

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS**

Разработка алгоритма вокселизации трехмерных моделей для задач печати методом селективного лазерного плавления

Обучающийся / Student Залялутдинова Карина Радиковна

Факультет/институт/клластер/ Faculty/Institute/Cluster факультет систем управления и робототехники

Группа/Group R34812

Направление подготовки/ Subject area 15.03.04 Автоматизация технологических процессов и производств

Образовательная программа / Educational program Цифровое производство 2020

Язык реализации ОП / Language of the educational program Русский

Квалификация/ Degree level Бакалавр

Руководитель ВКР/ Thesis supervisor Афанасьев Максим Яковлевич, кандидат технических наук, Университет ИТМО, факультет систем управления и робототехники, доцент (квалификационная категория "ординарный доцент")

Обучающийся/Student

Документ подписан	
Залялутдинова Карина Радиковна	
17.05.2024	
(эл. подпись/ signature)	

Залялутдинова
Карина
Радиковна

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Афанасьев Максим Яковлевич	
17.05.2024	
(эл. подпись/ signature)	

Афанасьев
Максим
Яковлевич

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Залялутдинова Карина Радиковна

Факультет/институт/клластер/ Faculty/Institute/Cluster факультет систем управления и робототехники

Группа/Group R34812

Направление подготовки/ Subject area 15.03.04 Автоматизация технологических процессов и производств

Образовательная программа / Educational program Цифровое производство 2020

Язык реализации ОП / Language of the educational program Русский

Квалификация/ Degree level Бакалавр

Тема ВКР/ Thesis topic Разработка алгоритма вокселизации трехмерных моделей для задач печати методом селективного лазерного плавления

Руководитель ВКР/ Thesis supervisor Афанасьев Максим Яковлевич, кандидат технических наук, Университет ИТМО, факультет систем управления и робототехники, доцент (квалификационная категория "ординарный доцент")

Характеристика темы ВКР / Description of thesis subject (topic)

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not

Тема в области прикладных исследований / Subject of applied research: да / yes

Основные вопросы, подлежащие разработке / Key issues to be analyzed

Анализ общей проблематики представления трехмерных компьютерных моделей;
Обоснование воксельного подхода;

Анализ существующих алгоритмов вокселизации трехмерных моделей;
Разработка программы вокселизации и заполнения модели решетчатой структурой;
Разработка графического интерфейса программы.

Дата выдачи задания / Assignment issued on: 05.02.2024

Срок представления готовой ВКР / Deadline for final edition of the thesis 23.05.2024

СОГЛАСОВАНО / AGREED:

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Афанасьев Максим	

Афанасьев

Яковлевич
17.05.2024

Максим
Яковлевич

Документ подписан
Залялутдинова Карина Радиковна
17.05.2024

Залялутдинова
Карина
Радиковна

Документ подписан
Пирогов Александр Владимирович
20.05.2024

Пирогов
Александр
Владимирович

Задание принял к
исполнению/ Objectives
assumed BY

Руководитель ОП/ Head
of educational program

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
SUMMARY OF A GRADUATION THESIS**

Обучающийся / Student Заялутдинова Карина Радиковна

Факультет/институт/кластер/ Faculty/Institute/Cluster факультет систем управления и робототехники

Группа/Group R34812

Направление подготовки/ Subject area 15.03.04 Автоматизация технологических процессов и производств

Образовательная программа / Educational program Цифровое производство 2020

Язык реализации ОП / Language of the educational program Русский

Квалификация/ Degree level Бакалавр

Тема ВКР/ Thesis topic Разработка алгоритма вокселизации трехмерных моделей для задач печати методом селективного лазерного плавления

Руководитель ВКР/ Thesis supervisor Афанасьев Максим Яковлевич, кандидат технических наук, Университет ИТМО, факультет систем управления и робототехники, доцент (квалификационная категория "ординарный доцент")

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
DESCRIPTION OF THE GRADUATION THESIS**

Цель исследования / Research goal

Разработка алгоритма вокселизации и заполнения трехмерной модели решетчатой структурой

Задачи, решаемые в ВКР / Research tasks

Анализ общей проблематики представления объемных компьютерных моделей; Анализ существующих алгоритмов вокселизации; Общий анализ применения решетчатых структур; Разработка алгоритма вокселизации и заполнения модели решетчатой структурой; Разработка графического интерфейса алгоритма

Краткая характеристика полученных результатов / Short summary of results/findings

Разработанный алгоритм отвечает поставленным в работе требованиям, а именно: Строит воксельное представление трехмерной модели (менее, чем за 1 минуту для воксельной сетки размером 256x256x256); Генерирует внутреннюю решетчатую структуру модели; Обладает графическим интерфейсом и др.

Обучающийся/Student

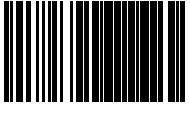
Документ подписан
Заялутдинова Карина Радиковна



Заялутдинова
Карина

17.05.2024
(эл. подпись/ signature)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Афанасьев Максим Яковлевич	
17.05.2024	

(эл. подпись/ signature)

Радикова
(Фамилия И.О./ name
and surname)

Афанасьев
Максим
Яковлевич

(Фамилия И.О./ name
and surname)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	2
1 Анализ общей проблематики представления объемных компьютерных моделей	5
1.1 Каркасное моделирование	5
1.2 Поверхностное моделирование	6
1.3 Твердотельное моделирование	7
2 Вокселизация трехмерной модели	7
2.1 Обоснование воксельного подхода для представления объемных компьютерных моделей	8
2.2 Решетчатые структуры в аддитивном производстве	11
3 Анализ существующих алгоритмов вокселизации	14
3.1 Поверхностная вокселизация (Surface Voxelization)	14
3.2 Сплошная вокселизация (Solid Voxelization)	16
3.2 Алгоритм вокселизации с помощью октодерева (Sparse Voxel Octree)	
19	
4 Разработка программы вокселизации трехмерных моделей	21
4.1 Требования, предъявляемые к программе вокселизации	21
4.2 Обзор open-source решения для алгоритма вокселизации трехмерных моделей	22
4.3 Обзор библиотек и модулей, используемых в программе	24
4.4 Описание архитектуры программы вокселизации	26
5 Разработка графического интерфейса программы вокселизации	33
ЗАКЛЮЧЕНИЕ	41
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	43
ПРИЛОЖЕНИЕ 1	45

ВВЕДЕНИЕ

Аддитивное производство представляет собой процесс изготовления деталей путем послойного добавления материала. Сегодня отрасль трехмерной печати активно развивается. По прогнозу Министерства экономического развития Российской Федерации, которое оно опубликовало в Белой книге «Развитие отдельных высокотехнологичных направлений», к 2030 году объем рынка 3D-печати и оборудования в России увеличится в 13 раз [1].

Селективное лазерное сплавление (SLM, Selective laser melting) – это технология трехмерной печати путем плавления металлических порошковых составов с помощью лазера.

Технология SLM-печати позволяет изготавливать детали сложного профиля, а также использовать широкий круг материалов (например, цинк, сталь, титановые сплавы, алюминиевые сплавы и другие).

Процесс изготовления деталей с помощью селективного лазерного сплавления включает в себя несколько этапов. Сначала на платформу наносится слой металлического порошка определенной высоты. Лазерный луч с помощью сканирующего зеркала расплавляет необходимые участки порошкового слоя, формируя тем самым форму участков, а также соединяя изготавливаемый слой с предыдущим. Затем насыпается новый слой порошка, и стол опускается, после чего цикл повторяется до полного изготовления детали. После окончания процесса неспеченный порошок удаляется. Принцип изготовления деталей методом SLM представлен на Рисунке 1.

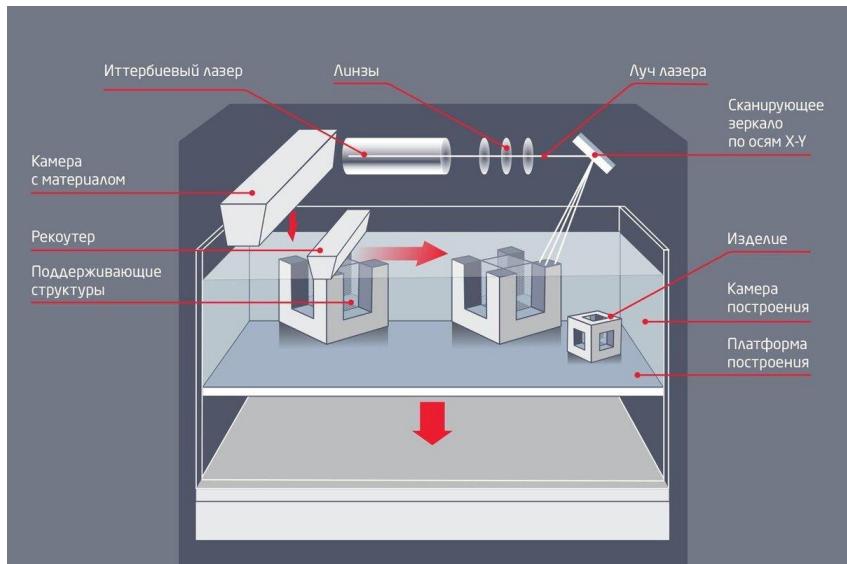


Рисунок 1 – Принцип изготовления деталей методом SLM

Особенность данной технологии заключается в том, что в процессе изготовления деталь окружена неспеченным металлическим порошком, который служит опорой элементов изготавляемой детали. Именно это – выборочное плавление материала – позволяет получать изготавливать детали сложных профилей, имеющих относительно тонкие внутренние перегородки.

Удобным способом моделирования трехмерных объектов для SLM печати является *воксельная модель*. Данная работа направлена на разработку алгоритма вокселизации трехмерной модели, представленной с помощью полигональной модели в STL-формате. Разработанный алгоритм можно будет использовать для 3D-печати методом селективного лазерного сплавления.

Целью исследования является разработка оптимизированного алгоритма вокселизации и заполнения трехмерных моделей решетчатой структурой.

Объектом исследования является алгоритм вокселизации трехмерных моделей.

Предметом исследования являются методики (подходы) к разбиению модели на воксели, обеспечивающие возможность заполнения модели решетчатой структурой для улучшения внутренней структуры изделия.

Задачами исследования являются:

- Анализ общей проблематики представления объемных компьютерных моделей;
- Анализ существующих алгоритмов вокселизации;
- Общий анализ применения решетчатых структур;
- Разработка алгоритма вокселизации и заполнения модели решетчатой структурой;
- Разработка графического интерфейса алгоритма.

Актуальность работы обусловлена тем, что разрабатываемый алгоритм позволит быстро создавать трехмерные модели с решетчатой структурой. Благодаря своим уникальным свойствам (сохранение прочности при уменьшении веса, проницаемость, экономичность, теплопередача) подобные изделия могут быть использованы в различных областях (например, в производстве электроники, в медицине, в аэрокосмической промышленности). Однако их производство затруднено самой внутренней структурой изделий, а также сложностью проектирования.

1 АНАЛИЗ ОБЩЕЙ ПРОБЛЕМАТИКИ ПРЕДСТАВЛЕНИЯ ОБЪЕМНЫХ КОМПЬЮТЕРНЫХ МОДЕЛЕЙ

Сегодня известны три основных подхода к трехмерному моделированию изделий:

- Каркасное моделирование;
- Поверхностное моделирование;
- Твердотельное моделирование.

1.1 Каркасное моделирование

Каркасное моделирование предполагает представление трехмерной модели в виде набора точек, узлов каркаса, и соединяющих линий. Математически 3D-модель представляет собой совокупность уравнений кривых, описывающих поверхность детали, координат точек и информации о связности кривых и точек.

Назначением каркасной 3D-модели является простейшее представление форм граней детали с помощью линий.

Такой способ представления трехмерного объекта прост в построении. Для модели, имеющей плоские грани, достаточно показать ее ребра. Для построения объекта с криволинейными поверхностями необходимо задать дополнительные точки и линии.

Существенным недостатком каркасного способа представления трехмерных объектов является неоднозначность изображения для восприятия наблюдателем. В каркасной модели трудно отличить видимые и невидимые грани, а также внешние и внутренние поверхности. Подобная неоднозначность не только затрудняет чтение изображения трехмерного объекта, но и делает невозможным расчет массы изделия, применение метода конечных элементов для инженерного анализа модели.

Из-за своих ограничений каркасное моделирование было вытеснено другими способами проектирования, и сегодня используется как вспомогательный способ представления трехмерных моделей.

1.2 Поверхностное моделирование

Поверхностное моделирование представляет собой описание не только точек и соединяющих их линий, как при каркасном моделировании, но и информации об образующих поверхностях модели и том, как они сопрягаются между собой.

Полигональная модель представляет собой трехмерный объект, поверхность которого описана с помощью *полигонов* или многоугольников, соединенных между собой. Чаще всего в качестве полигона выступает треугольник.

Стоит отметить, что такой способ представления трехмерной модели подразумевает описание только образующих плоскостей, без заполнения внутреннего объема. Однако многие трехмерные объекты не являются замкнутыми, когда часть поверхностей видны как снаружи, так и изнутри. В случае поверхностного моделирования внутренние поверхности должны быть описаны как внешние.

Также, полигональная модель может включать в себя информацию о физических свойствах образующих поверхностей, например, материал, цвет, коэффициент прозрачность и другое, для достижения более реалистичного изображения.

Как уже было сказано выше, полигональная модель не хранит информацию о внутреннем объеме объекта. Это не позволяет вычислить объем и массу изделия, применить метод конечных элементов для анализа.

Кроме того, полигональные модели содержит информацию о множестве граней полигонов, что делает применение булевых операций к модели более сложной и трудоемкой задачей.

Сегодня поверхностное моделирование используется для создания программ для обработки поверхностей изделий с помощью станков с ЧПУ, а также для представления фотореалистичных моделей изделия.

1.3 Твердотельное моделирование

Самым популярным способом представления трехмерных моделей сегодня является *твердотельное моделирование*.

Твердотельная (сплошная) модель представляет собой целостный замкнутый объект, заполненный точками или другими элементами. Каждая точка твердотельной модели определена как находящаяся внутри нее, снаружи или на границе объекта. Сегодня такой подход к моделированию является стандартом моделирования в САЕ- и CAD-системах.

Твердотельная модель позволяет производить *декомпозицию* объекта, то есть разделять единую сложную модель на несколько более простых элементов или *примитивов*. К примитивам относят кубы, сферы, пирамиды, призмы и так далее. Использование примитивов позволяет сократить объем используемой памяти для хранения модели и уменьшить трудоемкость построения объекта.

2 ВОКСЕЛИЗАЦИЯ ТРЕХМЕРНОЙ МОДЕЛИ

Воксельное представление модели представляет собой кубическую трехмерную сетку размером $n \times m \times l$, объектами которой являются кубики заданной величины – *воксели*.

Вокселизация позволяет проектировать объекты с полостями, разрывами и негладкими границами, что, как правило, непросто при представлении объекта с помощью полигональных моделей.

Выделяют два вида воксельного представления:

- *Бинарная* воксельная модель, когда деталь описывается с помощью трехмерной матрицы, содержащей только нули (воксель находится

снаружи объекта, он пуст) и единицы (воксель находится внутри объекта или на его границе);

- Воксельная модель, содержащая информацию о цвете voxеля и его альфа-канале (прозрачности). Цветовое разделение voxелей позволяет производить функциональную сегментацию различных частей моделей.

Сегодня воксельное представление данных получило широкое распространение во многих сферах и способно решать различные виды задач визуализации:

- Генерация ландшафтов для симуляции виртуального окружения, применяется как в игровой индустрии, так и в образовании;
- Трехмерное представление внутренних органов человека на основании показателей, полученных с помощью томографических сканеров;
- Создание моделей для естественнонаучных исследований.

Стоит отметить, что воксельная модель является лишь приближенным представлением изготавливаемой детали, то есть после применения voxелизации внешняя форма детали будет изменена в зависимости от величины voxелей.

2.1 Обоснование воксельного подхода для представления объемных компьютерных моделей

Как правило, исходными данными для трехмерного принтера является STL-файл, который содержит информацию о геометрической форме 3D-модели. Согласно ГОСТ Р 57591-2017, «STL – формат файлов, применяемый с 1987 г. в качестве базового для передачи данных компьютерной 3D-модели в аддитивную машину для построения физической модели». Данный формат поддерживается большинством современных CAD-систем.

STL-файл представляет трехмерную модель в виде набора треугольных граней, описывающих ее поверхность, а также вектор нормали, исходящий из треугольников в сторону направления материала. STL-файл может быть записан двумя способами:

- Текстовый формат (ASCII STL) – более понятный для чтения, но более объемный и долгий для чтения и записи вариант;
- Двоичный формат – более компактный вариант, однако человеку трудно его прочесть; поддерживается не всеми программами.

На Рисунке 2 представлен текстовый формат STL-файла.

```
solid ascii
facet normal -2.902847e -01 -9.569403e -01 0.000000e +00
outer loop
vertex 6.173166e +00 7.612047e -00 0.000000e +00
vertex 8.049097e +00 1.921472e -01 2.000000e +01
vertex 6.173166e +00 7.612047e -01 2.000000e +01
endloop
endfacet
facet normal -2.902847e -01 -9.569403e -01 0.000000e +00
outer loop
vertex 6.173166e +00 7.612047e -01 0.000000e +00
vertex 8.049097e +00 1.921472e -01 0.000000e +00
vertex 8.049097e +00 1.921472e -01 2.000000e +01
endloop
endfacet
facet normal 9.951847e -01 9.801714e -02 -0.000000e +00
outer loop
vertex 2.000000e +01 1.000000e +01 0.000000e +00
vertex 1.980785e +01 1.195090e +01 0.000000e +00
vertex 1.980785e +01 1.195090e +01 2.000000e +01
endloop
endfacet
end solid
```

Рисунок 2 – Структура ASCII STL-файла.

STL-файл содержит координаты трех точек, образующих треугольный полигон, а также координаты вектора нормали.

Как становится понятно из описания STL-файла, он представляет собой полигональную модель трехмерного объекта. Часто в STL-файлах содержатся

ошибки, которые обусловлены самой сутью данного формата – отсутствием топологической информации (описание расположения объектов друг относительно друга).

Наиболее распространенные ошибки в полигональной модели STL-файла:

- Отверстия или разрывы, то есть отсутствующие треугольники в сетке;
- Вырожденные треугольники, когда все ребра треугольника коллинеарны, и он имеет нулевую площадь;
- Пересекающиеся и перекрывающиеся грани;
- Инвертированные нормали, когда вектор нормали направлен в другую сторону.

На Рисунке 3 представлен пример ошибок в полигональной модели.

