OI: 10.15514/ISPRAS-2025-37(3)-22

Оптимизации генерации иерархических уровней детализации для масштабных полигональных сцен

[1] *В.Н. Шуткин, ORCID: 0000-0002-9238-5029 <v451ly@ispras.ru>*
[1] *Н.К. Морозкин, ORCID: 0000-0003-3949-7731 <nmzik@ispras.ru>*
[1,2] *В.А. Семенов, ORCID: 0000-0002-8766-8454 <sem@ispras.ru>*
[1,2,3] *О.А. Тарлапан, ORCID: 0000-0001-5559-2013 <oleg@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.*

[2] *Московский физико-технический институт, Россия, 141701, Московская область, г. Долгопрудный, Институтский пер., 9.*

[3] *Московский государственный университет имени М.В. Ломоносова, Россия, 119991, Москва, Ленинские горы, д. 1.*

Аннотация. Альтернативные уровни детализации (LOD) являются одним из наиболее перспективных подходов к эффективному рендерингу сложных пространственно-трехмерных сцен. Подход получил развитие в методах иерархических уровней детализации (HLOD) и иерархических динамических уровней детализации (HDLOD), которые в настоящее время хорошо проработаны и успешно применяются для консервативного и интерактивного рендеринга больших динамических сцен. Вместе с тем, вопросам эффективной генерации уровней детализации не уделялось должного внимания, а они оказываются критичными в ряде приложений, связанных с визуальным моделированием сложных индустриальных проектов и масштабных инфраструктурных программ. В работе рассматриваются перспективные техники ускорения генерации HLOD и HDLOD. Также обсуждается возможность быстрого обновления иерархических уровней детализации с учетом перманентных локальных изменений в трёхмерной модели, характерных для приложений совместной работы.

Ключевые слова: иерархические уровни детализации; иерархические динамические уровни детализации; кластеризация; параллельные вычисления; упрощение геометрии; инкрементальное обновление; удаление невидимой геометрии; интерактивный рендеринг.

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

Optimizations for Hierarchical Levels of Detail Generation on Large-Scale Polygonal Scenes

[1] *V.N. Shutkin, ORCID: 0000-0002-9238-5029 <v451ly@ispras.ru>*
[1] *N.K. Morozkin, ORCID: 0000-0003-3949-7731 <nmzik@ispras.ru>*
[1, 2] *V.A. Semenov, ORCID: 0000-0002-8766-8454 <sem@ispras.ru>*
[1, 2, 3] *O.A. Tarlapan, ORCID: 0000-0001-5559-2013 <oleg@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

[2] *Moscow Institute of Physics and Technology, 9, Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.*

[3] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia.*

Abstract. Alternative levels of detail (LOD) are one of the most promising approaches to effective rendering of complex spatial 3D scenes. The approach has been realized in the hierarchical levels of detail (HLOD) and hierarchical dynamic levels of detail (HDLOD) methods, which are currently well studied and successfully applied for conservative and interactive rendering of large dynamic scenes. At the same time, the issues of efficient generation of levels of detail, which are critical in a number of applications related to visual modeling of complex industrial projects and large-scale infrastructure programs, have not received due attention. The paper considers emerging techniques for accelerating HLOD and HDLOD generation. It also discusses the possibility of quickly updating hierarchical levels of detail, taking into account permanent local changes in the three-dimensional model, typical for collaborative applications.

Keywords: hierarchical levels of detail; hierarchical dynamic levels of detail; clustering; parallel computing; geometry simplification; incremental update; hidden geometry removal; interactive rendering.

For citation: Shutkin V.N., Morozkin N.K., Semenov V.A., Tarlapan O.A. Optimizations for hierarchical levels of detail generation on large-scale polygonal scenes. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 3, 2025, pp. 311-324 (in Russian). DOI: 10.15514/ISPRAS-2025-37(3)-22.

1. Введение

Альтернативные уровни детализации (англ. levels of detail, LOD) получили широкое применение в приложениях компьютерной графики, оперирующих масштабными полигональными сценами и требующих эффективных средств рендеринга. Благодаря заранее подготовленным, упрощенным представлениям объектов сцены и их контекстному применению в зависимости от положения камеры и требований точности удается обеспечить высокую скорость рендеринга с приемлемым визуальным качеством.

Впервые идея использования альтернативных представлений объектов при рендеринге изображений была высказана ещё в 1976 году в работе [1]. Однако термин «уровни детализации» активно начал использоваться только в 1990-е годы, когда с развитием персональных компьютеров трёхмерная графика «шагнула в массы». Идея создания нескольких уровней детализации и выбора подходящего уровня в зависимости от расстояния до объекта (или размера объекта на экране) была высказана в работах [2, 3]. Использование более упрощённых представлений для более дальних объектов позволяет существенно увеличить производительность рендеринга. Однако при этом общее число объектов в сцене не сокращается. Чтобы сократить число обрабатываемых объектов, и для дальних участков сцены оперировать не индивидуальными объектами, а группами объектов, в 2001 году были разработаны иерархические уровни детализации (hierarchical levels of detail, HLOD) [4]. Идея заключается в построении иерархии уровней детализации, в которой каждый узел представляет собой упрощённое представление для целой группы объектов. Объекты иерархически группируются во всё большие и большие группы с возрастающей степенью

