ПРОГРАММИРОВАНИЕ, 2020, № 3, с. 42–52

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.92

ВИЗУАЛИЗАЦИЯ БОЛЬШИХ СЦЕН С ДЕТЕРМИНИРОВАННОЙ ДИНАМИКОЙ

© 2020 г. В. А. Семенов[a,b,c,*], В. Н. Шуткин[a,], В. А. Золотов[a,***], С. В. Морозов[a,d,****], В. И. Гонахчян[a,*****]**

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

[b] *Московский физико-технический институт (национальный исследовательский университет)*
141701 Московская обл., Долгопрудный, Институтский пер., д. 9, Россия

[c] *Национальный исследовательский университет "Высшая школа экономики"*
101000 Москва, Мясницкая ул., д. 20, Россия

[d] *Московский государственный университет имени М.В. Ломоносова*
119991 Москва, ул. Ленинские Горы, д. 1, Россия

**e-mail: sem@ispras.ru*

***e-mail: v451ly@ispras.ru*

****e-mail: vladislav.zolotov@ispras.ru*

*****e-mail: serg@ispras.ru*

******e-mail: pusheax@ispras.ru*

Поступила в редакцию 20.12.2019 г.
После доработки 09.01.2020 г.
Принята к публикации 19.01.2020 г.

Визуализация больших динамических сцен является актуальной проблемой компьютерной графики. Существует множество подходов к решению этой проблемы: использование отсечений областью видимости, удаление невидимых поверхностей, упрощение полигональных представлений, оптимизация техник рендеринга. Одним из эффективных методов является использование уровней детализации (LOD) для объектов сцены. Для больших сцен хорошо зарекомендовал себя иерархический метод (HLOD), в котором уровни детализации создаются для больших групп объектов. Однако данный метод сталкивается с трудностями при работе с динамическими сценами. В данной работе предлагается метод визуализации сцен с детерминированным характером динамики, основанный на использовании иерархических динамических уровней детализации (HDLOD). Описываются алгоритмы генерации кластеров HDLOD и их визуализации. Результаты проведенных вычислительных экспериментов подтверждают эффективность и перспективность предложенного метода.

DOI: 10.31857/S0132347420030073

1. ВВЕДЕНИЕ

Визуализация больших трехмерных сцен была и остается серьезной проблемой компьютерной графики. Растут мощности графического оборудования, разрабатываются новые методы и алгоритмы, однако реалистичная или достоверная визуализация сложных сцен остается недостижимой целью для многих приложений компьютерной графики. Многие индустриальные программные приложения, такие как системы CAD/CAE/CAM, BIM и GIS, являются критичными по отношению к сложности сцен и детализации индивидуальных объектов, поскольку подразумевают интерактивную модель взаимодействия человека с компьютером и возможность эффективного рендеринга сцены на оборудовании пользователя. Сегодня сцены зачастую состоят из тысяч или миллионов полигональных моделей, созданных в приложениях 3D-моделирования или полученных в результате сканирования объектов реального мира. Более того, объекты могут проявлять динамическое поведение, определяемое детерминированными или случайно происходящими событиями их появления, исчезновения или передвижения по сцене.

Существует множество подходов к решению этой проблемы: использование отсечений областью видимости, удаление невидимых поверхностей, упрощение полигональных представлений, оптимизация техник рендеринга. Одним из наиболее эффективных на сегодняшний день подходов является применение методов упрощения полигональных представлений. Все они преследуют

одну цель: сократить количество полигонов, при этом сохранив основные особенности оригинальной модели, насколько это возможно. Данные методы различаются по основной операции прореживания (удаление вершины, стягивание ребра, объединение вершин и т.д.), по метрике ошибки и по требованиям к топологии полигональной модели. Заинтересованному читателю предлагаем обратиться к [1]. Следует особенно выделить метод [2]. За счет использования стягивания ребер и квадратичной метрики ошибки этот метод показывает хорошую производительность и качество упрощения. При этом, что немаловажно, он может применяться для моделей, которые не являются односвязными двумерными многообразиями.

Одним из перспективных методов визуализации сложных сцен является использование уровней детализации (levels of detail, LOD) для объектов сцены. Это направление исследований в компьютерной графике имеет давнюю историю, начавшуюся с введения концепции LOD Джеймсом Кларком в 1976 году. Данная концепция подразумевает, что для каждого объекта сцены создается несколько версий его представления с различной степенью детализации. При визуализации сцены подходящие уровни детализации выбираются таким образом, что более точные представления используются для близких объектов, а более грубые – для дальних.