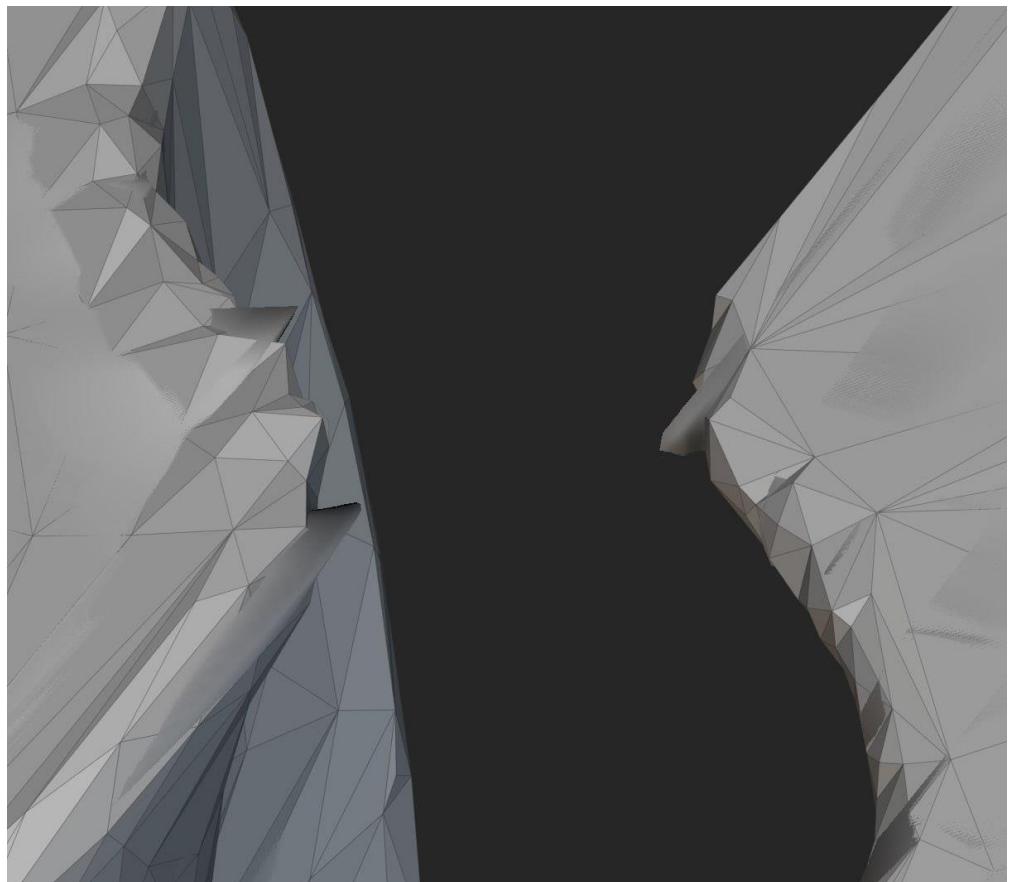


Рисунок 3 – Ошибки в полигональной модели.

Эти ошибки нарушают процесс *слайсинга* модели (процесс нарезания модели на слои для написания управляющего кода для 3D-принтера) и отражаются на конечном изделии. Применение булевых операций к полигональной модели, а также более сложная форма детали, естественно, увеличивают количество и шанс появления ошибки.

Эти ошибки чаще всего неизбежны. Более того, их не всегда легко заметить и устраниТЬ. Чтобы устранить эти коллизии, в процессе моделирования изделия для трехмерной печати необходимо дополнительно проводить проверки, верификации полигональных моделей, что увеличивает время проектирования.

Воксельная модель является удобным способом представления модели для 3D-печати, поскольку она лишена подобных проблем и позволяет легко применять геометрические операции к модели.

С другой стороны, сложность и объем вычислений будет напрямую зависеть от уровня детализации воксельной модели, то есть от количества вокселей, а также от габаритов и сложности самого объекта. Например, при увеличении детализации модели в N раз, объем памяти увеличится в N^3 раз. Такие затраты памяти сильно ограничивают разрешающую способность и точность моделирования.

Тем не менее, вокселизация является перспективным направлением компьютерной графики. Большое количество затрачиваемой памяти возможно компенсировать путем выбора оптимизированного алгоритма для создания воксельной модели, а также путем сжатия модели для ее хранения.

2.2 Решетчатые структуры в аддитивном производстве

С помощью технологии SLM печати можно создавать различные детали, обладающие сложной геометрией, в том числе изделия, содержащие *решетчатую структуру*, как показано на Рисунке 4.

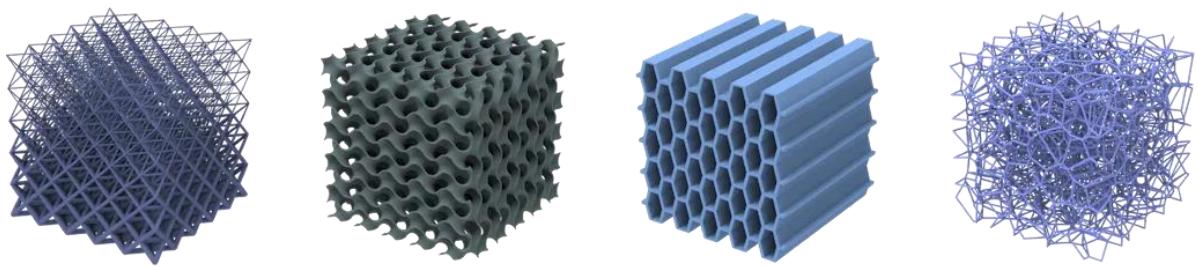


Рисунок 4 – Примеры решетчатых структур

Подобные структуры могут заполнять полость детали целиком или только его часть. Структура решетки, ее размер и плотность заполнения детали во многом определяют механические свойства полученного изделия. Например, как показано в исследовании «Evaluations of cellular lattice structures manufactured using selective laser melting» [8], предел текучести изделия с размером ячейки, равным 2 мм, примерно на 36% выше, чем у изделия с размером ячейки, равным 8 мм. Также, модуль упругости первого изделия примерно на 27% больше, чем у второго.

Главными преимуществами использования решетчатых структур в дизайне изделий являются:

- Уменьшение веса изделия с сохранением удовлетворительных механических и физических свойств. К примеру, исследование, проведенное в 2016 году [9], показало, что при испытании образцов на сжатие прочность решетчатого изделия по сравнению с цельным ниже, однако она удовлетворяет прикладным требованиям исследователей, что позволяет использовать подобные конструкции для уменьшения веса изделий;
- Экономия материала при изготовлении изделия (что может положительно повлиять на себестоимость продукции);
- Решетчатая структура изделия позволяет эффективно управлять температурой изделия (так как решетчатые конструкции обеспечивают

большую площадь поверхности, а скорость теплопередачи имеет пропорциональную зависимость от площади поверхности).

Все эти преимущества предоставляют новые возможности для производства изделий в различных областях, таких как аэрокосмическая отрасль, медицина, электроника, автомобилестроение и других.

Сегодня создать изделие с решетчатой структурой можно с помощью традиционных методов производства (например, путем листовой штамповки или с помощью литья), однако это является сложным процессом и серьезным вызовом для традиционного производства. Поэтому для создания решетчатых структур предприятия часто прибегают к технологиям трехмерной печати. Во многих случаях производство решетчатых структур с помощью трехмерной печати является наиболее эффективным методом производства подобных изделий.

Проектирование решетчатых структур. Современные CAD-системы не являются эффективными инструментами для проектирования решетчатых структур из-за лежащих в их основе методов обработки геометрии. Как уже было сказано выше, STL-модели решетчатых структур трудно редактировать и применять к ним логические операции.

Одним из подходов решения этой проблемы является вокселизация полигональных моделей. В результате данной работы будет разработана программа, позволяющая эффективно и быстро вокселизировать полигональные модели с решетчатой структурой.

Реализация в программе. В разрабатываемой программе представлены 3 вида решетчатой структуры: *гироид* (бесконечно связанная трижды периодическая минимальная поверхность) и поверхности Шварца Р (примитивная) и D (алмазная). Эти поверхности в программе описаны математическими формулами.

3 АНАЛИЗ СУЩЕСТВУЮЩИХ АЛГОРИТМОВ ВОКСЕЛИЗАЦИИ

Алгоритмы вокселизации трехмерных моделей, как правило, делятся на три основные группы:

- Поверхностная вокселизация;
- Сплошная вокселизация;
- Вокселизация с помощью октодерева.

Все три метода получили широкое распространение сегодня и могут быть использованы для решения разных задач. Остановимся на каждом из них подробнее.

3.1 Поверхностная вокселизация (Surface Voxelization)

Поверхностная вокселизация (Surface Voxelization) подразумевает вокселизацию только внешней, образующей поверхности трехмерной модели, то есть с помощью будет создана трехмерная полая модель.

Данный алгоритм основан на разделении модели на двумерные слои и их сканировании. Итак, при послойном подходе алгоритм разделяет объемную модель на слои, называемые *Z-плоскостью*, и обрабатывает каждый срез по отдельности.

Внутри каждого среза, на *Z-плоскости*, алгоритм строит пиксельную сетку (поскольку речь идет о двумерной плоскости). Затем шаг за шагом алгоритм проверяет каждый пиксель. Если пиксель пересекает линию, принадлежащую поверхности модели, то он отмечается как граничный, а если не пересекает – остается не тронутым.

На Рисунке 4 изображен двумерный срез модели, к которой был применен метод поверхностной вокселизации.

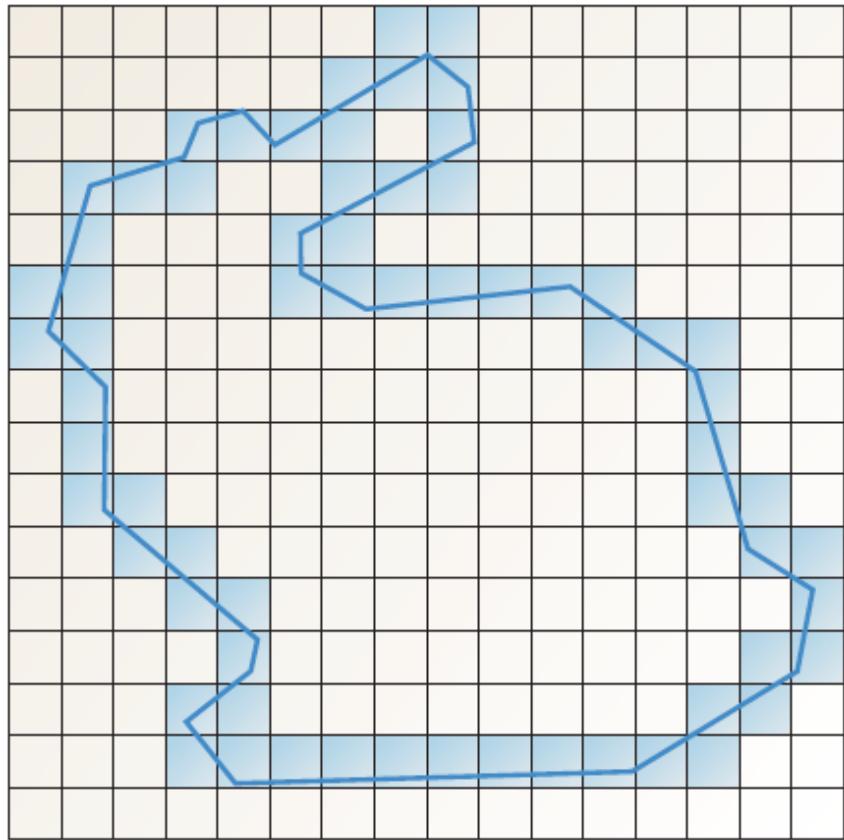


Рисунок 4 – Поверхностная вокселизация, двумерный срез.

Главное преимущество данного алгоритма в том, что он не использует большое количество памяти, поскольку генерирует воксели только на границе трехмерного объекта. По этой же причине поверхностная вокселизация выполняется быстрее сплошной. Более того, сегодня, благодаря развитию возможностей техники, существуют оптимизированные алгоритмы поверхностной вокселизации, которые подразумевают многопоточность, то есть обработку нескольких срезов одновременно. Многопоточная оптимизация данного алгоритма еще сильнее ускоряют его.

Однако данный метод чем-то похож на полигональное описание модели, использующее вместо плоских полигонов трехмерные воксели. К сожалению, он не лишен ошибок, которые возникают в полигональных моделях. Часто при методе поверхностной вокселизации образуются отверстия, разрывы поверхности, тогда модель становится незамкнутой.

Метод поверхностной вокселизации не подходит для решения задачи, поставленной в этом исследовании. Во-первых, воксельное представление трехмерной модели используется для того, чтобы избежать проблем полигональных моделей, а метод поверхностной вокселизации не всегда позволяет это сделать.

3.2 Сплошная вокселизация (Solid Voxelization)

Сплошная вокселизация (Solid Voxelization) предполагает заполнение вокселями не только внешних границ объекта, но и его внутреннего пространства.

Алгоритм сплошной вокселизации несколько похож на алгоритм поверхностной вокселизации, поскольку также обрабатывает модель послойно.

Каждый воксель определен по отношению к модели как лежащий внутри нее или снаружи. Проверка положения вокселя относительно модели проводится с помощью теоремы Жордана, которая гласит, что точка в пространстве считается внутренней, если число пересечений с моделью любого луча, исходящего из этой точки, нечетно; внешней – если четно. Использование этой теоремы накладывает ограничения для объекта вокселизации: модель должна иметь замкнутый контур.

Вместо построения бесконечного числа лучей из вокселя для проверки количества пересечений с границами, как правило, используется метод подсчета отрисованных фрагментов перед воксelem. Если число таких фрагментов нечетно, то воксель считается лежащим внутри объекта. Реализуется эта идея с помощью битовых масок.

Для простоты представим один ряд пикселей, лежащий горизонтально, как показано на Рисунке 5. Желтым показаны фрагменты, которые принадлежат модели (то есть находятся внутри нее), жирными черными линиями показаны границы этих фрагментов.

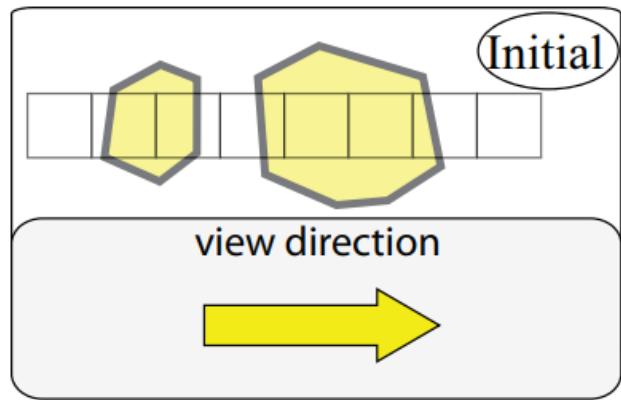


Рисунок 5 – Фрагменты, принадлежащие модели

Алгоритм перебирает все пиксели, которые лежат до первой границы, и проставляет в их битовой маске единицы. На Рисунке 6 красным выделен пиксель, битовую маску которого изменит алгоритм. Важно отметить, что сами воксели, лежащие на границе не помечаются единицей. К исходной битовой маске, заполненной нулями, и результирующей битовой маске применяется операция исключающего «ИЛИ» (XOR).

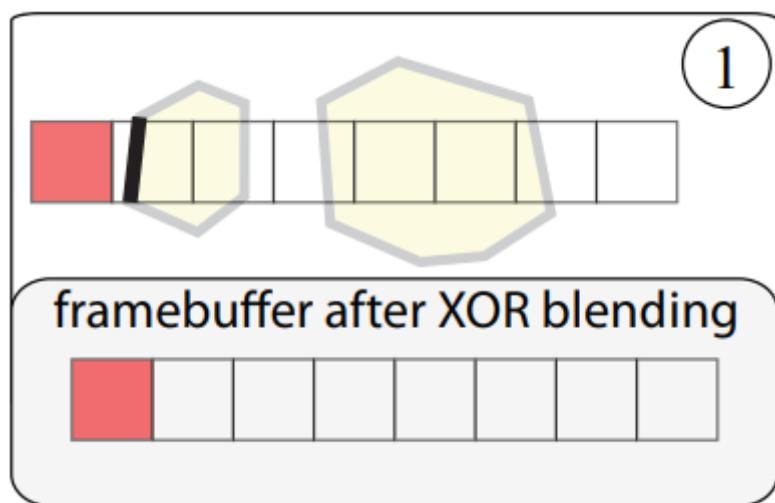


Рисунок 6 – изменение битовой маски для вокселей, лежащих перед первой границей.

Затем алгоритм снова проходит по строке пикселей до другой границы фрагмента, как показано на Рисунке 7. Алгоритм снова обновляет битовую маску и проставляет единицы у тех вокселей, которые находятся перед

выбранной границей. Затем операция XOR снова применяется к битовым маскам, полученной на предыдущем этапе и полученной на этом этапе.

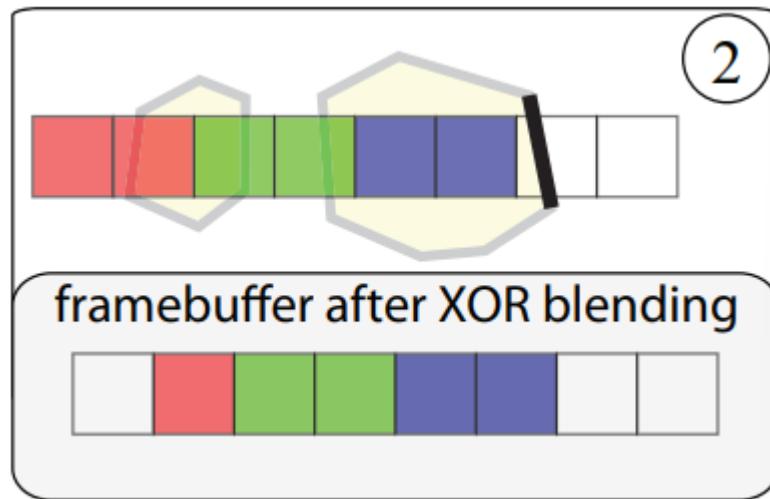


Рисунок 7 – Результат применения операции XOR.

Затем эти шаги повторяются до тех пор, пока не будут обработаны все границы, как показано на Рисунке 8.

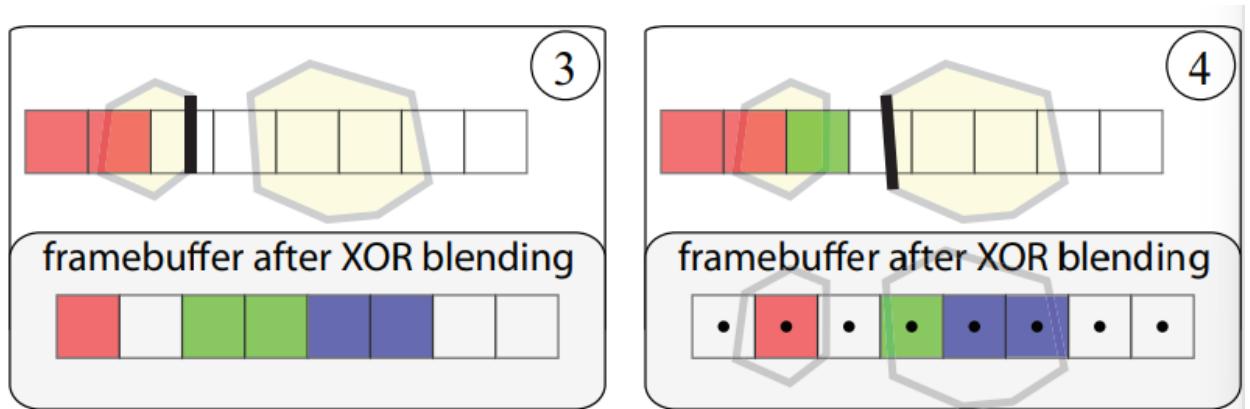


Рисунок 8 – Конечный результат применения алгоритма.