упрощения. В 2000-е годы основным направлением исследований в этой области стал рендеринг сверхбольших сцен, не помещающихся в оперативную память, с использованием иерархических уровней детализации [5, 6] (рендеринг во внешней памяти). В 2020-е годы уровни детализации получили дальнейшее развитие и были обобщены на случай динамических сцен. Иерархические динамические уровни детализации (hierarchical dynamic levels of detail, HDLOD) успешно применяются для консервативного (гарантирующего заданную пространственную и временную точность) и интерактивного (стремящегося обеспечить максимально возможную частоту генерации изображений при приемлемом уровне реализма) режимов отображения больших псевдо-динамических сцен [7-9]. Многовариантный метод HDLOD [10] обеспечивает одновременное отображение альтернативных сценариев без дополнительных вычислительных затрат на пересчет уровней детализации, характерных для основного метода.

Параллельно с развитием методов уровней детализации развивались методы упрощения геометрических представлений. Наиболее распространённым способом задания геометрического представления является полигональная сетка — набор полигонов, аппроксимирующих форму поверхности объекта. Основные подходы к упрощению полигональных сеток: стягивание рёбер [11], удаление вершины [12], кластеризация вершин [2, 13]. Подробный обзор методов упрощения полигональных представлений и уровней детализации представлен в книге [14]. В последнее время можно выделить три направления исследований по данной тематике: улучшение частных аспектов методов, адаптация методов для конкретных приложений и обобщение методов на новые типы геометрии. Работы [15, 16] предлагают улучшенные алгоритмы упрощения сеток на основе метода стягивания рёбер. Работа [17] развивает идею упрощения на основе изображений классической работы [18] с использованием дифференцируемого рендеринга и современных алгоритмов нелинейной оптимизации. Работа [19] посвящена вопросу предварительной загрузки данных при рендеринге во внешней памяти, а [20] предлагает схему размещения данных на диске для быстрого поиска. В работе [21] предлагается алгоритм упрощения сетки на графическом процессоре, а также подход к внешнему рендерингу, минимизирующий передачу данных между центральным и графическим процессором, а в следующей работе от тех же авторов [22] приводится подход, учитывающий количество видеопамяти.

Ряд работ посвящён генерации уровней детализации для моделей городов: в работах [23-25] приводятся алгоритмы упрощения, оптимизированные для обработки моделей зданий, работа [26] предлагает улучшения качества упрощения, размера текстур и производительности рендеринга, в работе [27] предлагается использовать гибридные представления уровней детализации (точки, линии, сплэты) для моделей городов, работа [28] посвящена веб-рендерингу больших моделей городов. Следующие работы описывают использование уровней детализации с другими типами геометрии: NeRF [29], 3D Gaussian [30], облако точек [31], воксельное представление [32], гибридное представление полигональная сетка + воксель [33].

Важно отметить, что в большинстве работ внимание исследователей сосредоточено на производительности рендеринга. Анализ времени подготовки уровней детализации проводится только для получения некоторой референсной информации. Вопросам же эффективной генерации уровней детализации не уделяется должного внимания, а они становятся критичными для приложений, связанных с визуализацией сложных индустриальных проектов и масштабных инфраструктурных программ на основе цифровых моделей. В этих приложениях время подготовки уровней детализации исчисляется часами и сутками процессорного времени на компьютерах типовой конфигурации, что является неприемлемым для формирования и визуального анализа цифровых моделей в условиях перманентных изменений, обусловленных, в частности групповой работой.

В статье рассматривается зарекомендовавшая себя вычислительная стратегия генерации иерархических уровней детализации с кластеризацией снизу-вверх. Для нее обсуждаются три

фактора, которые существенно влияют на производительность генерации уровней детализации и которые следует принимать во внимание в программных реализациях. Первый фактор – это методы и средства полигонального упрощения, которые имеют разную вычислительную сложность и которые следует рационально и избирательно применять в зависимости от контекста рендеринга, в частности, с учетом положения камеры, видимости объектов сцены и требуемого визуального качества. Второй важный фактор – это эффективная программная реализация методов и, прежде всего, возможность распараллеливания основной процедуры формирования индивидуальных уровней детализации и упрощения их полигонального представления. Наконец, третьим обсуждаемым фактором является возможность инкрементального обновления уровней детализации при локальных изменениях представления сцены вместо повторной генерации полной иерархии уровней детализации.

2. Генерация иерархических уровней детализации

Целью данной работы является исследование способов ускорения генерации иерархических уровней детализации с кластеризацией снизу-вверх. Данная вычислительная стратегия хорошо себя зарекомендовала как для статических, так детерминированных псевдодинамических сцен при использовании иерархических уровней детализации HLOD и иерархических динамических уровней детализации HDLOD соответственно [7-10]. Поэтому основные результаты работы в равной степени применимы к обоим случаям, а мы для краткости применяем аббревиатуру первого метода.

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 $g_c = \bigcup_{i=1,\dots n} g_{s_i}$.

