УДК 378:004
DOI: 10.25686/978-5-8158-2474-4-2025-261-275

Эффективная генерация иерархических уровней детализации: параллельная обработка, инкрементальные обновления, полигональные упрощения на основе видимости

В. Н. Шуткин [1], Н. К. Морозкин [1], В. А. Семенов [1,2], О. А. Тарлапан [1,2,3]
[1] Институт системного программирования им. В.П. Иванникова РАН, Москва, Россия
[2] Московский физико-технический институт, Долгопрудный, Россия
[3] Московский государственный университет имени М.В. Ломоносова, Москва, Россия

Аннотация. В работе рассматривается задача генерации иерархических уровней детализации (HLOD), широко применяемых в приложениях компьютерной графики для рендеринга больших сцен. Теория и практика уровней детализации HLOD являются хорошо проработанными темами исследований, однако вопросам их эффективного вычисления и оперативного обновления не уделялось должного внимания. Вместе с тем данные требования являются принципиальными для цифровых двойников сложных индустриальных объектов и масштабных инфраструктурных программ. Целью работы являлись анализ и оценка подходов к эффективному вычислению уровней детализации HLOD в рамках стратегии агломеративной кластеризации. Для этого разработаны и исследованы специальные техники крупноблочного распараллеливания вычислений, инкрементального обновления уровней детализации и полигональные упрощения с удалением невидимых элементов. Проведенные вычислительные эксперименты с реальными проектными моделями гражданского строительства подтверждают эффективность техник оптимизации, обеспечивающих увеличение производительности генерации HLOD в среднем на порядок.

***Ключевые слова*:** HLOD, 3D Tiles, кластеризация, параллельные вычисления, полигональные упрощения, инкрементальное обновление, удаление невидимой геометрии, интерактивный рендеринг

Effective generation of hierarchical levels of detail: parallel processing, incremental updates, visibility-based polygonal simplifications

V. N. Shutkin [1], N. K. Morozkin [1], V. A. Semenov [1,2], O. A. Tarlapan [1,2,3]
[1] Ivannikov Institute for System Programming RAS, Moscow, Russia
[2] Moscow Institute of Physics and Technology, Dolgoprudny, Russia
[3] Lomonosov Moscow State University, Moscow, Russia

Abstract. The paper considers the problem of generating hierarchical levels of detail (HLOD), widely used in computer graphics applications for rendering large scenes. The theory and practice of levels of detail are well-developed research topics, but the issues of their efficient computation and incremental updating have not received due attention. At the same time, these requirements are fundamental for digital twins of complex industrial objects and large-scale infrastructure programs. The aim of the work was to analyse and evaluate approaches to accelerated computation of HLOD within the framework of the agglomerative clustering strategy. For this purpose, optimization techniques for large-grained parallelization of computations, incremental updating of levels of detail and polygonal simplifications with the removal of invisible elements were developed and studied. The conducted computational experiments with real design models of civil construction confirm the effectiveness of the techniques that provide a total increase in the performance of HLOD generation by an order of magnitude on average.

***Keywords*:** HLOD, 3D Tiles, clustering, parallel computing, polygonal simplification, incremental update, occlusion culling, interactive rendering

Введение

В настоящее время иерархические уровни детализации (hierarchical levels of detail, HLOD) являются одним из наиболее распространенных и эффективных подходов к рендерингу больших трехмерных сцен. Теория и практика методов детализации сцен являются хорошо проработанными темами исследований, которым посвящено большое число работ. В частности, установлено, что иерархические уровни детализации обеспечивают хорошую масштабируемость приложений по отношению к сложности сцен и допускают интерактивный рендеринг больших сцен во внешней памяти. Вместе с тем широкое применение цифровых двойников сложных индустриальных объектов и масштабных инфраструктурных программ диктует новые требования не только к производительности рендеринга, но и к оперативности подготовки уровней детализации, необходимых для этого. Вопросам

эффективного вычисления и оперативного обновления уровней детализации в условиях перманентно изменяемых проектных данных не уделялось должного внимания, а они оказываются критичными для широкого класса приложений. Например, подготовка уровней детализации HLOD для цифрового двойника города с детальными информационными моделями зданий, сооружений и инфраструктуры может занимать сутки процессорного времени, что является неприемлемым для контроля и мониторинга оперативной ситуации.

Целью работы являлись анализ и оценка подходов к эффективному вычислению уровней детализации HLOD в рамках стратегии агломеративной кластеризации. Для этого разработаны и исследованы специальные техники крупноблочного распараллеливания вычислений, инкрементального обновления уровней детализации и полигональных упрощений с удалением невидимых элементов. Проведенные вычислительные эксперименты с реальными проектными моделями гражданского строительства подтверждают эффективность техник оптимизации, обеспечивающих увеличение производительности генерации HLOD в среднем на порядок. Заметим, что результаты работы распространяются и на иерархические динамические уровни детализации (hierarchical dynamic levels of detail, HDLOD), в основе которых лежит та же агломеративная стратегия [1–5]. В ходе работы были также выявлены дополнительные оптимизационные возможности, которые отмечены в соответствующих разделах статьи и которые предстоит исследовать в дальнейшем.

Применение альтернативных представлений геометрических объектов при отображении сцен имеет давнюю и богатую историю. Идея применения упрощённых представлений для дальних объектов сцены с целью снижения нагрузки на аппаратные средства и увеличения производительности рендеринга была высказана ещё в 1976 году [6]. Окончательно она оформилась в метод и связанный с ним термин «уровни детализации» (level of detail, LOD) со стремительным развитием персональных компьютеров и средств компьютерной графики в 1990-е годы. В это же время идея распространяется на случай подготовки нескольких уровней детализации для каждого индивидуального объекта сцены и выбора подходящего уровня в зависимости от расстояния от точки обзора до объекта [7,8]. Однако данный метод порождает ограничения, обусловленные дополнительным анализом расположения объектов при рендеринге сцены и хранением избыточных представлений объектов в видеопамяти.

Чтобы преодолеть данные ограничения и оперировать не большим числом индивидуальных объектов, а их группами, в 2001 году были разработаны иерархические уровни детализации HLOD [9]. В иерархии каждый узел представляет собой упрощённое представление для группы объектов. Как правило, листовые узлы соответствуют точному представлению группы, а по мере продвижения к корню иерархии и объединения объектов все в большие группы точность представления монотонно уменьшается, так что корень соответствует упрощенному представлению всей сцены. Иерархические уровни детализации расходуют больший объем памяти и поэтому критичными для них становятся вопросы организации хранения во внешней памяти и обеспечения быстрого доступа для обеспечения интерактивности приложений и плавной навигации по сцене [10,11]. Возможности предварительной загрузки уровней детализации [12], оптимизации схемы хранения на внешнем диске [13], минимизации трафика между центральным и графическим процессорами [14], а также кэширования наиболее используемых уровней детализации непосредственно в видеопамяти [15] хорошо проработаны и успешно применяются на практике. Необходимо отметить, что методы HLOD плохо подходят для динамических сцен из-за необходимости пересчета уровней детализации при каждом изменении модельного времени. Предварительная подготовка уровней для тысяч и миллионов временных интервалов, на которых происходят изменения сцены, не представляется возможной из-за требуемых огромных ресурсов, а пересчет уровней во время исполнения графического интерактивного приложения трудно реализуем из-за высокой вычислительной сложности.

