Architecture and Modern Information Technologies. 2025. №4(73). С. 308-320

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И АРХИТЕКТУРА

Научная статья

УДК/UDC 728.51.012:004.942
DOI: 10.24412/1998-4839-2025-4-308-320
EDN: ZFTQTV

Информационное моделирование фасадной системы средствами визуального программирования

Иван Борисович Авдюшкин[1]✉, Мария Игоревна Акимова[2]
[1,2]Новосибирский государственный архитектурно-строительный университет (Сибстрин), Новосибирск, Россия
[1]i_avdyushkin@mail.ru [2]m.akimova@sibstrin.ru

Аннотация. В статье рассматривается проблема интеграции сложной архитектурной геометрии в BIM-среду на примере проектирования фасадной системы здания гостиницы. Стандартные инструменты Autodesk Revit демонстрируют ограниченные возможности при работе со сложными формами, что требует разработки альтернативных подходов. Предложена методика автоматизации импорта и генерации адаптивных элементов с использованием визуального программирования в Dynamo. Особое внимание уделено применению искусственного интеллекта для автоматической генерации Python-скриптов, а также оптимизации процессов триангуляции и обработки полигональных сеток. Результаты тестирования показали значительное сокращение временных затрат по сравнению с ручным моделированием при сохранении высокой точности соответствия исходному архитектурному замыслу.

Ключевые слова: BIM, Autodesk Revit, Dynamo, Python-скрипт, искусственный интеллект, автоматизация проектирования, нестандартное архитектурное решение, архитектурное проектирование сложных форм

Для цитирования: Авдюшкин И.Б. Информационное моделирование фасадной системы средствами визуального программирования / И.Б. Авдюшкин, М.И. Акимова // Architecture and Modern Information Technologies. 2025. №4(73). С. 308-320. URL: https://marhi.ru/AMIT/2025/4kvart25/PDF/21_avdyushkin.pdf DOI: 10.24412/1998-4839-2025-4-308-320 EDN: ZFTQTV

INFORMATION TECHNOLOGIES AND ARCHITECTURE

Original article

The information modeling of a facade system using visual programming

Ivan B. Avdyushkin[1]✉, Maria I. Akimova[2]
[1,2]Novosibirsk State University of Architecture and Civil Engineering (Sibstrin), Novosibirsk, Russia
[1]i_avdyushkin@mail.ru [2]m.akimova@sibstrin.ru

Abstract. The article discusses the problem of integrating complex architectural geometry into the BIM environment using the example of designing the facade system of a hotel building. The standard tools of Autodesk Revit demonstrate limited capabilities when working with complex shapes, which require the development of alternative approaches. The article proposes a method for automating the import and generation of adaptive elements using visual programming in Dynamo. Special attention is paid to the use of artificial intelligence for automatically generating

Python scripts, as well as optimizing the processes of triangulation and processing of polygonal meshes. The test results showed a significant reduction in time costs compared to manual modeling, while maintaining high accuracy in accordance with the original architectural design.
Keywords: BIM, Autodesk Revit, Dynamo, Python script, artificial intelligence, design automation, non-standard architectural solution, architectural design of complex shapes
For citation: Avdyushkin I.B., Akimova M.I. The information modeling of a facade system using visual programming. Architecture and Modern Information Technologies, 2025, no. 4(73), pp. 308-320. Available at: https://marhi.ru/AMIT/2025/4kvart25/PDF/21_avdyushkin.pdf
DOI: 10.24412/1998-4839-2025-4-308-320 EDN: ZFTQTV

Введение

Разработка архитектурных объектов со сложной геометрией в последние годы стала устойчивой тенденцией в современном проектировании. Однако их реализация в рамках BIM-среды сопряжена со значительными техническими трудностями. В ходе проектирования фасадной системы гостиницы было установлено, что стандартные инструменты Autodesk Revit не обеспечивают достаточной точности воспроизведения исходной концепции: полученная модель не отражала всех нюансов задуманной геометрии. Одним из возможных решений является использование специализированного стороннего программного обеспечения (например, Rhino, Grasshopper или Blender), обладающего более гибкими инструментами для работы со свободными формами. Однако при импорте такой геометрии в Revit она воспринимается лишь как статическая оболочка – без параметрических свойств, возможности редактирования и интеграции в BIM-процессы. Это делает невозможным автоматизированную генерацию адаптивных компонентов, необходимых для дальнейшего проектирования, количественного анализа и строительства.

Для преодоления указанных ограничений всё большее распространение получают средства визуального программирования, в частности платформа Dynamo. Dynamo представляет собой среду визуального программирования с открытым исходным кодом, разрабатываемую Autodesk – ведущим мировым поставщиком программного обеспечения для архитектуры, инженерии и строительства. Основное назначение Dynamo – расширение функциональности BIM-платформ за счёт создания пользовательских алгоритмов, позволяющих автоматизировать рутинные операции, обрабатывать большие объёмы данных и генерировать сложную геометрию непосредственно внутри Revit. Процесс программирования в Dynamo строится на компоновке визуальных блоков – узлов (англ. *nodes*), каждый из которых реализует определённую функцию. Последовательное соединение узлов формирует сценарий (англ. *script*), который выполняется как единый алгоритм, обрабатывающий входные данные и генерирующий требуемый результат [1].

Актуальность исследования

Актуальность данного исследования обусловлена необходимостью расширения возможностей концептуального проектирования в BIM-среде. В современной проектной практике импортированная геометрическая модель, как правило, используется лишь в качестве визуального референса, на основе которого впоследствии осуществляется ручное моделирование в Revit. Такой подход не только трудоёмок, но и не позволяет в полной мере использовать потенциал исходной геометрии. Между тем, при правильной интеграции она может стать основой для формирования параметрически управляемой фасадной системы, сохраняющей точное соответствие архитектурной концепции и пригодной для дальнейшего инженерного анализа и строительства.

