y+vector.y+dY)/ zoom, (trian.point2.z+vector.z) / zoom,1);
               glVertex4f((trian.point3.x+vector.x+dX) / zoom,
(trian.point3.y+vector.y+dY)/ zoom, (trian.point3.z+vector.z) / zoom,1);
          glEnd();
     }
     glPopMatrix();
     }

     void GLRenderSystem::drawGCode(const GCode& pCode, double progres) {
     GLfloat zoom = printSet.getRMax() * 2;
     glLineWidth(5);

     progres = (progres + 100) / 200;
     COLORREF color(viewSet.getObjColor());

     double oldZ = 0;
     double fAbs = 0;

     for (size_t i = 0; i< (pCode.size()* progres)-1; i++) {
          Vec p1 = Converter::toDec(pCode[i], fAbs);
          fAbs += pCode[i].polarPoint.y;
          Vec p2 = Converter::toDec(pCode[i + 1], fAbs);
          if (is_equal(pCode[i+1].e, .0)) {
               //drawLine(p1 / zoom, p2 / zoom, RGB(255, 0, 0));
          } else {
               drawLine(p1 / zoom, p2 / zoom, color);
          }
     }
     }

     void GLRenderSystem::drawLine(Vec p1, Vec p2, COLORREF color) {

     glColor4f(
          GetRValue(color) / (GLfloat)255,
          GetGValue(color) / (GLfloat)255,
          GetBValue(color) / (GLfloat)255, 1);
```

```
     glBegin(GL_LINES);
     glVertex3f(p1.x, p1.y, p1.z);
     glVertex3f(p2.x, p2.y, p2.z);
     glEnd();
     }


     void GLRenderSystem::drawLines(const LineSalat& salat,
std::vector<size_t> indices){

     GLfloat zoom = printSet.getRMax() * 2;
     glLineWidth(1.5);

     COLORREF color(viewSet.getObjColor());

     GLfloat r = GetRValue(color) / (GLfloat)255,
          g = GetGValue(color) / (GLfloat)255,
          b = GetBValue(color) / (GLfloat)255;

     for (size_t i = 0; i<salat.size();i++) {

          const Line& l = salat[i];
          if (std::find(indices.begin(), indices.end(), i)==
indices.end()) {
               glColor4f(r,g,b, 1);
               glLineWidth(1.5);
          }else {
               glColor4f(1-r,1-g,1-b, 1);
               glLineWidth(2);
          }
          glBegin(GL_LINES);
          glVertex3f(l.point1.x / zoom, l.point1.y / zoom, l.point1.z /
zoom + 0.01);
          glVertex3f(l.point2.x / zoom, l.point2.y / zoom, l.point2.z /
zoom + 0.01);
          glEnd();
     }
     }

     void GLRenderSystem::drawPoint(Vec p, double z, uint32_t width,
uint32_t height) {

     double s = (height > width) ? height : width;

     p = p / s;

     p.x -= width / s / 2.f;
     p.y -= height / s / 2.f;
```

```
p.y = -p.y;
p.z = 0.01;
p = p*2.f*z;

glPointSize(5.0f);
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_POINTS);
glVertex3f(p.x,p.y, p.z);
glEnd();
}

void GLRenderSystem::drawPoint(Vec p) {

GLfloat zoom = printSet.getRMax() * 2;

glPointSize(5.0f);
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_POINTS);
glVertex3f(p.x / zoom, p.y / zoom, p.z / zoom + 0.01);
glEnd();
}
```

Model.cpp

```
#include "pch.h"
#include "Model.h"

Model::Model()
:minV(9999999, 9999999, 9999999),maxV(-9999999, -9999999, -9999999),
centerV(0,0,0){
}

double Model::getZ() {

return maxV.z;
}

Model::~Model(){

}

void Model::setToNull() {
maxV = Vec(-9999999, -9999999, -9999999);
minV = Vec(9999999, 9999999, 9999999);
centerV = Vec(0,0,0);
}

void Model::up(TriangleSalat& salat) {
for (Triangle& trian : salat) {
     upTurn(trian.point1);
     upTurn(trian.point2);
```