Ряд работ посвящён уровням детализации для динамической геометрии, под которой подразумевается деформация полигональных моделей. Деформируемые модели с пространственно-временными уровнями детализации рассматриваются в [16]. В [17] представлена концепция «прогрессивной сетки» с учетом временного фактора. Рендеринг деформируемых изоповерхностей в приложениях моделирования динамики жидкости обсуждается в [18]. Непрерывные и иерархические уровни детализации для толп персонажей со скелетной анимацией рассматриваются в [19] и [20] соответственно.

В 2020-е годы иерархические уровни детализации получили дальнейшее развитие в серии работ, адресованных проблеме рендеринга больших динамических сцен. С этой целью был выделен класс детерминированных псевдодинамических сцен с доминированием дискретных событий. Данный класс находит важные приложения, связанные, в частности, с цифровыми двойниками сложных индустриальных проектов и масштабных инфраструктурных программ [21, 22]. В подобных приложениях динамика сцены определяется заранее подготовленным календарно-сетевым графиком проектных работ, в соответствии с которым элементы конструкций и оборудования устанавливаются в предопределенное положение или удаляются. В сцене могут присутствовать объекты с иным характером динамики, например техника или персонал, перемещаемые по строительной площадке, однако доля таких объектов относительно невелика. Для больших псевдодинамических сцен был разработан метод иерархических динамических уровней детализации во внешней памяти HDLOD с поддержкой консервативного (гарантирующего заданную пространственную и временную точность) и интерактивного (стремящегося обеспечить максимально возможную частоту генерации изображений при приемлемом уровне реализма) режимов отображения [1–3]. Также был предложен многовариантный пакетный метод, обеспечивающий одновременный рендеринг нескольких динамических сценариев [4].

Методы упрощения геометрических моделей, прежде всего полигональных сеток как наиболее распространенного вида объектов в приложениях компьютерной графики, играют ключевую роль в методах детализации LOD, HLOD и HDLOD. Полигональные упрощения представляют и самостоятельный интерес. В монографии [23] и тематическом обзоре [24] систематически изложены базовые идеи и принципы, лежащие в их основе, а именно: стягивание рёбер [25], удаление вершин [26] и кластеризация вершин [7,27]. Иногда особенности сцены, например, обусловленные фотограмметрической реконструкцией здания или целого города, позволяют выполнить дополнительные упрощения материалов и текстур и тем самым оптимизировать представление уровней детализации для последующего рендеринга [28–31].

В последние годы теория и практика уровней детализации получили дальнейшее развитие. Как результат иерархические уровни детализации были распространены на нейронные поля излучения NeRF [32], гауссианы 3DGS [33], облака точек [34], воксельные и гибридные представления [35, 36]. В работе [37] предложено гибридное представление полигональная сетка-NeRF.

В последние годы графические интерфейсы (API) претерпевают значительное развитие, благодаря чему становится возможным выбирать необходимое геометрическое представление объекта непосредственно на графическом процессоре. Определенную известность получили Unreal Engine Nanite [38] и подобные ей технологии [39]. Они обеспечивают рендеринг больших полигональных сеток, как правило, представляющих собой топологически замкнутые двумерные поверхности, на основе подготовленных деревьев кластеров. Кластеризация сеток осуществляется сверху-вниз с последующим упрощением фрагментов и исключением разрывов на границах. Обход загруженного в видеопамять дерева кластеров и рендеринг отобранных кластеров осуществляются непосредственно на графическом процессоре, причем загрузка данных кластеров происходит напрямую с диска в видеопамять. В статье [40] предлагается схема управления памятью графического процессора при рендеринге больших полигональных сеток. Статья [41] адресована выборочному уточнению большой сетки при трассировке лучей.

В настоящей работе обсуждается задача рендеринга больших сцен в более общей постановке. Предполагается, что сцена представлена набором полигональных моделей в виде так называемого «полигонального супа» без каких-либо дополнительных требований к топологии, а применяемые методы иерархических уровней детализации HLOD носят универсальный характер, не требуют аппаратной поддержки специальных функций и могут быть применены, в частности, для визуализации сложных сцен в web-браузерах на мобильных устройствах.

Постановка задачи

Задачами работы являлись разработка и экспериментальное исследование техник оптимизации вычисления уровней детализации HLOD. Поскольку техники разработаны применительно к

конкретной стратегии агломеративной кластеризации объектов [5] со своими особенностями представления уровней детализации и правилами их формирования, приведем ее краткое описание.

Пусть сцена S определена в трёхмерном евклидовом пространстве E^3 и представлена как набор объектов $s(g_s, \iota_s) \in S$. Каждый объект имеет фиксированное геометрическое представление g_s, а также уникальный идентификатор ι_s. В рамках данной работы будем считать, что g_s задаётся в виде полигональной сетки треугольников, а ι_s может быть задан любым способом: строка, номер, GUID или UUID.

Будем рассматривать генерацию HLOD как построение дерева кластеров. Каждый кластер $c(g_c, b_c, \varepsilon_c)$ имеет фиксированное геометрическое представление g_c, ограничивающий объём b_c и геометрическую погрешность ε_c. Для кластеров определено отношение агломерации $\prec$. Запись $c' \prec c$ означает, что кластер c' является ребёнком кластера c в дереве, а также то, что кластер c' вошёл в кластер c во время кластеризации. Пусть у кластера c имеется n детей $\forall i = 1, \dots n\ c_i \prec c$. Это означает, что геометрическое представление g_c получено путём объединения и упрощения представлений всех дочерних кластеров, а именно $g_c = \bigcup_{i=1,\dots n} g_{c_i}$, после чего g_c упрощено с геометрической погрешностью ε_c. Листовые кластеры получаются путём объединения объектов, а кластеры последующих уровней – путём объединения кластеров (и, возможно, объектов). Для листового кластера его геометрическое представление получается путём объединения представлений вошедших в него объектов без упрощения. Если в листовой кластер c вошли n объектов $s_1, \dots s_n$, то $g_c = \bigcup_{i=1,\dots n} g_{s_i}$. Идентификатор объекта ι_s сохраняется для каждого треугольника в геометрии кластера. Ограничивающий объём кластера всегда охватывает ограничивающие объёмы дочерних кластеров: $\forall c' \prec c\ b_{c'} \subseteq b_c$. Геометрическая погрешность дочернего кластера не может превышать геометрическую погрешность родительского кластера: $\forall c' \prec c\ \varepsilon_{c'} \leq \varepsilon_c$. Листовые кластеры содержат точную (неупрощённую) геометрию: $\forall c\colon \nexists c' \prec c\ \varepsilon_c = 0$. Пример дерева кластеров представлен на рис. 1.