Заполнение внутреннего объема позволяет полностью избавиться от ошибок, которые возникают у полигональных моделей. Это делает сплошную воксельную модель удобной для решения задач моделирования, симуляций, анализа и трехмерной печати.

Недостатком данного алгоритма, как уже было сказано, является большое количество затрачиваемой памяти.

3.2 Алгоритм вокселизации с помощью октодерева (Sparse Voxel Octree)

Октодерево или восьмеричное дерево – это тип структуры данных, который представляет собой дерево с восьмью ответвлениями или *потомками*. У каждого потомка, также, может быть по восемь своих потомков.

Алгоритм, использующий октодеревья для вокселизации (Sparse Voxel Octree, SVO), проиллюстрирован на Рисунке 9.

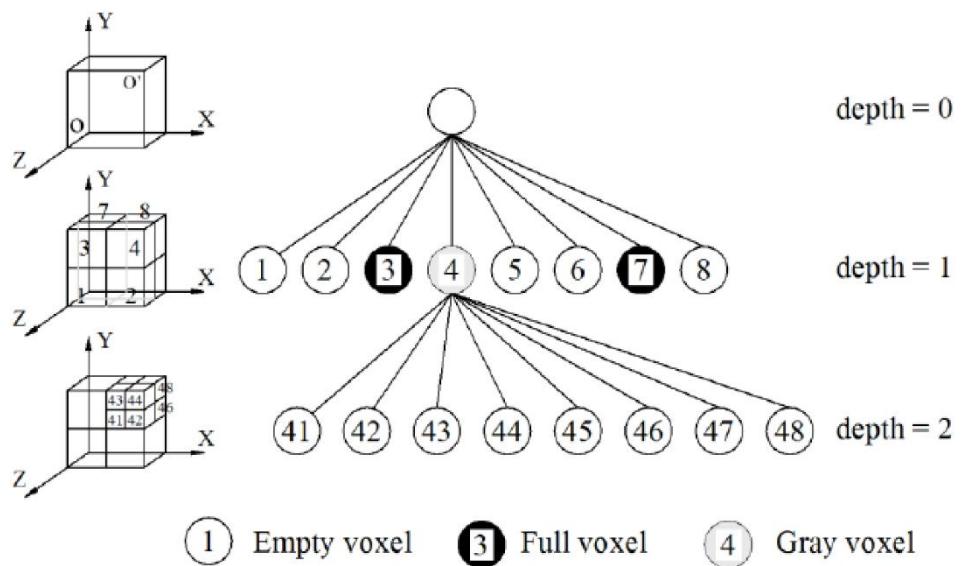


Рисунок 9 – Алгоритм Sparse Voxel Octree

Корнем октодерева является воксель, содержащий в себе все ограничивающие поверхности модели, то есть ограничивающий объект. Корень разбивается на 8 потомков, то есть на 8 более меленьких вокселий. Затем алгоритм проверяет каждого потомка, пересекает ли хотя бы один из них границу модели. Потомок, пересекающий границы объекта, делится еще на 8 потомков. Так, алгоритм разбивает большие воксели на более мелкие до тех пор, пока не будет достигнута желаемая точность.

Стоит отметить, что в результате этого метода будут разделены не все воксели. В случае, когда воксель находится полностью внутри модели, он не разделяется на более мелкие части, как показано на Рисунке 10.

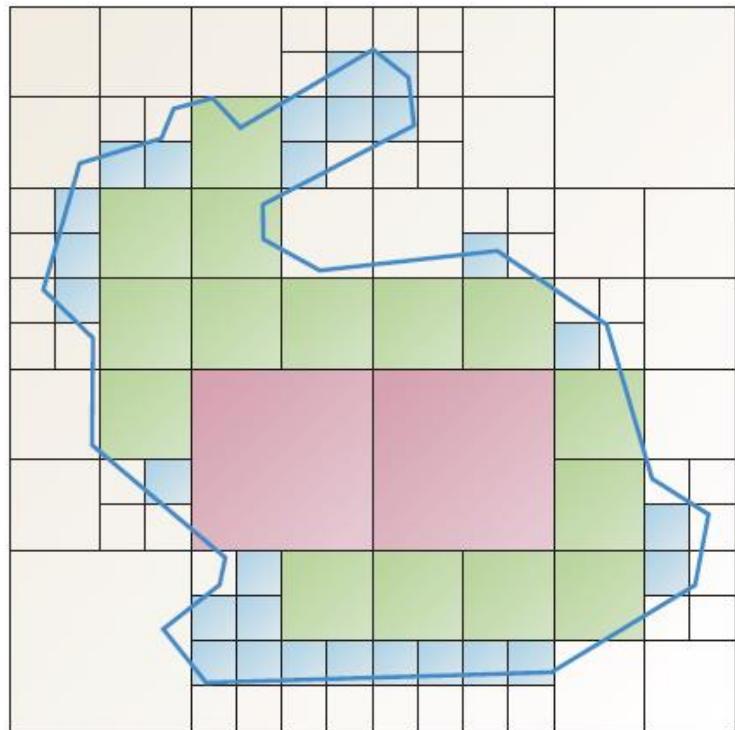


Рисунок 10 – Результат вокселизации методом SVO

4 РАЗРАБОТКА ПРОГРАММЫ ВОКСЕЛИЗАЦИИ ТРЕХМЕРНЫХ МОДЕЛЕЙ

4.1 Требования, предъявляемые к программе вокселизации

К разрабатываемой программе предъявляются следующие функциональные требования:

- Чтение STL-файла, содержащего трехмерную модель изделия;
- Преобразование полигональной сетки, описанной в STL-файле, в вокельное представление;
- Реализация сплошной вокселизации модели;
- Реализация заполнения воксельной модели решетчатой структурой;
- Обратное преобразование вокельного представления в STL-файл для слайсинга и генерации G-кода для 3D-принтера.

Таким образом, разрабатываемый алгоритм должен преобразовывать STL-файл в вокселизированное представление, заполнять модель решетчатой структурой и затем сохранять полученный массив данных обратно в STL-файл.

К разрабатываемой программе предъявляются следующие нефункциональные требования:

- Оптимизация: программа должна выполнять вокселизацию модели быстро (не более 1 минуты для воксельной сетки размером $256 \times 256 \times 256$ вокселей);
- Генерация решетчатой структуры должна быть реализована с помощью математических функций;
- Должен быть разработан графический интерфейс;
- Методология программирования – объектно-ориентированное программирование.

4.2 Обзор open-source решения для алгоритма вокселизации трехмерных моделей

Open-source software или *открытое программное обеспечение* – это программное обеспечение с открытым исходным кодом. Код таких программ можно свободно просматривать, использовать для собственных целей, а также модифицировать.

В качестве алгоритма вокселизации модели был использован инструмент «Cuda Voxelizer» [14], написанный Джеруном Баертом и опубликованный на платформе GitHub в 2017 году. Алгоритм реализует оптимизированную версию метода вокселизации, представленного в работе «Fast parallel surface and solid voxelization on GPUs» [15] на языке программирования C++. Сущность алгоритма была описана в пункте 3.2 данной работы.

Вокселизация и запуск программы осуществляется с помощью вызова команды в командной строке. Пример запроса: «cuda_voxelizer -f <имя файла> -s <размер воксельной сетки> -o <формат выходного файла> -solid». Опция «solid» означает выбор сплошной вокселизации, а не поверхностной (по умолчанию, то есть без этой опции, программа выполняет поверхностную вокселизацию).

«Cuda Voxelizer» позволяет реализовать вокселизацию с помощью как центрального, так и графического процессоров. При отсутствии на компьютере пользователя видеокарты программа использует центральный процессор. Стоит отметить, что данный инструмент оптимизирован: программа считывает количество ядер процессора и создает отдельный поток для каждого ядра. Таким образом, вокселизация нескольких слоев выполняется параллельно, что ускоряет процесс преобразования.

Данный инструмент поддерживает чтение файлов различных форматов: PLY, OFF, OBJ, 3DS, SM, RAY и STL. Выходные данные также могут быть записаны в файлы различных форматов, таких как VOX, BINVOX, OBJ.

Поскольку в данной работе выходной файл, получаемый в результате работы «Cuda Voxelizer», не является конечным (а будет в дальнейшем обрабатываться для заполнения решетчатой структурой), может быть использован любой выходной формат файла.

Параметр «-s» определяет размер voxelной сетки. Сначала программа определяет рамки, ограничивающие модель. Затем полученные измерения делит на указанное количество voxелей по каждой из осей. Таким образом, формируется размер каждого voxеля.

Тестирование программы. «Cuda Voxelizer» был протестирован на компьютере с центральным процессором Intel Core i5-12500H с 16 ядрами, без видеокарты. Входной файл – трехмерная модель Стенфордского кролика в формате STL (86 632 треугольника, размер 4,13 МБ). Выходной файл представлен в формате BINVOX. Вокселизация была выбрана сплошной. Размер voxelной сетки – 256^3 voxелей.

На Рисунке 11 представлена визуализация модели после вокселизации.

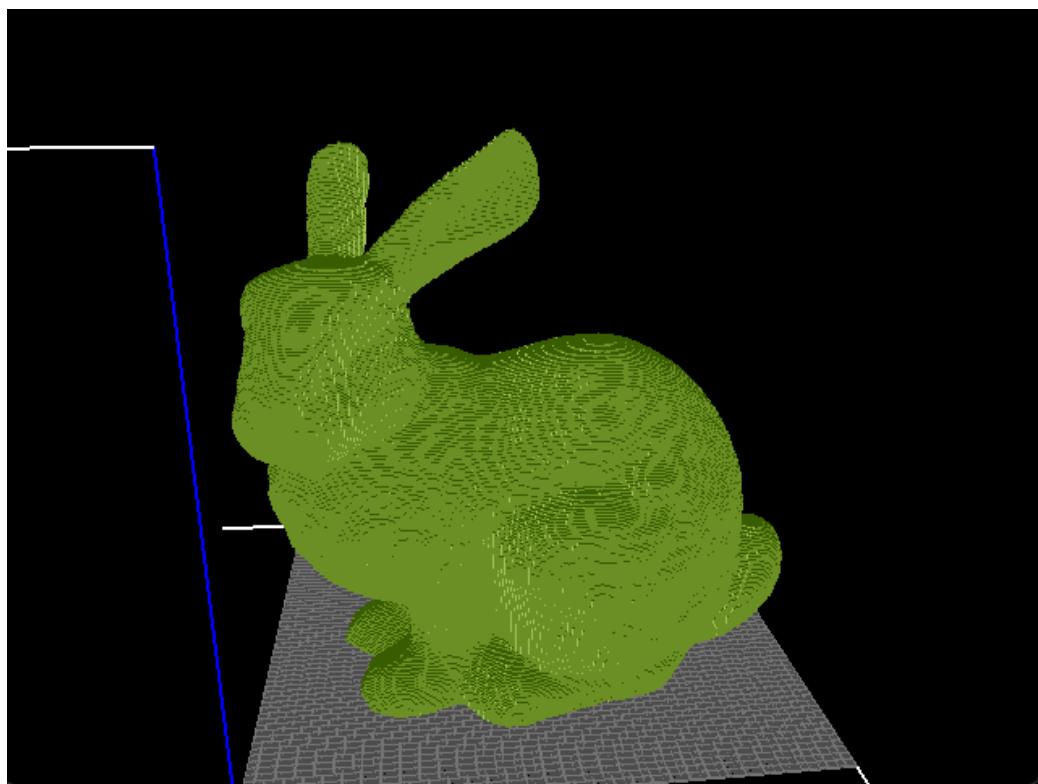


Рисунок 11 – Визуализация модели после вокселизации

Результат тестирования. Полный процесс вокселизации (включая чтение исходного файла и запись выходного файла) занял 2,7 секунды. Сам процесс вокселизации занял 2,3 секунды. Размер полученного файла – 208 КБ. На Рисунке 11 представлены выходные данные, написанные в терминале после вызова программы.

```
PS C:\Users\karin\PycharmProjects\pythonProject1> ./cuda_voxelizer -f 'StanfordBunny.stl' -s 256 -o bivox -solid
## CUDA VOXELIZER
CUDA Voxelizer v0.6 by Jeroen Baert
https://github.com/Forceflow/cuda_voxelizer - mail (at) jeroen-baert (dot) be

## PROGRAM PARAMETERS
[Info] Filename: StanfordBunny.stl
[Info] Grid size: 256
[Info] Output format: bivox file
[Info] Using CPU-based voxelization: No (default: No)
[Info] Using Solid Voxelization: Yes (default: No)

## READ MESH
[I/O] Reading mesh from StanfordBunny.stl
Reading StanfordBunny.stl... Done.
[Mesh] Number of triangles: 86632
[Mesh] Number of vertices: 259896
[Mesh] Computing bbox
Computing bounding box... Done.
x = -43.1485 .. 43.1485, y = -33.4337 .. 33.4337, z = 0 .. 83.7481

## VOXELISATION SETUP
[Voxelization] Bounding Box: (-43.157097,-43.157097,-1.283031)-(43.157097,43.157097,85.031166)
[Voxelization] Grid size: 256 256 256
[Voxelization] Triangles: 86632
[Voxelization] Unit length: x: 0.337165 y: 0.337165 z: 0.337165

## CUDA INIT
[CUDA] First call to CUDA Runtime API failed. Are the drivers installed?
[Info] CUDA GPU not found

## CPU VOXELISATION
[Info] No suitable CUDA GPU was found: Falling back to CPU voxelization
[Info] Using 16 threads
[Perf] CPU voxelization time: 2360.0 ms

## FILE OUTPUT
[I/O] Writing data in bivox format to StanfordBunny.stl_256.bivox

## STATS
[Perf] Total runtime: 2708.3 ms
```

Рисунок 12 – Данные полученные после вызова программы «Cuda Voxelizer»

Таким образом, данный инструмент показывает хорошее время вокселизации, которое удовлетворяет требованиям.

4.3 Обзор библиотек и модулей, используемых в программе

Модуль «Os». «Os» является модулем из стандартной библиотеки языка программирования Python. Данный модуль, как правило, используется для работы с операционной системой компьютера и его файловой системой.

В программе вокселизации используется одна функция из модуля «os» – os.system(<command>). Данная функция вызывает терминал и прописывает команду, написанную в скобках, в терминал. Таким образом, вызывается скомпилированная программа «Cuda Voxelizer». На Рисунке 13 представлен пример использования этой функции, реализованный в программе вокселизации.

```
command = f'cuda_voxelizer -f {self.inp["stl_file"].value} -s {self.inp["size"].value} -o binvox -solid'
os.system(command)           # Вызов программы Cuda Voxelizer с помощью терминала
```

Рисунок 13 – Вызов программы «Cuda Voxelizer» с помощью модуля «Os»

Библиотека «Trimesh». «Trimesh» — это популярная библиотека на Python для работы с полигональными сетками. Она предоставляет широкий функционал для чтения, анализа и изменения полигональных представлений с помощью Python.

Для алгоритма вокселизации используется метод trimesh.voxel.ops.matrix_to_marching_cubes, который преобразовывает многомерную бинарную матрицу в полигональное представление с помощью алгоритма «Marching Cubes» [16]. Данный алгоритм строит многоугольники, представляющие поверхность, проходящую через область voxеля. Затем эти многоугольники объединяются для формирования необходимой поверхности. После чего строятся нормали к поверхностям, которые определяют направление материала.

Модуль «Binvox_rw». Данный инструмент представляет собой очень простой парсер для чтения и записи Binvox-файлов. Для алгоритма вокселизации он используется по своему прямому назначению: считывает воксельный массив, а также другие параметры (масштаб, направление осей, размер массива и др.) из файла.

Библиотека «Open 3D». Данная библиотека доступна на двух языках программирования Python и C++. «Open 3D» предназначена для визуализации

3D-объектов, в том числе и для визуализации полигональных моделей, записанных в STL-файле. Именно для этой цели данная библиотека и применяется в программе вокселизации.

Стоит отметить, что эта библиотека не включена в структуру классов программы. Это было реализовано осознанно: на данном этапе «Open 3D» используется как инструмент отладки и проверки выполнения программой правильной вокселизации и заполнения решетчатой структурой, и, возможно, в будущем будет заменена на другую для удобной интеграции с графическим интерфейсом.

4.4 Описание архитектуры программы вокселизации

Выбранная архитектура программы была вдохновлена подходом к моделированию, реализованным в программном обеспечении Blender.

Blender – свободное ПО для создания графических объектов и сцен. Данный продукт является популярным среди 3D-дизайнеров. Данное ПО предоставляет возможность использовать *узлы* или *ноды* для построения геометрии. Суть такого подхода заключается в том, что пользователь создает не каждый объект по отдельности, а задает свойства группе объектов с помощью нод. Таким образом, при изменении параметров узла, меняется все объекты, которые относятся к узлу.

На Рисунке 14 представлен пример проекта, использующий геометрические узлы.

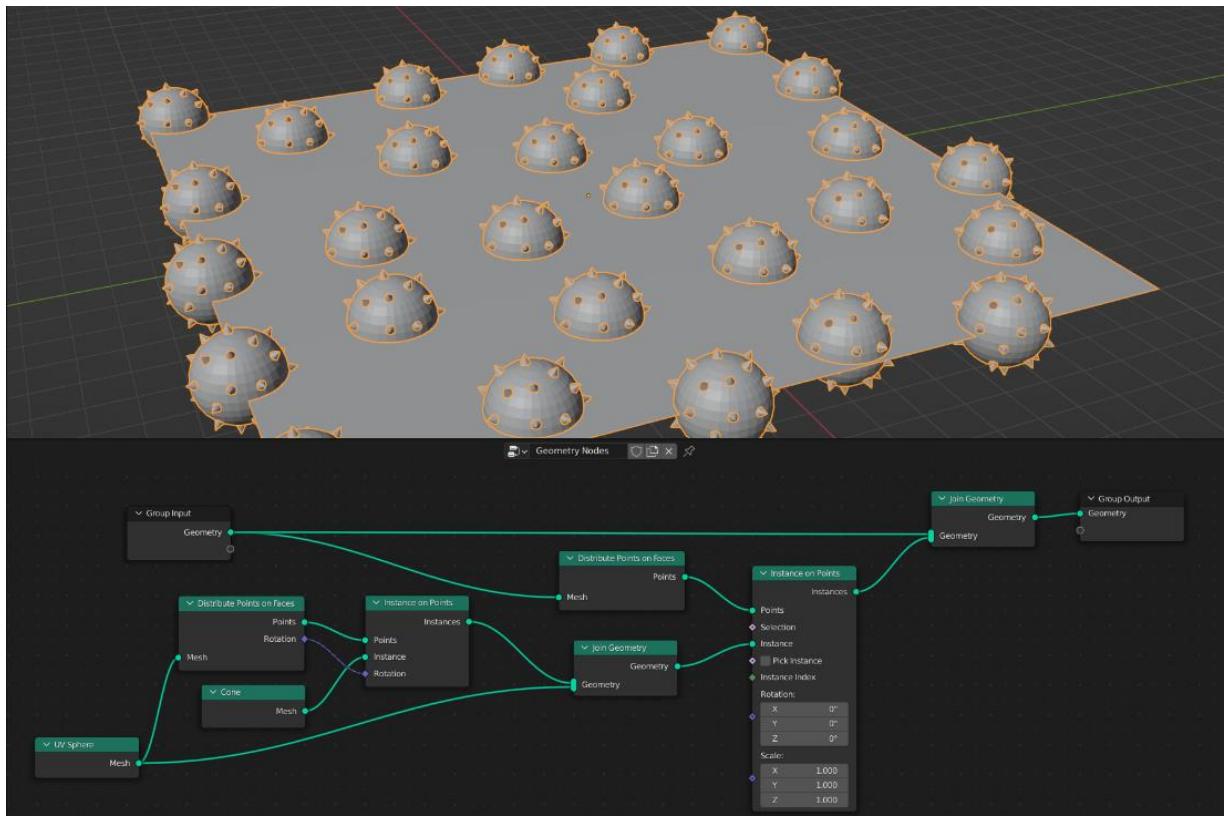


Рисунок 14 – Пример проекта, использующий геометрические узлы в Blender

Логика алгоритма. Сам алгоритм вокспараеализации и заполнения модели решетчатой структуры достаточно прост. Его можно представить в виде древовидной структуры, представленной в проекте Blender.

