Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.925.8

ВИЗУАЛИЗАЦИЯ ГЕОМЕТРИИ СЛОЁВ ИЗДЕЛИЙ АДДИТИВНОГО ПРОИЗВОДСТВА СРЕДСТВАМИ QT И OPENGL

Фролов Денис Александрович,
Национальный исследовательский университет «МИЭТ», Москва, Зеленоград
den.frolov2004@mail.ru

Бабенков Юрий Михайлович,
Национальный исследовательский университет «МИЭТ», Москва, Зеленоград
e-mail:; kisheryt2007@gmail.com

Аннотация

Рассмотрена организация визуализации геометрии слоёв изделий аддитивного производства в настольном приложении на базе C++ и Qt. Описаны представление внешних и внутренних контуров, траекторий заполнения и холостых переходов, преобразование технологических координат в координаты графической сцены, а также формирование вершинных буферов OpenGL. Предложена архитектура, разделяющая загрузку данных, модель слоя и графический модуль. Приведены способы отображения элементов различного назначения, интерактивного масштабирования и визуального контроля геометрии до передачи задания технологическому оборудованию. Результаты функциональной проверки показывают применимость подхода для оперативного выявления ошибок геометрии и подготовки задания.

Ключевые слова: аддитивное производство, слой изделия, визуализация, Qt, OpenGL, C++, траектория сканирования, операторский интерфейс.

VISUALIZATION OF LAYER GEOMETRY IN ADDITIVELY MANUFACTURED PARTS USING QT AND OPENGL

Denis A. Frolov,
National Research University of Electronic Technology (MIET), Moscow, Zelenograd, Russian Federation

Yuri M. Babenkov,
National Research University of Electronic Technology (MIET), Moscow, Zelenograd, Russian Federation

ABSTRACT

The paper considers the organization of layer geometry visualization for additive manufacturing parts in a desktop application based on C++ and Qt. The representation of outer

and inner contours, filling trajectories and non-processing moves is described. The conversion of technological coordinates into graphical scene coordinates and the generation of OpenGL vertex buffers are considered. An architecture separating data loading, the layer model and the rendering module is proposed. Methods for displaying elements of different purposes, interactive scaling and visual geometry verification before transferring a job to technological equipment are presented. Functional testing demonstrates that the approach can be used to detect geometry errors and verify prepared layer data.

Keywords: additive manufacturing, part layer, visualization, Qt, OpenGL, C++, scanning trajectory, operator interface.

Актуальность

В аддитивном производстве трёхмерная модель изделия преобразуется в последовательность слоёв, каждый из которых содержит геометрические контуры и технологические траектории. До передачи задания оборудованию оператору необходимо проверить расположение объектов, замкнутость контуров, наличие внутренних областей, корректность заполнения и последовательность выполнения переходов. Текстовое представление координат не позволяет быстро обнаруживать геометрические ошибки, поэтому программное обеспечение установки должно включать модуль визуализации слоя.

Стандарты ISO/ASTM определяют аддитивное производство как формирование физической геометрии последовательным добавлением материала [1]. Обмен геометрическими данными и подготовка информации для производства рассматриваются в ISO/ASTM 52950 и ISO/ASTM 52915 [2, 3]. На уровне операторского приложения требуется представить подготовленные двумерные данные слоя в форме, удобной для анализа, без изменения исходной технологической информации. Исследования методов формирования и анализа траекторий показывают, что выбранная структура перемещений и стратегия сканирования влияют на точность, качество и эффективность аддитивного процесса [9–11], поэтому визуальная проверка контуров и последовательности переходов имеет самостоятельное практическое значение.

При небольшом числе графических примитивов отображение может выполняться средствами высокоуровневого двумерного API. Однако для сложных слоёв, содержащих десятки и сотни тысяч отрезков, требуется подход, уменьшающий число отдельных вызовов рисования и обеспечивающий интерактивное изменение масштаба. В экосистеме Qt такую задачу целесообразно решать с применением QOpenGLWidget и аппаратно ускоряемого рендеринга OpenGL [4–7].

Цель исследования

Целью работы является формирование архитектуры модуля визуализации геометрии слоёв изделий аддитивного производства средствами Qt и OpenGL. Для достижения цели решаются задачи представления данных слоя, преобразования координат, группировки элементов по назначению, организации графического конвейера и функциональной проверки отображения типовых геометрических случаев.

Материалы и методы исследования

Слой представляется как набор частей изделия. Каждая часть содержит внешний контур, ноль или несколько внутренних контуров, а также последовательности отрезков заполнения. Минимальной единицей данных является точка с координатами x и y и признаком состояния технологического воздействия. Последовательность точек образует ломаную линию; переход между двумя соседними точками интерпретируется как рабочий или холостой участок траектории.

Для каждого элемента хранится категория отображения: внешний контур, внутренний контур, заполнение, опорная структура, холостой переход либо выделенный объект. Категория определяет способ группировки и визуального различения элементов. Модель слоя не должна зависеть от конкретного графического API: она хранит координаты в технологической системе единиц и передаёт графическому модулю неизменяемый набор данных. Такое разделение позволяет использовать одну модель для OpenGL-визуализации, экспорта и вычислительных процедур.