Особую значимость предложенный алгоритм приобретает при проектировании уникальных фасадных систем, где необходимо одновременно сохранить точное соответствие сложной

архитектурной геометрии и обеспечить возможность автоматизированной генерации адаптивных элементов. Разработанный скрипт на платформе Dynamo устанавливает эффективную взаимосвязь между гибкими инструментами формообразования, характерными для сред генеративного проектирования, и строгой параметрической средой BIM-моделирования, тем самым преодолевая разрыв между концептуальным замыслом и его технической реализацией.

Подготовка модели к импорту

В качестве базового инструмента для создания исходной трёхмерной модели была выбрана программа Blender. Такой выбор обусловлен наличием в её функционале мощных и гибких средств моделирования, позволяющих эффективно работать со сложными концептуальными формами и обеспечивать высокую степень художественной выразительности. Благодаря развитой системе работы с полигональными сетками, поддержке скульптинга и неограниченной свободе в манипуляциях с геометрией, Blender представляет собой оптимальную платформу для формирования архитектурных концепций на ранних этапах проектирования.

Хотя конкретные этапы моделирования в Blender могут варьироваться в зависимости от специфики проектной задачи и не являются предметом детального рассмотрения в данной статье, целесообразно выделить ключевые аспекты формирования геометрии, оказывающие непосредственное влияние на последующую интеграцию модели в программную среду Autodesk Revit.

Топология сетки и процесс триангуляции

Требованием при создании модели является обеспечение корректной топологии полигональной сетки. В программе Blender исходная геометрия, как правило, формируется с использованием преимущественно четырёхугольных полигонов (quads) или, в меньшей степени, треугольных (triangles), что способствует гладкости поверхности и удобству редактирования на этапе концептуального проектирования. Однако при экспорте и последующем импорте в Autodesk Revit вся геометрия автоматически подвергается триангуляции – процессу разбиения поверхностей на треугольные элементы, что обусловлено особенностями внутреннего геометрического ядра Revit. Этот аспект оказывает существенное влияние на качество и точность передачи формы, особенно в участках с высокой кривизной или сложной детализацией, и требует учёта уже на этапе моделирования в Blender.

Триангуляция представляет собой процесс декомпозиции сложной полигональной поверхности на набор примитивных треугольных поверхностей. Поскольку треугольник является простейшим многоугольником, он обладает геометрической жёсткостью и однозначно определяет плоскость: любые три точки в трёхмерном пространстве всегда лежат в одной плоскости. Это свойство критически важно для вычислительных алгоритмов, используемых в BIM-системах, так как обеспечивает стабильность геометрических расчётов, корректность визуализации и надёжную обработку форм.

Вследствие этого, независимо от исходной топологии – будь то четырёхугольная или многоугольная сетка – при импорте в Autodesk Revit вся геометрия автоматически преобразуется в треугольную. Следовательно, уже на этапе моделирования в Blender стремиться к созданию «чистой», регулярной и равномерной полигональной сетки. Такой подход минимизирует риск возникновения артефактов, разрывов или искажений в ходе автоматической триангуляции и повышает качество передачи формы в BIM-среду.

На рис. 1 представлен пример равномерной четырёхугольной сетки, оптимизированной для последующей интеграции в Revit.

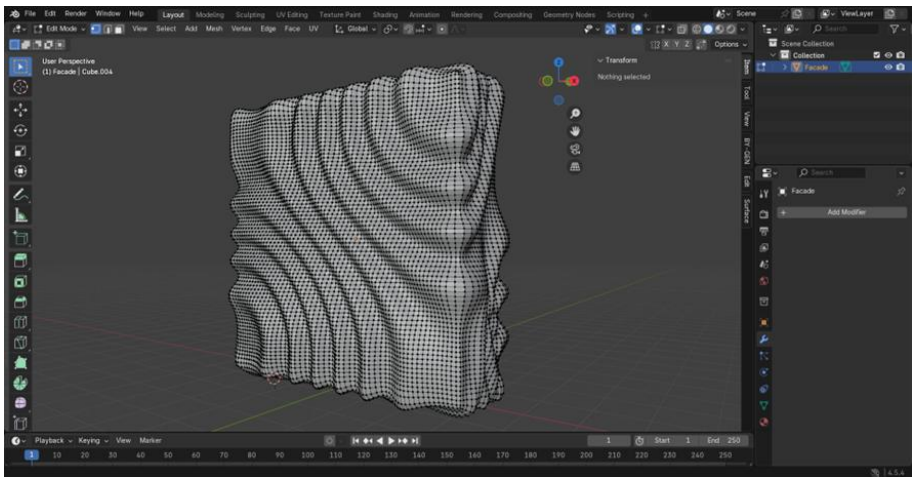


Рис. 1. Геометрическая сетка модели, сформированная в программной среде Blender

Оптимизация геометрической модели

Значимым критерием при подготовке модели к импорту в BIM-среду является объём файла. Эмпирически установлено, что оптимальный размер геометрического файла не должен превышать 1 МБ. Хотя превышение этого порога не делает импорт технически невозможным, но приводит к существенному увеличению времени обработки геометрии со стороны Revit, а также может негативно сказаться на общей производительности проекта – вплоть до замедления работы программы, увеличения времени пересчёта зависимостей и риска нестабильности сессии.