```
     upTurn(trian.point3);
     upTurn(trian.normal);
}
calcModelParam(salat);
}

void Model::down(TriangleSalat& salat) {
for (Triangle& trian : salat) {
     downTurn(trian.point1);
     downTurn(trian.point2);
     downTurn(trian.point3);
     downTurn(trian.normal);
}
calcModelParam(salat);
}

void Model::left(TriangleSalat& salat) {
for (Triangle& trian : salat) {
     leftTurn(trian.point1);
     leftTurn(trian.point2);
     leftTurn(trian.point3);
     leftTurn(trian.normal);
}
calcModelParam(salat);
}

void Model::right(TriangleSalat& salat) {
for (Triangle& trian : salat) {
     rightTurn(trian.point1);
     rightTurn(trian.point2);
     rightTurn(trian.point3);
     rightTurn(trian.normal);
}
calcModelParam(salat);
}

void Model::downTurn(Vec& point) {
double t = point.y;
point.y = -point.z;
point.z = t;
}

void Model::leftTurn(Vec& point) {
double t = point.x;
point.x = -point.z;
point.z = t;
}

void Model::rightTurn(Vec& point) {
```

```
double t = point.x;
point.x = point.z;
point.z = -t;
}

void Model::upTurn(Vec& point) {
double t = point.y;
point.y = point.z;
point.z = -t;
}

void Model::calcModelParam(TriangleSalat& salat) {
setToNull();
for (const Triangle& trian : salat) {
     findParam(trian);
}

centerV.x = (minV.x+maxV.x)/2;
centerV.y = (minV.y + maxV.y) / 2;;
centerV.z = (minV.z + maxV.z) / 2;;

for (Triangle& trian : salat) {
     trian.point1 -= centerV;
     trian.point2 -= centerV;
     trian.point3 -= centerV;
}

minV -= centerV;
maxV -= centerV;
}

void Model::findParam(const Triangle& trian) {

findMin(trian.point1);
findMin(trian.point2);
findMin(trian.point3);

findMax(trian.point1);
findMax(trian.point2);
findMax(trian.point3);
}

void Model::findMin(const Vec& point) {
if (point.x < minV.x) { minV.x = point.x; }
if (point.y < minV.y) { minV.y = point.y; }
if (point.z < minV.z) { minV.z = point.z; }
}

void Model::findMax(const Vec& point) {
```

```
    if (point.x > maxV.x) { maxV.x = point.x; }
    if (point.y > maxV.y) { maxV.y = point.y; }
    if (point.z > maxV.z) { maxV.z = point.z; }
    }

    void Model::findCenter(const Vec& point) {
    centerV.x += point.x;
    centerV.y += point.y;
    centerV.z += point.z;
    }
```

GCodeParser.cpp

```
    #include "pch.h"
    #include "GCodeParser.h"
    #include<iomanip>


    void GCodeParser::write(const GCode& gCode, const std::string&
filename) {

    std::ofstream fout;
    fout.open(getPath(filename));
    double oldZ = 0;

    for (const GCodePoint& point : gCode){
        if (!is_equal(oldZ,point.polarPoint.z)){
            oldZ = point.polarPoint.z;
            fout << std::fixed<< "G0 Z" << std::setfill('0') <<
std::setw(3)<<point.polarPoint.z << std::endl;
        }

        if (is_equal(point.e,0)) {
            fout << "G0 R" << std::setfill('0') << std::setw(9)<<
std::setprecision(5)<<
                point.polarPoint.x << " F" << std::setfill('0') <<
std::setw(9) << std::setprecision(5) <<
                point.polarPoint.y << std::endl;
        }
        else{
            fout << "G1 R" << std::setfill('0') << std::setw(9) <<
                point.polarPoint.x << " F" << std::setfill('0') <<
std::setw(9) << std::setprecision(5)<<
                point.polarPoint.y << " E" << std::setfill('0') <<
std::setw(9) << std::setprecision(5)<<  point.e << std::endl;
        }
    }
    fout.close();
    }
```

```
	const std::string GCodeParser::getPath(const std::string& filename)
{
	auto pos = filename.find_last_of('.txt');

	if (pos==std::string::npos) {
		return filename+".txt";
	}
	else {
		return filename;
	}
	}
```

Додаток В