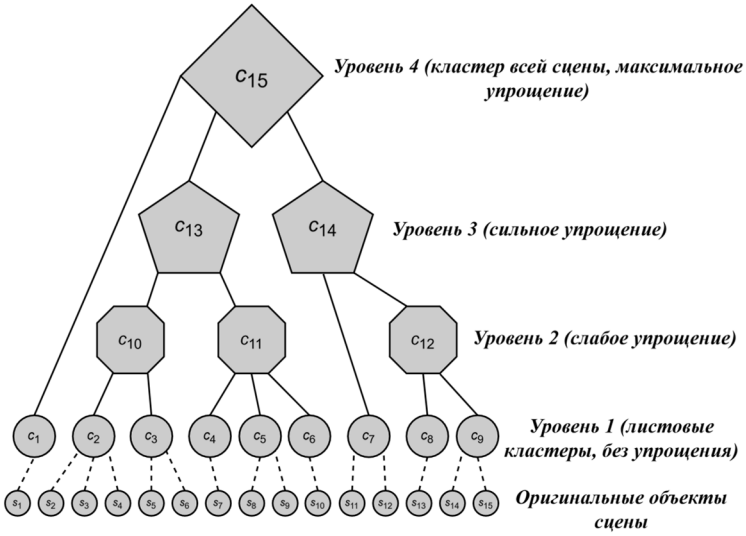


Рисунок 1. Пример дерева кластеров

Опишем общую стратегию построения иерархии уровней детализации HLOD. На первом этапе строится дерево кластеров в результате агломеративной кластеризации объектов снизу вверх. Процедура кластеризации стартует с оригинальных объектов сцены и затем распространяется на сформированные листовые и промежуточные кластеры вплоть до достижения единого корневого кластера всей сцены. На втором этапе вычисляются геометрические представления кластеров путем объединения представлений дочерних кластеров и их последующего упрощения. Данный этап является самым вычислительно затратным.

Теория

Перечислим основные подходы к эффективному вычислению уровней детализации HLOD на основе описанной выше стратегии агломеративной кластеризации.

Прежде всего, это ускорение полигонального упрощения, которое должно обеспечить меньшие вычислительные затраты на генерацию кластеров приемлемого качества. Причем качество следует оценивать в многокритериальной постановке, которая бы учитывала геометрическую погрешность упрощенной сетки, сохранность ее визуальных особенностей, а также сложность представления, влияющую на производительность рендеринга. К сожалению, на практике не удается контролировать все критерии из-за особенностей методов упрощения, поэтому представляется более перспективным выбор одного из известных методов, (1) обладающего невысокой вычислительной сложностью, (2) гарантирующего соблюдение заданной геометрической погрешности и (3) допускающего высокую степень упрощения (4) при сохранении визуальных особенностей полигональной сетки.

Важным подходом к оптимизации вычисления уровней детализации HLOD, безусловно, является распараллеливание. Видятся две основные схемы распараллеливания: так называемая мелкозернистая (fine-grained) на уровне подготовки упрощенного полигонального представления для одного обрабатываемого кластера и крупноблочная (coarse-grained) на уровне подготовки представлений для целых групп кластеров. Обе схемы имеют свои достоинства и недостатки.

Мелкозернистая схема предполагает применение методов полигонального упрощения, допускающих эффективное распараллеливание. Это налагает жесткие ограничения на выбор самих методов, а также требует эффективных процедур декомпозиции исходной сетки и объединения результатов упрощения, которые обычно требуют значительных вычислительных ресурсов и выполняются преимущественно последовательным образом. Как результат эффект от распараллеливания вычислений для частей сетки нивелируется затратами на их пред- и постобработку.

Более перспективной представляется крупноблочная схема, поскольку ее реализация не зависит от особенностей применяемых методов полигонального упрощения и не ограничивает в их выборе, диктуемом иными перечисленными выше факторами. Данная схема распараллеливания применима к группам кластеров, относящихся к одному уровню детализации, поскольку их геометрические представления зависят только представлений дочерних кластеров. Данная схема не требует какой-либо пред- и постобработки геометрических представлений и реализуется на доступных мультиядерных и мультипроцессорных архитектурах. В настоящей работе рассматривается одна из возможных реализаций данной схемы.

Наконец, третий обсуждаемый подход состоит в инкрементальном обновлении уровней детализации HLOD. Данная возможность важна для графических приложений поддержки коллективной работы, в ходе которой участниками перманентно вносятся изменения в общие проектные данные. Инкрементальный пересчет уровней детализации целесообразен для цифровых двойников сложных индустриальных объектов, параметры которых актуализируются и уточняются на протяжении всего их жизненного цикла. Применение подхода мотивировано для приложений САПР, в которых сложные 3D-модели подлежат постоянным редакторским правкам и нуждаются в оперативном отображении результатов. Если изменения носят локальный характер и распространяются на относительное небольшое число кластеров, инкрементальное обновление имеет хорошую перспективу.

Опишем более подробно техники оптимизации, реализующие перечисленные подходы к ускорению вычисления уровней детализации HLOD.

1. Полигональные упрощения

Методы полигонального упрощения различаются не только указанными выше базовыми принципами (стягивание ребер, удаление вершин и кластеризация), но и параметризацией, способами организации вычислений (итерационные, регулярные), условиями прерывания [23]. Подобные различия приводят к ситуации, когда методы имеют разную вычислительную сложность и генерируют сетки разного качества. В рамках проводимого исследования не ставилась цель комплексного сравнения методов с учетом сложности многокритериального сравнения результатов упрощения.

Вместо этого было принято решение об использовании популярного метода стягивания рёбер с квадратичной метрикой погрешности [25]. В сочетании с процедурой удаления невидимых граней данный метод удовлетворяет всем перечисленным выше требованиям. Метод реализует итеративное стягивание рёбер полигональной сетки, пока не будет достигнуто целевое число треугольников. Для

минимизации геометрической погрешности рёбра обрабатываются в определённом порядке: в первую очередь стягиваются те ребра, которые вносят наименьшие изменения в сетку. Для этого каждому ребру приписывается стоимость и рёбра сортируются по возрастанию стоимости. В данном случае используется квадратичная метрика, оценивающая кривизну поверхности. Стягивание ребра приводит к вырождению опирающихся на него треугольников и, следовательно, к сокращению числа треугольников в сетке.

Для оптимизации базового метода было предложено дополнительно использовать процедуру удаления невидимых граней. Действительно, в сценах гражданских зданий и индустриальных сооружений обычно большая часть элементов невидима снаружи из-за наличия стен, перекрытий, коробов инженерного оборудования. Другим фактором учета невидимых граней является сама организация стратегии агломеративной кластеризации. Поскольку при объединении полигональных представлений кластеров значительная часть граней может быть перекрыта другой и впоследствии удалена, процесс объединения кластеров предшествует их упрощению.

Предлагаемая техника удаления невидимых граней в уровнях детализации HLOD налагает дополнительное ограничение на их использование в ходе рендеринга – камера всегда должна находиться снаружи отображаемых кластеров. Данное требование естественным образом реализуется в случае консервативного рендеринга сцены, поскольку отображение ближайших объектов предполагает использование нижних листовых кластеров с точной и полной геометрией. В случае интерактивного рендеринга необходимые кластеры могут быть еще не подгружены и тогда используются доступные верхние кластеры с упрощенным представлением. Когда камера оказывается внутри них, наблюдатель может не увидеть важные удаленные артефакты. Данное ограничение не является столь критичным, если навигация по сцене осуществляется плавно и приложение успевает подгружать необходимые кластеры. Ограничение оказывается принципиальным, если приложение предусматривает функции пространственного анализа сцены с использованием отсекающих плоскостей и объектных фильтров. Удаленные грани и объекты не могут быть реконструированы в представлении HLOD и наблюдатель, возможно, упустит что-то важное в деталях отображаемой сцены.