На Рисунке 15 представлена логика алгоритма вокселизации.

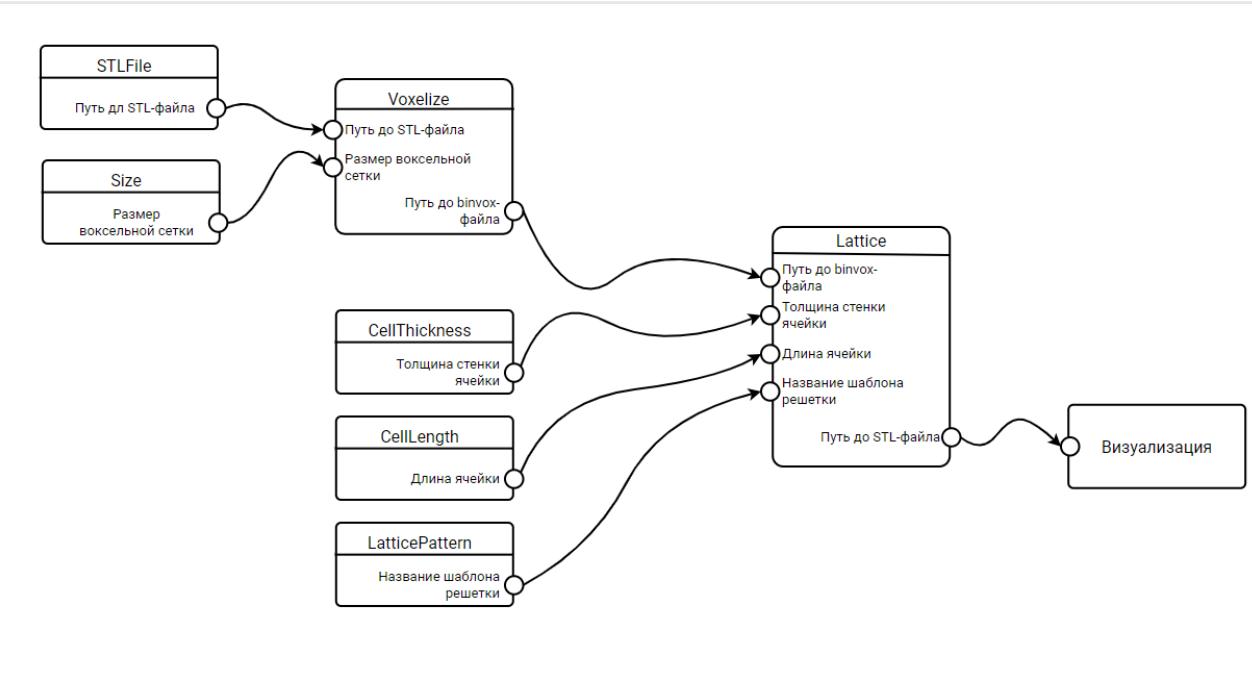


Рисунок 15 – Логика алгоритма voxelизации и заполнения решетчатой структурой

Программа имеет несколько нод. Более подробное описание нод представлено ниже. Каждая нода имеет входные параметры и (или) выходные параметры. Они соединяются между собой, и выходной параметр одной ноды становится входным параметром другой.

Таким образом, при изменении ноды Size, будет перестраиваться воксельная модель, а затем перестроенная воксельная модель будет заполняться решетчатой структурой. Если изменить ноду CellThickness, то нет необходимости перестраивать воксельную модель в ноде Voxelize, нужно лишь перезаполнить имеющуюся модель решеткой с измененными параметрами.

Структура. В программе voxelизации описаны несколько «главных» классов: Project, Input и Node.

Класс Project является объединяющим классом для всех остальных. Он содержит функции добавления нод в проект, отображения созданных нод, а также записи нод в файл формата JSON и чтение входного JSON-файла, содержащего проект. Главное назначение этого класса в том, что для

обновления информации о нодах в случае изменения каких-либо параметров не нужно обновлять каждую ноду вручную, а лишь обновить весь проект.

Класс `Input` содержит информацию обо всех входных данных нод, а также позволяет отслеживать, были ли изменены входные данные. Если они были изменены, то с помощью класса `Project` необходимо обновить весь проект.

Класс `Node` является родительским классом для следующих классов или узлов:

- `BINVOXFile` – нода, имеющая один выходной параметр – путь до файла в формате `Binvox`. Позволяет избежать этапа вокселизации, если она воксельная модель была создана ранее.
- `STLFile` – нода, имеющая один выходной параметр – путь до файла в формате `STL`.
- `Size` – нода, имеющая один выходной параметр – размер воксельной сетки.
- `CellLength` – нода, имеющая один выходной параметр – длину ячейки решетчатой структуры.
- `CellThickness` – нода, имеющая один выходной параметр – толщину стенки ячейки решетчатой структуры.
- `LatticePattern` – нода, имеющая один выходной параметр – название шаблона решетчатой структуры.
- `Voxelize` – нода, реализующая вокселизацию модели из `STL`-файла в `Binvox`-файл. Нода принимает 2 входных параметра: путь до `STL`-файла и размер воксельной сетки. Выходной параметр ноды – путь до выходного `Binvox`-файла.
- `Lattice` – нода, выполняющая заполнение воксельной модели решетчатой структуры. Нода принимает 4 входных параметра: путь до `Binvox`-файла, имя шаблона решетчатой структуры, длину ячейки решетки и толщину стенки ячейки. Выходным параметром

ноды является путь до STL-файла, который содержит модель с заполнением решеткой.

Как видно из описания нод, их можно условно разделить на 2 категории: «простые», содержащие только выходной параметр, и «сложные», содержащие как входы, так и выходы.

Выходные данные «простых» узлов задаются пользователем. В сущности, «простые» узлы являются просто контейнерами для определенных параметров.

Стоит отдельно описать «сложные» ноды: Voxelize и Lattice.

Нода Voxelize получает путь до STL-файла и размер воксельной сетки. Эти данные используются для построения модели с помощью программы «Cuda Voxelizer». На Рисунке 16 представлен текст программы, описывающей этот узел.

```
329     2 usages
330     class Voxelize(Node):
331         def init(self, *arg):
332             # Принимает 2 входных параметра: путь до STL-файла и размер воксельной сетки
333             self.inp = {'stl_file': Input(arg[0][0]),
334                         'size': Input(arg[0][1]),
335                         }
336             self.out = {'out': 0}
337
338         def update_func(self):
339             last = self.out['out']
340             command = f'cuda_voxelizer -f {self.inp["stl_file"].value} -s {self.inp["size"].value} -o binvox -solid'
341             os.system(command)          # Вызов программы Cuda Voxelizer с помощью терминала
342             self.out['out'] = f'{self.inp["stl_file"].value}_{self.inp["size"].value}.binvox'
343             return last != self.out['out']
344
345         def __repr__(self):
346             return "%s (%s)\n" % (self.class_name(), self.out["out"])
```

Рисунок 16 – Текст программы, описывающей ноду Voxelize

Как уже было сказано выше, узел Lattice принимает 4 параметра. Сначала нода принимает имя шаблона решетчатой структуры и устанавливает соответствующую математическую формулу, описывающую этот шаблон. В программе реализованы 3 шаблона: гироид и поверхности Шварца Р и D. На Рисунке 17 представлены функции, описывающие эти структуры.

```

9   # Функции, описывающие решетчатую структуру
10  # Гироид
11  func_gyroid = lambda x_coord, y_coord, z_coord, l: (np.sin(x_coord / l) * np.cos(y_coord / l) +
12                                              np.sin(y_coord / l) * np.cos(z_coord / l) +
13                                              np.sin(z_coord / l) * np.cos(x_coord / l))
14  # Поверхность Шварца P
15  func_primitive = lambda x_coord, y_coord, z_coord, l: np.cos(x_coord / l) + np.cos(y_coord / l) + np.cos(z_coord / l)
16
17  # Поверхность Шварца D
18  func_diamond = lambda x_coord, y_coord, z_coord, l: (np.sin(x_coord / l) * np.sin(y_coord / l) * np.sin(z_coord / l) +
19                                              np.sin(x_coord / l) * np.cos(y_coord / l) * np.cos(z_coord / l) +
20                                              np.cos(x_coord / l) * np.sin(y_coord / l) * np.cos(z_coord / l) +
21                                              np.cos(x_coord / l) * np.cos(y_coord / l) * np.sin(z_coord / l))
22

```

Рисунок 17 – Функции, описывающие решетчатые структуры.

Затем нода принимает параметры длины ячейки решетки и толщины ее стенки. После чего происходит пересечение каждого вокселя исходной модели с решетчатой структурой, согласно правилу, которое задается математической формулой решетки. Функция построения решетки представлена на Рисунке 18.

```

274      # Функция создания решетчатой структуры внутри модели
275      1 usage
276      def make_structure(self):
277          self.pattern = self.set_pattern()
278          for x in range(self.dims[0]):
279              for y in range(self.dims[1]):
280                  for z in range(self.dims[2]):
281                      if self.voxel_array[x][y][z] == 0 \
282                          if abs(self.pattern(x, y, z, self.cell_length)) > self.cell_thickness else 1
283

```

Рисунок 18 – Функция построения решетки

Получив новый массив вокселей, нода преобразовывает его в полигональную сетку и записывает в STL-файл.

Результат. Результатом работы программы является полигональная модель, заполненная решетчатой структурой. Визуализация полигональной модели представлена на Рисунке 19.

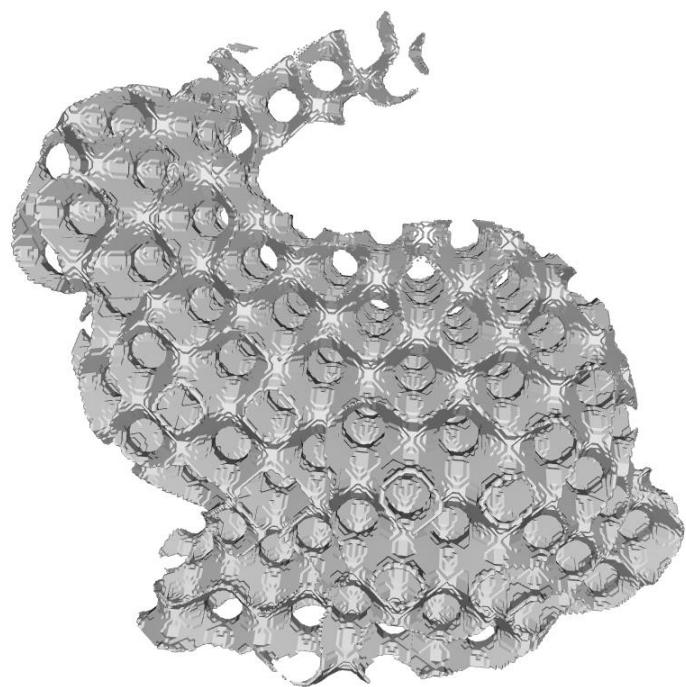


Рисунок 19 – Визуализация полигональной модели, заполненной гироидом

5 РАЗРАБОТКА ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПРОГРАММЫ ВОКСЕЛИЗАЦИИ

Для программы вокселизации был разработан графический интерфейс на основе библиотеки DearPyGUI (DPG).

DearPyGUI (DPG) – это инструментарий для создания графических приложений на языке Python. Эта библиотека была выбрана, потому что она позволяет строить и редактировать графы, содержащие ноды и соединения между атрибутами нод. Ноды представлены в виде окон, которые пользователь может свободно перемещать внутри главного окна приложения. Различные входные и выходные параметры нод содержатся в атрибутах. Атрибуты представлены в трех вариантах: ввод, вывод, статический.

Структура программы была частично изменена и адаптирована под особенности библиотеки DPG.

В программу был добавлен класс *Main_Window*, который содержит информацию о главном окне программы. В этом классе реализованы строка меню в верхней части окна и выпадающие меню со всплывающими подсказками, как показано на Рисунке 20. Все ноды создаются путем нажатия на название ноды в меню. Подсказки появляются на экране при наведении курсора на название ноды в меню.

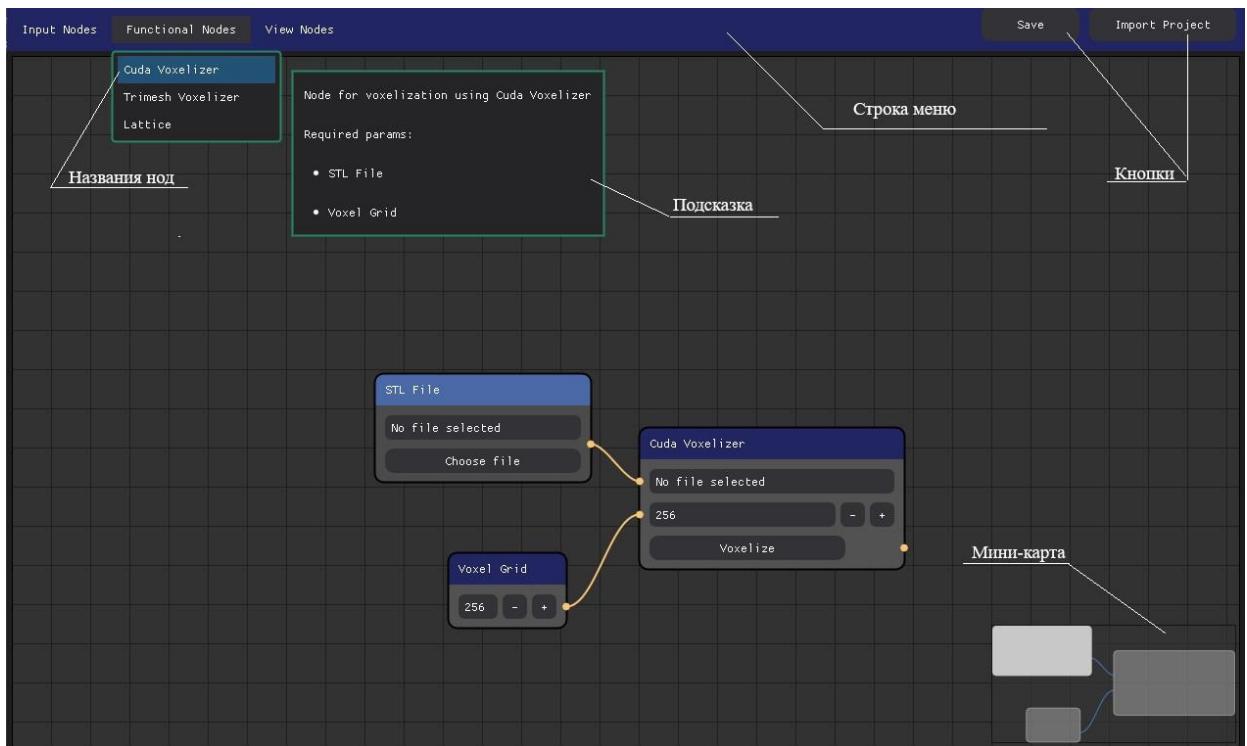


Рисунок 20 – Главное окно программы

Также, были реализованы кнопки сохранения и импорта проекта, они находятся в верхнем правом углу. Проект сохраняется в файле формата JSON. Для импорта проекта также необходимо загрузить JSON-файл, который содержит информацию о нодах, использованных в проекте, их расположении на главном экране, значениях, которые они хранят, и связях между ними. При нажатии на кнопку Import Project открывается окно выбора файла, как показано на Рисунке 21. В этом окне показаны только папки и JSON-файлы. Таким образом, выбрать файлы другого формата невозможно.

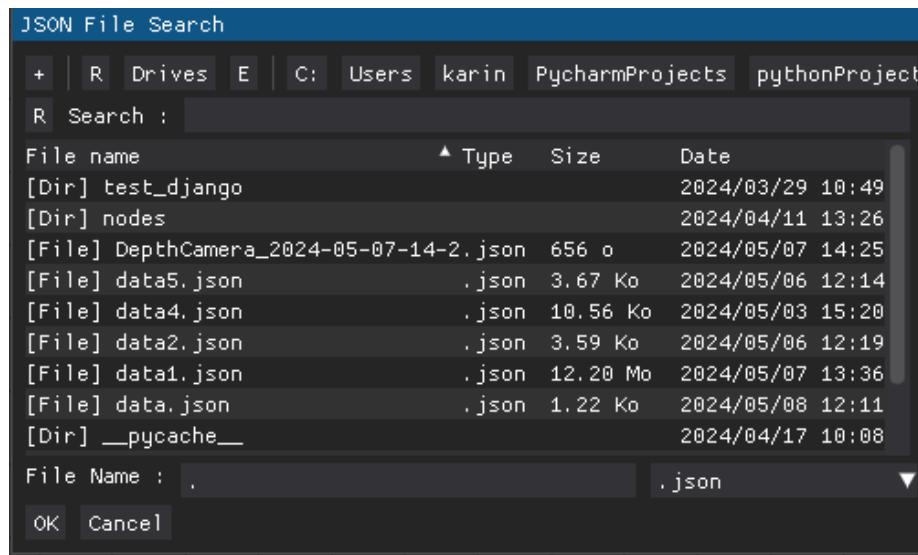


Рисунок 21 – Окно выбора файла

Кроме того, данный класс содержит информацию о теме и оформлении всех элементов программы. В правом нижнем углу показана интерактивная мини-карта проекта.

В процессе создания проекта программа отслеживает историю операций и сохраняет ее с помощью класса *History*. Это позволяет отменять последние несколько действий (с помощью стандартной комбинации клавиш Ctrl+Z) и возобновлять их (с помощью стандартной комбинации клавиш Ctrl+Shift+Z).