Геометрическое представление кластера должно сохранять идентификаторы объектов для своих частей. Идентификатор объекта ι_s должен храниться для каждого треугольника (или вершины). Это необходимо, чтобы во время рендеринга иметь возможность связать часть геометрии кластера с оригинальным объектом сцены. Например, чтобы реализовать выделение оригинальных объектов по клику пользователя в 3D окне, чтобы применять цвета и стили к оригинальным объектам при отображении HLOD представления, или для реализации динамики в виде появления-исчезновения объектов (сокрытие/показ частей геометрии кластера). Поэтому во время генерации HLOD важно сохранять связь треугольников с идентификаторами объектов, в том числе не терять идентификацию треугольников в процессе упрощения полигональных сеток. Можно было бы представить геометрию кластера как набор полигональных сеток с идентификаторами, однако практика показывает, что для эффективности рендеринга необходимо минимизировать число

обрабатываемых сущностей, сокращать число вызовов отрисовки и т.д. Поэтому оптимальным представляется использование одной сетки на кластер с сокрытием или покраской её частей в шейдере.

Ограничивающий объём кластера всегда охватывает ограничивающие объёмы дочерних кластеров: $\forall c' \prec c\ b_{c'} \subseteq b_c$. Это свойство позволяет использовать HLOD как иерархию ограничивающих объёмов для эффективных проверок видимости во время рендеринга. А именно, если кластер невидим, то не видны и все его дети. С другой стороны, если b_c кластера полностью попадает в область видимости, то все дочерние кластеры тоже будут попадать в область видимости и проверки для них можно уже не проводить.

Геометрическая погрешность дочернего кластера не может превышать геометрическую погрешность родительского кластера: $\forall c' \prec c\ \varepsilon_{c'} \leq \varepsilon_c$. Это свойство необходимо во время рендеринга, чтобы выбор более детальных представлений осуществлялся как обход вглубь дерева.

Листовые кластеры содержат точную (неупрощённую) геометрию: $\forall c: \nexists c' \prec c\ \varepsilon_c = 0$. Это свойство необходимо, чтобы всегда иметь возможность отобразить точную (оригинальную) геометрию объекта.

На рис. 1 (а) приведён пример дерева кластеров: объекты сцены объединяются в листовые кластеры без упрощения, после чего кластеры объединяются во всё большие и более упрощённые кластеры пока не процесс не сойдется к одному корневому кластеру, который будет являться наиболее упрощённым представлением для всей сцены целиком.

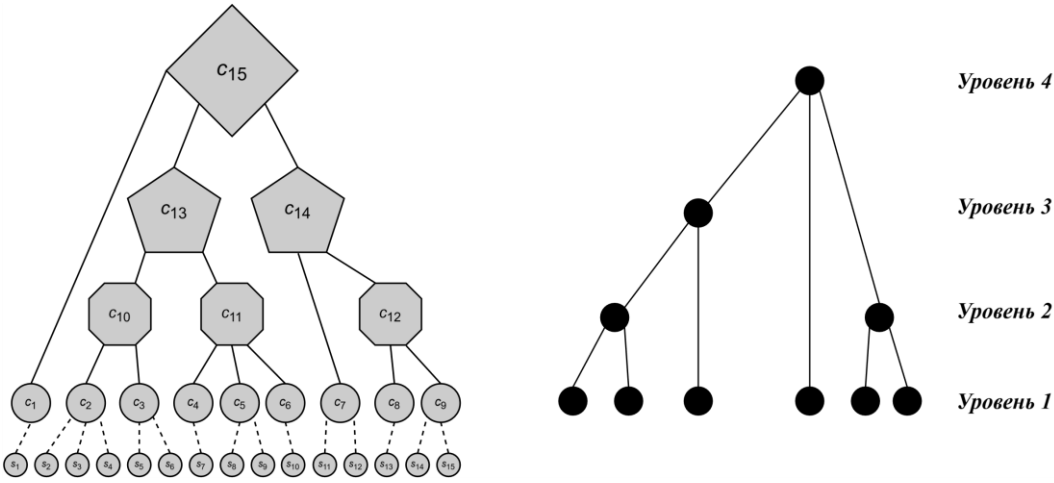

Рис. 1. а) пример дерева кластеров; б) нумерация уровней в дереве кластеров.
Fig. 1. a) an example of a cluster tree; b) numbering of levels in a cluster tree.

Опишем общую стратегию построения HLOD. Первым этапом является построение дерева кластеров. На данном этапе используется только информация об ограничивающих объёмах объектов сцены. Поскольку ограничивающий объём представляет собой достаточно компактную структуру данных, можно не беспокоиться об ограничениях по оперативной памяти и загрузить ограничивающие объёмы всех объектов сцены в оперативную память даже для очень больших сцен. Дерево кластеров строится снизу вверх по шагам. На первом шаге запускается процедура кластеризации, которая группирует объекты сцены в кластеры. Эти кластеры становятся листовыми узлами дерева. На втором шаге та же процедура кластеризации используется уже для группировки этих кластеров в новые (родительские) кластеры. И так далее, пока процесс не сойдётся к единому корневому кластеру.