Для эффективной оптимизации топологии и снижения полигональной нагрузки применялись следующие встроенные модификаторы Blender:
1. Модификатор Decimate позволяет уменьшить количество вершин/граней в сетке с минимальными изменениями формы[3].
2. Модификатор Remesh – это инструмент для создания новой топологии сетки. Результат повторяет кривизну поверхности исходного объекта, но его топология будет состоять только из четырёхугольников[4].

Применение этих инструментов на финальном этапе подготовки модели позволило достичь компромисса между визуальным качеством геометрии, количеством полигонов и требованиями к производительности при работе в BIM-среде.

Финальным этапом подготовки является экспорт модели в формат .obj. Важно отметить, что скрипт работает исключительно с этим типом файла.

Создание скрипта для импорта модели в Revit

В рамках проведённого исследования был разработан специализированный скрипт, обеспечивающий импорт и последующее преобразование геометрических данных для их интеграции в среду Autodesk Revit. Работа выполнялась в контексте проекта семейства, созданного на основе шаблона «Метрическая система, формообразующий элемент» в категории «Концептуальные формы», что позволило сохранить геометрию в параметрически управляемом виде.

[3] Decimate Modifier // Blender Documentation - blender.org. URL: https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/decimate.html (дата обращения: 22.10.2025).
[4] Remesh Modifier // Blender Documentation - blender.org. URL: https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/remesh.html (дата обращения: 22.10.2025).

Реализация скрипта потребовала выполнения ряда подготовительных условий, включая установку дополнительного программного пакета – Dynamo Mesh Toolkit. Этот инструментарий предоставляет расширенные функции для работы с полигональными сетками: импорт геометрии из внешних форматов (в частности, .obj), генерацию сеток на основе нативных объектов Dynamo, а также ручное построение сеток по заданным массивам вершин и индексов граней. Кроме того, библиотека включает средства для модификации топологии сетки, её восстановления при повреждениях, а также извлечения горизонтальных сечений, что может быть полезно на этапах координации и производства[5].

Процесс создания скрипта начинается с использования узла File Path, который позволяет пользователю указать путь к исходному файлу геометрии. На выходе данный узел генерирует строковое значение, содержащее полный путь к выбранному файлу. Это значение передаётся на вход узлу Mesh.ImportFile. Указанный узел выполняет импорт внешнего файла и преобразует его в полигональную сетку, представленную в среде Dynamo как объект типа Mesh.

Полученный массив сеток поступает на вход узла Mesh.Triangles, который осуществляет критически важное преобразование полигональных данных в набор параметрических поверхностей (Surface). Далее массив поверхностей передается узлу Solid.ByJoinedSurfaces, где выполняется генерация объемного твердого тела (Solid) через объединение смежных поверхностей в замкнутую оболочку.

Параллельно с процессом геометрического моделирования активируется узел FamilyType.ByGeometry, который создает новые типы семейств на основе объемной геометрии. Этот узел характеризуется шестью входными параметрами: solidGeometry, name, category, templatePath, material и subcategory. Процесс начинается с подключения к входному параметру name узла Code Block, предназначенного для программного определения текстовых и числовых значений. В рамках исследования этому параметру было присвоено значение «Фасад». Следующим этапом осуществляется подключение к входу category узла Categories, предоставляющего доступ к полной библиотеке категорий проекта Revit. В данном случае была выбрана категория «Формы, остекление по формообразующим». Важным элементом настройки является определение шаблона семейства через входной параметр templatePath. К этому входу подключается экземпляр узла File Path, содержащий ссылку на файл шаблона семейств «Метрическая система, адаптивная типовая модель.rft».

Следует отметить, что в работе входные параметры material и subcategory не были задействованы, так как их функциональность не являлась необходимой для решения поставленных задач.

Следующим этапом является установление связи между сгенерированным твердым телом и семейством Revit. Для этого осуществляется прямое соединение выхода Solid узла Solid.ByJoinedSurfaces со входом solidGeometry узла FamilyType.ByGeometry. При этом узел FamilyType.ByGeometry выполняет преобразование статичной геометрической формы в семейство Revit с сохранением: топологических характеристик исходной модели, пространственной ориентации и возможностей для последующего параметрического редактирования.

Процедура масштабирования импортируемой модели представляет собой важный этап, обеспечивающий соответствие геометрии проектным параметрам в среде Revit. Для выполнения этой операции используется узел Geometry.Scale, осуществляющий пропорциональное преобразование геометрических объектов относительно глобальной системы координат. Реализация процесса масштабирования предполагает

[5] Dynamo Mesh Toolkit // Github.com. URL: https://github.com/DynamoDS/Dynamo/wiki/Dynamo-Mesh-Toolkit (дата обращения: 22.10.2025).

последовательное соединение выхода Solid узла Solid.ByJoinedSurfaces со входом geometry узла Geometry.Scale, при этом входной параметр amount соединяется с блоком Code Block, где установлено значение коэффициента масштабирования 1000.

Финальная операция завершается узлом ImportInstance.ByGeometry, который импортирует подготовленную геометрию непосредственно в активный проект Revit. Важной особенностью этого узла является сохранение параметрических связей и обеспечение корректного позиционирования элементов в координационном пространстве проекта (рис. 2).

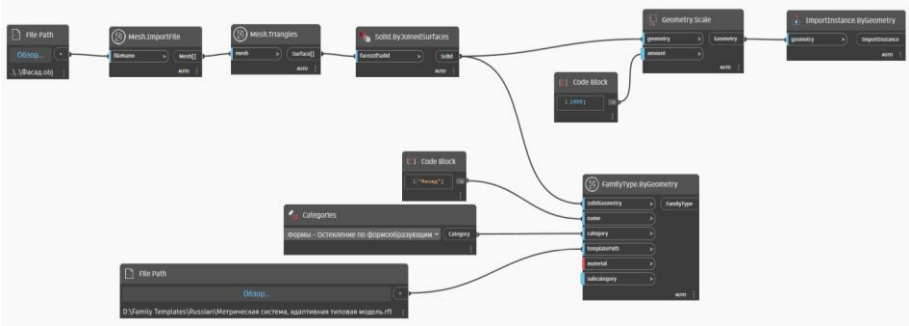


Рис. 2. Скрипт для импорта модели в Revit

После запуска разработанного скрипта осуществляется процедура импорта геометрической модели в проект Autodesk Revit. Данный процесс включает последовательное преобразование исходной полигональной сетки в твердое тело с последующей его интеграцией в качестве управляемого семейства (рис. 3).