Перемещать ноды внутри главного окна проекта можно двумя способами. Первый способ – с помощью выделения конкретной ноды или нескольких нод и нажатия на кнопку D на клавиатуре. В таком случае нода переместится на 50 пикселей влево и вниз. Второй способ – с помощью зажатия левой кнопки мыши и перемещения ноды в нужное положение. Оба этих действия можно отменить и возобновить с помощью стандартных комбинаций клавиш, описанных выше.

Удалить ноды можно путем выделения ноды или нескольких нод и нажатием на клавишу Delete на клавиатуре.

Для создания связей между нодами необходимо навести курсор мыши на желтый кружок, который обозначает выходной параметр ноды. Затем

необходимо зажать левую кнопку мыши и вести желтую линию до желтого кружочка другой ноды, который символизирует входной параметр другой ноды.

Если две ноды соединены между собой и значение одной ноды изменяется, то это значение автоматически обновляется в другой ноде, никаких дополнительных действий не требуется.

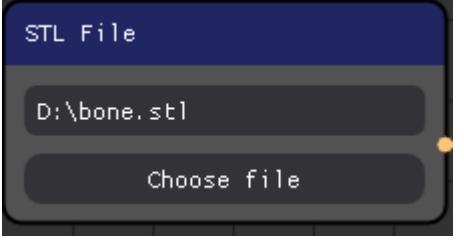
Стоит отметить, что выходить из желтой точки ноды (то есть из выходного параметра) может несколько соединений, однако входить в желтую точку ноды (то есть во входной параметр) может только одна линия.

Удалить связь между нодами можно двумя способами. Первый способ – путем выделения нужной линии или нескольких линий и нажатием кнопки Delete. Таким образом, если выделить некоторую область проекта левой кнопкой мыши и затем нажать на кнопку Delete, будут удалены все ноды и соединения между ними. Также, можно удалить одну конкретную связь. Для этого необходимо навести курсор мыши на нужную линию, после чего нажать на кнопку Ctrl и на левую кнопку мыши.

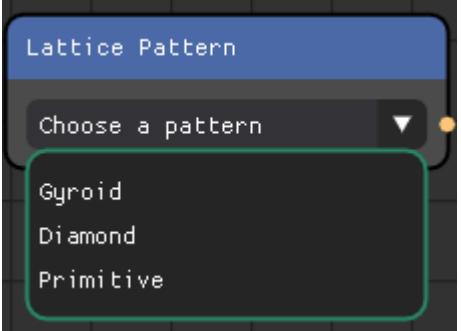
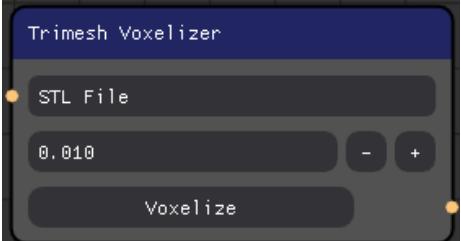
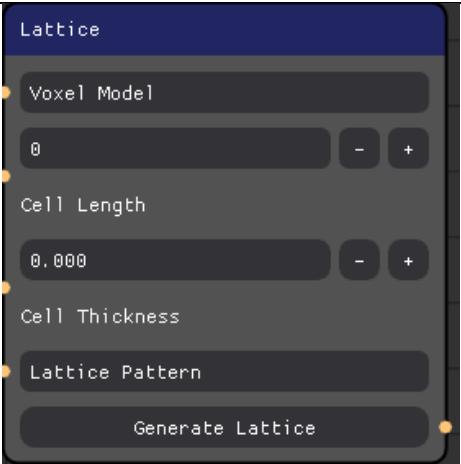
Удаление нод и связей между ними можно также отменить с помощью комбинации клавиш Ctrl + Z.

Список всех нод, реализованных в программе, представлен в Таблице 1. Полный текст программы представлен в Приложении 1.

Таблица 1 – Ноды, реализованные в программе

Input Nodes	STL File	Хранит информацию о расположении STL-файла. Имя файла можно ввести вручную или выбрать с помощью окна выбора	
----------------	----------	--	---

		файла (при нажатии на кнопку Choose file)	
Input Nodes	Voxel Grid	Хранит значение размера воксельной сетки (натуральное число). Значение по умолчанию – 256. Число можно ввести вручную или с помощью кнопок +/– (± 10)	
	Cell Thickness	Хранит значение толщины стенки ячейки решетчатой структуры. Десятичная дробь. Значение по умолчанию – 0,2. Число можно ввести вручную или с помощью кнопок +/– ($\pm 0,1$)	
	Cell Length	Хранит значение длины ячейки решетчатой структуры. Натуральное число от 0 до 50. Значение по умолчанию – 5. Число можно ввести вручную или с	

		помощью кнопок $+/-$ (± 1)	
	Lattice Pattern	Хранит название шаблона решетчатой структуры. Выбрать шаблон можно с помощью выпадающего списка.	
Functional Nodes	Cuda Voxelizer	Вокселизирует модель с помощью Cuda Voxelizer'a. Вокселизация происходит после нажатия на кнопку Voxelize	
	Trimesh Voxelizer	Вокселизирует модель с помощью библиотеки Trimesh. Вокселизация происходит после нажатия на кнопку Voxelize	
	Lattice	Заполняет воксельную модель решетчатой структурой. Заполнение происходит после нажатия на кнопку Generate Lattice	

View Nodes	View STL	<p>Визуализирует STL-файл. Визуализация происходит после нажатия на кнопку View.</p>	
---------------	----------	--	---

Из Таблицы 1 видно, что ноды и их функционал был сохранен тем же, как и описано в пункте 4.

Пример проекта, созданный в окне программы, представлен на Рисунке 22.

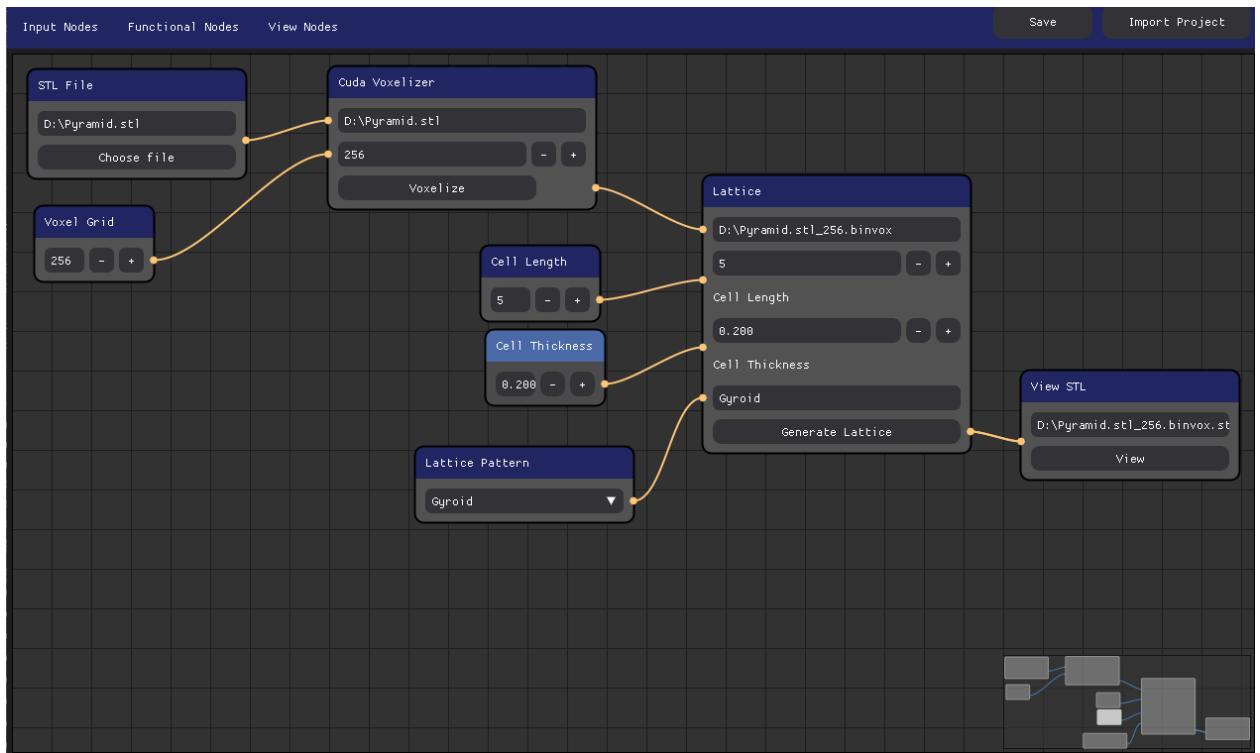


Рисунок 22 – Пример проекта

Результат, полученный после выполнение данного проекта, представлен на Рисунке 23.

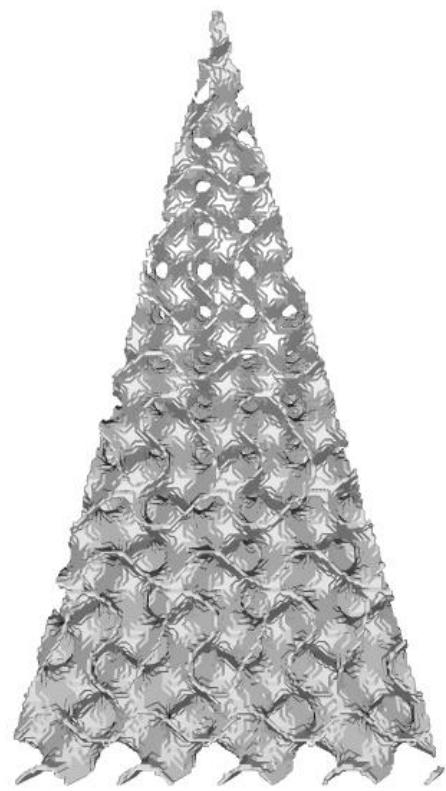


Рисунок 23 – Результат выполнения проекта

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы был реализован алгоритм вокселизации 3D-объектов и заполнения трехмерной модели решетчатой структурой. Для разработанного алгоритма был создан графический интерфейс, позволяющий пользователю, не обладающему навыками программирования, создавать собственные проекты с индивидуальными параметрами. Для удобства пользователя интерфейс адаптирован под использование стандартных комбинаций клавиш.

Данный алгоритм использует оптимизированные библиотеки и инструменты вокселизации, что позволяет вокселизировать модели, загружать и читать файлы с отвечающей требованиям скоростью.

Область применения результатов включает в себя предприятия, использующие технологии селективного лазерного плавления для производства изделий. Изделия, модели которых были созданы с помощью данного алгоритма и произведенны с помощью аддитивных технологий, могут быть использованы в различных областях, например, в медицине, электронике и аэрокосмической отрасли.

Важным направлением развития данной работы является создание медицинских имплантов. Планируется, что разработанный алгоритм будет использоваться для проектирования индивидуальных имплантов из металла. Важным аспектом подобной программы является удобство и простота ее использования, которая позволяет пользователю без инженерного или технического образования легко создавать трехмерные модели.

Кроме того, необходимо реализовать более сложные операции заполнения модели решетчатой структурой: возможность выбора определенной области заполнения, неравномерное заполнение.

Таким образом, алгоритм, разработанный в ходе выпускной квалификационной работы, имеет потенциал развития в области аддитивного производства индивидуальных имплантов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Белая книга. Развитие отдельных высокотехнологичных направлений / А. М. Абакумов, В. Е. Авербах, В. Р. Анпилогов [и др.]; Национальный исследовательский университет «Высшая школа экономики». – Москва: Изд-во Национального исследовательского университета «Высшая школа экономики», 2022. – 187 с.
2. Проектирование автоматизированных станков и комплексов: учебник в 2 томах / В. М. Утенков, Г. Н. Васильев, Б. М. Дмитриев [и др.]; под редакцией П. М. Чернянского. — Москва: МГТУ им. Баумана, [б. г.]. — Том 2 — 2014. — 303 с. — ISBN 978-5-7038-3811-2. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/106452> (дата обращения: 16.02.2024). — Режим доступа: для авториз. пользователей.
3. Bacciaglia, Antonio, Alessandro Ceruti, and Alfredo Liverani. "A systematic review of voxelization method in additive manufacturing." *Mechanics & Industry* 20.6 (2019): 630.
4. Основы автоматизации технологических процессов и производств: учебное пособие: в 2 томах / под редакцией Г. Б. Евгенева. — Москва: МГТУ им. Баумана, 2015 — Том 1: Информационные модели — 2015. — 441 с. — ISBN 978-5-7038-4138-9. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/106342> (дата обращения: 19.02.2024). — Режим доступа: для авториз. пользователей.
5. Лазерные аддитивные технологии в машиностроении: учебное пособие / А. Г. Григорьянц, И. Н. Шиганов, А. И. Мисюров, Р. С. Третьяков; под редакцией А. Г. Григорьянца. — Москва: МГТУ им. Баумана, 2018. — 278 с. — ISBN 978-5-7038-4976-7. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/172807> (дата обращения: 19.02.2024). — Режим доступа: для авториз. пользователей.
6. Leong, K. F., C. K. Chua, and Y. M. Ng. "A study of stereolithography file errors and repair. Part 1. Generic solution." *The International Journal of Advanced Manufacturing Technology* 12 (1996): 407–414.
7. ГОСТ Р 57591–2017. Аддитивные технологические процессы. Базовые принципы: утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии № 847-ст от 9 августа 2017 г. – URL: <http://gost.gtsever.ru/Data/649/64967.pdf> (дата обращения: 24.02.2024).

8. Szilvsi-Nagy, Mrta, and G. Y. Matyasi. "Analysis of STL files." *Mathematical and computer modelling* 38.7-9 (2003): 945–960.
9. Elmar Eisemann, Xavier D'ecoret. Single-pass GPU Solid Voxelization and Applications. GI '08: Proceedings of Graphics Interface 2008, May 2008, Windsor, ONT, Canada. Canadian Information Processing Society, 322, pp.73-80, 2008.
10. Yan C. et al. Evaluations of cellular lattice structures manufactured using selective laser melting //International Journal of Machine Tools and Manufacture. – 2012. – T. 62. – C. 32–38.
11. Mahshid R., Hansen H. N., Højbjørre K. L. Strength analysis and modeling of cellular lattice structures manufactured using selective laser melting for tooling applications //Materials & Design. – 2016. – T. 104. – C. 276–283.
12. Aremu A. O. et al. A voxel-based method of constructing and skinning conformal and functionally graded lattice structures suitable for additive manufacturing //Additive Manufacturing. – 2017. – T. 13. – C. 1–13.
13. Hao L. et al. Design and additive manufacturing of cellular lattice structures //The International Conference on Advanced Research in Virtual and Rapid Prototyping (VRAP). Taylor & Francis Group, Leiria. – 2011. – C. 249–254.
14. Репозиторий «Cuda Voxelizer» [Электронный ресурс] – URL: https://github.com/Forceflow/cuda_voxelizer (дата обращения: 05.04.2024). – Режим доступа: свободный.
15. Schwarz M., Seidel H. P. Fast parallel surface and solid voxelization on GPUs //ACM transactions on graphics (TOG). – 2010. – T. 29. – №. 6. – C. 1-10.
16. Lorensen W. E., Cline H. E. Marching cubes: A high resolution 3D surface construction algorithm //Seminal graphics: pioneering efforts that shaped the field. – 1998. – C. 347-353.

ПРИЛОЖЕНИЕ 1

Текст программы

```
1 import os
2
3 import dearpygui.dearpygui as dpg
4 import open3d as o3d
5 from time import time, sleep
6 import numpy as np
7 import json
8 import binvox_file
9 import trimesh
10
11 dpg.create_context()
12
13 # Функции, описывающие решетчатую структуру
14 # Гироид
15 func_gyroid = lambda x_coord, y_coord, z_coord, l: (
16     np.sin(x_coord / l) * np.cos(y_coord / l) +
17     np.sin(y_coord / l) * np.cos(z_coord / l) +
18     np.sin(z_coord / l) * np.cos(x_coord / l))
19 # Поверхность Шварца P
20 func_primitive = lambda x_coord, y_coord, z_coord, l:
21     np.cos(x_coord / l) + np.cos(y_coord / l) + np.cos(
22         z_coord / l)
23
24 # Поверхность Шварца D
25 func_diamond = lambda x_coord, y_coord, z_coord, l: (
26     np.sin(x_coord / l) * np.sin(y_coord / l) * np.sin(
27         z_coord / l) +
28     np.sin(x_coord / l) * np.cos(y_coord / l) * np.cos(
29         z_coord / l) +
30     np.cos(x_coord / l) * np.sin(y_coord / l) * np.sin(
31         z_coord / l))
```

```

32         self.sender = sender
33         self.links = None
34         self.target = app_data
35         self.from_child = None
36         self.to_child = None
37         self.status = True
38
39     def redo(self):
40         self.links = dpg.add_node_link(self.target[0],
41             self.target[1], parent=self.sender)
42         # print(self.links, self.target)
43         self.from_child = dpg.get_item_children(item=
44             self.target[0], slot=1)[0]
45         self.to_child = dpg.get_item_children(item=
46             self.target[1], slot=1)[0]
47         dpg.set_value(item=self.to_child, value=dpg.
48             get_value(self.from_child))
49         Create_Connection.instances.append(self)
50         # print(Create_Connection.instances)
51
52     def undo(self):
53         dpg.delete_item(self.links)
54
55     @classmethod
56     def get_connection_by_attr(cls, target):
57         return [inst for inst in cls.instances if
58             inst.target[0] == target]
59
60 class Delete_Connection:
61     def __init__(self, sender, app_data):
62         self.sender = sender
63         self.links = app_data
64         self.target = None
65         self.selected_links = dpg.get_selected_links(
66             node_editor=self.sender)
67
68     def redo(self):
69         # print(self.selected_links)

```

```

69         if not self.selected_links:
70             for item in history.actions:
71                 if item.__class__ ==
72                     Create_Connection:
73                         if item.links == self.links:
74                             item.status = False
75                             dpg.hide_item(self.links)
76                             # dpg.configure_item(item=
77                             self.links, enabled=False)
78                         else:
79                             for link in self.selected_links:
80                                 # print(self.selected_links)
81                                 for item in history.actions:
82                                     if item.__class__ ==
83                                         Create_Connection:
84                                             if item.links == link:
85                                                 item.status = False
86                                                 dpg.hide_item(link)
87
88         def undo(self):
89             if not self.selected_links:
90                 dpg.configure_item(item=self.links, show
91 =True)
92                 for item in history.actions:
93                     if item.__class__ ==
94                         Create_Connection:
95                             if item.links == self.links:
96                                 item.status = True
97                             else:
98                                 for link in self.selected_links:
99                                     for item in history.actions:
100                                         if item.__class__ ==
101                                             Create_Connection:
102                                                 if item.links == link:
103                                                     item.status = True
104                                                     dpg.configure_item(item=
105                                                     link, show=True)
106
107     class History:
108         actions = [] # [action object]
109         nodes = [] # [node object]
110         index = 0
111         n_index = 0