После того, как дерево построено, запускается генерация кластеров, а именно, формирование их геометрических представлений. При этом происходит объединение полигональных сеток

дочерних кластеров и упрощение полученной сетки до целевой геометрической погрешности или требуемой сложности, выражаемой количеством вершин и граней в результирующей сетке. Данный этап является самым вычислительно затратным. Поскольку полигональные сетки занимают много места в памяти, необходимо подгружать их в оперативную память только по мере необходимости, и освобождать память, когда они становятся не нужны. Также, этап генерации кластеров может быть распараллелен, так как кластеры одного уровня не зависят друг от друга, а зависят только от дочерних кластеров. Алгоритм упрощения полигональной сетки также заслуживает отдельного внимания, поскольку качество и время генерации в основном зависят именно от него.

Зафиксируем эту общую вычислительную стратегию и в рамках неё исследуем возможности для ускорения генерации.

Описанная выше стратегия генерации HLOD допускает три ключевых способа оптимизации:

1. ускорение процедур упрощения;
2. распараллеливание;
3. инкрементальные обновления.

В разделе 3 приводится краткий обзор методов полигонального упрощения, которые в сочетании с методами кластеризации являются ключевыми компонентами формирования иерархических уровней детализации HLOD, HDLOD. Важное внимание при этом уделяется возможностям идентификации и удаления внутренних элементов кластера, которые будучи невидимыми при внешнем расположении камеры, занимают дополнительный объем RAM оперативной памяти (видеопамяти) и расходуют значительное время GPU графического процессора при рендеринге. В разделе 4 описывается схема крупноблочного распараллеливания процедуры кластеризации и упрощения полигональных представлений, которая допускает эффективную реализацию на мульти-ядерных, мультипроцессорных архитектурах и вычислительных кластерах. Раздел 5 посвящен частной, но практически важной возможности оптимизации вычислений при повторном формировании уровней детализации.

3. Ускорение упрощения геометрии

Существует множество алгоритмов упрощения полигональных сеток, различающихся по скорости и качеству. Ограничимся двумя популярными алгоритмами.

Стягивание рёбер с квадратичной метрикой погрешности [11] является одним из наиболее широко используемых алгоритмов упрощения. Алгоритм заключается в итеративном стягивании рёбер сетки пока не будет достигнуто целевое число треугольников и кратко может быть описан следующим псевдокодом [15]

- оценить кривизну поверхности в каждой из исходных вершин x_i;
- для каждого ребра (x_1, x_2) вычислить оптимальную результирующую вершину x_α и «стоимость» стягивания;
- отсортировать все рёбра по стоимости от минимальной к максимальной;
- последовательно стягивать рёбра и обновлять стоимости.

Стоимость — это мера того, насколько сильную погрешность внесёт стягивание ребра. Таким образом, сначала стягиваются рёбра, которые наименьшим образом исказят форму сетки. При стягивании ребра некоторые треугольники вырождаются, что приводит к сокращению числа треугольников в сетке. x_α может быть как новой вершиной (лучшее качество упрощения), так и одной из вершин x_1, x_2 (для экономии памяти при хранении нескольких уровней детализации для одной сетки). При добавлении ряда проверок можно гарантировать сохранение топологии. Также, есть вариация алгоритма, в которой рассматриваются не

только рёбра, но и любые пары вершин, что позволяет лучше упрощать несвязные сетки (с нарушением топологии).

Кластеризация вершин [2, 13] — это один из самых быстрых и простых в реализации алгоритмов упрощения. Алгоритм использует регулярную сетку. Все вершины, попавшие в одну ячейку сетки, объединяются в одну вершину. Таким образом сокращается число вершин (в каждой ячейке остаётся одна вершина), что приводит к вырождению части треугольников в точку или линию, что в свою очередь приводит к сокращению общего числа треугольников. Данный алгоритм нарушает топологию. Какие-то треугольники могут увеличиться в размерах, а какие-то уменьшиться, но размер ячейки ε гарантирует, что никакая вершина не сдвинется более, чем на $\varepsilon\sqrt{3}$ (диагональ куба), что позволяет контролировать погрешность.

От выбранного алгоритма существенным образом будут зависеть скорость генерации уровней детализации и их качество. Если требуется, чтобы все кластеры имели приблизительно одинаковую сложность и размер, необходим алгоритм упрощения до целевого числа треугольников. Для этой цели подходит, например, алгоритм стягивания рёбер с квадратичной метрикой погрешности. Сгенерированные уровни детализации будут удобны с точки зрения предсказуемости объёмов данных, кэшируемых, загружаемых в основную память и передаваемых по сети, что особенно важно для рендеринга больших сцен веб-приложениями. С другой стороны, при таком способе генерации иерархических уровней детализации не контролируется геометрическая погрешность кластеров, которая может влиять на эффективность рендеринга в ходе навигации по сцене из-за необходимого замещения кластеров с надлежащей точностью. Для этой цели можно порекомендовать алгоритм кластеризации вершин с целевой геометрической погрешностью, заданной относительно габаритов кластера и учитывающей его иерархический уровень.