В настоящее время для визуализации больших сцен широко применяются иерархические уровни детализации (HLOD) [3, 4]. В отличие от традиционных уровней детализации, иерархические предоставляют упрощенные представления не только для индивидуальных объектов, но и для целых групп объектов, организованных в многоуровневые иерархии. Вместо выбора подходящего уровня детализации для каждого объекта становится возможным обрабатывать сразу их группы. Это позволяет достичь большей степени упрощения, сократить время обхода дерева сцены и количество вызовов отрисовки.

Однако иерархические уровни детализации затруднительно применять для произвольных динамических сцен. Каждый раз, когда объекты появляются, исчезают или двигаются, их представления должны быть пересчитаны. Время, необходимое для пересчетов, как правило, больше времени, необходимого для рендеринга сцены, что делает иерархические уровни детализации бесполезными для сцен с большим количеством динамических объектов. Известные попытки оптимизировать пересчеты благодаря использованию инкрементальных обновлений и их параллельному вычислению не привели к значимому успеху [3].

В данной статье рассматриваются динамические сцены с детерминированным характером событий, определяющих появление объектов, их исчезновение или передвижение по сцене. Под детерминированностью понимается наличие априорного знания о том, когда и с какими объектами происходят события. Для эффективной визуализации сцен такого типа предлагается новый метод, названный иерархическими динамическими уровнями детализации (HDLOD). Метод позволяет создавать иерархические уровни детализации, не требующие пересчета при анимации динамической сцены. Он принципиально отличается от методов использования уровней детализации при рендеринге сцен с типовыми геометрическими моделями и предопределенными шаблонами поведения, например, сцен моделирования пешеходных потоков [5].

Метод HDLOD не препятствует использованию разных методов рендеринга, включая методы отсечения конусом видимости (frustum culling) и методы удаления невидимых поверхностей (occlusion culling) 5. В предположении, что упрощенное представление сцены загружается в видеопамять, допустимо применение распространенных методов выполнения проверок видимости на графическом процессоре [7–9]. В статье описываются алгоритмы генерации кластеров HDLOD и их визуализации с использованием различных методов рендеринга, а также приводятся результаты проведенных вычислительных экспериментов, которые подтверждают эффективность и перспективность предложенного метода.

2. ИЕРАРХИЧЕСКИЕ ДИНАМИЧЕСКИЕ УРОВНИ ДЕТАЛИЗАЦИИ

Пусть сцена $S(t)$ определена в трехмерном евклидовом пространстве E^3 на моделируемом временном периоде $t \in [0, T]$ и представлена как линейный список объектов $s(g_s, v_s, p_s) \in S$. Каждый объект имеет неизменное геометрическое представление $g_s \subseteq E^3$. Статус присутствия объектов в сцене определяется их функциями видимости $v_s(t) : [0,T] \to \{0, 1\}$ таким образом, что функция $v_s(t)$ принимает единичное значение, если объект s присутствует в сцене в момент времени t, и нулевое значение – если отсутствует. Положение объекта определяется функцией положения $p_s(t) : [0,T] \to M$, где M – множество матриц размерности 4×4. Следует отметить, что движение объектов может быть задано множеством различных способов, например, как набор ключевых точек, в которых заданы позиция и ориентация объекта, с указанием времени, когда объект находится в этой ключевой точке. Положение объекта в каждый момент времени может определяться как результат интерполяции положений в ближайших ключевых точках. Однако, для целей рендеринга, как правило, используется задание положения объекта с помощью модельной матрицы преобразова-