Обработка начинается с чтения файла задания или внутренней структуры технологического процесса. Парсер формирует объекты слоя и выполняет базовую проверку целостности: наличие координат, корректность индексов, допустимость числовых значений и согласованность последовательностей точек. Затем модель сцены вычисляет габаритный прямоугольник слоя, формирует категории элементов и задаёт порядок их отрисовки.

На следующем этапе координаты преобразуются в формат, используемый графическим конвейером, и группируются в массивы вершин. Для каждой категории формируется отдельный вершинный буфер либо диапазон общего буфера. Такой подход сокращает количество вызовов отрисовки и позволяет изменять параметры отображения категории без повторного разбора исходных данных.

Компонент QOpenGLWidget интегрирует OpenGL-контекст с интерфейсом приложения и предоставляет функции initializeGL(), resizeGL() и paintGL() [4]. Ресурсы, не изменяющиеся при каждом кадре, создаются в initializeGL(), а обновление вершинных данных выполняется после загрузки слоя или изменения его содержимого. Шейдерная программа создаётся средствами QOpenGLShaderProgram [5]. Матрицы вида и проекции могут формироваться классом QMatrix4x4 [8].

Технологические координаты слоя обычно выражаются в миллиметрах и могут принимать произвольные значения. Для первоначального вписывания слоя в окно вычисляются минимальные и максимальные координаты. После этого координаты приводятся к нормализованному диапазону OpenGL:

$$x_n = 2\frac{x - x_{min}}{x_{max} - x_{min}} - 1, \quad y_n = 2\frac{y - y_{min}}{y_{max} - y_{min}} - 1.$$

где x, y — исходные технологические координаты; x_{min}, x_{max}, y_{min}, y_{max} — границы слоя; x_n, y_n — нормализованные координаты. Если диапазон по одной из осей равен нулю, для этой оси задаётся отдельная обработка, исключающая деление на ноль. Для сохранения геометрических пропорций применяется единый масштаб, вычисляемый по меньшему из доступных размеров области вывода.

Масштабирование выполняется изменением матрицы проекции либо коэффициента преобразования. Перемещение сцены реализуется сдвигом центра вида. При изменении размеров окна необходимо сохранять одинаковый масштаб по осям, иначе геометрия будет визуально искажена.

Контуры отображаются замкнутыми или открытыми ломаными в соответствии с исходными данными. Внутренние контуры должны визуально отличаться от внешних не только цветом, но при необходимости толщиной или типом линии.

Результаты и их обсуждение

Для небольших объёмов данных геометрию можно отображать средствами QPainter. Такой способ обладает низкой начальной сложностью и напрямую интегрируется с QWidget. При росте числа отрезков увеличиваются затраты на преобразование координат и выполнение большого количества операций рисования. OpenGL требует дополнительной подготовки ресурсов, однако позволяет хранить вершины в буферах и выполнять повторное отображение средствами графического процессора. Поэтому для просмотра

сложных слоёв и интерактивного масштабирования целесообразно использовать OpenGL, сохраняя QPainter для подписей и вспомогательных элементов.

Таблица 1.

Качественное сравнение средств отображения геометрии

Критерий	QPainter	OpenGL
Начальная сложность реализации	Низкая	Средняя
Большое число отрезков	Производительность ограничена обработкой на CPU	Буферизация вершин и отрисовка на стороне GPU
Интерактивное масштабирование	Пересчёт координат и повторное рисование	Матричные преобразования сцены
Интеграция с Qt Widgets	Непосредственная	Через QOpenGLWidget
Шейдерная обработка	Не используется	Поддерживается

Функциональная проверка графического модуля проводилась на слоях с различной геометрией: одним внешним контуром, внешним контуром и внутренним отверстием, несколькими отдельными объектами, частичными линиями заполнения и координатами, выходящими за исходно ожидаемый диапазон. Оценивались соответствие габаритов, отсутствие инверсии осей, корректность замыкания линий, сохранение пропорций при изменении размера окна и визуальное разделение категорий геометрии.

Таблица 2.

Результаты функциональной проверки

Тестовый случай	Проверяемый критерий	Результат
Один внешний контур	Замкнутость, отсутствие инверсии осей	Соответствует
Внешний и внутренний контуры	Различимость внешней границы и отверстия	Соответствует
Несколько отдельных объектов	Сохранение взаимного положения частей	Соответствует
Частичные линии заполнения	Корректное отображение начала и конца отрезков	Соответствует
Изменение размера окна	Сохранение пропорций и положения центра сцены	Соответствует

Проведённая проверка показала, что разделение модели данных и графического модуля упрощает обработку различных типов геометрии. После формирования вершинных буферов повторное отображение при масштабировании и перемещении сцены не требует повторного разбора технологических данных. Наиболее существенным ограничением остаётся необходимость контролировать объём графической памяти и обновлять буферы только при изменении содержимого слоя.