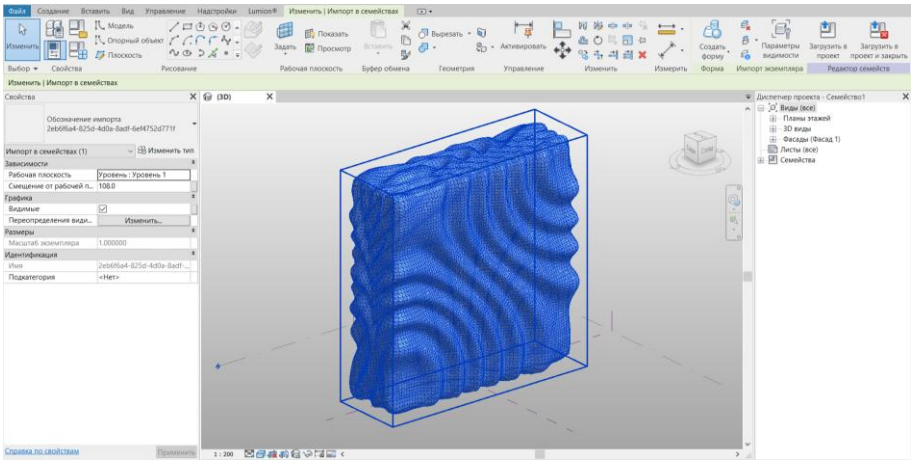


Рис. 3. Импортированная модель в Revit

Для перехода к непосредственному редактированию импортированной модели необходимо выполнить финальную операцию. Была выделена модель, а затем в интерфейсе Revit активировалась вкладка «Изменить | Импорт в семействах», где был выбран раздел «Импорт экземпляра» и выполнена операция «Расчленить». Важным является выбор опции «Полное расчленение», которая обеспечивает преобразование единого импортированного объекта в набор отдельных редактируемых элементов.

Разработанный скрипт успешно реализовал задачу импорта внешней геометрической модели в среду Autodesk Revit. Важным достижением является не только техническая возможность переноса геометрии, но и возможность редактирования модели с помощью

инструментов Revit. Результат импорта приведен на рис. 4, где отчетливо прослеживается соответствие исходной геометрии и её адаптированного представления в Revit.

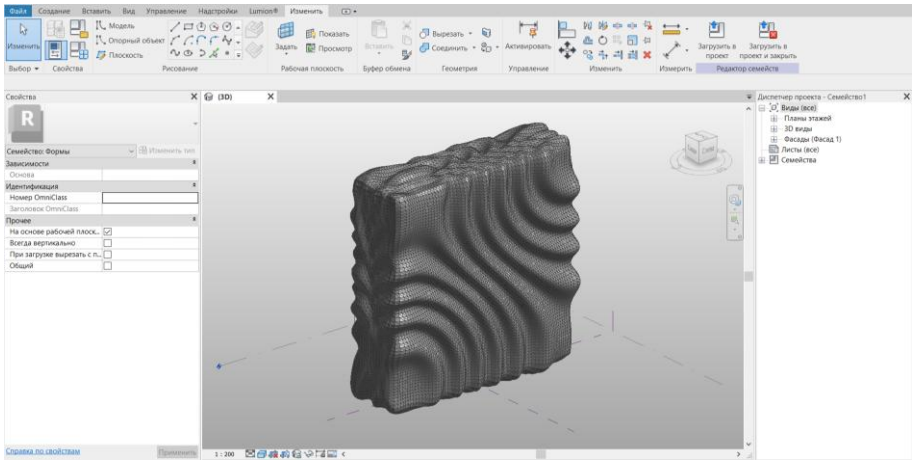
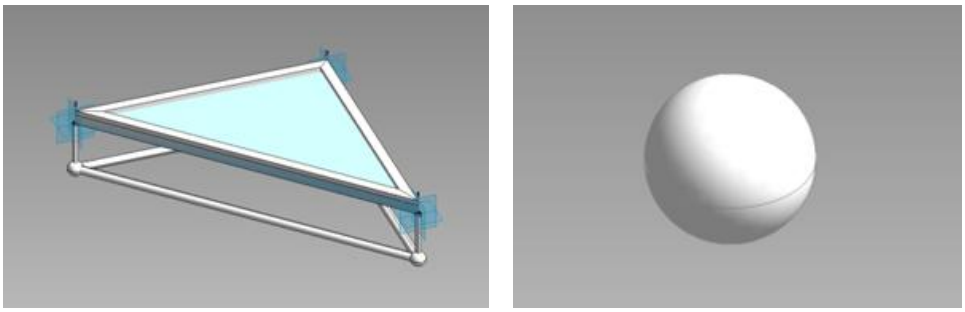


Рис. 4. Финальный вариант импортированной модели в Revit

Создание скрипта для автоматизации генерации адаптивных элементов фасада

На следующем этапе исследования была реализована автоматизация расстановки элементов фасадной системы посредством разработки специализированного скрипта. Хотя ручное размещение компонентов технически возможно, но оно сопряжено со значительными временными издержками и повышенным риском позиционных ошибок. В этой связи создание алгоритма автоматизированного размещения становится ключевым условием повышения точности проектирования и сокращения трудозатрат.