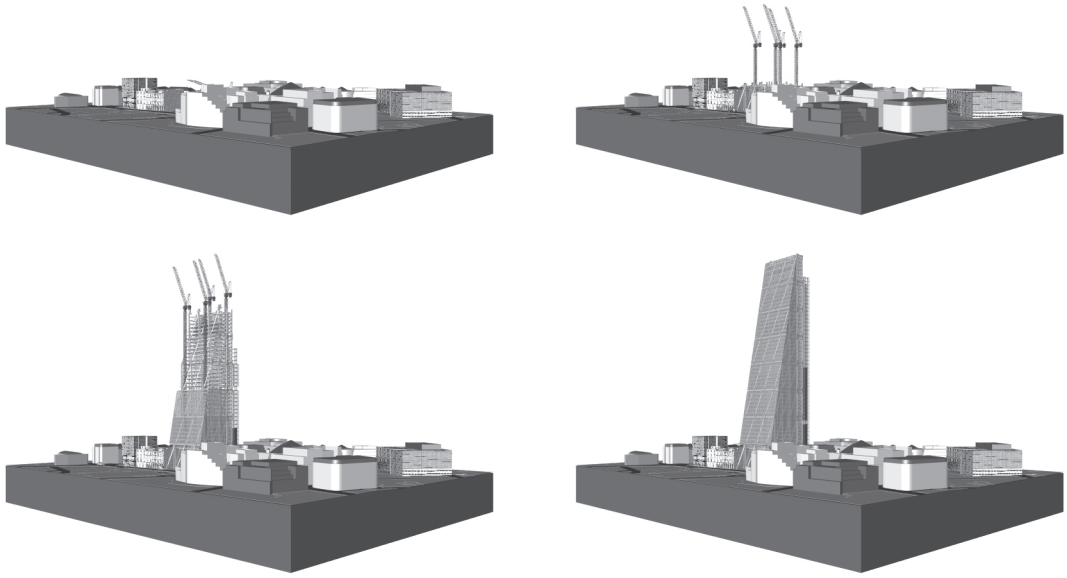

Рис. 1. Пример динамической сцены, моделирующей строительство небоскреба.

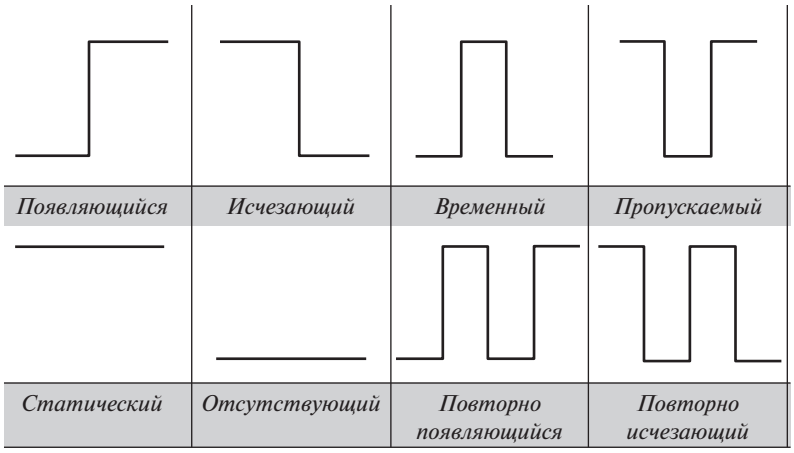

Рис. 2. Некоторые типичные функции видимости.

ния, более того, любое представление положения объекта может быть выражено при помощи матрицы. Поэтому без ограничения общности можно утверждать, что в каждый момент времени $t \in [0,T]$ положение объекта может быть представлено некоторой матрицей.

На рис. 1 показан пример динамической сцены, моделирующей строительство небоскреба в соответствии с предварительно подготовленным планом проекта. По мере изменения времени моделирования элементы конструкций и оборудование устанавливаются на строительной площадке или удаляются. Такие появления и исчезновения можно воспроизвести при помощи функций видимости, представленных на рис. 2. Элементы ландшафта остаются неизменными на протяжении всего моделируемого периода. Также в ходе строительства по строительной площадке перемещается различная техника (рис. 3).

В дальнейшем будем предполагать, что геометрическое представление объектов сцены задано набором треугольников. Не будем делать никаких предположений о топологии граничных представлений объектов и не будем требовать, чтобы граничные представления были связными или являлись многообразиями, поскольку часто для целей рендеринга предоставляется лишь так называемый “полигональный суп” без гарантий каких-либо топологических свойств.

По динамическому поведению все объекты сцены могут быть поделены на три класса.

Рис. 3. Пример динамических объектов в сцене – строительная техника и оборудование.

В класс *статических* объектов попадают объекты, видимость и положение которых не изменяется на протяжении всего моделируемого периода, то есть $\forall t \in [0,T]\, v_s(t) = 1, p_s(t) = I$, где I – единичная матрица.

Класс *псевдодинамических* объектов состоит из объектов, которые появляются и исчезают, однако их положение остается неизменным: $\forall t \in [0, T] p_s(t) = I$, но $\exists t_1, t_2 \in [0,T]$, такие что $v_s(t_1) \neq v_s(t_2)$.

Наконец, класс *динамических* объектов подразумевает, что и функция видимости, и функция положения объекта меняются с течением времени: $\exists t_1, t_2, t_3, t_4 \in [0,T]$, такие что $v_s(t_1) \neq v_s(t_2)$, $p_s(t_3) \neq p_s(t_4)$.