Заключение

Визуализацию геометрии слоёв изделий аддитивного производства в приложении на базе C++ и Qt целесообразно строить на независимой модели данных, отдельном этапе преобразования координат и OpenGL-модуле отображения. Группировка элементов по назначению и формирование вершинных буферов уменьшают число операций рисования и обеспечивают интерактивный просмотр сложных слоёв. Преобразование координат с

единым масштабом по осям сохраняет геометрические пропорции при изменении размеров окна.

Модуль позволяет отображать внешние и внутренние контуры, рабочие траектории, холостые переходы и выделенные объекты, а также выполнять предварительную визуальную проверку задания до передачи технологическому оборудованию. Дальнейшее развитие может включать выбор элементов по координате курсора, отображение параметров траекторий, профилирование времени отрисовки и автоматическое выделение потенциальных нарушений целостности геометрии.

Список литературы:

1. ISO/ASTM 52900:2021. Additive manufacturing — General principles — Fundamentals and vocabulary. Geneva: ISO, 2021.
2. ISO/ASTM 52950:2021. Additive manufacturing — General principles — Overview of data processing. Geneva: ISO, 2021.
3. ISO/ASTM 52915:2020. Specification for additive manufacturing file format (AMF) Version 1.2. Geneva: ISO, 2020.
4. Qt Group. QOpenGLWidget Class. Qt 6 Documentation. URL: https://doc.qt.io/qt-6/qopenglwidget.html (дата обращения: 23.07.2026).
5. Qt Group. QOpenGLShaderProgram Class. Qt 6 Documentation. URL: https://doc.qt.io/qt-6/qopenglshaderprogram.html (дата обращения: 23.07.2026).
6. Khronos Group. OpenGL Registry. URL: https://registry.khronos.org/OpenGL/ (дата обращения: 23.07.2026).
7. Khronos Group. OpenGL 4.6 Core Profile Specification. 2022.
8. Qt Group. QMatrix4x4 Class. Qt 6 Documentation. URL: https://doc.qt.io/qt-6/qmatrix4x4.html (дата обращения: 23.07.2026).
9. Jin Y.-A., He Y., Fu J.-Z., Gan W.-F., Lin Z.-W. Optimization of tool-path generation for material extrusion-based additive manufacturing technology // Additive Manufacturing. 2014. Vol. 1–4. P. 32–47. DOI: 10.1016/j.addma.2014.08.004.
10. Ding D., Pan Z., Cuiuri D., Li H., Larkin N. Adaptive path planning for wire-feed additive manufacturing using medial axis transformation // Journal of Cleaner Production. 2016. Vol. 133. P. 942–952. DOI: 10.1016/j.jclepro.2016.06.036.
11. Foteinopoulos P., Papacharalampopoulos A., Angelopoulos K., Stavropoulos P. Development of a simulation approach for laser powder bed fusion based on scanning strategy selection // The International Journal of Advanced Manufacturing Technology. 2020. Vol. 108. P. 3085–3100. DOI: 10.1007/s00170-020-05603-4.

References:

1. ISO/ASTM 52900:2021. Additive manufacturing — General principles — Fundamentals and vocabulary. Geneva: ISO, 2021.
2. ISO/ASTM 52950:2021. Additive manufacturing — General principles — Overview of data processing. Geneva: ISO, 2021.
3. ISO/ASTM 52915:2020. Specification for additive manufacturing file format (AMF) Version 1.2. Geneva: ISO, 2020.
4. Qt Group. QOpenGLWidget Class. Qt 6 Documentation. Available at: https://doc.qt.io/qt-6/qopenglwidget.html (accessed 23 July 2026).

5. Qt Group. QOpenGLShaderProgram Class. Qt 6 Documentation. Available at: https://doc.qt.io/qt-6/qopenglshaderprogram.html (accessed 23 July 2026).
6. Khronos Group. OpenGL Registry. Available at: https://registry.khronos.org/OpenGL/ (accessed 23 July 2026).
7. Khronos Group. OpenGL 4.6 Core Profile Specification. 2022.
8. Qt Group. QMatrix4x4 Class. Qt 6 Documentation. Available at: https://doc.qt.io/qt-6/qmatrix4x4.html (accessed 23 July 2026).
9. Jin Y.-A., He Y., Fu J.-Z., Gan W.-F., Lin Z.-W. Optimization of tool-path generation for material extrusion-based additive manufacturing technology. Additive Manufacturing. 2014;1–4:32–47. DOI: 10.1016/j.addma.2014.08.004.
10. Ding D., Pan Z., Cuiuri D., Li H., Larkin N. Adaptive path planning for wire-feed additive manufacturing using medial axis transformation. Journal of Cleaner Production. 2016;133:942–952. DOI: 10.1016/j.jclepro.2016.06.036.
11. Foteinopoulos P., Papacharalampopoulos A., Angelopoulos K., Stavropoulos P. Development of a simulation approach for laser powder bed fusion based on scanning strategy selection. The International Journal of Advanced Manufacturing Technology. 2020;108:3085–3100. DOI: 10.1007/s00170-020-05603-4.