```

```

106     out = {'nodes': [], 'links': []}
107
108     def undo(self):
109         if self.index > 0:
110             self.actions[self.index - 1].undo()
111             self.index -= 1
112
113     def redo(self):
114         if self.index < len(self.actions):
115             if self.index > 30:
116                 self.actions.pop(0)
117             self.actions[self.index].redo()
118             self.index += 1
119
120     def add(self, a):
121
122         if self.index >= 30:
123             self.actions.pop(0)
124             self.index -= 1
125         self.actions.append(a)
126         self.index += 1
127         if a.__class__ == Create_Node:
128             self.nodes.append(a.node_object)
129
130     def to_json(self):
131         out_1 = {'nodes': [], 'links': []}
132         for i in range(len(self.actions)):
133             item = {}
134
135             if self.actions[i].__class__ ==
136                 Create_Node:
137                     if self.actions[i].node_object.
138                         status:
139                             item['fields'] = self.actions[i].
140                             .node_object.__dict__
141                             try:
142                                 item['fields']['value'] =
143                                     dpg.get_value(self.actions[i].node_object.out)
144                                 except AttributeError:
145                                     item['fields']['value'] =
146                                         dpg.get_value(self.actions[i].node_object.inp_1)
147                                         item['position'] = dpg.
148                                         get_item_pos(self.actions[i].node_id)
149                                         self.out['nodes'].append(item)
150

```

```

144                     out_1[ 'nodes' ].append(item)
145             elif self.actions[i].__class__ ==
146                 Create_Connection and self.actions[i].status:
147                     item[ 'fields' ] = self.actions[i].
148                         __dict__
149                     self.out[ 'links' ].append(item)
150                     out_1[ 'links' ].append(item)
151                     out_2 = json.dumps(out_1, indent=4)
152                     with open('data.json', "w") as f:
153                         f.truncate()
154                         f.write(out_2)
155
156
157
158
159
160
161     def from_json(self, file):
162         with open(file, 'r') as f:
163             data = json.load(f)
164             for i in data.keys():
165                 self.out[i] = data[i]
166             self.recover_project_from_json()
167
168
169     def recover_project_from_json(self):
170         link_targets = []
171
172         for link in self.out[ 'links' ]:
173             link_fields = link[ 'fields' ]
174             link_targets += link_fields[ 'target' ]
175             new_targets = [ 0 ] * len(link_targets)
176
177             for item in self.out[ 'nodes' ]:
178                 item_fields = item[ 'fields' ]
179                 node_id = item_fields[ 'node_id' ]
180                 label = item_fields[ 'label' ]
181                 position = item[ 'position' ]
182                 value = item_fields[ 'value' ]
183
184                 node = Create_Node(label, node_id)
185                 node.node_object.value = value
186                 history.add(node)
187
188                 dpg.set_item_pos(item=node.node_id, pos=
189                                 position)
190                 children = dpg.get_item_children(item=
191                                 node.node_id, slot=1)
192
193                 if len(children) == 1:

```

```

184             grandchild = dpg.get_item_children(
185                 item=children[0], slot=1)[0]
186             dpg.set_value(item=grandchild, value
187 =value)
188         else:
189             grandchild = dpg.get_item_children(
190                 item=children[-1], slot=1)[-2]
191             dpg.set_value(item=grandchild, value
192 =value)
193
194         key_list = list(item_fields.keys())
195         val_list = list(item_fields.values())
196
197         for i in range(len(link_targets)):
198             try:
199                 attr_position = val_list.index(
200                     link_targets[i])
201                 new_targets[i] = node.
202                 node_object.__dict__[key_list[attr_position]]
203             except ValueError:
204                 pass
205             for i in range(0, len(new_targets) - 1, 2):
206                 try:
207                     con = Create_Connection(sender='
208                     node_editor', app_data=[new_targets[i], new_targets[
209                     i + 1]])
210                     con.redo()
211                 except Exception:
212                     pass
213                 history.add(con)
214
215
216
217     class Move_Node:
218         def __init__(self, dx=50, dy=50):
219             self.selected_nodes = dpg.get_selected_nodes
220             (node_editor='node_editor')
221             self.dx = dx
222             self.dy = dy
223             self.current_position = [15, 15]
224             self.redo()
225
226
227         def redo(self):
228             for node in self.selected_nodes:

```

```

219         current_position = dpg.get_item_pos(node
20      )
220         current_position[0] += self.dx
221         current_position[1] += self.dy
222         self.current_position = current_position
223         dpg.set_item_pos(item=node, pos=
224           current_position)
225     def undo(self):
226       for node in self.selected_nodes:
227         current_position = dpg.get_item_pos(node
20      )
228         current_position[0] -= self.dx
229         current_position[1] -= self.dy
230         dpg.set_item_pos(item=node, pos=
231           current_position)
232
233 class Drop_Node:
234   def __init__(self, node_id):
235     self.node_id = node_id
236     self.current_position = None
237     self.previous_position = None
238     self.redo()
239
240   def redo(self):
241     for item in history.nodes:
242       if item.node_id == self.node_id:
243         if self.current_position == dpg.
244           get_item_pos(self.node_id):
245             self.current_position = self.
246             previous_position
247             self.previous_position = dpg.
248             get_item_pos(self.node_id)
249             else:
250               self.current_position = dpg.
251               get_item_pos(self.node_id)
252               self.previous_position = item.
253                 position
254                 item.position = self.
255                 current_position
256                 dpg.set_item_pos(item=self.
257                   node_id, pos=self.current_position)
258

```

```

252     def undo(self):
253         for item in history.nodes:
254             if item.node_id == self.node_id:
255                 pos = dpg.get_item_pos(self.node_id)
256                 self.current_position = self.
257                     previous_position
258                     self.previous_position = pos
259                     item.position = self.
260                     current_position
261                     dpg.set_item_pos(item=self.node_id,
262                                     pos=self.current_position)
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285

```

```

286         elif self.label == 'Cell Length':
287             self.node_object = Node_Cell_Length(
288                 node_id=self.node_id, label=self.label)
289         elif self.label == 'Lattice Pattern':
290             self.node_object = Node_Lattice_Pattern(
291                 node_id=self.node_id, label=self.label)
292         elif self.label == 'Lattice':
293             self.node_object = Node_Lattice(node_id=
294                 self.node_id, label=self.label)
295
296     def undo(self):
297         try:
298             dpg.hide_item(item=self.node_id)
299             self.node_object.status = False
300         except SystemError:
301             pass
302
303 class Delete_Node:
304     def __init__(self):
305         self.selected_nodes = dpg.get_selected_nodes
306         (node_editor='node_editor')
307         self.redo()
308
309     def redo(self):
310         for node in self.selected_nodes:
311             for item in history.nodes:
312                 if item.node_id == node:
313                     item.status = False
314                     dpg.hide_item(node)
315
316     def undo(self):
317         for node in self.selected_nodes:
318             for item in history.nodes:
319                 if item.node_id == node:
320                     item.status = True
321                     dpg.configure_item(item=node,
322                         pos=dpg.get_item_pos(node), show=True)
323
324 class Node_Cell_Thickness:
325     def __init__(self, node_id, label):
326         self.node_id = node_id
327         self.position = [15, 15]

```

```

325         self.label = label
326         self.attr_1 = None
327         self.out = None
328         self.value = 0.2
329         self.status = True
330         self.create_node()
331
332     def create_node(self):
333         node = dpg.add_node(label=self.label,
334                             tag=self.node_id,
335                             pos=self.position,
336                             parent='node_editor',
337                             draggable=True,
338                             drag_callback=
339                             drag_n_drop,
340                             )
341
342         self.attr_1 = dpg.add_node_attribute(
343             attribute_type=dpg.mvNode_Attr_Output,
344             parent=
345             node)
346
347         self.out = dpg.add_input_float(default_value
348             =self.value,
349             tag=dpg.
350             generate_uuid(),
351             max_value=1,
352             min_value=0,
353             min_clamped=
354             True,
355             max_clamped=
356             True,
357             width=100,
358             callback=self
359             .update,
360             parent=self.
361             attr_1)
362
363     def update(self, *args):
364         current_value = dpg.get_value(item=self.out)
365         if current_value != self.value:
366             self.value = current_value
367             connections = Create_Connection.
368             get_connection_by_attr(self.attr_1)
369             for con in connections:

```

```

359             dpg.set_value(item=con.to_child,
      value=self.value)
360
361
362 class Node_STL_File:
363     def __init__(self, node_id, label):
364         self.node_id = node_id
365         self.position = [15, 15]
366         self.label = label
367         self.attr_1 = None
368         self.out = None
369         self.value = 'No file selected'
370         self.status = True
371         self.create_node()
372
373     def create_node(self):
374
375         node = dpg.add_node(label=self.label,
376                             tag=self.node_id,
377                             pos=self.position,
378                             parent='node_editor',
379                             draggable=True,
380                             )
381
382         self.attr_1 = dpg.add_node_attribute(label='
383             STL File -> STL File',
384             attribute_type=dpg.mvNode_Attr_Output,
385             parent=
386             node)
387
388         self.out = dpg.add_input_text(default_value=
389             self.value,
390             width=200,
391             parent=self.
392             attr_1,
393             tag=dpg.
394             generate_uuid(),
395             )
396         button = dpg.add_button(label='Choose file'
397             , tag=dpg.generate_uuid(), parent=self.attr_1,
398             width=200, callback=
399             self.create_file_dialog, user_data=self.out)
400

```

```

394     def create_file_dialog(self, sender, app_data,
395         user_data):
395         file_dialog = dpg.add_file_dialog(label='STL
396             File Search', tag=dpg.generate_uuid(),
396                 width=500
397                 , height=300, callback=self.set_value, user_data=
397                     user_data)
397             extension = dpg.add_file_extension(label='
398                 STL File Extension', extension='.stl',
398                     tag=dpg.
399             generate_uuid(), parent=file_dialog)
399
400     def set_value(self, sender, app_data, user_data
400 ):
401         dpg.set_value(item=user_data, value=app_data
401 ['file_path_name'])
402         self.update()
403
404     def update(self):
405         current_value = dpg.get_value(item=self.out)
406         if current_value != self.value:
407             self.value = current_value
408             connections = Create_Connection.
408 get_connection_by_attr(self.attr_1)
409             for con in connections:
410                 if dpg.get_item_configuration(con.
410 links)[ "show" ]:
411                     dpg.set_value(item=con.to_child
411 , value=self.value)
412
413
414 class Node_View_STL:
415     def __init__(self, node_id, label):
416         self.node_id = node_id
417         self.position = [15, 15]
418         self.label = label
419         self.attr_1 = None
420         self.inp_1 = None
421         self.value = 'STL File Path'
422         self.status = True
423         self.create_node()
424
425     def create_node(self):
426         node = dpg.add_node(label=self.label,

```

```

427                         tag=self.node_id,
428                         pos=self.position,
429                         parent='node_editor',
430                         draggable=True,
431                         )
432             self.attr_1 = dpg.add_node_attribute(
433                 attribute_type=dpg.mvNode_Attr_Input,
434                         parent=
435                         node,
436                         tag=dpg
437                         .generate_uuid())
438             self.inp_1 = dpg.add_input_text(
439                 default_value=self.value, tag=dpg.generate_uuid(),
440                         parent=self.
441                         attr_1, width=200)
442             button = dpg.add_button(label='View', tag=
443                         dpg.generate_uuid(), parent=self.attr_1,
444                         width=200, callback=
445                         self.view, user_data=self.inp_1)
446
447 class Node_Trimesh_Voxelizer:
448     def __init__(self, node_id, label):
449         self.node_id = node_id
450         self.label = label
451         self.position = [15, 15]
452         self.attr_1 = None
453         self.attr_2 = None
454         self.attr_3 = None
455         self.inp_1 = None
456         self.inp_2 = None
457         self.out = None
458         self.value = None
459         self.status = True
460         self.create_node()
461

```

```

462     def create_node(self):
463         node = dpg.add_node(label=self.label,
464                             tag=self.node_id,
465                             pos=self.position,
466                             parent='node_editor',
467                             draggable=True,
468                             )
469         self.attr_1 = dpg.add_node_attribute(label='
470             Trimesh Voxelizer -> STL File',
471             attribute_type=dpg.mvNode_Attr_Input,
472             parent=
473             node,
474             tag=dpg
475             .generate_uuid())
476         self.attr_2 = dpg.add_node_attribute(label='
477             Trimesh Voxelizer -> Pitch',
478             attribute_type=dpg.mvNode_Attr_Static,
479             parent=
480             node,
481             tag=dpg
482             .generate_uuid())
483         self.attr_3 = dpg.add_node_attribute(label='
484             Voxel Grid instance',
485             attribute_type=dpg.mvNode_Attr_Output,
486             parent=
487             node,
488             tag=dpg
489             .generate_uuid())
490         self.inp_1 = dpg.add_input_text(
491             default_value='STL File',
492             tag=dpg.
493             generate_uuid(),
494             parent=self.
495             attr_1,
496             width=250,
497             readonly=
498             True)
499         self.inp_2 = dpg.add_input_float(
500             default_value=0.01,
501             tag=dpg.
502             generate_uuid(),
503             width=250,
504             readonly=True)

```

```

488                               parent=self
489                               .attr_2,
490                               width=250,
491                               readonly=
492                               False)
493                               self.out = dpg.add_text(default_value=' ',
494                               tag=dpg.generate_uuid(),
495                               parent=self.attr_3,
496                               show=False)
497                               button = dpg.add_button(label='Voxelize',
498                               tag=dpg.generate_uuid(), parent=self.attr_3,
499                               width=200, callback=
500                               self.voxelize)
501
502                               def voxelize(self):
503                               file_path = dpg.get_value(self.inp_1)
504                               pitch = dpg.get_value(self.inp_2)
505                               mesh = trimesh.exchange.load.load_mesh(
506                               file_obj=file_path, file_type='stl')
507                               voxel_array = trimesh.voxel.creation.
508                               voxelize(mesh=mesh, pitch=pitch,
509 )
510                               voxel = trimesh.voxel.morphology.fill(
511                               encoding=voxel_array.matrix, method='orthographic')
512                               # print(voxel.dense)
513                               val = json.dumps(voxel.dense.tolist())
514                               with open('data1.json', "w") as f:
515                                   f.truncate()
516                                   f.write(val)
517                               self.value = 'data1.json'
518                               dpg.set_value(item=self.out, value=self.
519                               value)
520
521
522 class Node_Cuda_Voxelizer:
523     def __init__(self, node_id, label):
524         self.node_id = node_id
525         self.label = label
526         self.position = [15, 15]
527         self.attr_1 = None
528         self.attr_2 = None
529         self.attr_3 = None
530         self.inp_1 = None

```

```

521         self.inp_2 = None
522         self.out = None
523         self.value = None
524         self.status = True
525         self.create_node()
526
527     def create_node(self):
528         node = dpg.add_node(label=self.label,
529                             tag=self.node_id,
530                             pos=self.position,
531                             parent='node_editor',
532                             draggable=True,
533                             )
534         self.attr_1 = dpg.add_node_attribute(label='
535             Cuda Voxelizer -> STL File',
536             attribute_type=dpg.mvNode_Attr_Input,
537             parent=
538             node,
539             tag=dpg
540             .generate_uuid())
541         self.attr_2 = dpg.add_node_attribute(label='
542             Cuda Voxelizer -> Voxel Grid',
543             attribute_type=dpg.mvNode_Attr_Input,
544             parent=
545             node,
546             tag=dpg
547             .generate_uuid())
548         self.inp_1 = dpg.add_input_text(
549             default_value='STL File',
550             generate_uuid(),
551             parent=self.
552             attr_1,
553             width=250,

```

```

550                                         readonly=
551             True)
551             self.inp_2 = dpg.add_input_int(default_value
551             =0,
552                                         tag=dpg.
552             generate_uuid(),
553                                         parent=self.
553             attr_2,
554                                         width=250,
555                                         readonly=True
555             )
556             self.out = dpg.add_text(default_value='
556             Binvox File Path', tag=dpg.generate_uuid(),
557                                         parent=self.attr_3,
557             show=False)
558             button = dpg.add_button(label='Voxelize',
558             tag=dpg.generate_uuid(), parent=self.attr_3,
559                                         width=200, callback=
559             self.voxelize, user_data=[self.inp_1, self.inp_2,
559             self.out])
560
561     def voxelize(self, *args):
562         file_path = dpg.get_value(item=self.inp_1)
563         voxel_grid = dpg.get_value(item=self.inp_2)
564         out = self.out
565         command = f'cuda_voxelizer -f {file_path} -s
565             {voxel_grid} -o binvox -solid'
566         os.system(command)
567         dpg.set_value(item=out, value=f'{file_path}_
567             {voxel_grid}.binvox')
568         self.update()
569
570     def update(self):
571         current_value = dpg.get_value(item=self.out)
572         if current_value != self.value:
573             self.value = current_value
574             connections = Create_Connection.
574             get_connection_by_attr(self.attr_3)
575             for con in connections:
576                 dpg.set_value(item=con.to_child,
576                 value=self.value)
577
578
579 class Node_Voxel_Grid:

```

```

580     def __init__(self, node_id, label):
581         self.node_id = node_id
582         self.label = label
583         self.position = [15, 15]
584         self.attr_1 = None
585         self.out = None
586         self.value = 256
587         self.status = True
588         self.create_node()
589
590     def create_node(self):
591         node = dpg.add_node(label=self.label,
592                             tag=self.node_id,
593                             pos=self.position,
594                             parent='node_editor',
595                             draggable=True,
596                             )
597         self.attr_1 = dpg.add_node_attribute(label='
598             Voxel Grid -> Voxel Grid',
599             attribute_type=dpg.mvNode_Attr_Output,
600             parent=
601             node)
602         self.out = dpg.add_input_int(default_value=
603             self.value,
604             min_value=0,
605             max_value=4096,
606             min_clamped=
607             True,
608             max_clamped=
609             True,
610             width=100,
611             step=10,
612             step_fast=100,
613             parent=self.
614             attr_1,
615             update,
616             generate_uuid(),
617             callback=self.
618             tag=dpg.
619             )
620
621     def update(self, *args):
622         current_value = dpg.get_value(item=self.out)

```

```

615         if current_value != self.value:
616             self.value = current_value
617             connections = Create_Connection.
618                 get_connection_by_attr(self.attr_1)
619                     for con in connections:
620                         dpg.set_value(item=con.to_child,
621                         value=self.value)
620
621
622 class Node_Cell_Length:
623     def __init__(self, node_id, label):
624         self.node_id = node_id
625         self.label = label
626         self.position = [15, 15]
627         self.attr_1 = None
628         self.out = None
629         self.value = 5
630         self.status = True
631         self.create_node()
632
633     def create_node(self):
634         # print(sender)
635         node = dpg.add_node(label=self.label,
636                             tag=self.node_id,
637                             pos=self.position,
638                             parent='node_editor',
639                             draggable=True,
640                             )
641         self.attr_1 = dpg.add_node_attribute(label='
Cell Length -> Length',
642                                         attribute_type=dpg.mvNode_Attr_Output,
643                                         parent=
node,
644                                         tag=dpg
.generate_uuid())
645         self.out = dpg.add_input_int(default_value=5
,
646                                         min_value=0,
647                                         max_value=50,
648                                         min_clamped=
True,
649                                         max_clamped=
True,