На подготовительном этапе были разработаны и загружены в проект два типа параметрических семейств фасадных элементов (рис. 5): фасадная панель, основанная на трех адаптивных точках позиционирования, и соединительный узловой профиль с одной адаптивной точкой.



а) б)

Рис. 5. Семейства элементов фасада на основе адаптивных точек: а) фасадная панель; б) соединительный узловой профиль

Разработка скрипта начиналась с узла Python.Script, выполняющего роль вычислительного ядра для обработки сценария на языке Python. Конфигурация узла предусматривает создание двух входных параметров: IN[0] и IN[1], обеспечивающих интерфейс для передачи данных.

Для выбора геометрических элементов импортированной модели был задействован узел Select Model Element, выходной параметр Element которого соединяется с входом IN[0] узла

Python.Script. Параметризация процесса осуществляется через узел Integer Slider, функционирующий в качестве регулятора целочисленных значений. Этот параметр передается через вход IN[1] узла Python.Script.

Ключевым элементом скрипта является узел AdaptiveComponent.ByPoints, преобразующий двумерный массив координат в иерархическую структуру адаптивных компонентов. Его вход points соединяется с выходом OUT узла Python.Script, обеспечивая передачу обработанных геометрических данных.

Завершающим этапом становится конфигурация входного параметра familyTypes узла AdaptiveComponent.ByPoints, который соединяется с узлом Family Types. Эта связь обеспечивает доступ к библиотеке загруженных семейств проекта и позволяет выбрать требуемые типы компонентов для последующей генерации фасадной системы (рис. 6).

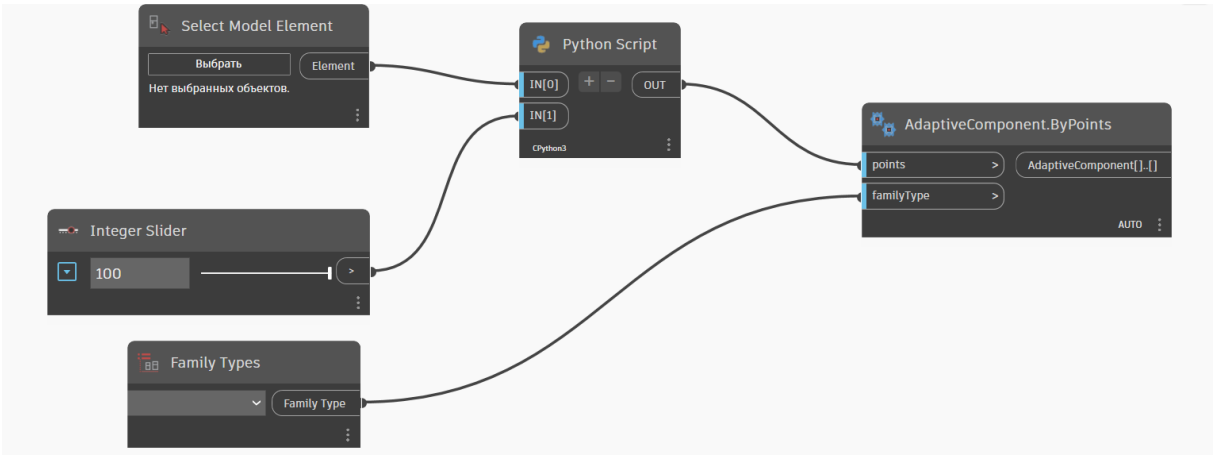


Рис. 6. Финальный вид скрипта для автоматической генерации элементов фасада

Разработанный скрипт представляет собой базовый шаблон, который требовал дальнейшей программной реализации для полноценной автоматизации генерации фасадных элементов. На текущем этапе была идентифицирована ключевая проблема: для преобразования концептуального алгоритма в работоспособный инструмент необходимо было разработать специализированный программный код на языке Python.

Таким образом, последующий этап исследования был посвящен разработке и интеграции программного кода, который обеспечит преобразование базового шаблона в функциональный инструмент автоматизированного проектирования.

Применение искусственного интеллекта при создании программного кода

Современное развитие технологий искусственного интеллекта (ИИ) оказало значительное влияние на множество областей, включая архитектурное и градостроительное проектирование. Сегодня упоминание технологий ИИ стало неотъемлемой составляющей на профильных конференциях, круглых столах и профессиональных обсуждениях, связанных с цифровыми технологиями в архитектуре. Понятия, такие как машинное обучение, глубокое обучение и нейронные сети, стали привычными, а термин «генеративный искусственный интеллект» приобретает все большую популярность, формируя отдельное направление исследований и разработок [2].

Стоит рассмотреть, как именно искусственный интеллект применяется в программировании:

1. *Улучшение процесса разработки ПО*. Одним из ключевых преимуществ использования искусственного интеллекта в программировании является улучшение процесса разработки

программного обеспечения. ИИ может быть использован для автоматизации многих этапов разработки, начиная от анализа требований и проектирования архитектуры программы до тестирования и оптимизации кода. Например, системы машинного обучения могут помочь разработчикам в оптимизации производительности программы, выявлении уязвимостей безопасности и создании более эффективного кода.

2. *Создание инновационных продуктов*. Искусственный интеллект также играет важную роль в создании инновационных продуктов и сервисов. Благодаря возможностям машинного обучения и нейронных сетей разработчики могут создавать «умные» приложения, способные адаптироваться к потребностям пользователей. Примером таких продуктов являются голосовые помощники, рекомендательные системы, системы автоматизации рутинных задач и многое другое.