Будем называть иерархическими динамическими уровнями детализации (HDLOD) дерево кластеров $C(G, V, P) = \{c(g_c, v_c, p_c), \prec\}$, представленное множеством кластеров $c(g_c, v_c, p_c)$ с приписанными геометрическими представлениями g_c, функциями видимости $v_c(t) : [0,T] \to [0,1]$ и функциями положения $p_c(t) : [0,T] \to M$. В отличие от функций видимости объектов $v_s(t)$, которые принимают значения 0 или 1, функции видимости кластеров $v_c(t)$ принимают значения в диапазоне от 0 до 1, используя таким образом понятие частичной истины. Действительно, поскольку некоторые из объектов кластера могут присутствовать в сцене в некоторый момент времени, в то время как другие могут в этот же момент отсутствовать, невозможно вынести однозначный вердикт о статусе присутствия всего кластера. Для кластеров также определено отношение агломерации $\prec$ таким образом, что $c' \prec c$ тогда и только тогда, когда кластер $c' \in C$ является прямым потомком (в дереве) кластера $c \in C$.

Первым этапом генерации HDLOD является классификация объектов сцены на статические, псевдодинамические и динамические. Для каждого класса используется свой метод формирования кластеров.

Статические и псевдодинамические деревья кластеров имеют похожую структуру. В них листья дерева являются точно заданными индивидуальными объектами. Внутренние узлы представляют собой кластеры, агрегирующие геометрию (а в случае с псевдодинамическими – еще и видимость) соответствующих поддеревьев. При этом с ростом уровня в дереве геометрия и функция видимости кластеров становятся все более упрощенными, а корень представляет собой наиболее упрощенное представление огромной группы объектов.

Динамические объекты могут иметь различные траектории движения и различные моменты появления и исчезновения. Это существенно осложняет анализ и препятствует их эффективному объединению в кластеры. Поэтому для каждого динамического объекта создается отдельный кластер. В типичных сценах различные объекты могут быть представлены одной и той же моделью. Например, по строительной площадке могут перемещаться несколько экскаваторов, которые имеют одинаковое геометрическое представление, то есть являются экземплярами одной общей

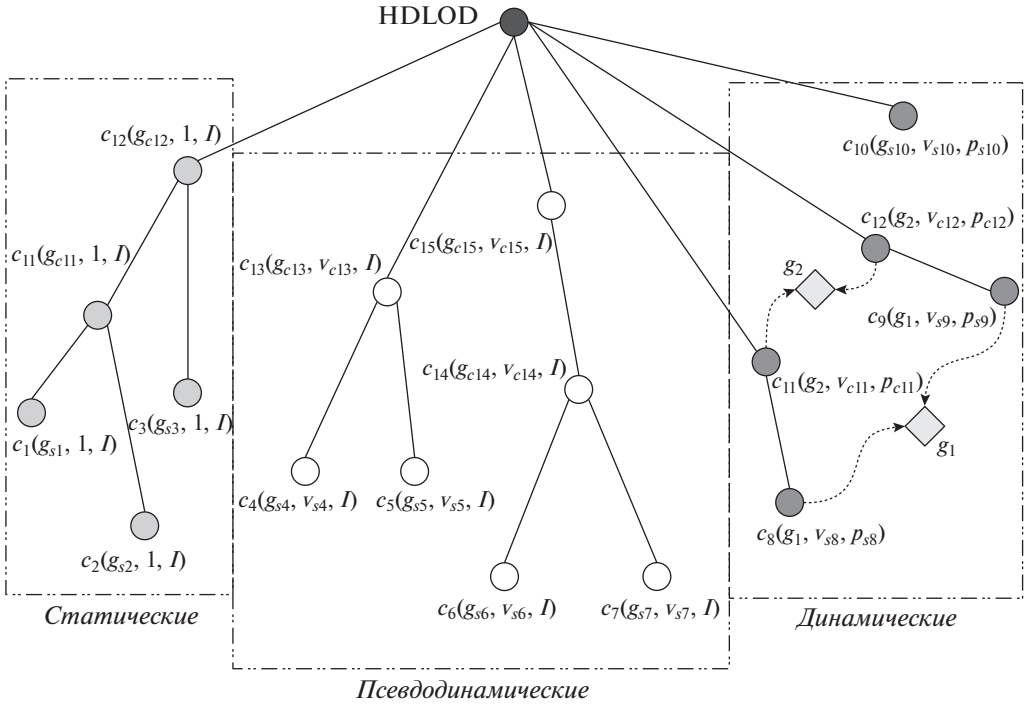

Рис. 4. Пример дерева HDLOD.

модели экскаватора. Это учитывается при формировании кластеров для динамических объектов: они могут ссылаться на общее геометрическое представление. Следует отметить, что для кластеров динамических объектов возможно дополнительно создать родительские кластеры, которые будут содержать упрощенные геометрию, функцию видимости и функцию положения.