```

```

650                                     width=100,
651                                     step=1,
652                                     step_fast=10,
653                                     parent=self.
654                                     attr_1,
655                                     update,
656                                     generate_uuid(),
657                                     )
658     def update(self, *args):
659         current_value = dpg.get_value(item=self.out)
660         if current_value != self.value:
661             self.value = current_value
662             connections = Create_Connection.
663             get_connection_by_attr(self.attr_1)
664             for con in connections:
665                 dpg.set_value(item=con.to_child,
666                               value=self.value)
667 class Node_Lattice_Pattern:
668     def __init__(self, node_id, label):
669         self.node_id = node_id
670         self.label = label
671         self.position = [15, 15]
672         self.attr_1 = None
673         self.out = None
674         self.value = 'Choose a pattern'
675         self.status = True
676         self.create_node()
677
678     def create_node(self):
679         node = dpg.add_node(label=self.label,
680                             tag=self.node_id,
681                             pos=self.position,
682                             parent='node_editor',
683                             draggable=True,
684                             )
685         self.attr_1 = dpg.add_node_attribute(label='
686 Lattice Pattern -> Pattern',
687                                     attribute_type=dpg.mvNode_Attr_Output,

```

```

687                                     parent=
688             node,
689             .generate_uuid())
690             self.out = dpg.add_combo(items=[ 'Gyroid', '
691                                         Diamond', 'Primitive'],
692                                         default_value=self.
693                                         value,
694                                         width=200,
695                                         parent=self.attr_1,
696                                         callback=self.
697                                         update,
698                                         tag=dpg.
699                                         generate_uuid())
700
701     def update(self, *args):
702         current_value = dpg.get_value(item=self.out)
703         if current_value != self.value:
704             self.value = current_value
705             connections = Create_Connection.
706             get_connection_by_attr(self.attr_1)
707             for con in connections:
708                 dpg.set_value(item=con.to_child,
709                               value=self.value)
710
711     class Node_Lattice:
712         def __init__(self, node_id, label):
713             self.node_id = node_id
714             self.label = label
715             self.position = [15, 15]
716             self.pattern_func = None
717             self.cell_thickness = None
718             self.cell_length = None
719             self.voxel_array = []
720             self.dims = []
721             self.output_file = None
722             self.attr_1 = None
723             self.attr_2 = None
724             self.attr_3 = None
725             self.attr_4 = None
726             self.attr_5 = None
727             self.inp_1 = None

```

```

723         self.inp_2 = None
724         self.inp_3 = None
725         self.inp_4 = None
726         self.out = None
727         self.value = 'STL File Path'
728         self.status = True
729         self.create_node()
730
731     def create_node(self):
732         node = dpg.add_node(label=self.label,
733                             tag=self.node_id,
734                             pos=self.position,
735                             parent='node_editor',
736                             draggable=True
737                             )
738         self.attr_1 = dpg.add_node_attribute(label='
739             Lattice -> Voxel Model',
740             attribute_type=dpg.mvNode_Attr_Input,
741             parent=
742             node,
743             tag=dpg
744             .generate_uuid())
745         self.attr_2 = dpg.add_node_attribute(label='
746             Lattice -> Cell Length',
747             attribute_type=dpg.mvNode_Attr_Input,
748             parent=
749             node,
750             tag=dpg
751             .generate_uuid())
752         self.attr_3 = dpg.add_node_attribute(label='

```

```

752 node,
753                                     tag=dpg
754         .generate_uuid())
755         self.attr_5 = dpg.add_node_attribute(label='
756             Lattice -> STL File',
757             attribute_type=dpg.mvNode_Attr_Output,
758             parent=
759             node,
760             tag=dpg
761             .generate_uuid())
762             self.inp_1 = dpg.add_input_text(
763                 default_value='Voxel Model',
764                 tag=dpg.
765                 generate_uuid(),
766                 parent=self.
767                 attr_1,
768                 width=250,
769                 readonly=
770                 True)
771             self.inp_2 = dpg.add_input_int(default_value
772                 =0,
773                 tag=dpg.
774                 generate_uuid(),
775                 parent=self.
776                 attr_2,
777                 width=250,
778                 readonly=True)
779             helping_text_1 = dpg.add_text(default_value=
780                 'Cell Length', parent=self.attr_2)
781             self.inp_3 = dpg.add_input_float(
782                 default_value=0,
783                 tag=dpg.
784                 generate_uuid(),
785                 parent=self.
786                 attr_3,
787                 width=250,
788                 readonly=
789                 True)
790             helping_text_2 = dpg.add_text(default_value=
791                 'Cell Thickness', parent=self.attr_3)
792
793
794
795
796

```

```

777         self.inp_4 = dpg.add_input_text(
778             default_value='Lattice Pattern',
779             generate_uuid(),
780             tag=dpg.
781             parent=self.
782             attr_4,
783             width=250,
784             readonly=
785             True)
786             self.out = dpg.add_text(default_value=self.
787             value, tag=dpg.generate_uuid(), parent=self.attr_5,
788             show=False)
789             button = dpg.add_button(label='Generate
790             Lattice', tag=dpg.generate_uuid(), parent=self.
791             attr_5,
792             width=250, callback=
793             self.lattice)
794
795     def lattice(self, *args):
796         file_path = dpg.get_value(item=self.inp_1)
797         self.cell_length = dpg.get_value(item=self.
798         inp_2)
799         self.cell_thickness = dpg.get_value(self.
800         inp_3)
801         pattern = dpg.get_value(item=self.inp_4)
802
803         self.set_pattern(pattern)
804
805         if file_path.split('.')[ -1] == 'json':
806             self.voxel_model_trimesh(file_path)
807             elif file_path.split('.')[ -1] == 'binvox':
808                 self.read_binvox(file_path)
809
810         self.make_structure()
811
812         self.write_to_stl(file_path)
813         dpg.set_value(item=self.out, value=f'{
814             file_path}.stl')
815         self.update()
816
817     def set_pattern(self, pattern):
818         if pattern == 'Gyroid':

```

```

809         self.pattern_func = func_gyroid
810     elif pattern == 'Diamond':
811         self.pattern_func = func_diamond
812     else:
813         self.pattern_func = func_primitive
814
815     def read_binvox(self, file_path):
816         with open(file_path, 'rb') as f:
817             model = binvox_file.read_as_3d_array(f)
818             # Сохранение всех параметров модели
819             self.voxel_array = model.data
820             self.dims = model.dims
821
822     def voxel_model_trimesh(self, file_path):
823         with open(file_path, "r") as f:
824             self.voxel_array = np.array(json.load(f),
825             dtype=float)
826             self.dims = np.shape(self.voxel_array)
827
828     def make_structure(self):
829         for x in range(self.dims[0]):
830             for y in range(self.dims[1]):
831                 for z in range(self.dims[2]):
832                     if self.voxel_array[x][y][z]:
833                         self.voxel_array[x][y][z] =
834                         0 \
835                         if abs(self.pattern_func
836 (x, y, z, self.cell_length)) > self.cell_thickness
837                         else 1
838
839     def write_to_stl(self, path):
840         print(self.voxel_array)
841         # Используется метод
842         # matrix_to_marching_cubes из библиотеки trimesh
843         mesh = trimesh.voxel.ops.
844         matrix_to_marching_cubes(
845             matrix=self.voxel_array,
846             pitch=.20)
847         print("Merging vertices closer than a pre-
848 set constant...")
849         mesh.merge_vertices()
850         print("Removing duplicate faces...")
851         mesh.update_faces(mesh.unique_faces())
852         # mesh.remove_duplicate_faces()

```

```

846         print("Scaling...")
847         mesh.apply_scale(scaling=1.0)
848         print("Making the mesh watertight...")
849         trimesh.repair.fill_holes(mesh)
850         print("Fixing inversion and winding...")
851         trimesh.repair.fix_inversion(mesh)
852         trimesh.repair.fix_winding(mesh)
853
854     # Сохранение STL-файла
855     print("Generating the STL mesh file")
856     trimesh.exchange.export.export_mesh(
857         mesh=mesh,
858         file_obj=f'{path}.stl',
859
860         file_type="stl"
861     )
862
863     def update(self):
864         current_value = dpg.get_value(item=self.out)
865         if current_value != self.value:
866             self.value = current_value
867             connections = Create_Connection.
868             get_connection_by_attr(self.attr_5)
869             for con in connections:
870                 dpg.set_value(item=con.to_child,
871                               value=self.value)
872
873     """
874     class Node_Subtraction(Node_Base):
875         def __init__(self, node_id, label):
876             self.node_id = node_id
877             self.label = label
878             self.position = [15, 15]
879             self.attr_1 = None
880             self.inp_1 = None
881             self.attr_2 = None
882             self.inp_2 = None
883             self.attr_3 = None
884             self.out = None
885             self.value = 'Voxel Model'
886             self.status = True
887             self.create_node()

```

```

888     def create_node(self):
889         node = dpg.add_node(label=self.label,
890                             tag=self.node_id,
891                             pos=self.position,
892                             parent='node_editor',
893                             draggable=True,
894                             )
895         self.attr_1 = dpg.add_node_attribute(label='
896             Voxel Model 1',
897             attribute_type=dpg.mvNode_Attr_Input,
898             parent=
899             node,
900             tag=dpg
901             .generate_uuid())
902
903         self.inp_1 = dpg.add_input_text(
904             default_value=self.value,
905             width=250,
906             parent=self.
907             attr_1,
908             # callback=
909             self.update,
910             tag=dpg.
911             generate_uuid(),
912             readonly=
913             True)
914         self.attr_2 = dpg.add_node_attribute(label='
915             Voxel Model 2',
916             attribute_type=dpg.mvNode_Attr_Input,
917             parent=
918             node,
919             tag=dpg
920             .generate_uuid())
921
922         self.inp_2 = dpg.add_input_text(
923             default_value=self.value,
924             width=250,
925             parent=self.
926             attr_2,
927             # callback=
928             self.update,
929             tag=dpg.

```

```

915     generate_uuid(),
916                                         readonly=
917         True)
918             self.attr_3 = dpg.add_node_attribute(label='
919             Voxel Model 2',
920             attribute_type=dpg.mvNode_Attr_Output,
921             parent=
922             node,
923                                         tag=dpg
924             .generate_uuid())
925
926             self.out = dpg.add_text(default_value='Voxel
927             Model', parent=self.attr_3,
928                                         tag=dpg.
929             generate_uuid(), show=True)
930             button = dpg.add_button(label='Subtract',
931             tag=dpg.generate_uuid(), parent=self.attr_3,
932                                         width=200, callback=
933             self.subtraction, user_data=[self.inp_1, self.inp_2
934             , self.out])
935
936             def subtraction(self):
937                 pass
938
939             """
940             class Main_Window:
941                 def __init__(self):
942                     dpg.add_window(label="Tutorial", width=800,
943                     height=800, tag='main_window', no_close=False,
944                     no_move=False)
945                     dpg.bind_item_theme(item='main_window',
946                     theme=Main_Window.createTheme())
947
948                     dpg.add_template_registry(tag='
949                     template_registry')
950                     dpg.add_node_editor(callback=link,
951                     delink_callback=delink,
952                                         tag='node_editor',
953                     parent='main_window',
954                                         minimap=True,
955                     minimap_location=1)

```

```

942         # dpg.add_value_registry(tag='value_registry
')
943
944         self.global_handler()
945         dpg.add_item_handler_registry(tag='
movement_handler')
946             dpg.add_item_handler_registry(tag='
del_handler')
947                 dpg.add_menu_bar(tag='main_menu_bar', parent
='main_window')
948                     dpg.add_menu(label="Input Nodes", parent='
main_menu_bar', tag='menu_input_nodes')
949
950             dpg.add_menu_item(label='STL File', tag='STL
File Menu', user_data='STL File',
parent='menu_input_nodes'
, callback=create_node)
952                 dpg.add_tooltip(tag='STL File tooltip',
parent='STL File Menu')
953                     dpg.add_text(default_value='Node to choose
STL File from directory', parent='STL File tooltip')
954
955             dpg.add_menu_item(label='Voxel Grid', tag='
Voxel Grid Menu', user_data='Voxel Grid',
parent='menu_input_nodes'
, callback=create_node)
957                 dpg.add_tooltip(tag='Voxel Grid tooltip',
parent='Voxel Grid Menu')
958                     dpg.add_text(default_value='Node to set
voxel grid for voxelization', parent='Voxel Grid
tooltip')
959
960             dpg.add_menu_item(label='Cell Length', tag='
Cell Length Menu', user_data='Cell Length',
parent='menu_input_nodes'
, callback=create_node)
962                 dpg.add_tooltip(tag='Cell Length tooltip',
parent='Cell Length Menu')
963                     dpg.add_text(default_value='Node to set cell
length to generate lattice structure',
parent='Cell Length tooltip')
964
965             dpg.add_menu_item(label='Cell Thickness',
tag='Cell Thickness Menu', user_data='Cell Thickness'

```

```

966 ,
967                                     parent='menu_input_nodes'
968             , callback=create_node)
969             dpg.add_tooltip(tag='Cell Thickness tooltip'
970             , parent='Cell Thickness Menu')
971             dpg.add_text(default_value='Node to set
972             cell thickness to generate lattice structure',
973             parent='Cell Thickness tooltip'
974             )
975
976             dpg.add_menu_item(label='Lattice Pattern',
977             tag='Lattice Pattern Menu', user_data='Lattice
978             Pattern',
979             parent='menu_input_nodes'
980             , callback=create_node)
981             dpg.add_tooltip(tag='Lattice Pattern
982             tooltip', parent='Lattice Pattern Menu')
983             dpg.add_text(default_value='Node to set
984             lattice pattern to generate lattice structure',
985             parent='Lattice Pattern
986             tooltip')
987
988             dpg.add_menu(label="Functional Nodes",
989             parent='main_menu_bar', tag='menu_functional_nodes'
990             )
991
992             dpg.add_menu_item(label='Cuda Voxelizer',
993             tag='Cuda Voxelizer Menu', user_data='Cuda
994             Voxelizer',
995             parent='
996             menu_functional_nodes', callback=create_node)
997             dpg.add_tooltip(tag='Cuda Voxelizer tooltip'
998             , parent='Cuda Voxelizer Menu')
999             with dpg.group(parent='Cuda Voxelizer
1000             tooltip'):
1001                 dpg.add_text(default_value='Node for
1002                 voxelization using Cuda Voxelizer')
1003                 dpg.add_text(default_value='Required
1004                 params:')
1005                 dpg.add_text(default_value='STL File',
1006                 bullet=True)
1007                 dpg.add_text(default_value='Voxel Grid'
1008                 , bullet=True)
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3
```

```

989         dpg.add_menu_item(label='Trimesh Voxelizer'
990             , tag='Trimesh Voxelizer Menu', user_data='Trimesh
991                 Voxelizer',
992                     parent='
993                         menu_functional_nodes' , callback=create_node)
994             dpg.add_tooltip(tag='Trimesh Voxelizer
995                 tooltip', parent='Trimesh Voxelizer Menu')
996             with dpg.group(parent='Trimesh Voxelizer
997                 tooltip'):
998                 dpg.add_text(default_value='Node for
999                     voxelization using Trimesh Voxelizer')
1000                 dpg.add_text(default_value='Required
1001                     params:')
1002                 dpg.add_text(default_value='STL File',
1003                     bullet=True)
1004                 dpg.add_text(default_value='Pitch -
1005                     size of a voxel side', bullet=True)
1006
1007         dpg.add_menu_item(label='Lattice', tag='
1008             Lattice Menu', user_data='Lattice',
1009                     parent='
1010                         menu_functional_nodes' , callback=create_node)
1011             dpg.add_tooltip(tag='Lattice tooltip',
1012                 parent='Lattice Menu')
1013             with dpg.group(parent='Lattice tooltip'):
1014                 dpg.add_text(default_value='Node to
1015                     generate lattice structure')
1016                 dpg.add_text(default_value='Required
1017                     params:')
1018                 dpg.add_text(default_value='Binvox File
1019                     ', bullet=True)
1020                 dpg.add_text(default_value='Cell Length
1021                     ', bullet=True)
1022                 dpg.add_text(default_value='Cell
1023                     Thickness', bullet=True)
1024                 dpg.add_text(default_value='Lattice
1025                     Pattern', bullet=True)
1026
1027         dpg.add_menu(label="View Nodes", parent='
1028             main_menu_bar', tag='menu_view_nodes')
1029             dpg.add_menu_item(label='View STL', tag='
1030                 View STL Menu', user_data='View STL',
1031                     parent='menu_view_nodes'
1032                     , callback=create_node)

```

```

1012         dpg.add_tooltip(tag='View STL tooltip',
1013                         parent='View STL Menu')
1014         dpg.add_text(default_value='Node to
1015                         visualize STL File', parent='View STL tooltip')
1016         dpg.add_button(label='Save', tag='
1017                         save_button', parent='main_menu_bar', indent=990,
1018                         callback=save,
1019                         width=100, height=35)
1020
1021         self.import_file_path = dpg.add_text(
1022             default_value='No path', parent='main_menu_bar',
1023             show=False)
1024         dpg.add_button(label='Import Project', tag=
1025                         dpg.generate_uuid(),
1026                         parent='main_menu_bar',
1027                         indent=1100, width=150, height=35, track_offset=0.1
1028
1029         ,
1030             callback=Main_Window.
1031             import_project_file_dialog, user_data=self.
1032             import_file_path)
1033
1034     @staticmethod
1035     def import_project_file_dialog(sender, app_data
1036 , user_data):
1037         file_dialog = dpg.add_file_dialog(label='
1038             JSON File Search', tag=dpg.generate_uuid(),
1039                         width=500
1040 , height=300, user_data=user_data,
1041                         callback=
1042             Main_Window.set_value)
1043         extension = dpg.add_file_extension(label='
1044             JSON File Extension', extension='.json',
1045                         tag=dpg.
1046             generate_uuid(), parent=file_dialog)
1047
1048     @staticmethod
1049     def set_value(sender, app_data, user_data):
1050         dpg.set_value(item=user_data, value=
1051             app_data['file_path_name'])
1052         history.from_json(file=app_data[ '
1053             file_path_name'])
1054
1055     def global_handler(self):

```

```

1037         dpg.add_handler_registry(tag='
1038             global_handler')
1039             kph = dpg.add_key_press_handler(callback=
1040                 parent='global_handler')
1041             mph = dpg.add_mouse_release_handler(button=
1042                 dpg.mvMouseButton_Left, parent='global_handler',
1043                 callback=drag_n_drop)
1044
1045     @staticmethod
1046     def createTheme():
1047         with dpg.theme() as themeTag:
1048             with dpg.theme_component(dpg.mvAll):
1049                 dpg.add_theme_style(target=dpg.
1050                     mvStyleVar_WindowPadding, x=5, y=5, category=dpg.
1051                     mvThemeCat_Core)
1052                 dpg.add_theme_style(target=dpg.
1053                     mvStyleVar_WindowBorderSize, x=0, y=0, category=dpg.
1054                     mvThemeCat_Core)
1055                 dpg.add_theme_style(target=dpg.
1056                     mvStyleVar_FramePadding, x=12, y=16, category=dpg.
1057                     mvThemeCat_Core)
1058                 dpg.add_theme_style(target=dpg.
1059                     mvStyleVar_FrameRounding, x=6, category=dpg.
1060                     mvThemeCat_Core)
1061                 dpg.add_theme_style(target=dpg.
1062                     mvStyleVar_FrameBorderSize, x=0, category=dpg.
1063                     mvThemeCat_Core)
1064                 dpg.add_theme_style(target=dpg.
1065                     mvStyleVar_ItemSpacing, x=8, y=9, category=dpg.
1066                     mvThemeCat_Core)
1067                 dpg.add_theme_style(target=dpg.
1068                     mvStyleVar_ItemInnerSpacing, x=5, y=9, category=dpg.
1069                     mvThemeCat_Core)
1070                 dpg.add_theme_style(target=dpg.
1071                     mvStyleVar_CellPadding, x=5, y=9, category=dpg.
1072                     mvThemeCat_Core)
1073                 dpg.add_theme_style(target=dpg.
1074                     mvStyleVar_PopupBorderSize, x=2, category=dpg.
1075                     mvThemeCat_Core)
1076                 dpg.add_theme_style(target=dpg.
1077                     mvStyleVar_PopupRounding, x=3, category=dpg.
1078                     mvThemeCat_Core)
1079             dpg.add_theme_style(target=dpg.