Указанное ограничение компенсируется важными преимуществами. Во-первых, сокращается время вычисления всей иерархии HLOD за счет удаления значительной части внутренних граней в полигональных представлениях кластеров. Во-вторых, при обеспечении того же внешнего визуального качества кластеров уменьшается их сложность и возрастает скорость рендеринга. Меньшее число полигонов в представлении кластеров снижает требования и к объемам основной памяти и видеопамяти, требуемым для их размещения, что уменьшает число обменов и также способствует повышению производительности рендеринга.

Удаление внутренних граней может быть организовано несколькими способами:

– применением техники, аналогичной затенению фонового освещения (англ. ambient occlusion) [42]. Из каждой грани кластера выпускаются лучи, и если хотя бы один луч достигает «неба» (не встретил препятствий на своем пути), то эта грань считается видимой снаружи. В оригинальной технике фонового затенения для оценки степени освещённости определяется доля лучей, достигших неба. В нашем случае достаточно провести один непрерывный луч, чтобы сделать вывод о видимости грани;

– построением ограничивающей сферы вокруг кластера и бросанием лучей из точек сферы (распределенных равномерно или каким-либо другим способом) к ее центру с целью обнаружения внешних граней. Грани, не идентифицированные как внешние, удаляются из представления кластера;

– пакетным рендерингом представления кластера с разных ракурсов и применением идентификаторов граней в качестве цветовых индексов. Грани, идентификаторы которых попали хотя бы в один фрейм буфер, считаются внешними, а остальные – внутренними и подлежат удалению.

Отметим, что идентификация и удаление внутренних граней кластеров требуют дополнительных затрат и могут, в конечном счете, нивелировать выигрыш в скорости подготовки уровней детализации HLOD от упрощения кластеров. Поэтому алгоритмические и реализационные детали техник удаления невидимых граней исключительно важны для оценки суммарного эффекта. Прежде всего, достоверность определения внутренних граней, напрямую связанная с числом построенных и пересекаемых лучей или количеством ракурсов для пакетного рендеринга, может радикально влиять

на соотношение оптимизационных факторов. При этом сам выбор базовой техники не играет принципиальной роли. Как показали результаты профилирования программ, реализующих данные техники, основные затраты приходятся не на массовые пересечения лучей или пакетный рендеринг параллельными средствами GPU, а на формирование единого массива идентификаторов внешних и внутренних объектов на CPU. В силу указанных факторов целесообразность применения данной оптимизации нуждается в экспериментальном подтверждении.

2. Распараллеливание

После того как кластеризация объектов сцены завершена и дерево HLOD построено, начинается процесс формирования геометрических представлений кластеров с использованием описанных выше методов полигонального упрощения и удаления невидимых граней. Из-за объема данных геометрические представления хранятся во внешней памяти и загружаются в оперативную лишь по мере необходимости. Дерево кластеров целиком размещается в оперативной памяти, поскольку содержит лишь информацию об идентификаторах кластеров, их ограничивающих объёмах и связях родитель-ребёнок.

Напомним, что кластеры формируются снизу-вверх в соответствии с уровнями в дереве. Сначала обрабатываются кластеры первого нижнего уровня, в результате чего формируются точные и полные геометрические представления из оригинальных моделей объектов сцены. Геометрические представления кластеров второго уровня формируются из представлений кластеров первого уровня. Геометрические представления кластеров третьего уровня формируются из представлений кластеров второго и, возможно, первого уровня (рис. 1). Процесс продолжается до тех пор, пока не будет сформировано представление самого верхнего и единственного корневого кластера, соответствующего всей сцене.

Описанный порядок формирования кластеров позволяет применить схему крупноблочного распараллеливания вычислений. Обработка кластеров одного уровня может осуществляться в нескольких параллельных потоках, поскольку между ними нет зависимостей по данным, а они зависят только от ранее подготовленных представлений дочерних кластеров нижних уровней.

При обработке каждого кластера выполняются следующие действия:

– полигональные сетки дочерних кластеров загружаются в оперативную память и добавляются в кэш, если не находились там ранее;

– полигональные сетки объединяются в единое представление новой сетки;

– к новой сетке применяется процедура упрощения с соответствующими целевыми параметрами сложности и визуального качества;

– полученная полигональная сетка кластера добавляется в кэш и сохраняется во внешней памяти;

– сетки дочерних кластеров удаляются из кэша, проверяется свободный объем кэша и при необходимости применяется политика вытеснения для размещения новых сеток и обработки следующих кластеров.

Описанная схема распараллеливания допускает разные способы реализации. Во-первых, можно отказаться от реализации кэша и подгружать сетки дочерних кластеров из внешней памяти непосредственно перед обработкой родительского и высвобождать сразу после ее завершения. Это обеспечит неблокируемый доступ к сеткам, но может привести к необходимости их повторного чтения с внешнего диска. Во-вторых, планирование заданий на обработку кластеров и их распределение по вычислительным узлам (ядрам или процессорам) может быть осуществлено разными способами. Наиболее простой состоит в предварительном назначении обрабатываемых кластеров одного уровня отдельным вычислительным узлам. Поскольку число листовых кластеров обычно много больше, чем вычислительных узлов, степень параллелизма при обработке кластеров нижних уровней высока, однако падает по мере продвижения к верхним уровням и корню дерева. Другим фактором, препятствующим эффективному распараллеливанию, является неоднородность кластеров, выражающаяся в разной сложности полигональных сеток и приводящая к разному времени их обработки. В зависимости от метода кластеризации степень неоднородности может существенно различаться, и для обеспечения максимальной загрузки вычислительных узлов целесообразно переназначать их на новые задания по мере завершения предыдущих, используя пул тредов. Обычно

программные средства распараллеливания вычислений позволяют реализовать оба варианта относительно просто [43].

3. Инкрементальные обновления

Как отмечено выше, существует довольно широкий класс графических интерактивных приложений, оперирующих большими 3D-моделями и нуждающихся в быстрых средствах пересчета уровней детализации HLOD для их отображения в оперативном режиме. Обсудим возможности и алгоритмические детали применения техники инкрементального обновления для этих целей.

Эффективность применения инкрементальных обновлений во многом зависит от характера изменений в сцене. В случае локальных изменений, затрагивающих небольшое число объектов сцены и не меняющих их пространственное положение, габариты и ограничивающие объемы существенно, появляется возможность обновления небольшого числа существующих кластеров без пересчета всей HLOD-иерархии. Прежде всего, это листовые кластеры с изменившимися объектами, а также их родители вплоть до корня дерева при условии, что их представления действительно чувствительны к исходным изменениям. Если это не так, то процесс распространения изменений вверх по дереву и пересчет родительских кластеров не имеет смысла.

В случае глобальных изменений, затрагивающих относительно большое число объектов сцены и существенно меняющих ограничивающие объемы кластеров и плотность заполнения, применение прежней HLOD-иерархии становится неэффективным. При нарушенной пространственной кластеризации отсечение конусом видимости не приведет к желаемому результату и заведомо лишние объекты будут включены в процесс рендеринга. При деградации иерархии ее следует пересчитать заново, проведя повторную кластеризацию объектов и формируя геометрические представления кластеров. К сожалению, трудно выработать критерии для принятия решения о пересчете иерархии, который, с одной стороны, необходим для последующего производительного рендеринга сцены и обеспечения интерактивности графического приложения, а с другой стороны, потребует значительных вычислительных затрат и, скорее всего, прервет работу пользователей.