```
G0 Z0
G0 Z10
G0 F-28.855 R2669.43
G0 Z0
G1 F-5.48483 R2348.75
G1 F-4.93879 R2607.68
G1 F-46.018 R3741.19
G1 F-58.6808 R3151.18
G1 F-56.039 R2932.18
G1 F-46.129 R3373.55
G1 F-29.6015 R2688.96
G0 Z10
G0 F38.68 R2994.94
G0 Z0
G1 F19.2289 R2476.14
G1 F17.4268 R2722.98
G1 F51.5159 R4174.86
G1 F148.639 R6279.4
G1 F171.352 R5423.66
G1 F170.919 R5166.76
G1 F149.478 R5922.71
G1 F52.1435 R3809.76
G1 F39.2724 R3020.1
G0 Z10
G0 F147.549 R2437.64
G0 Z0
G1 F146.444 R2366.36
G1 F123.61 R3562.57
G1 F124.928 R3618.88
G1 F145.802 R2505.12
G1 F147.639 R2630.54
G1 F149.516 R2578.4
G1 F148.046 R2471.49
G0 Z10
G0 F157.372 R3425.69
G0 Z0
G1 F157.527 R3421.86
G1 F160.945 R4006.52
G1 F159.605 R4040.28
G1 F157.576 R3691.1
G1 F138.99 R4521.6
G1 F138.145 R4446.62
G1 F156.969 R3598.86
G1 F155.997 R3461.32
G1 F156.758 R3441.28
G0 Z10
G0 F164.42 R4907.31
```

```
G0 Z0
G1 F164.533 R4904.63
G1 F165.509 R5227.29
G1 F164.453 R5253.21
G1 F163.413 R4932.24
G1 F163.971 R4918.2
G0 Z10
G0 F194.171 R5262.13
G0 Z0
G1 F182.52 R5106.94
G1 F182.397 R5366.7
G1 F206.659 R5999.83
G1 F211.447 R5159.89
G1 F208.92 R5029.14
G1 F205.486 R5651.99
G1 F194.592 R5272.06
G0 Z10
G0 F0 R2797.24
G0 Z10
G0 F-120.315 R2817.22
G0 Z0
G1 F-127.439 R3062.95
G1 F-124.671 R3273.21
G1 F-108.524 R2839.1
G1 F-110.349 R2593.88
G1 F-119.608 R2797.24
G0 Z10
G0 F-57.0316 R3065.18
G0 Z0
G1 F-54.7353 R3015.33
G1 F-52.7229 R3094.07
G1 F-52.9851 R3333.84
G1 F-51.2024 R3415.6
G1 F-47.2835 R3350.94
G1 F-46.4588 R3396.43
G1 F-46.3467 R3679.18
G1 F-45.7238 R3678.46
G1 F-45.7841 R3395.64
G1 F-44.9718 R3347.21
G1 F-40.9651 R3400.84
G1 F-39.1878 R3313.21
G1 F-39.5775 R3072.28
G1 F-37.5654 R2987.42
G1 F-33.2748 R3071.59
G1 F-31.1007 R2999.09
G1 F-30.8261 R2757.57
G1 F-28.3366 R2690.38
G1 F-23.9544 R2810.03
G1 F-21.3696 R2757.59
G1 F-20.1462 R2522.32
G1 F-17.1917 R2478.75
G1 F-13.0781 R2636.38
G1 F-10.1658 R2608.96
```

```
G1 F-7.80024 R2390.12
G1 F-6.13427 R2381.64
G0 Z10
G0 F170.645 R5201.18
G0 Z0
G1 F170.029 R5210.71
G1 F168.976 R5432.24
G1 F167.57 R5459.99
G1 F165.662 R5296.99
G1 F164.241 R5332.43
G1 F163.447 R5562.52
G1 F162.109 R5602.94
G1 F160.1 R5457.9
G1 F158.764 R5505.86
G1 F158.218 R5741.98
G1 F156.964 R5794.02
G1 F154.902 R5667.05
G1 F153.665 R5726.28
G1 F153.343 R5966.17
G1 F152.183 R6028.65
G1 F150.111 R5919.33
G1 F149.541 R5953.63
G1 F150.489 R6126.87
G1 F148.731 R6238.18
G1 F147.649 R6051.21
G1 F149.277 R5946.59
G1 F148.947 R5889.5
G1 F146.609 R5883.56
G1 F145.882 R5772.93
G1 F146.821 R5551.33
G1 F146.054 R5440.44
G1 F143.526 R5447.04
G1 F142.678 R5340.62
G1 F143.556 R5114.14
G1 F142.651 R5007.72
G1 F139.919 R5028.99
G1 F138.923 R4927.92
G1 F139.696 R4696.62
G1 F138.623 R4595.95
G1 F135.678 R4634.42
G1 F134.505 R4540.18
G1 F135.108 R4304.5
G1 F133.83 R4211.28
G1 F130.682 R4269.85
G1 F129.3 R4184.33
G1 F129.644 R3945.35
G1 F128.124 R3861.81
G1 F124.807 R3943.6
G1 F123.189 R3869.19
G1 F123.159 R3628.97
G1 F121.365 R3557.93
G1 F117.96 R3665.92
G1 F116.092 R3605.44
```