В конечном итоге все корневые узлы деревьев статических, псевдодинамических и динамических кластеров прикрепляются к виртуальному узлу, образуя единое дерево HDLOD. На рис. 4 показан пример такого дерева.

Каждый кластер $c \in C$ хранит не только геометрическое и поведенческое представление, но также и такие производные атрибуты, как: ограничивающий параллелепипед b_c, размер или мощность w_c, а также пространственную погрешность ε_c и временную погрешность γ_c, которые будут определены ниже. Фактически эти атрибуты насчитываются при генерации иерархических динамических уровней детализации и используются при их отображении. Таким образом, каждый HDLOD-кластер представляется как кортеж $c(g_c, v_c, p_c, b_c, w_c, \varepsilon_c, \gamma_c)$.

Пространственную погрешность ε_c можно определить как абсолютную погрешность, которая устанавливает допустимое максимальное локальное отклонение геометрического представления кластера c от агрегированного представления объектов $o \prec .. \prec c' \prec c$:

$$\varepsilon_c = \max_{c' \prec c}(\varepsilon_{c'}) + D_H\left(g_c, \bigcup_{c' \prec c} g_{c'}\right).$$

Здесь погрешность ε_c определена рекуррентно с использованием метрики Хаусдорфа $D_H(A, B)$, которая представляет собой наибольшее из всех расстояний от точки из одного множества до ближайшей точки другого множества:

$$D_H(A, B) = \max\{\max_{x \in A} \min_{y \in B} D(x, y), \max_{y \in B} \min_{x \in A} D(x, y)\},$$

где A, B – замкнутые множества точек и $D(x, y)$ – функция метрики в Евклидовом пространстве.

Временная погрешность γ_c определяется для псевдодинамических кластеров как максимальное отклонение функции видимости кластера от функций видимости оригинальных объектов:

$$\gamma_c = \max_{c' \prec c}(\gamma_{c'} + D_V(v_c, v_{c'})),$$

где расстояние $D_V(v_A, v_B)$ вычисляется с использованием функциональной метрики:

$$D_V(v_A(t), v_B(t)) = \frac{1}{T}\int_0^T |v_A(t) - v_B(t)|\, dt.$$

Данные параметры используются для оценки отклонения геометрии и близости временных поведений. Пространственные погрешности вы-

числяются в процессе упрощения геометрии кластера. Временные погрешности могут быть вычислены вместе с функцией видимости кластера. Для вычисления функции видимости кластера предлагается использовать следующую формулу:

$$v_c(t) = \frac{\sum_{c' \prec c} w_{c'} v_{c'}(t)}{\sum_{c' \prec c} w_{c'}}.$$

Использование взвешенных сумм позволяет в большей степени учитывать поведение значимых объектов и надлежащим образом воспроизводить поведение кластера. Функция $v_c(t)$ выражает взвешенную долю видимых объектов кластера в момент времени t. Если она принимает единичное значение, это означает, что все объекты кластера присутствуют в сцене в момент времени t, если нулевое – отсутствуют в момент времени t.

Для каждого кластера необходимо принять решение о том, стоит ли отображать его, проигнорировать, или использовать более точные дочерние представления. Для этого вычисляется зависящая от времени пространственная погрешность $\delta_c(t)$ (пространственная ошибка, вызванная как геометрическим, так и временным упрощением):

$$\delta_c(t) = \begin{cases} 0, & \text{при} \quad v_c(t) = 0 \\ \varepsilon_c, & \text{при} \quad v_c(t) = 1 \\ \varepsilon_c + (1 - v_c(t)) \sum_{c' \prec c} w_{c'}, & \text{при} \quad \frac{1}{2} \le v_c(t) < 1 \\ \varepsilon_c + v_c(t) \sum_{c' \prec c} w_{c'}, & \text{при} \quad 0 < v_c(t) < \frac{1}{2} \end{cases}$$

При визуализации сцены совершается обход дерева кластеров. В каждом узле атрибуты кластера анализируются для определения необходимости дальнейшего обхода поддерева. Проверяется, присутствует ли кластер в сцене в указанное время моделирования ($v_c(t) > 0.5$), попадает ли ограничивающий параллелепипед b_c кластера в конус видимости, а также является ли погрешность $\delta_c(t)$ кластера достаточной для получения необходимого качества. Если все условия удовлетворены, представление кластера немедленно выбирается для отображения. В данном случае все поддерево может быть исключено из обхода. Если третье условие не удовлетворяется, продолжается обход поддерева кластеров и дочерние узлы проходят такие же проверки, пока не будут достигнуты листовые узлы с точно заданными объектами сцены. Псевдокод алгоритма визуализации HDLOD представлен в приложении 1.