В нашей работе [5] предлагается подход к пересчету HLOD-иерархии, основанный на оценке ее качества непосредственно в процессе инкрементального обновления. В качестве критериев предлагается использовать показатели плотности кластеров (количество объектов в единице ограничивающего объема кластера) и скорости упрощения кластеров (количество удаленных полигонов или треугольников при единичном увеличении погрешности). Сама иерархия строится при ограничениях на объем кластера, число дочерних кластеров и сложность геометрического представления. Решение о пересчете принимается, если не удается обновить кластеры иерархии с требуемым качеством при соблюдении указанных ограничений.

Приведем краткое описание ранее предложенной техники инкрементального обновления HLOD-иерархии, используемой в проведенных вычислительных экспериментах. Инкрементальное обновление реализуется в виде процесса обработки и распространения изменений следующих типов: добавление новых объектов, удаление объектов и изменение геометрии или положения существующих объектов.

При добавлении объектов в сцену анализируется возможность их включения в существующие листовые кластеры и распространения изменений вверх по иерархии с соблюдением ограничений построения иерархии. Если это оказывается невозможным, изменения считаются глобальными и выносится вердикт о пересчете иерархии.

При удалении объектов оценивается качество листовых кластеров с учетом возможного уменьшения ограничивающих объемов и сокращения полигонов в них. Вырожденные кластеры с нарушенными критериями качества удаляются, а приписанные им объекты перераспределяются по соседним кластерам. Если подобные преобразования не удается реализовать с перечисленными выше требованиями, процесс прерывается и принимается решение о полном пересчете иерархии.

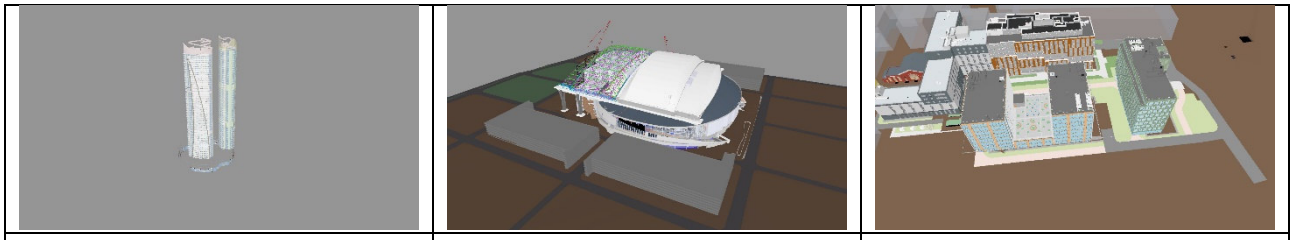
Изменения геометрии или положения удобно интерпретировать как удаление соответствующего объекта и его последующее добавление с новой геометрией и новым расположением в сцене. Поэтому обработка подобных изменений включает все вышеописанные действия. В случаях, когда изменения затрагивают только геометрическое представление объекта, а его положение и габариты не меняются, обработка сводится к анализу сложности обновленного кластера и переоценке скорости упрощения.

Процесс распространения изменений реализуется с помощью меток обновления, которые позволяют контролировать согласованность изменений для всех кластеров дерева.

Результаты экспериментов

Для каждого из описанных выше трех способов оптимизации была проведена серия вычислительных экспериментов, которая позволила количественно оценить эффект как результат сокращения процессорного времени генерации иерархических уровней детализации HLOD.

Все эксперименты проводились на компьютере с типовой конфигурацией Intel(R) Core(TM) i9-9880 CPU @ 3.60GHz (8 ядер, 16 потоков), 32 ГБ ОЗУ, 2 ТБ NVME SSD. В экспериментах использовался единый набор тестовых сцен, реконструированных по реальным цифровым моделям строительных проектов с соответствующими архитектурными, конструктивными и инженерными элементами (рис. 2). Сцена многоэтажного здания содержала 146 461 объект с тесселированной геометрией в виде полигональных сеток и общим числом треугольников 3 426 267. Сцены стадиона и отеля содержали 41 451 и 47 160 объектов, 7 960 512 и 29 131 139 треугольников соответственно.



(а) (б) (в)

Рисунок 2. Цифровые модели строительных объектов: а) многоэтажное здание; б) стадион; в) отель

Эксперименты были организованы следующим образом: проводилась генерация иерархических уровней детализации HLOD в соответствии с описанной выше стратегией, замерялось время генерации упрощённых представлений кластеров. На первом уровне дерева кластеры формируются из оригинальных представлений объектов сцены, поэтому интерес представляют уровни дерева, начиная со второго, на которых применяются вычислительно затратные упрощения полигональных сеток.

1. Распараллеливание

Первая серия вычислительных экспериментов была связана с применением описанной выше техники распараллеливания. Напомним, что обработка уровней выполняется последовательно. При этом кластеры одного уровня обрабатываются параллельно в несколько потоков, поскольку их представления зависят только от кластеров нижних уровней. После завершения всех потоков на очередном уровне начинается параллельная обработка кластеров следующего уровня. В таблице 1 приведено сравнение времени генерации уровней детализации без распараллеливания и с распараллеливанием для трёх тестовых сцен. В ней же указано количество кластеров на каждом уровне дерева, начиная со второго.

Таблица 1. **Эксперименты с распараллеливанием**

Модель	Кол-во кластеров на уровнях иерархии	Последовательное выполнение	Параллельное выполнение	Ускорение
Здание	17, 9, 4, 2, 1, 1	7м 57с	3м 2с	2.6x
Стадион	27, 15, 8, 5, 2, 1, 1	36м 28с	19м 56с	1.8x
Отель	127, 64, 32, 16, 8, 4, 2, 1	12ч 51м 38с	3ч 37м 18с	3.6x

2. Удаление внутренних граней

Вторая серия экспериментов предназначалась для оценки техники полигонального упрощения с удалением внутренних граней. Обнаружение внутренних граней было реализовано с использованием описанной выше процедуры бросания лучей. Лучи выпускались из точек, равномерно распределенных по ограничивающей сфере кластера, в направлении ее центра. Для поиска пересечений лучей с гранями полигональной сетки применялась библиотека Embree [44], в которой данные операции эффективно

реализованы с использованием пространственного индексирования на основе иерархий ограничивающих объёмов. Грани, в которые попал хотя бы один луч, помечались как внешние, остальные – как внутренние. После завершения процесса бросания лучей и поиска пересечений все грани, помеченные как внутренние, удалялись. Для каждого кластера поиск и удаление внутренних граней осуществлялись сразу после формирования его полигональной сетки (путём объединения сеток дочерних кластеров) и перед запуском процедуры упрощения полигональной сетки.

Как отмечалось выше, достоверность идентификации внутренних граней описанным способом существенно зависит от числа выпущенных лучей. С увеличением числа лучей достоверность будет монотонно повышаться, однако процесс потребует большее время обработки. При этом будет обнаружено больше внешних граней и меньше внутренних. Поэтому число лучей в проводимых экспериментах варьировалось, начиная от десяти тысяч и заканчивая миллиардом лучей (на каждый кластер). Для каждой тестовой сцены замерялось время генерации всех уровней (начиная со второго) с использованием техники распараллеливания. Также определялись общее количество удалённых граней во всех кластерах, процент от общего числа граней в кластерах без удаления внутренних граней и размер полученных иерархических уровней детализации, сохранённых на диске в формате 3D Tiles [45]. Полученные результаты приведены в таблице 2. В качестве референсной информации также укажем общее число граней в кластерах HLOD-иерархии без применения процедуры удаления внутренних граней: 6 826 213 (здание), 13 853 526 (стадион) и 57 705 028 (отель).