3. *Оптимизация производительности и увеличение эффективности*. Применение искусственного интеллекта в программировании позволяет существенно повысить производительность и увеличить эффективность работы разработчиков. Автоматизация процессов, улучшенная обработка данных и принятие автоматизированных решений на основе алгоритмов машинного обучения позволяют сократить время, затрачиваемое на разработку программного обеспечения, а также повысить качество и надежность кода [3].

В рамках данного исследования применялась система искусственного интеллекта DeepSeek для генерации специализированных программных кодов на языке Python. С помощью ИИ были разработаны два программных кода, обеспечивающих автоматизацию размещения различных адаптивных элементов: код для распределения элементов с тремя точками позиционирования и код для размещения элементов с одной точкой позиционирования в вершинах треугольников.

Архитектура решения основана на использовании скрипта Dynamo в качестве параметрического шаблона. Для реализации различных сценариев генерации были созданы два файла в формате .dyn, идентичных по структуре, но содержащих разные Python-скрипты, а также различные фасадные элементы.

Применение скрипта в автоматизации расстановки адаптивных элементов

Готовое семейство геометрии было успешно интегрировано в архитектурный проект Autodesk Revit, что обеспечивает его совместимость с общепринятыми стандартами проектной деятельности и позволяет использовать созданные элементы в составе комплексной BIM-модели.

Процесс формирования фасадной системы может быть реализован по модульному принципу с последовательной генерацией различных типов компонентов. Каждому этапу выполнения предшествует тестирование алгоритма с установкой значения узла Integer Slider на уровень 100, тогда как для финальной реализации применяется значение 1, обеспечивающее полномасштабное размещение элементов. После завершения генерации каждого типа компонентов выполняется сохранение проекта с последующим переходом к следующему этапу путем активации соответствующего файла Dynamo с модифицированными Python-скриптами, показанными на рис. 7.

Визуальные результаты выполнения алгоритма демонстрируют следующую последовательность генерации:

```python
import clr
clr.AddReference('RevitNodes')
import Revit
clr.ImportExtensions(Revit.GeometryConversion)
clr.ImportExtensions(Revit.Elements)

clr.AddReference('RevitAPI')
from Autodesk.Revit.DB import *

clr.AddReference('RevitServices')
import RevitServices
from RevitServices.Persistence import DocumentManager
from RevitServices.Transactions import TransactionManager

doc = DocumentManager.Instance.CurrentDBDocument

def create_points_for_adaptive(mass_element, sample_rate=100):
    """
    Создает точки в правильном формате для AdaptiveComponent.ByPoints
    """
    TransactionManager.Instance.EnsureInTransaction(doc)

    try:
        mass = UnwrapElement(mass_element)

        options = Options()
        options.ComputeReferences = True
        geometry_element = mass.get_Geometry(options)

        # Создает список наборов точек (каждый набор - 3 точки для одной панели)
        points_sets = []

        for geom_obj in geometry_element:
            if isinstance(geom_obj, Solid):
                solid = geom_obj
                for i, face in enumerate(solid.Faces):
                    if i % sample_rate == 0:
                        try:
                            mesh = face.Triangulate()
                            if mesh and mesh.NumTriangles > 0:
                                triangle = mesh.get_Triangle(0)

                                point1 = triangle.get_Vertex(0)
                                point2 = triangle.get_Vertex(1)
                                point3 = triangle.get_Vertex(2)

                                # Проверяет расстояния
                                if (point1.DistanceTo(point2) > 0.1 and
                                    point2.DistanceTo(point3) > 0.1):

                                    # Конвертирует точки в Dynamo геометрие
                                    point1_dyn = point1.ToPoint()
                                    point2_dyn = point2.ToPoint()
                                    point3_dyn = point3.ToPoint()

                                    # Создает набор из 3 точек для одной панели
                                    point_set = [point1_dyn, point2_dyn, point3_dyn]
                                    points_sets.append(point_set)

                        except:
                            continue

        TransactionManager.Instance.TransactionTaskDone()

        # Возвращает список наборов точек
        return points_sets

    except Exception as e:
        TransactionManager.Instance.TransactionTaskDone()
        return f"Ошибка: {str(e)}"

# Входные данные
mass = IN[0]
sample_rate = IN[1] if len(IN) > 1 else 100

OUT = create_points_for_adaptive(mass, sample_rate)
```

а)