Радикальной техникой упрощения, сочетаемой с основными алгоритмами, является **удаление внутренних невидимых граней**. Это особенно важно при подготовке HLOD для цифровых индустриальных моделей, содержащих большое число конструктивных и инженерных элементов, перекрывающих друг друга и не видимых снаружи генерируемых кластеров. При этом важно заметить, что упрощённые представления используются только тогда, когда камера располагается достаточно далеко от объекта, то есть кластеры HLOD всегда просматриваются снаружи. Исключение могут составлять только два случая: интерактивный внешний рендеринг, когда необходимое детальное представление ещё не успело загрузиться из внешней памяти, и поэтому используется упрощённое представление, и прозрачные объекты (или возможность изменять прозрачность объектов в ходе визуализации). Случай с интерактивным рендерингом не является столь критичным, если навигация по сцене осуществляется плавно, и геометрия успевает загружаться. При этом удаление внутренних граней способно не только уменьшить объем данных, но и существенно повысить скорость генерации и рендеринга HLOD за счет сокращения общего числа обрабатываемых и растеризуемых полигонов, а также ускорения времени подгрузки данных из внешней памяти.

Удаление внутренних граней может быть осуществлено одним из нескольких способов:

- алгоритм наподобие затенения фонового освещения (англ. ambient occlusion), когда из грани выпускаются лучи, и если хотя бы один луч достиг «неба» (не встретил на пути геометрию), то эту грань можно считать видимой снаружи. Различие с затенением заключается в том, что там вычисляется доля лучей, достигших неба, для расчёта степени освещённости, а в нашем случае достаточно одного луча для вынесения вердикта о видимости грани;
- выпускание лучей снаружи к центру геометрии (равномерно по сфере) с целью обнаружения внешних граней;

- рендеринг геометрии с разных ракурсов, когда в качестве цвета пикселя выводится идентификатор грани; идентификаторы, попавшие во фрейм буфер – это обнаруженные внешние грани.

4. Распараллеливание

После того, как дерево кластеров построено, начинается процесс генерации кластеров, который заключается в создании и упрощении геометрических представлений. Поскольку геометрические представления занимают много памяти, они хранятся во внешней памяти и загружаются в оперативную память в процессе генерации по мере необходимости. Дерево кластеров хранится в оперативной памяти, поскольку оно содержит лишь информацию об ограничивающих объёмах кластеров, их идентификаторах и связях родитель-ребёнок, и поэтому не занимает много места.

Генерация кластеров осуществляется по уровням дерева. Сначала обрабатывается первый уровень, при этом происходит генерация геометрических представлений кластеров из представлений оригинальных объектов. Затем обрабатывается второй уровень, при этом генерируются геометрические представления кластеров второго уровня из представлений кластеров первого уровня. Затем обрабатывается третий уровень, при этом генерируются геометрические представления кластеров третьего уровня из представлений кластеров второго и, возможно, первого уровней (рис. 1 (б)). И так далее, пока не будет обработан последний уровень, содержащий корневой кластер.

Обработка кластеров происходит по уровням, поскольку это позволяет произвести распараллеливание. А именно, кластеры одного уровня не зависят друг от друга, а только от кластеров нижних уровней, поэтому обработка кластеров одного уровня может осуществляться в нескольких параллельных потоках. Обработка одного кластера заключается в следующем:

- полигональные сетки дочерних кластеров загружаются в оперативную память;
- полигональные сетки объединяются в одну новую сетку;
- освобождается оперативная память, выделенная под полигональные сетки дочерних кластеров;
- новая сетка упрощается;
- полученная полигональная сетка кластера сохраняется во внешней памяти;
- освобождается оперативная память, выделенная под полигональную сетку.

При этом между кластерами одного уровня нет зависимости по данным, что позволяет произвести распараллеливание. Следует заметить, что, степень параллелизма падает с ростом уровня в дереве, поскольку на каждом следующем уровне становится меньше кластеров. Также, параллельное исполнение будет увеличивать рабочий расход оперативной памяти. Фактором, который может препятствовать распараллеливанию, является способ хранения геометрии оригинальных объектов сцены и кластеров. Например, если для хранения используется база данных, не предусматривающая параллельное исполнение запросов.

5. Инкрементальные обновления

Для широкого ряда приложений, например, для приложений просмотра 3D моделей, может быть достаточно однократной генерации HLOD с последующим многократным использованием этих данных. В таких приложениях сцена может быть подготовлена один раз (в момент публикации модели) и затем она может многократно просматриваться множеством пользователей на различных клиентских устройствах. Однако, в приложениях для редактирования 3D моделей возникает необходимость быстро обновлять HLOD

представления, чтобы не заставлять пользователя проводить время в ожидании увидеть результат внесённых им изменений.

Если изменения в сцене *глобальные* и меняется кластеризация объектов (дерево кластеров), то осуществить инкрементальные обновления не представляется возможным и требуется заново проводить кластеризацию и построение дерева кластеров, после чего генерировать представления всех кластеров. Если же изменения в сцене *локальные*, например, изменились геометрические представления некоторых объектов сцены, но их положения и ограничивающие объёмы остались близки к прежним, то становится возможным не проводить повторную кластеризацию, а использовать прежнее дерево кластеров. В нём необходимо найти кластеры, в которые попали изменившиеся объекты, и произвести перегенерацию представлений только для этих кластеров. Вообще говоря, оценка степени изменений возможна лишь в контексте *ограничений* на HLOD и *метрики качества* HLOD. Так, если HLOD нельзя обновить без нарушения ограничений, требуется полная перегенерация. С другой стороны, если обновление HLOD повлечёт ухудшение качества HLOD, это негативно скажется на скорости рендеринга, что может стать поводом провести полную перегенерацию.