Таблица 2. **Эксперименты с удалением внутренних граней**

		Без удаления внутренних граней	10 000 лучей	100 000 лучей	1 000 000 лучей	10 000 000 лучей	100 000 000 лучей	1 000 000 000 лучей
Здание	Время генерации	3м 2с	0.58с	2.41с	16.68с	1м 11с	4м 5с	24м 57с
	Кол-во удалённых граней	0	3416063 (50%)	3378397 (49%)	3267249 (48%)	3072979 (45%)	2815698 (41%)	2589898 (38%)
	Размер файлов	969 МБ	482 МБ	508 МБ	554 МБ	613 МБ	654 МБ	714 МБ
Стадион	Время генерации	19м 56с	0.86с	1.56с	13.77с	57с	5м 59с	1ч 3м 22с
	Кол-во удалённых граней	0	7942012 (57%)	7905560 (57%)	7778853 (56%)	7567357 (55%)	7267930 (52%)	6889677 (50%)
	Размер файлов	2.24 ГБ	1.26 ГБ	1.27 ГБ	1.3 ГБ	1.35 ГБ	1.41 ГБ	1.47 ГБ
Отель	Время генерации	3ч 37м 18с	2.72с	3.41с	10.81с	1м 4с	7м 44с	1ч 7м 59с
	Кол-во удалённых граней	0	29103008 (50%)	29044720 (50%)	28876971 (50%)	28480811 (49%)	27715526 (48%)	26732145 (46%)
	Размер файлов	11.4 ГБ	5.78 ГБ	5.81 ГБ	5.86 ГБ	5.98 ГБ	6.18 ГБ	6.42 ГБ

3. Инкрементальные обновления

Финальная серия экспериментов предназначалась для оценки техники инкрементального обновления иерархических уровней детализации HLOD при существенно локальных изменениях в сцене. Эксперименты были организованы следующим образом: некоторые объекты сцены помечались как изменившиеся, после чего запускалось обновление уровней детализации. По набору изменившихся объектов определялись листовые кластеры первого уровня, требующие обновления. Затем итеративно определялись кластеры, требующие обновления на следующих уровнях, согласно следующему

принципу: кластер требует обновления, если хотя бы один из его детей изменился. После чего запускался процесс генерации представлений для помеченных кластеров. Заметим, что положение объектов, их габариты и ограничивающие объемы не менялись, поэтому все вычисления выполнялись с прежней иерархией кластеров. Было реализовано два способа моделирования локальных изменений в сцене: путем случайного отбора объектов и путем выбора объектов, компактно расположенных в центре сцены при минимизации их общего огранивающего объёма. Эксперименты проводились для следующих случаев: когда изменился один случайный объект, когда изменилось 5 % объектов сцены (смежных и случайных) и когда поменялось 10 % объектов сцены (смежных и случайных). Замерялось время генерации кластеров всех уровней, начиная со второго, а также число кластеров второго уровня, которые потребовали обновления. В таблице 3 приведены полученные результаты. В экспериментах полигональные упрощения кластеров проводились параллельно. Техника удаления внутренних граней не применялась, чтобы не исказить эффект от самих инкрементальных обновлений.

Таблица 3. **Эксперименты с инкрементальными обновлениями**

		Полная генерация	Изменения 1 объекта	5 % смежных изменений	5 % случайных изменений	10 % смежных изменений	10 % случайных изменений
Здание	Время	3м 2с	1м 6с	1м 20с	2м 16с	1м 18с	2м 26с
	Кол-во объектов	146464	1	7323	7323	14646	14646
	Кол-во кластеров уровня 2	17	1	3	12	3	12
Стадион	Время	19м 56с	3м 1с	12м 53с	20м 1с	12м 59с	20м 32с
	Кол-во объектов	41451	1	2072	2072	4145	4145
	Кол-во кластеров уровня 2	27	1	7	24	8	26
Отель	Время	3ч 37м 18с	10м 27с	44м 17с	1ч 36м 1с	1ч 48с	2ч 9м 20с
	Кол-во объектов	47160	1	2358	2358	4716	4716
	Кол-во кластеров уровня 2	127	1	9	39	13	53

Обсуждение результатов

Обсудим результаты в порядке описания проведенных вычислительных экспериментов. Главное внимание при этом уделим применяемым техникам оптимизации и факторам, влияющим на производительность генерации иерархических уровней детализации HLOD.

1. Распараллеливание

Прежде всего, отметим не столь значительное ускорение вычислительных процедур обработки кластеров в результате их распараллеливания. На компьютерной архитектуре с 8-ядерным процессором ускорение на тестовых сценах варьировалось в диапазоне 1.8–3.6. С одной стороны, даже двух- и трёхкратное ускорение может быть значимым, особенно в тех случаях, когда подготовка HLOD занимает часы процессорного времени (например, тест с отелем в таблице 1) и визуализация является частью непрерывного рабочего процесса. С другой стороны, применяемая схема крупноблочного распараллеливания допускает лучшее масштабирование. Полученные результаты объясняются тем, что для экспериментов намеренно были выбраны сцены среднего размера (50000 – 150000 объектов). Построенные иерархии ограничивались 7-9 уровнями; верхние четыре уровня, обрабатываемые существенно последовательно, имели низкое среднее число кластеров (2, 2.25 и 3.75 для сцен здания, стадиона и отеля соответственно). Это приводило к деградации применяемой схемы

распараллеливания с предварительным планированием заданий из-за низкой загрузки вычислительных узлов на верхних уровнях иерархии. Для больших сцен, соответствующих, например, проектам промышленного строительства и содержащих миллионы объектов, следует ожидать лучшие результаты из-за более глубокой иерархии и эффективной обработки существенно большей части уровней. Более сложный тест со сценой отеля хорошо демонстрирует эту закономерность.

Вместе с тем существуют и другие факторы, препятствующие эффективному распараллеливанию процедур генерации HLOD. В теоретическом разделе среди них обсуждались неоднородность кластеров, обусловленная особенностями оригинальной сцены и ограничениями метода кластеризации, а также повторное чтение полигональных сеток из внешней памяти при обработке каждого следующего уровня иерархии. Организация пула тредов с динамическим назначением заданий на обработку кластеров и кэширование полигональных сеток в оперативной памяти могут обеспечить более полную загрузку вычислительных узлов и достичь большей производительности. Данные возможности нуждаются в дополнительном экспериментальном подтверждении, основанном на конкретных программных реализациях.

2. Полигональные упрощения с удалением внутренних граней

Полигональные упрощения с удалением невидимых граней приводят к более значительному и предсказуемому результату, который выражается как в ускорении времени генерации HLOD, так и в меньшем объеме оперативной памяти и видеопамяти, необходимой для хранения и отображения уровней детализации.