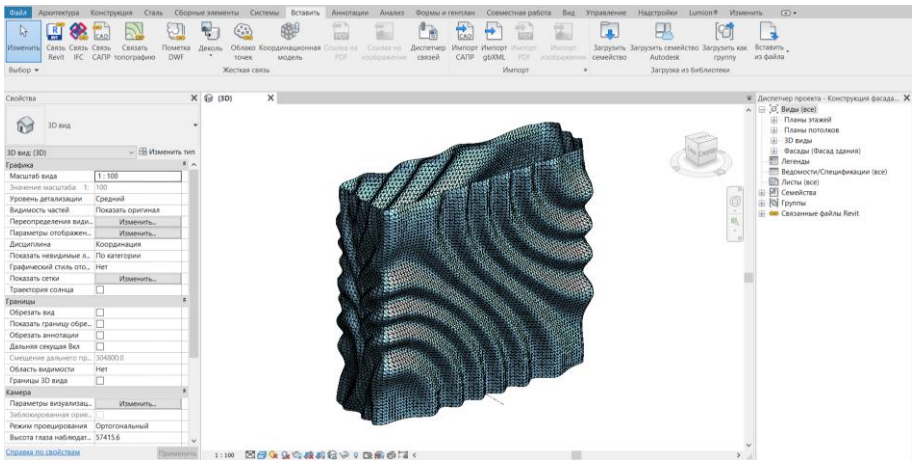
```python
import clr
clr.AddReference('RevitNodes')
import Revit
clr.ImportExtensions(Revit.GeometryConversion)
clr.ImportExtensions(Revit.Elements)

clr.AddReference('RevitAPI')
from Autodesk.Revit.DB import *

clr.AddReference('RevitServices')
import RevitServices
from RevitServices.Persistence import DocumentManager

doc = DocumentManager.Instance.CurrentDBDocument

def simple_vertex_points(mass_element, sample_rate=100):
    """
    Простой скрипт для получения точек в вершинах
    """
    mass = UnwrapElement(mass_element)

    options = Options()
    options.ComputeReferences = True
    geometry_element = mass.get_Geometry(options)

    # Список списков с одной точкой
    single_point_sets = []
    processed_vertices = set()

    for geom_obj in geometry_element:
        if isinstance(geom_obj, Solid):
            solid = geom_obj
            for i, face in enumerate(solid.Faces):
                if i % sample_rate == 0:
                    try:
                        mesh = face.Triangulate()
                        if mesh:
                            for tri_idx in range(min(1, mesh.NumTriangles)):  # Только первый треугольник
                                triangle = mesh.get_Triangle(tri_idx)

                                for vertex_idx in range(3):
                                    vertex = triangle.get Vertex(vertex idx)

                                    vertex_key = (
                                        round(vertex.X, 3),
                                        round(vertex.Y, 3),
                                        round(vertex.Z, 3)
                                    )

                                    if vertex_key not in processed_vertices:
                                        processed_vertices.add(vertex_key)
                                        vertex_dyn = vertex.ToPoint()
                                        single_point_sets.append([vertex_dyn])  # Одна точка в списке

                    except:
                        continue

    return single_point_sets

# Входные данные
mass = IN[0]
sample_rate = IN[1] if len(IN) > 1 else 100

OUT = simple_vertex_points(mass, sample_rate)
```

б)

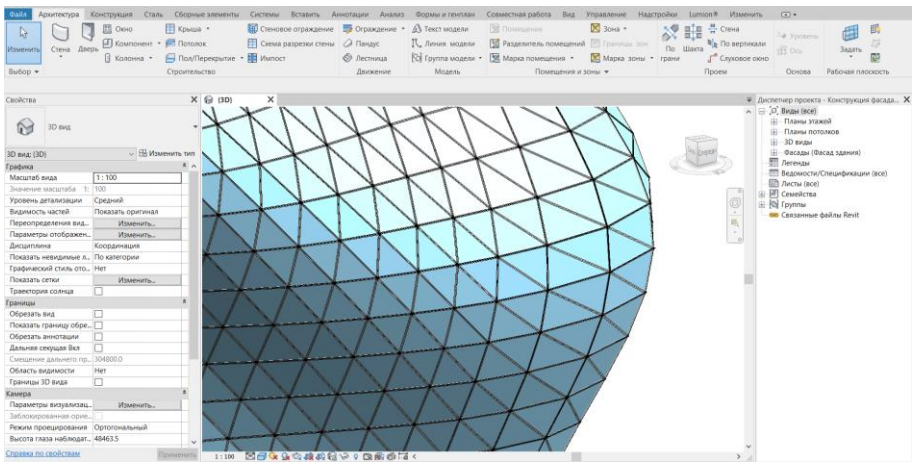
Рис. 7. Python-скрипты для автоматической генерации фасадных элементов:
а) программный код для генерации адаптивных элементов на основе трёх точек;
б) программный код для генерации адаптивных элементов на основе одной точки

На рис. 8 представлен результат работы первого скрипта, обеспечившего распределение фасадных панелей по поверхности геометрии.

На рис. 9 отображен финальный результат комплексной генерации фасадной системы.



а)



б)

Рис. 8. Модель Revit с расставленными фасадными панелями: а) полномасштабное изображение модели; б) увеличенное изображение фасада

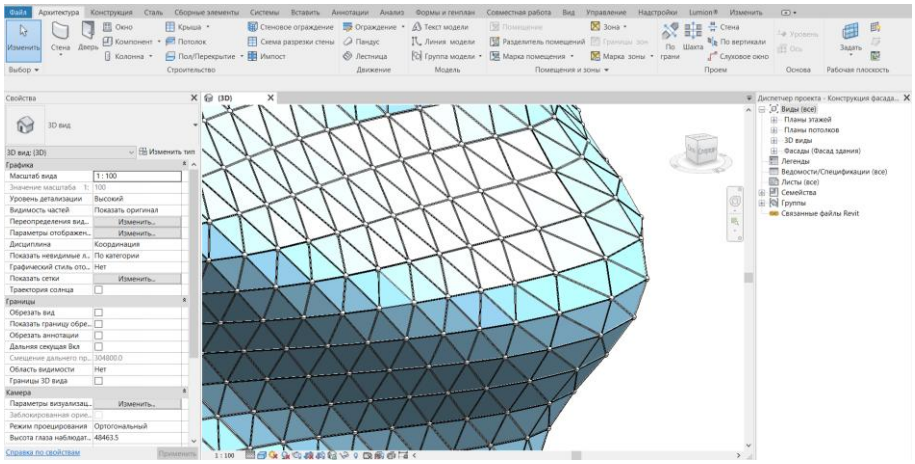


Рис. 9. Увеличенное изображение модели Revit со всеми адаптивными фасадными элементами

Несмотря на то, что поставленные задачи были успешно решены, проведённый анализ выявил ряд ограничений предложенного метода. Временные затраты на полную генерацию фасадной системы модели составили приблизительно два часа. Хотя этот результат значительно превосходит по эффективности ручное размещение элементов, он указывает на необходимость дальнейшей оптимизации вычислительных алгоритмов, особенно при масштабировании на более крупные или детализированные объекты. Кроме того, функциональность разработанного алгоритма на текущем этапе ограничена поддержкой лишь двух типов фасадных компонентов. Это сужает область его применения в проектах, требующих разнообразия элементов или сложной логики замены их типов. Тем не менее, полученные результаты подтверждают практическую ценность предложенного решения: оно обеспечивает точную передачу концептуальной геометрии, автоматизирует рутинные операции и интегрирует свободные формы в параметрическую BIM-среду. В то же время отмечается значительный потенциал для дальнейшего совершенствования, в частности, за счёт оптимизации программного кода, повышения эффективности обработки полигональных данных и расширения библиотеки поддерживаемых адаптивных семейств.