Набор ограничений зависит от конкретной реализации, но может включать следующие ограничения:

- ограничение на размер кластера в зависимости от уровня, то есть размер ограничивающего объёма кластера на уровне l не должен превышать некоторого порогового значения $w(l)$;
- ограничение на число детей кластера (кластер может иметь не более X детей);
- ограничение на число треугольников в кластере (геометрическое представление кластера может содержать не более X треугольников);

Оценка качества HLOD — это более нетривиальный вопрос. Можно выделить несколько показателей:

- плотность кластеров (минимизация ограничивающего объёма кластера при максимизации числа объектов, вошедших в кластер);
- качество упрощения кластера (минимизация числа треугольников при незначительном увеличении геометрической погрешности);
- лучше иметь небольшое число крупных кластеров, чем много мелких.

Общая стратегия реакции на изменения может быть описана следующим образом:

1. необходимо произвести попытку обновить дерево кластеров (пока без пересчёта геометрии);
2. произвести проверку ограничений для обновлённого дерева;
3. оценить качество обновлённого дерева;
4. если обновлённое дерево удовлетворяет ограничениям и требованиям к качеству, изменения можно считать локальными, в противном случае изменения считаются глобальными.

При *локальных* изменениях осуществляется поиск «устаревших» кластеров. После чего запускается процедура, которая генерирует геометрию только для этих кластеров, не затрагивая остальные. При *глобальных* изменениях осуществляется удаление всех прежних HLOD данных и запускается новая генерация HLOD.

Рассмотрим возможные реакции на следующие типы изменений: добавление новых объектов, удаление объектов, изменение геометрии существующих объектов.

Добавление объектов. Пусть ранее имелось N объектов, которые вошли в K листовых кластеров. Пусть теперь добавилось M объектов. Реакция: если эти M объектов можно

добавить в K кластеров без нарушения ограничений, такие изменения можно считать локальными. В противном случае изменения считаются глобальными.

Удаление объектов. Как правило, удаление объектов не способно нарушить ограничений, поскольку оно ослабляет, а не усиливает характеристики кластеров (уменьшается ограничивающий объём, число треугольников и т.д.). Какие-то кластеры могут опустеть и потребуется их удаление, однако это не является жёстким требованием для повторной кластеризации. Тем не менее, кластеризация может стать менее оптимальной в терминах плотности кластеров (минимизация ограничивающего объёма при максимизации числа детей), числа треугольников в кластере (минимизация общего числа кластеров) и т.д. Число детей кластера может сократиться до одного, что является нежелательным, так как кластер перестаёт быть агрегированным представлением и иерархические уровни детализации вырождаются до обычных уровней детализации. Реакция на изменения: удалить вырожденные кластеры, вычислить новые параметры кластеров (ограничивающий объём, число треугольников, число детей и т.д.) и на основе них вычислить метрики качества HLOD. Если метрики качества ниже допустимых значений, изменения считаются глобальными, в противном случае локальными.

Изменение геометрии существующих объектов. Если изменения не нарушают ограничения, то их можно считать локальными. В противном случае (например, объект передвинулся, что влечёт существенное увеличение ограничивающего объёма кластера, и теперь было бы оптимальней включить этот объект в другой кластер, или же в геометрическом представлении объекта увеличилось число треугольников, что приводит к нарушению ограничения на число треугольников в кластере) изменения считаются глобальными. Если объект передвинулся, можно произвести попытку удалить его кластера и добавить в некоторый другой кластер, с целью не допустить нарушение ограничений и ухудшение качества. Если это удаётся сделать для всех передвинувшихся объектов, то изменения можно считать локальными.

Для реализации инкрементальных обновлений HLOD также необходимо уметь отслеживать изменения каждого объекта. Для этого будем считать, что у каждого объекта $s \in S$ имеется временная метка t_s, хранящая дату и время последней модификации объекта. Далее, у каждого HLOD кластера c тоже должна быть временная метка t_c, чтобы можно было понять, что кластер «устарел» и требуется его обновление. В дереве кластеров листовые кластеры формируются из оригинальных объектов сцены, а внутренние кластеры формируются из дочерних кластеров. Таким образом, временные метки нужны только в листовых кластерах, чтобы путём их сравнения с временными метками объектов, вошедших в кластер, определить, «устарел» ли кластер, или нет. Для внутренних кластеров их признак «устарелости» можно определить следующим образом: если хотя бы один и дочерних кластеров является «устаревшим», то этот кластер тоже является «устаревшим». Временную метку кластера можно вычислить как максимум (наиболее позднюю) из временных меток вошедших в него объектов:

$$t_c = \max_{s \prec c} t_s.$$

Она присваивается каждому листовому кластеру при генерации или обновлении. В начале обновления, при поиске кластеров, требующих обновления, кластер считается «устаревшим», если: $\forall s \prec c \; t_s > t_c$ (для листовых кластеров) или $\exists c' \prec c$, такой что c' является устаревшим (для внутренних кластеров). Используя эти правила, признак «устарелости» может быть вычислен для кластеров итеративно по уровням дерева. Сначала он вычисляется для листовых кластеров, затем для их родителей, затем для родителей этих родителей и т.д.