3. ГЕНЕРАЦИЯ HDLOD

Иерархические динамические уровни детализации могут быть сгенерированы автоматически при помощи предложенного метода. Первым этапом генерации HDLOD является классификация объектов сцены на статические, псевдодинамические и динамические. Для каждого класса используется свой метод формирования кластеров.

Для начала рассмотрим метод обработки псевдодинамических объектов как наиболее сложный. Данный метод использует иерархическую (снизу-вверх) кластеризацию и многоуровневый контроль точности, он начинает с индивидуальных объектов и последовательно группирует их во все большие и большие кластеры, пока не будет достигнуто желаемое количество уровней. Корневые кластеры могут быть в дальнейшем еще сильнее упрощены, если потребуется отображать эту сцену как часть более сложной композиции. Кластеризация должна проводиться при строгом контроле точности, результирующее дерево кластеров должно удовлетворять множеству требований касательно ожидаемого количества уровней детализации, степени узлов, пространственной плотности и перекрытия дочерних кластеров, их временной близости и снижения сложности кластеров с повышением уровня.

К сожалению, классические методы кластеризации не могут быть напрямую применены к проблемам генерации HDLOD. Предъявляемые требования достаточно сложны и могут противоречить друг другу. Это препятствует математической формализации метрических функций и критериев связности, необходимых для методов кластеризации [10]. Неприемлемо высокая вычислительная сложность этих методов является другой причиной, препятствующей адаптации классических результатов. Например, наивная реализация агломеративной кластеризации имеет временную сложность $O(n^3)$ и требует $O(n^3)$ памяти, что делает ее неприменимой даже для достаточно простых сцен. Более быстрая иерархическая кластеризация с временной сложностью $O(n^2)$ и потреблением памяти $O(n^2)$ также не подходит для решения обсуждаемых проблем [10].

Поэтому предлагается формировать дерево, руководствуясь другими принципами. Процесс кластеризации разделяется на шаги, на каждом из которых сформированные кластеры удовлетворяют определенным требованиям по точности. По мере того как делаются новые шаги, эти требования ослабляются таким образом, чтобы гарантировать завершение процесса после определенного количества шагов, равного числу уровней детализации L.

На каждом запланированном шаге метода l ($1 \le l \le L$) делается попытка сформировать новые

кластеры с размерами $w_c \le w(l)$ и погрешностями $\varepsilon_c \le \varepsilon(l)$, $\gamma_c \le \gamma(l)$. Для этого пороговые значения $w(l)$, $\varepsilon(l)$ и $\gamma(l)$ выбираются таким образом, чтобы они монотонно увеличивались с увеличением уровня. Например, можно предложить следующие значения:

$$w(l) = \frac{l}{L}W, \quad \varepsilon(l) = 0.01\frac{l}{L}W = 0.01w(l),$$
$$\gamma(l) = 0.25\frac{l}{L}T,$$

где W – это размер всей сцены.

Метод на каждом шаге формирует список активных кластеров, которые будут участвовать в следующем шаге. Изначально в список активных кластеров помещаются все оригинальные объекты. На каждом шаге метода представители из числа активных кластеров выбираются с использованием кривых пространственного заполнения Гильберта [11]. С одной стороны, это позволяет выбирать представителей во всем объеме сцены, с другой – локализовать их в плотно заполненных областях. Далее для каждого представителя осуществляется поиск соседей. Соседи должны удовлетворять условиям по пространственной и временной близости, а также гарантировать формирование родительского кластера с запланированной точностью. Из представителя и его соседей формируется новый кластер, а они становятся его детьми. Геометрическое представление нового кластера получается путем объединения представлений детей. Оно должно быть упрощено для достижения необходимой погрешности $\varepsilon_c \le \varepsilon(l)$, например, с использованием упомянутого алгоритма [2]. Функция видимости нового кластера определяется путем вычисления взвешенной функции видимости $v_c(t)$, приведенной выше. Новый кластер добавляется в список активных, а его дети – удаляются оттуда. Если же для текущего представителя не удалось найти подходящих соседей, он исключается из анализа на текущем шаге, но будет участвовать в следующих. Псевдокод описанного алгоритма представлен в приложении 2.