```
G1 F115.556 R3367.45
G1 F113.479 R3312.22
G1 F110.125 R3448.55
G1 F108.022 R3405.06
G1 F106.855 R3174.37
G1 F104.529 R3138.37
G1 F101.423 R3303.44
G1 F99.1443 R3279.68
G1 F97.2808 R3062.69
G1 F94.7994 R3048.69
G1 F92.158 R3240.3
G1 F89.8044 R3238.02
G1 F87.284 R3041.42
G1 F84.7868 R3050.62
G1 F82.7791 R3263.89
G1 F80.4748 R3283.27
G1 F77.4483 R3112.39
G1 F75.0806 R3143.99
G1 F73.7707 R3372.39
G1 F71.6248 R3411.97
G1 F68.3051 R3269.6
G1 F66.1716 R3321.09
G1 F65.5126 R3558.04
G1 F63.593 R3615.22
G1 F60.1853 R3501.46
G1 F58.3317 R3569.49
G1 F58.2076 R3809.58
G1 F56.5378 R3881.33
G1 F53.1943 R3794.31
G1 F52.3983 R3834.54
G1 F51.8912 R4115.19
G1 F51.3377 R4110.58
G1 F51.8049 R3829.6
G1 F51.1665 R3776.36
G1 F47.5316 R3803.41
G1 F46.1102 R3704.17
G1 F46.9329 R3467.8
G1 F45.3526 R3369.66
G1 F41.3654 R3421.67
G1 F39.6096 R3333.3
G1 F40.0296 R3092.54
G1 F38.0436 R3006.82
G1 F33.7657 R3089.07
G1 F31.6157 R3015.56
G1 F31.3876 R2773.93
G1 F28.9266 R2705.54
G1 F24.5416 R2823.03
G1 F21.9795 R2769.28
G1 F20.8183 R2533.4
G1 F19.6492 R2514.42
G0 Z10
G0 F-149.756 R5130.29
G0 Z0
```

```
G1 F-151.387 R5141.05
G1 F-152.081 R5258.26
G1 F-150.883 R5470.67
G1 F-151.547 R5587.3
G1 F-153.989 R5614.11
G1 F-154.284 R5673.97
G1 F-152.492 R5763.57
G1 F-153.469 R5959.57
G1 F-155.393 R5864.6
G1 F-154.553 R5683.38
G1 F-155.178 R5654.39
G1 F-157.238 R5782.34
G1 F-158.496 R5730.91
G1 F-159.056 R5495.06
G1 F-160.397 R5447.76
G1 F-162.402 R5593.8
G1 F-163.744 R5554.05
G1 F-164.552 R5324.36
G1 F-165.977 R5289.64
G1 F-167.878 R5453.61
G1 F-169.287 R5426.58
G1 F-170.354 R5205.6
G1 F-171.841 R5184.47
G1 F-173.588 R5365.57
G1 F-175.04 R5352.04
G1 F-176.363 R5142.36
G1 F-177.161 R5138.31
G0 Z10
G0 F-110.498 R2628.42
G0 Z0
G1 F-110.601 R2648.8
G1 F-111.591 R2862.88
G1 F-114.024 R2914.47
G1 F-118.191 R2793.35
G1 F-120.542 R2858.62
G1 F-120.782 R3098.52
G1 F-122.85 R3168.68
G1 F-126.323 R3092.84
G1 F-126.95 R3080.73
G0 Z25
```