Ускорение достигается за счёт сокращения числа граней в полигональной сетке и более быстрой последующей работы алгоритма упрощения (с меньшим числом итераций, необходимым для достижения целевого числа треугольников, и меньшим размером вспомогательных индексных структур, поиск в которых, как правило, осуществляется за логарифмическое время от числа элементов). Удаление внутренних граней имеет накопительный эффект в рамках применяемой стратегии кластеризации. Поскольку при объединении представлений кластеров часть треугольников перекрывается и удаляется, на каждом следующем уровне иерархии всё меньше треугольников участвует в полигональных упрощениях. Упрощение уровней детализации HLOD при их прежнем визуальном качестве улучшает также и производительность рендеринга.

Конкретные значения ускорения существенно зависят от достоверности идентификации внутренних граней и варьируются в диапазоне от нескольких единиц до трех порядков (см. таблицу 2). Максимальное ускорение в 4 793 раза достигается на тестовой сцене отеля при числе лучей 10 000, однако достоверность определения внутренних граней невелика и, по крайней мере, 4 % граней идентифицируется неверно (оценка получается путем сравнения доли внутренних или внешних граней, обнаруженных при числе лучей 10 000 и 1 000 000 000). Это может привести к нежелательным артефактам, например к дырам во внешних поверхностях или удаленным видимым элементам внутри сцены. При числе лучей 10 000 000 вычислительные затраты на определение пересечений возрастают и частично нивелируют эффект от сокращения числа полигонов. Ускорение на тестовой сцене отеля составляет уже 203. Дальнейшее увеличение числа лучей до 1 000 000 000 приводит к снижению ускорения, а для сцен со зданием и стадионом эффект от применяемой оптимизации и вовсе становится отрицательным.

Результаты экспериментов показывают, что для разных тестовых сцен и разных кластеров одни и те же оценки достоверности достигаются при существенно разном числе лучей. Поэтому на практике их довольно сложно контролировать. Возможная итеративная схема для получения оценки заключается в кратном увеличении числа лучей для каждого обрабатываемого кластера до тех пор, пока изменения в доле внутренних или внешних граней не зафиксируются с заданной погрешностью. В любом случае, компромисс между производительностью генерации уровней детализации HLOD и достоверностью их отображения должен искаться с учетом требований, предъявляемых к графическому приложению.

3. Инкрементальные обновления

Наконец, проведенные эксперименты с инкрементальными обновлениями HLOD подтверждают перспективность данной техники для приложений, нуждающихся в оперативном пересчете уровней

детализации. Ускорение на тестовых сценах здания, стадиона и отеля с одним изменившимся объектом составило 2.75, 6.6 и 20.8 соответственно (см. таблицу 3).

Ключевым фактором, влияющим на эффективность инкрементальных обновлений, оказывается пространственная локализация изменений, а не просто их число. В самом деле, если изменения затронули объекты одного листового кластера и не привели к его вырождению, то обновление будет осуществлено относительно быстро в результате распространения изменений по родительской ветке вверх по иерархии. И наоборот, если изменившиеся объекты находятся в разных листовых кластерах, то потребуется инкрементальная обработка большой части кластеров, что может оказаться более затратным, чем полный пересчет всей иерархии HLOD заново.

Так, в экспериментах со сценой здания время обновления при 5 % изменившихся объектов и при 10 % не отличается, поскольку изменения затрагивают одинаковое число кластеров (3 в случае смежных изменений и 12 в случае изменения случайных объектов). Более того, на всех сценах 10 % изменившихся смежных объектов затрагивают меньше кластеров, чем 5 % изменившихся случайных объектов, и, как следствие, обновление для 10 % смежных происходит быстрее, чем для 5 % случайных.

Необходимо также учитывать фактор параллельности в проведенных экспериментах, который несколько завуалирует эффект от инкрементальной обработки кластеров. Если изменился, к примеру, один листовой кластер, это потребует обработать его и всех его родителей (по пути от листа до корня). При этом на каждом уровне будет обрабатываться только один кластер, поэтому параллельность не будет задействована. Если изменилось несколько кластеров на одном уровне дерева, включится фактор параллельности.

В целом, инкрементальные обновления необходимы в приложениях, в которых большие сцены подвергаются перманентным изменениям и изменения локализованы в пространстве.

Выводы

В данной работе анализируются проблемы эффективной генерации иерархических уровней детализации HLOD и их оперативного обновления в условиях перманентно изменяемых проектных данных. Данные требования характерны, в частности, для цифровых двойников сложных индустриальных объектов и масштабных инфраструктурных программ.

Для рассмотренной стратегии вычисления уровней детализации HLOD на основе агломеративной кластеризации предложены и исследованы техники крупноблочного распараллеливания вычислений, инкрементального обновления уровней детализации и полигональные упрощения с удалением невидимых элементов. Проведены серии вычислительных экспериментов с моделями гражданского строительства, которые подтверждают перспективность данных оптимизаций, обеспечивающих увеличение производительности генерации HLOD в среднем на порядок. Показана зависимость полученных результатов, прежде всего производительности подготовки уровней детализации, от множества факторов, таких как сложность сцены, характер изменений, степень достоверности отображения.

Выявлены возможности для дальнейших оптимизаций, касающиеся, в частности, динамического планирования заданий при параллельной обработке кластеров и итеративного оценивания достоверности отображения внешних и внутренних граней. Данные возможности планируется исследовать в ходе дальнейшей работы.

Список литературы

1. Расширение метода иерархических уровней детализации для динамических сцен с детерминированным характером событий / Семёнов В.А., Шуткин В.Н., Золотов В.А., Морозов С.В. // Труды конференции ГрафиКон-2019. С. 37-41. DOI: 10.30987/graphicon-2019-1-37-41.

2. Visualization of Large Scenes with Deterministic Dynamics / Semenov V.A., Shutkin V.N., Zolotov V.A., Morozov S.V., Gonakhchyan V.I. // Programming and Computer Software. 2020. 46(3). P. 223-232.

3. Semenov V., Shutkin V., Zolotov V. Conservative Out-of-Core Rendering of Large Dynamic Scenes Using HDLODs //Proceedings of the 31st International Conference on Computer Graphics and Vision (GraphiCon 2021), 2021, Nizhny Novgorod, Russia, 2021. P. 105-115. DOI:10.20948/graphicon-2021-3027-105-115.

4. Alternative HDLOD method for large scenes with multiple dynamic behaviors / Shutkin V., Semenov V., Zolotov V., Morozkin N. // Proceedings of 17th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, Porto, Portugal 16 – 18 July 2023. P. 141-148.

5. Оптимизации генерации иерархических уровней детализации для масштабных полигональных сцен / Шуткин В.Н., Морозкин Н.К., Семенов В.А., Тарлапан О.А. // Труды ИСП РАН. 2025. Т. 37, вып. 3. С. 311-324. DOI: 10.15514/ISPRAS–2025–37(3)–22.

6. Clark J. H. Hierarchical geometric models for visible surface algorithms // Communications of the ACM. 1976. Vol. 19, no. 10. P. 547-554.

7. Rossignac J., Borrel P. Multi-resolution 3D approximations for rendering complex scenes // Modeling in computer graphics: methods and applications. Berlin, Heidelberg: Springer, 1993. P. 455-465.

8. Heckbert P., Garland M. Multiresolution modeling for fast rendering // Graphics Interface. Canadian Information Processing Society, 1994. P. 43-43.

9. Erikson C., Manocha D., Baxter III W. V. HLODs for faster display of large static and dynamic environments // Proceedings of the 2001 symposium on Interactive 3D graphics. 2001. P. 111-120.