Заключение

Проведённое исследование продемонстрировало эффективность применения связки программных средств – Blender, Dynamo и Autodesk Revit – для решения актуальной задачи интеграции сложной архитектурной геометрии в BIM-среду. Разработанная методика позволила преодолеть ключевое ограничение, возникающее при проектировании концептуально сложной фасадной системы.

Основные результаты работы включают:

1. Создание комплексного алгоритма преобразования полигональных моделей из Blender в семейства Revit с сохранением возможности дальнейшего редактирования стандартными инструментами платформы.

2. Разработка автоматизированных скриптов для генерации адаптивных фасадных элементов позволила значительно сократить временные затраты по сравнению с ручным моделированием.

3. Доказана целесообразность применения искусственного интеллекта для генерации специализированных Python-скриптов, что открывает новые возможности для оптимизации процессов параметрического проектирования.

Несмотря на определенные ограничения, связанные с временными затратами на генерацию и ограниченным выбором типов элементов, предложенная методика демонстрирует существенное преимущество перед традиционными подходами к проектированию сложных фасадных систем. Дальнейшие исследования могут быть направлены на оптимизацию вычислительных алгоритмов и расширение функциональности разработанных скриптов для работы с более сложными геометрическими формами.

Полученные результаты подтверждают практическую значимость методологии и ее потенциал для внедрения в реальные процессы проектирования архитектурных объектов со сложной геометрией.

Источники иллюстраций

Рис. 1. Изображение, выполнено авторами в программе Blender.
Рис. 2, 6. Изображение, выполнено авторами в программной среде Dynamo.
Рис. 3-5, 8-9. Изображения, выполнены е авторами в программной среде Autodesk Revit.
Рис. 7. Изображение, выполнено авторами в программной среде Visual Code.

Список источников

1. Смакаев Р.М. Применение среды визуального программирования Dynamo при разработке проекта здания в Autodesk Revit / Р.М. Смакаев, Т.А. Низина // Основы экономики, управления и права. 2020. №2(21). С. 48-55.

2. Велижанин И.А. Терминологическая неопределенность понятия «искусственный интеллект» в архитектурном проектировании // Architecture and Modern Information Technologies. 2025. №2(71). С. 352-361. URL: https://marhi.ru/AMIT/2025/2kvart25/PDF/21_velizhanin.pdf DOI: 10.24412/1998-4839-2025-2-352-361 EDN: TQKLKI

3. Грызлов Д.В. Использование искусственного интеллекта в программировании / Д.В. Грызлов, Е.Н. Куваева // Научный аспект. 2024. №8 том 17. С. 2121-2127.

References

1. Smakaev R.M., Nizina T.A. *Primenenie sredy vizual'nogo programmirovaniya Dynamo pri razrabotke proekta zdaniya v Autodesk Revit* [Use of Dynamo visual programming environment when developing a building project in Autodesk Revit]. Economy, governance and law basis, 2020, no. 2(21), pp. 48-55.

2. Velizhanin I.A. Terminological ambiguity of the concept of «artificial intelligence» in architectural design. Architecture and Modern Information Technologies, 2025, no. 2(71), pp. 352-361. Available at: https://marhi.ru/AMIT/2025/2kvart25/PDF/21_velizhanin.pdf DOI: 10.24412/1998-4839-2025-2-352-361 EDN: TQKLKI

3. Gryzlov D.V., Kuvayeva E.N. *Ispol'zovanie iskusstvennogo intellekta v programmirovanii. Zhurnal "Nauchnyj aspekt"* [Use of artificial intelligence in programming]. Scientific aspect, 2024, vol. 17, no. 8, pp. 2121-2127.

ОБ АВТОРАХ

Авдюшкин Иван Борисович
Магистрант кафедры «Архитектурного проектирования зданий и сооружений», Новосибирский государственный архитектурно-строительный университет (Сибстрин), Новосибирск, Россия
i_avdyushkin@mail.ru

Акимова Мария Игоревна
Кандидат искусствоведения, доцент, заведующая кафедрой «Архитектурного проектирования зданий и сооружений», Новосибирский государственный архитектурно-строительный университет (Сибстрин), Новосибирск, Россия
m.akimova@sibstrin.ru

ABOUT THE AUTHORS

Avdyushkin Ivan B.
Master's Student of the Department of «Architectural Design of Buildings and Structures», Novosibirsk State University of Architecture and Civil Engineering (Sibstrin), Novosibirsk, Russia
i_avdyushkin@mail.ru

Akimova Maria I.
PhD of Art History, Associate Professor, Head of «Architectural Design of Buildings and Structures», Novosibirsk State University of Architecture and Civil Engineering (Sibstrin), Novosibirsk, Russia
m.akimova@sibstrin.ru

Статья поступила в редакцию 14.10.2025; одобрена после рецензирования 05.12.2025; принята к публикации 10.12.2025.