```

```

1056 mvStyleVar_ScrollbarRounding, x=10, category=dpg.
    mvThemeCat_Core)
1057             dpg.add_theme_style(target=dpg.
    mvStyleVar_GrabMinSize, x=16, category=dpg.
    mvThemeCat_Core)
1058             dpg.add_theme_style(target=dpg.
    mvStyleVar_GrabRounding, x=5, category=dpg.
    mvThemeCat_Core)
1059             dpg.add_theme_color(target=dpg.
    mvThemeCol_FrameBgHovered, value=(40, 40, 40),
1060                                         category=dpg.
    mvThemeCat_Core)
1061             dpg.add_theme_color(target=dpg.
    mvThemeCol_FrameBgActive, value=(30, 30, 30),
1062                                         category=dpg.
    mvThemeCat_Core)
1063             dpg.add_theme_color(target=dpg.
    mvThemeCol_ButtonHovered, value=(40, 40, 40),
1064                                         category=dpg.
    mvThemeCat_Core)
1065             dpg.add_theme_color(target=dpg.
    mvThemeCol_ButtonActive, value=(30, 30, 30),
1066                                         category=dpg.
    mvThemeCat_Core)
1067             dpg.add_theme_color(target=dpg.
    mvThemeCol_WindowBg, value=(30, 30, 30),
1068                                         category=dpg.
    mvThemeCat_Core)
1069             dpg.add_theme_color(target=dpg.
    mvThemeCol_MenuBarBg, value=(33, 38, 99),
1070                                         category=dpg.
    mvThemeCat_Core)
1071             dpg.add_theme_color(target=dpg.
    mvThemeCol_Border, value=(0, 0, 0), category=dpg.
    mvThemeCat_Core)
1072             dpg.add_theme_color(target=dpg.
    mvThemeCol_BorderShadow, value=(0, 0, 0),
1073                                         category=dpg.
    mvThemeCat_Core)
1074             dpg.add_theme_color(target=dpg.
    mvThemeCol_SliderGrab, value=(25, 107, 52),
1075                                         category=dpg.
    mvThemeCat_Core)
1076             dpg.add_theme_color(target=dpg.

```

```

1076 mvThemeCol_SliderGrabActive, value=(44, 150, 79),
1077                                         category=dpg.
    mvThemeCat_Core)
1078
1079             dpg.add_theme_style(target=dpg.
    mvNodeStyleVar_NodeCornerRounding, x=9,
1080                                         category=dpg.
    mvThemeCat_Nodes)
1081             dpg.add_theme_style(target=dpg.
    mvNodeStyleVar_NodePadding, x=11, y=10,
1082                                         category=dpg.
    mvThemeCat_Nodes)
1083             dpg.add_theme_style(target=dpg.
    mvNodeStyleVar_NodeBorderThickness, x=2,
1084                                         category=dpg.
    mvThemeCat_Nodes)
1085             dpg.add_theme_style(target=dpg.
    mvNodeStyleVar_PinQuadSideLength, x=8, category=dpg
    .mvThemeCat_Nodes)
1086             dpg.add_theme_style(target=dpg.
    mvNodeStyleVar_PinTriangleSideLength, x=8,
1087                                         category=dpg.
    mvThemeCat_Nodes)
1088             dpg.add_theme_style(target=dpg.
    mvNodeStyleVar_PinCircleRadius, x=4, category=dpg.
    mvThemeCat_Nodes)
1089             dpg.add_theme_style(target=dpg.
    mvNodeStyleVar_PinOffset, x=-1, category=dpg.
    mvThemeCat_Nodes)
1090             dpg.add_theme_style(target=dpg.
    mvNodeStyleVar_LinkThickness, x=2, category=dpg.
    mvThemeCat_Nodes)
1091             dpg.add_theme_style(target=dpg.
    mvNodeStyleVar_LinkLineSegmentsPerLength, x=2,
1092                                         category=dpg.
    mvThemeCat_Nodes)
1093             dpg.add_theme_style(target=dpg.
    mvNodesStyleVar_MiniMapPadding, x=1, y=1,
1094                                         category=dpg.
    mvThemeCat_Nodes)
1095             dpg.add_theme_style(target=dpg.
    mvNodeStyleVar_GridSpacing, x=40, y=40,
1096                                         category=dpg.
    mvThemeCat_Nodes)

```

```

1097             # dpg.add_theme_color(target=dpg.
1098                         mvThemeCol_Button, value=(0, 0, 0),
1099                         # category=dpg.mvThemeCat_Core)
1100                         dpg.add_theme_color(target=dpg.
1101                           mvNodeCol_NodeOutline, value=(0, 0, 0),
1102                           category=dpg.
1103                           mvThemeCat_Nodes)
1104                         dpg.add_theme_color(target=dpg.
1105                           mvNodeCol_TitleBar, value=(33, 38, 99), # (80, 40
1106                           , 60),
1107                           category=dpg.
1108                           mvThemeCat_Nodes)
1109                         dpg.add_theme_color(target=dpg.
1110                           mvNodeCol_TitleBarHovered, value=(105, 137, 204),
1111                           # (100, 60, 80),
1112                           category=dpg.
1113                           mvThemeCat_Nodes)
1114                         dpg.add_theme_color(target=dpg.
1115                           mvNodeCol_NodeBackground, value=(82, 82, 82),
1116                           category=dpg.
1117                           mvThemeCat_Nodes)
1118                         dpg.add_theme_color(target=dpg.
1119                           mvNodeCol_Pin, value=(252, 197, 119),
1120                           category=dpg.
1121                           mvThemeCat_Nodes)
1122                         dpg.add_theme_color(target=dpg.
1123                           mvNodeCol_PinHovered, value=(252, 186, 93),
1124                           category=dpg.
1125                           mvThemeCat_Nodes)
1126                         dpg.add_theme_color(target=dpg.
1127                           mvNodeCol_Link, value=(252, 197, 119),
1128                           category=dpg.
1129                           mvThemeCat_Nodes)
1130                         dpg.add_theme_color(target=dpg.
1131                           mvNodeCol_GridLine, value=(30, 30, 30, 255),
1132                           category=dpg.
1133                           mvThemeCat_Nodes)
1134                         dpg.add_theme_color(target=dpg.
1135                           mvNodeCol_GridBackground, value=(50, 50, 50, 255),
1136                           category=dpg.

```

```

1118 mvThemeCat_Nodes)
1119             dpg.add_theme_color(target=dpg.
1120                         mvNodesCol_MiniMapOutline, value=(0, 0, 0),
1121                                         category=dpg.
1122                         mvThemeCat_Nodes)
1123             # dpg.add_theme_color(target=dpg.
1124                         mvNodesCol_MiniMapCanvas, value=(0, 0, 0, 50),
1125                                         category=dpg.
1126                         mvThemeCat_Nodes)
1127             # dpg.add_theme_color(target=dpg.
1128                         mvNodesCol_MiniMapCanvasOutline, value=(0, 0, 0),
1129                                         category=dpg.
1130                         mvThemeCat_Nodes)
1131             # dpg.add_theme_color(target=dpg.
1132                         mvNodesCol_MiniMapBackground, value=(0, 0, 0, 120),
1133                                         category=dpg.
1134                         mvThemeCat_Nodes)
1135             # dpg.add_theme_color(target=dpg.
1136                         mvNodesCol_NodeBackground, value=(128, 128,
1137                                         category=dpg.
1138                         mvThemeCat_Nodes)
1139             # dpg.add_theme_color(target=dpg.
1140                         mvStyleVar_SelectableTextAlign, x=-1, y=-1,
1141                                         category=dpg.mvThemeCat_Core)
1142             with dpg.theme_component(dpg.mvText):
1143                 dpg.add_theme_style(target=dpg.
1144                         mvStyleVar_WindowPadding, x=12, y=12, category=dpg.
1145                         mvThemeCat_Core)
1146                 # dpg.add_theme_style(target=dpg.
1147                         mvStyleVar_FramePadding, x=12, y=12, category=dpg.
1148                         mvThemeCat_Core)
1149                 dpg.add_theme_color(target=dpg.
1150                         mvThemeCol_Border, value=(40, 136, 101),
1151                                         category=dpg.
1152                                         mvThemeCat_Core)
1153                 dpg.add_theme_style(target=dpg.
1154                         mvStyleVar_ItemSpacing, x=15, y=15, category=dpg.
1155                         mvThemeCat_Core)
1156             with dpg.theme_component(dpg.mvCombo):

```

```

1141             dpg.add_theme_style(target=dpg.
    mvStyleVar_FramePadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1142             dpg.add_theme_style(target=dpg.
    mvStyleVar_FrameRounding, x=6, category=dpg.
    mvThemeCat_Core)
1143             dpg.add_theme_style(target=dpg.
    mvStyleVar_WindowPadding, x=14, y=14, category=dpg.
    mvThemeCat_Core)
1144             dpg.add_theme_style(target=dpg.
    mvStyleVar_PopupBorderSize, x=2, category=dpg.
    mvThemeCat_Core)
1145             dpg.add_theme_style(target=dpg.
    mvStyleVar_PopupRounding, x=10, category=dpg.
    mvThemeCat_Core)
1146             dpg.add_theme_color(target=dpg.
    mvThemeCol_Border, value=(40, 136, 101),
1147                                         category=dpg.
    mvThemeCat_Core)
1148
1149             with dpg.theme_component(dpg.mvInputInt
    ):
1150                 dpg.add_theme_style(target=dpg.
    mvStyleVar_FramePadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1151                 dpg.add_theme_style(target=dpg.
    mvStyleVar_FrameRounding, x=6, category=dpg.
    mvThemeCat_Core)
1152                 dpg.add_theme_style(target=dpg.
    mvStyleVar_WindowPadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1153
1154             with dpg.theme_component(dpg.
    mvInputIntMulti):
1155                 dpg.add_theme_style(target=dpg.
    mvStyleVar_FramePadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1156                 dpg.add_theme_style(target=dpg.
    mvStyleVar_FrameRounding, x=6, category=dpg.
    mvThemeCat_Core)
1157                 dpg.add_theme_style(target=dpg.
    mvStyleVar_WindowPadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1158

```

```

1159             with dpg.theme_component(dpg.
1160                 mvInputFloatMulti):
1161                     dpg.add_theme_style(target=dpg.
1162                         mvStyleVar_FramePadding, x=6, y=6, category=dpg.
1163                             mvThemeCat_Core)
1164                     dpg.add_theme_style(target=dpg.
1165                         mvStyleVar_FrameRounding, x=6, category=dpg.
1166                             mvThemeCat_Core)
1167                     dpg.add_theme_style(target=dpg.
1168                         mvStyleVar_WindowPadding, x=6, y=6, category=dpg.
1169                             mvThemeCat_Core)
1170                     with dpg.theme_component(dpg.
1171                         mvDragIntMulti):
1172                         dpg.add_theme_style(target=dpg.
1173                             mvStyleVar_FramePadding, x=6, y=6, category=dpg.
1174                             mvThemeCat_Core)
1175                         dpg.add_theme_style(target=dpg.
1176                             mvStyleVar_FrameRounding, x=6, category=dpg.
1177                             mvThemeCat_Core)
1178                     with dpg.theme_component(dpg.
1179                         mvDragFloat):
1180                         dpg.add_theme_style(target=dpg.
1181                             mvStyleVar_FramePadding, x=6, y=6, category=dpg.
1182                             mvThemeCat_Core)
1183                         dpg.add_theme_style(target=dpg.
1184                             mvStyleVar_FrameRounding, x=6, category=dpg.
1185                             mvThemeCat_Core)

```

```

1177                     dpg.add_theme_style(target=dpg.
    mvStyleVar_WindowPadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1178
1179             with dpg.theme_component(dpg.
    mvDragFloatMulti):
1180                 dpg.add_theme_style(target=dpg.
    mvStyleVar_FramePadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1181                     dpg.add_theme_style(target=dpg.
    mvStyleVar_FrameRounding, x=6, category=dpg.
    mvThemeCat_Core)
1182                     dpg.add_theme_style(target=dpg.
    mvStyleVar_WindowPadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1183
1184             with dpg.theme_component(dpg.
    mvInputText):
1185                 dpg.add_theme_style(target=dpg.
    mvStyleVar_FramePadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1186                     dpg.add_theme_style(target=dpg.
    mvStyleVar_FrameRounding, x=6, category=dpg.
    mvThemeCat_Core)
1187                     dpg.add_theme_style(target=dpg.
    mvStyleVar_WindowPadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1188
1189             with dpg.theme_component(dpg.mvText):
1190                 dpg.add_theme_style(target=dpg.
    mvStyleVar_FramePadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1191
1192             with dpg.theme_component(dpg.
    mvInputFloat):
1193                 dpg.add_theme_style(target=dpg.
    mvStyleVar_FramePadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)
1194                     dpg.add_theme_style(target=dpg.
    mvStyleVar_FrameRounding, x=8, category=dpg.
    mvThemeCat_Core)
1195                     dpg.add_theme_style(target=dpg.
    mvStyleVar_WindowPadding, x=6, y=6, category=dpg.
    mvThemeCat_Core)

```

```

1196
1197         with dpg.theme_component(dpg.
1198             mvColorEdit):
1199                 dpg.add_theme_style(target=dpg.
1200                     mvStyleVar_FramePadding, x=6, y=6, category=dpg.
1201                     mvThemeCat_Core)
1202                     dpg.add_theme_style(target=dpg.
1203                         mvStyleVar_FrameRounding, x=8, category=dpg.
1204                         mvThemeCat_Core)
1205                     dpg.add_theme_style(target=dpg.
1206                         mvStyleVar_WindowPadding, x=6, y=6, category=dpg.
1207                         mvThemeCat_Core)
1208             with dpg.theme_component(dpg.mvButton):
1209                 dpg.add_theme_style(target=dpg.
1210                     mvStyleVar_FramePadding, x=6, y=6, category=dpg.
1211                     mvThemeCat_Core)
1212                     dpg.add_theme_style(target=dpg.
1213                         mvStyleVar_ButtonTextAlign, x=6, y=6, category=dpg.
1214                         mvThemeCat_Core)
1215             with dpg.theme_component(dpg.
1216                 mvCheckbox):
1217                     dpg.add_theme_style(target=dpg.
1218                         mvStyleVar_FramePadding, x=4, y=4, category=dpg.
1219                         mvThemeCat_Core)
1220                     dpg.add_theme_color(target=dpg.
1221                         mvThemeCol_CheckMark, value=(201, 99, 30),
1222                                         category=dpg.
1223                                         mvThemeCat_Core)
1224             with dpg.theme_component(dpg.
1225                 mvSliderInt):
1226                     dpg.add_theme_style(target=dpg.
1227                         mvStyleVar_FrameRounding, x=5, category=dpg.
1228                         mvThemeCat_Core)
1229             with dpg.theme_component(dpg.

```

```

1216 mvSliderFloat):
1217         dpg.add_theme_style(target=dpg.
1218             mvStyleVar_FrameRounding, x=5, category=dpg.
1219             mvThemeCat_Core)
1220
1221
1222 keytimer = time()
1223
1224
1225 def key(*arg):
1226     global keytimer
1227     if keytimer < time():
1228         # print("asd")
1229         if dpg.is_key_down(dpg.mvKey_Z) and dpg.
1230             is_key_down(dpg.mvKey_Control) and dpg.is_key_down(
1231                 dpg.mvKey_Shift):
1232                     history.redo()
1233                     keytimer = time() + 0.2
1234             elif dpg.is_key_down(dpg.mvKey_Z) and dpg.
1235                 is_key_down(dpg.mvKey_Control):
1236                     history.undo()
1237                     keytimer = time() + 0.2
1238             elif dpg.is_key_down(dpg.mvKey_D) and dpg.
1239                 get_selected_nodes(node_editor='node_editor'
1240             ) != []:
1241                     move_node()
1242                     keytimer = time() + 0.2
1243             elif dpg.is_key_down(dpg.mvKey_Delete) and
1244                 (dpg.get_selected_nodes(node_editor='node_editor')
1245
1246             or dpg.get_selected_links(node_editor='node_editor
1247             ')):
1248                     if dpg.get_selected_nodes(node_editor='
1249                         node_editor'):
1250                             delete_node()
1251                     if dpg.get_selected_links(node_editor='
1252                         node_editor'):
1253                         delink(sender='node_editor',
1254                             app_data=None)
1255                     keytimer = time() + 0.2
1256
1257
1258

```

```

1247 def move_node(*arg):
1248     history.add(Move_Node())
1249
1250
1251 def delete_node(*arg):
1252     history.add(Delete_Node())
1253
1254
1255 def create_node(sender, app_data, user_data, *arg):
1256     node = Create_Node(user_data)
1257     history.add(node)
1258
1259
1260 def link(sender, app_data, *arg):
1261     connection = Create_Connection(sender, app_data
1262 )
1263     history.add(connection)
1264     connection.redo()
1265
1266
1267 def delink(sender, app_data, *arg):
1268     connection = Delete_Connection(sender, app_data
1269 )
1270     history.add(connection)
1271     connection.redo()
1272
1273
1274
1275
1276 def drag_n_drop(*arg):
1277     if dpg.get_selected_nodes(node_editor='
1278         node_editor'):
1279         for node in dpg.get_selected_nodes(
1280             node_editor='node_editor'):
1281             history.add(Drop_Node(node))
1282
1283
1284 window = Main_Window()
1285 history = History()
1286

```

```
1287 dpg.setup_dearpygui()
1288 dpg.show_item_registry()
1289 dpg.show_style_editor()
1290 dpg.show_viewport()
1291 dpg.set_primary_window(window='main_window', value=
    True)
1292
1293 while dpg.is_dearpygui_running():
1294     dpg.render_dearpygui_frame()
1295     sleep(0.033)
1296
1297 dpg.destroy_context()
1298
```