Перегенерация «устаревших» кластеров осуществляется итеративно по уровням дерева. На первой итерации запускается генерация листовых кластеров. При этом те кластеры, у которых признак «устарелости» не выставлен, игнорируются. На следующей итерации

обрабатывается следующий уровень дерева (родители листовых кластеров) и так далее. В целом процесс отличается от обычной генерации кластеров только наличием проверки признака «устарелости» кластера, поэтому в программной реализации первичная генерация кластеров и перегенерация «устаревших» кластеров могут быть реализованы одной процедурой. Если кластеры хранятся на диске, каждый кластер в отдельном файле, то обновление кластеров реализуется естественным образом. «Устаревшие» кластеры генерируются и их файлы перезаписываются, в то время как остальные файлы не затрагиваются, но читаются при генерации родительских кластеров.

6. Заключение

Таким образом, предложены и описаны перспективные техники ускорения генерации иерархических уровней детализации HLOD и HDLOD в приложениях компьютерной графики, связанных с консервативным и интерактивным рендерингом больших динамических сцен и нуждающихся в оперативном вычислении и обновлении уровней детализации. В частности, подобные критические требования возникают в приложениях визуального моделирования сложных индустриальных проектов и масштабных инфраструктурных программ. Главное внимание было уделено зарекомендовавшей себя вычислительной стратегии генерации уровней детализации с кластеризацией снизу-вверх, которая имеет потенциал для многочисленных оптимизаций. Для нее предложены и детально описаны техники распараллеливания вычислений кластеров на разных уровнях и инкрементального обновления кластеров при локальных изменениях в трехмерной модели. Также проведен анализ возможностей ускорения вычислений за счет грамотного выбора основного алгоритма упрощения полигональной геометрии, в том числе, с использованием техник удаления невидимых внутренних граней. Данные оптимизации могут применяться изолированно или совместно, а их эффективность может варьироваться в зависимости от специфики приложения и среды исполнения. Параллельная обработка кластеров на отдельных уровнях иерархии позволяет эффективно масштабировать процедуры упрощения геометрии и удаления невидимых граней, выполняя их в параллельных потоках на современных многоядерных процессорах. В приложениях для совместной работы с трехмерными моделями ключевым фактором ускорения может стать применение техники инкрементальных обновлений, позволяющей отследить локальные изменения в кластерах и не тратить вычислительные ресурсы на повторное вычисление всей иерархии уровней детализации. В дальнейших планах авторов программная реализация предложенных техник оптимизации и проведение серий вычислительных экспериментов для подтверждения их релевантности требованиям эффективности и масштабируемости. Эксперименты позволят также определить области применимости техник оптимизации и выработать рекомендации по их совместному использованию.

Список литературы / References

[1]. Clark J. H. Hierarchical geometric models for visible surface algorithms. Communications of the ACM. – 1976. – Т. 19. – №. 10. – С. 547-554.

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

[3]. Heckbert P., Garland M. Multiresolution modeling for fast rendering. Graphics Interface. – Canadian Information Processing Society, 1994. – С. 43-43.

[4]. 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. – С. 111-120.

[5]. Varadhan G., Manocha D. Out-of-core rendering of massive geometric environments. IEEE Visualization, 2002. VIS 2002. – IEEE, 2002. – С. 69-76.