10. Varadhan G., Manocha D. Out-of-core rendering of massive geometric environments // IEEE Visualization, 2002. VIS 2002. IEEE, 2002. P. 69-76.

11. Lakhia A. Efficient interactive rendering of detailed models with hierarchical levels of detail // Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. IEEE, 2004. P. 275-282.

12. Zhou Z., Chen K., Zhang J. Efficient 3-D scene prefetching from learning user access patterns // IEEE Transactions on Multimedia. 2015. Vol. 17, no. 7. P. 1081-1095.

13. Optimally redundant, seek-time minimizing data layout for interactive rendering / Chen J. et al. // The Visual Computer. 2017. Vol. 33. P. 139-149.

14. Peng C., Cao Y. A GPU-based approach for massive model rendering with frame-to-frame coherence // Computer Graphics Forum. Oxford, UK : Blackwell Publishing Ltd, 2012. Vol. 31, no. 2pt2. P. 393-402.

15. Peng C., Cao Y. Parallel LOD for CAD model rendering with effective GPU memory usage // Computer-Aided Design and Applications. 2016. Vol. 13, no. 2. P. 173-183.

16. Shamir A., Pascucci V. Temporal and spatial level of details for dynamic meshes // Proceedings of the ACM symposium on Virtual reality software and technology. 2001. P. 77-84

17. Kircher S., Garland M. Progressive multiresolution meshes for deforming surfaces // Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation. 2005. P. 191-200.

18. Ponchio F., Hormann K. Interactive rendering of dynamic geometry // IEEE Transactions on Visualization and Computer Graphics. 2008. Vol. 14, no. 4. P. 914-925.

19. Ramos F., Ripolles O., Chover M. Continuous level of detail for large scale rendering of 3d animated polygonal models // International Conference on Articulated Motion and Deformable Objects. Berlin, Heidelberg : Springer, 2012. P. 194-203.

20. Toledo L., De Gyves O., Rudomín I. Hierarchical level of detail for varied animated crowds // The Visual Computer. 2014. Vol. 30. P. 949-961.

21. 4D modeling of large industrial projects using spatio-temporal decomposition / V.A. Semenov, K.A. Kazakov, S.V. Morozov, et al. // eWork and eBusiness in Architecture, Engineering and Construction / eds. K. Menzel and R. Scherer. CRC Press, Taylor & Francis Group, London, UK, 2010. P. 89-95.

22. Visual Planning and Scheduling of Industrial Projects with Spatial Factors / V.A. Semenov, A.S. Anichkin, S.V. Morozov, et al. // Proceedings of 20th ISPE International Conference on Concurrent Engineering / eds. C. Bil, J. Mo, J. Stjepandic. IOS Press, Melbourne, Australia, 2013. P. 343-352, ISBN 978-1-61499-301-8.

23. Level of detail for 3D graphics / Luebke D., et al. Elsevier, 2002. ISBN 9780123991812.

24. Гонахчян В.И. Обзор методов упрощения полигональных моделей на графическом процессоре // Труды Института системного программирования РАН. 2014. Т. 26, вып. 2. С. 159-174.

25. Garland M., Heckbert P. S. Surface simplification using quadric error metrics // Proceedings of the 24th annual conference on Computer graphics and interactive techniques. 1997. P. 209-216.

26. Schroeder W. J., Zarge J. A., Lorensen W. E. Decimation of triangle meshes // Proceedings of the 19th annual conference on Computer graphics and interactive techniques. 1992. P. 65-70.

27. Low K. L., Tan T. S. Model simplification using vertex-clustering // Proceedings of the 1997 symposium on Interactive 3D graphics. 1997. P. 75-ff.

28. Zhao T., Jiang J., Guo X. A Novel Quadratic Error Metric Mesh Simplification Algorithm For 3D Building Models Based On ‘Local-Vertex’ Texture Features // The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2022. Vol. 48. P. 109-115.

29. Salinas D., Lafarge F., Alliez P. Structure-aware mesh decimation // Computer Graphics Forum, 2015. Vol. 34, no. 6. P. 211-227.

30. Li M., Nan L. Feature-preserving 3D mesh simplification for urban buildings // ISPRS Journal of Photogrammetry and Remote Sensing. 2021. Vol. 173. P. 135-150.

31. A Novel LOD Rendering Method with Multi-level Structure Keeping Mesh Simplification and Fast Texture Alignment for Realistic 3D Models / Ge Y., et al. // IEEE Transactions on Geoscience and Remote Sensing. 2024.

32. InfNeRF: Towards Infinite Scale NeRF Rendering with O (log n) Space Complexity / Liang J., et al. // SIGGRAPH Asia 2024 Conference Papers. 2024. P. 1-11.

33. A hierarchical 3d gaussian representation for real-time rendering of very large datasets / Kerbl B., et al. // ACM Transactions on Graphics (TOG). 2024. Vol. 43, no. 4. P. 1-15.

34. Pečnik S., Žalik B. Real-time visualization using GPU-accelerated filtering of lidar data // World Acad. Sci., Eng. Technol., Int. J. Comput., Elect., Automat., Control Inf. Eng. 2014. Vol. 8, no. 12. P. 2108-2112.

35. Efficient Scene Appearance Aggregation for Level-of-Detail Rendering / Zhou Y., et al. // arXiv preprint arXiv:2409.03761. 2024.

36. Loubet G., Neyret F. Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets // Computer Graphics Forum. 2017. Vol. 36, no. 2. P. 431-442.

37. Edavamadathil Sivaram V., Li T. M., Ramamoorthi R. Neural geometry fields for meshes // ACM SIGGRAPH 2024 Conference Papers. 2024. P. 1-11.

38. Презентация «Nanite: A Deep Dive» с коференции Siggraph 2021. URL: https://advances.realtimerendering.com/s2021/Karis_Nanite_SIGGRAPH_Advances_2021_final.pdf (дата обращения: 27.08.2025).

39. Continuous level of detail mesh library. URL: https://github.com/nvpro-samples/nv_cluster_lod_builder (дата обращения: 27.08.2025).

40. Zhang H., Cao L., Peng C. UltraMeshRenderer: Efficient Structure and Management of GPU Out-of-core Memory for Real-time Rendering of Gigantic 3D Meshes //ACM Transactions on Graphics (TOG). 2025. Vol. 44, no. 4. P. 1-19.

41. Haydel J., Yuksel C., Seiler L. Locally-adaptive level-of-detail for hardware-accelerated ray tracing // ACM Transactions on Graphics (TOG). 2023. Vol. 42, no. 6. P. 1-15.

42. Laine S., Karras T. Two methods for fast ray-cast ambient occlusion // EGSR'10: Proceedings of the 21st Eurographics conference on rendering. P.1325-1333. DOI: 10.1111/j.1467-8659.2010.01728.

43. Michael Klemm et al. OpenMP in a Modern World: From Multi-device Support to Meta Programming // Proceedings of 18th International Workshop on OpenMP, IWOMP 2022. Chattanooga, TN, USA, September 27–30, 2022.

44. Intel® Embree open-source high-performance ray tracing library. URL: https://www.embree.org/ (дата обращения: 14.07.2025).

45. 3D Tiles Standards Specification. URL: https://www.ogc.org/standards/3dtiles/ (дата обращения: 14.07.2025).