[6]. 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. – С. 275-282.
[7]. Семёнов В.А., Шуткин В.Н., Золотов В.А., Морозов С.В. Расширение метода иерархических уровней детализации для динамических сцен с детерминированным характером событий. Труды конференции ГрафиКон-2019, стр. 37-41, DOI: 10.30987/graphicon-2019-1-37-41.
[8]. Semenov V.A., Shutkin V.N., Zolotov V.A., Morozov S.V., Gonakhchyan V.I. Visualization of Large Scenes with Deterministic Dynamics. Programming and Computer Software, 2020, 46(3), pp. 223–232.
[9]. V. Semenov, V. Shutkin, V. Zolotov. 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, pp. 105-115, DOI:10.20948/graphicon-2021-3027-105-115.
[10]. Shutkin V., Semenov V., Zolotov V., Morozkin N. Alternative HDLOD method for large scenes with multiple dynamic behaviors. Proceedings of 17th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, Porto, Portugal 16 – 18 July 2023, pp. 141-148.
[11]. Garland M., Heckbert P. S. Surface simplification using quadric error metrics. Proceedings of the 24th annual conference on Computer graphics and interactive techniques. – 1997. – С. 209-216.
[12]. 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. – С. 65-70.
[13]. Low K. L., Tan T. S. Model simplification using vertex-clustering. Proceedings of the 1997 symposium on Interactive 3D graphics. – 1997. – С. 75-ff.
[14]. Luebke D. et al. Level of detail for 3D graphics. – Elsevier, 2002.
[15]. Michaud C., Mellado N., Paulin M. Mesh simplification with curvature error metric. Eurographics 2017. – 2017.
[16]. Liang Y., He F., Zeng X. 3D mesh simplification with feature preservation based on whale optimization algorithm and differential evolution. Integrated Computer-Aided Engineering. – 2020. – Т. 27. – №. 4. – С. 417-435.
[17]. Hasselgren J. et al. Appearance-Driven Automatic 3D Model Simplification. EGSR (DL). – 2021. – С. 85 -97.
[18]. Lindstrom P., Turk G. Image-driven simplification. ACM Transactions on Graphics (ToG). – 2000. – Т. 19. – №. 3. – С. 204-241.
[19]. Zhou Z., Chen K., Zhang J. Efficient 3-D scene prefetching from learning user access patterns. IEEE Transactions on Multimedia. – 2015. – Т. 17. – №. 7. – С. 1081-1095.
[20]. Chen J. et al. Optimally redundant, seek-time minimizing data layout for interactive rendering. The Visual Computer. – 2017. – Т. 33. – С. 139-149.
[21]. 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. – Т. 31. – №. 2pt2. – С. 393- 402.
[22]. Peng C., Cao Y. Parallel LOD for CAD model rendering with effective GPU memory usage. Computer-Aided Design and Applications. – 2016. – Т. 13. – №. 2. – С. 173-183.
[23]. Zhao T., Jiang J., Guo X. A Novel Quadratic Error Metric Mesh Simplification Algorithm For 3D Building Models Based On ‘Local-Vertex’ Texture Featues. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. – 2022. – Т. 48. – С. 109-115.
[24]. Salinas D., Lafarge F., Alliez P. Structure-aware mesh decimation. Computer Graphics Forum. – 2015. – Т. 34. – №. 6. – С. 211-227.
[25]. Li M., Nan L. Feature-preserving 3D mesh simplification for urban buildings. ISPRS Journal of Photogrammetry and Remote Sensing. – 2021. – Т. 173. – С. 135-150.
[26]. Ge Y. et al. A Novel LOD Rendering Method with Multi-level Structure Keeping Mesh Simplification and Fast Texture Alignment for Realistic 3D Models. IEEE Transactions on Geoscience and Remote Sensing. – 2024.
[27]. Zhou S. et al. A hybrid level-of-detail representation for large-scale urban scenes rendering. Computer Animation and Virtual Worlds. – 2014. – Т. 25. – №. 3-4. – С. 243-253.
[28]. Zhang L. et al. Web-based visualization of large 3D urban building models. International Journal of Digital Earth. – 2014. – Т. 7. – №. 1. – С. 53-67.
[29]. Liang J. et al. InfNeRF: Towards Infinite Scale NeRF Rendering with O (log n) Space Complexity. SIGGRAPH Asia 2024 Conference Papers. – 2024. – С. 1-11.

[30]. Kerbl B. et al. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. ACM Transactions on Graphics (TOG). – 2024. – Т. 43. – №. 4. – С. 1-15.
[31]. 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. – Т. 8. – №. 12. – С. 2108- 2112.
[32]. Zhou Y. et al. Efficient Scene Appearance Aggregation for Level-of-Detail Rendering. arXiv preprint arXiv:2409.03761. – 2024.
[33]. Loubet G., Neyret F. Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets. Computer Graphics Forum. – 2017. – Т. 36. – №. 2. – С. 431-442.

Информация об авторах / Information about authors

Василий Николаевич ШУТКИН – инженер ИСП РАН. Сфера научных интересов: компьютерная графика, рендеринг трехмерных сцен, полигональные упрощения, информационное моделирование в области архитектуры и строительства.

Vasily Nikolayevich SHUTKIN – engineer at ISP RAS. Research interests: computer graphics, rendering of 3D scenes, polygonal geometry simplification, building information modeling.

Никита Константинович МОРОЗКИН – инженер ИСП РАН. Сфера научных интересов: компьютерная графика, рендеринг трехмерных сцен, игровые движки, технологии виртуальной и смешанной реальности.

Nikita Konstantinovich MOROZKIN – engineer at ISP RAS. Research interests: computer graphics, rendering of 3D scenes, game engines, virtual and mixed reality.

Виталий Адольфович СЕМЕНОВ – доктор физико-математических наук, профессор, заведующий отделом системной интеграции и прикладных программных комплексов Института системного программирования им. В.П. Иванникова РАН. Сфера научных интересов: модельно-ориентированные технологии программной инженерии, системная интеграция, визуализация и компьютерная графика, вычислительная геометрия, информационное моделирование в области архитектуры и строительства, проектное управление и календарно-сетевое планирование.

Vitaly Adolfovich SEMENOV – Dr. Sci. (Phys.-Math.), Prof., Head of the Department of System Integration and Multi-disciplinary Applied Systems of the Ivannikov Institute for System Programming of the RAS. Research interests: model-driven methodologies and CASE toolkits, system integration, visualization and computer graphics, computational geometry, building information modeling, project management and scheduling.

Олег Анатольевич ТАРЛАПАН – кандидат физико-математических наук, ведущий научный сотрудник ИСП РАН, доцент кафедры системного программирования МГУ им. М.В. Ломоносова. Сфера научных интересов: компьютерная графика и научная визуализация, системы управления базами данных, информационное моделирование в области архитектуры и строительства.

Oleg Anatolevich TARLAPAN – Cand. Sci. (Phys.-Math.), leading researcher of ISP RAS, assistant professor of the department of system programming of Lomonosov Moscow State University. Scientific interests: computer graphics and scientific visualization, database management systems, building information modeling.