С использованием пространственной индексации кластеризация может быть осуществлена за $O(n\log n)$, где n – количество объектов сцены. В сравнении с классическими методами кластеризации, упомянутыми выше, этот результат является гораздо более приемлемым для больших сцен. Для быстрого поиска соседей могут применяться различные индексные структуры [11], в частности, для псевдодинамических сцен показывают хорошую производительность регулярные динамические окто-деревья [12]. В данной реализации использовались упорядочения по каждой из координат.

Для статических объектов применяется этот же метод кластеризации, однако для них учитывается лишь пространственный аспект. В данном случае статические объекты можно рассматривать как частный случай псевдодинамических.

Для каждого динамического объекта создается отдельный кластер. При этом осуществляется анализ геометрических представлений объектов. Если обнаружены объекты, имеющие одинаковое представление, то представление их кластеров задается при помощи ссылки на данное представление. Таким образом удается избежать дублирования данных.

4. РЕНДЕРИНГ

Эффективность визуализации HDLOD кластеров зависит от используемых методов рендеринга на графическом оборудовании. В данной работе предполагается применение двух параллельно работающих подсистем, первая из которых определяет видимые кластеры (visible surface determination), а вторая выполняет загрузку полигональных представлений кластеров в видеопамять и отправку команд на графический процессор, используя программный интерфейс OpenGL.

При визуализации HDLOD дерева осуществляется его обход с проверками видимости кластера на текущее модельное время с заданного положения камеры с учетом точности его полигонального представления. Для удовлетворяющих проверкам кластеров сообщения об их отображении передаются второй подсистеме. Получив сообщение, вторая подсистема отправляет триангулированное представление кластера и команду его рендеринга на графический процессор. После того, как все необходимые данные буферизованы в памяти графического процессора, производится вызов отрисовки (draw call), который запускает выполнение команд рендеринга. Выполнение каждой команды состоит из следующих этапов: преобразование вершин (вершинный шейдер), растеризация треугольников, расчет цвета фрагментов (фрагментный шейдер), композиция и вывод на экран (см. рис. 5).

Применение метода HDLOD позволяет уменьшить время, необходимое для выполнения всего описанного процесса. Во-первых, уменьшается количество треугольников в геометрических представлениях кластеров, что в свою очередь ускоряет передачу данных в видеопамять и сокращает время этапа преобразования вершин. Во-вторых, уменьшается общее количество команд рендеринга, что приводит к сокращению накладных расходов на их обработку.

При обработке большого количества команд расходуются ресурсы центрального процессора для валидации и записи текущего состояния графического

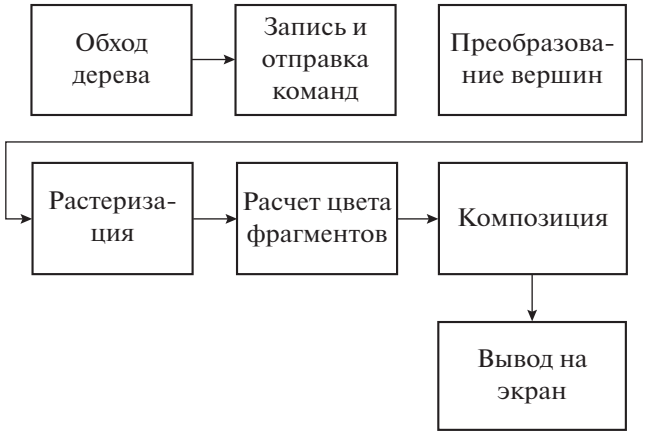

Рис. 5. Основные этапы рендеринга объектов сцены.

конвейера. Для снижения расходов можно использовать расширение OpenGL NV_Command_List, которое позволяет предварительно сформировать массив команд рендеринга вместе с текущим состоянием конвейера [13]. Это расширение является достаточно эффективным, но имеет ограниченную поддержку на современном графическом оборудовании.

В программной реализации метода визуализации HDLOD дерева использовалась процедура OpenGL MultiDrawElementsIndirect, которая позволяет выполнить массив буферизованных команд с помощью одного вызова [14]. Однако для этого требуется составить массивы команд, матриц преобразований и материалов. Для сокращения затрат на буферизацию и удаления невидимых поверхностей применяется метод пространственной декомпозиции на основе окто-деревьев (single reference octree) [15]. С каждым октантом ассоциированы соответствующие массивы, которые поддерживаются в согласованном с HDLOD деревом состоянии и обновляются по мере появления, удаления или передвижения кластеров. Если кластер переместился внутри одного октанта, то обновляется его матрица в соответствующем массиве. Если кластер переместился в другой октант, то происходит обновление массивов обоих октантов.

5. ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ

Для того чтобы апробировать введенную концепцию HDLOD, а также предложенный метод для автоматической генерации и визуализации HDLOD, была проведена серия вычислительных экспериментов. Были измерены значения времени рендеринга кадра при навигации по заданной сцене при фиксированных моментах времени моделирования, а также среднее время рендеринга кадра для анимации на протяжении всего периода моделирования. В качестве тестовой была выбрана представленная на рис. 1 динамическая сцена строительства небоскреба. Данная сцена является примером типичного проекта в строительной индустрии: в ней имеются статическое окружение, псевдодинамическая модель строящегося небоскреба, конструктивные элементы которого устанавливаются согласно плану проекта, а также динамическое оборудование, использующееся при строительстве. Все объекты сцены представлены полигональными сетками. Характеристики сцены приведены в таблице 1.

Для оценки эффективности HDLOD осуществлялась визуализация сцены при различных положениях камеры. В первом положении камера была размещена на максимально близком расстоянии, при котором сцена была видна полностью. В других положениях камера размещалась на таких расстояниях, что сцена занимала одну четвертую и одну шестнадцатую области экрана. Были выбраны положения времени в начале, конце, а также в одной третьей и в двух третях моделируемого периода. Выбор таких положений камеры, при которых сцена полностью помещается на экране, обусловлен желанием исключить фактор отсечения конусом видимости. Вычислительные эксперименты проводились на компьютере типичной конфигурации: Intel Core i7-4790 CPU (3.6 GHz), 16 GB of RAM, GeForce GTX 750 Ti (2 GB).

В таблице 2 приведены результаты измерений производительности в ходе описанных экспериментов. В последней колонке содержатся результаты, полученные при рендеринге сцены без использования упрощенных представлений, а в первых трех столбцах – с использованием HDLOD дерева. Видно, что применение HDLOD значительно повышает производительность. По мере удаления камеры от сцены достигаемый эффект растет, поскольку для отображения выбираются все более крупные (и упрощенные) кластеры.

Таблица 1. Количество объектов и треугольников в сцене

	Статические	Псевдо-динамические	Динамические	Всего
Объекты	498	40010	15649	56157
Треугольники	66784	2879506	1313767	4260057

Таблица 2. Время рендеринга кадра (в миллисекундах) при визуализации сцены строительства небоскреба

	1/1 экрана	1/4 экрана	1/16 экрана	Без HDLOD
Начало	3.81	3.58	3.29	16.9
1/3 периода	17.47	17.31	15.59	35.2
2/3 периода	8.5	7.87	7.53	54.22
Конец	3.72	3.58	3.25	61.56
Анимация	9.59	8.51	7.94	47.43

Данная зависимость наблюдается как при навигации по статической сцене, зафиксированной в выбранные моменты времени, так и при анимации сцены. В данном эксперименте уровни детализации для динамических объектов не применялись, если же использовать упрощения и для динамических объектов, то рост производительности с ростом расстояния до сцены будет более значительным. С ростом сложности сцены к концу времени моделирования (строящийся небоскреб) можно наблюдать увеличение времени кадра при визуализации сцены без применения HDLOD. Однако можно заметить, что HDLOD показывают наихудшую производительность для положения времени в 1/3 модельного периода. Это обуславливается тем, что в данный момент времени в сцене происходит большое количество изменений и для многих кластеров функция видимости $f_c(t)$ оказывается близкой к 0.5, а их пространственная погрешность, зависящая от времени, $\delta_c(t)$ оказывается в этот момент очень большой, что вынуждает использовать дочерние представления. Тем не менее, производительность остается выше, чем без использования HDLOD.

Аналогичная серия экспериментов была проведена для других задач визуального моделирования проектов строительства, городских инфраструктурных программ, машиностроительных процессов. Ее результаты подтверждают высокую эффективность и перспективность предложенного метода.

6. ЗАКЛЮЧЕНИЕ

Таким образом, в работе предложен метод визуализации сцен с детерминированным характером динамики, основанный на использовании иерархических динамических уровней детализации (HDLOD). В отличие от распространенных методов уровней детализации (LOD) и их иерархических расширений (HLOD), предлагаемый метод применим к широкому классу больших динамических сцен. Для метода описаны алгоритмы генерации дерева HDLOD и его эффективной визуализации с заданной точностью. Результаты проведенных вычислительных экспериментов подтверждают эффективность и перспективность предложенного метода. Дальнейшая работа будет посвящена исследованию алгоритмических вариантов разработанного метода, а также его применению в новых индустриальных приложениях.

7. Приложение 1. Псевдокод алгоритма визуализации HDLOD

```
PROCEDURE DISPLAY_CLUSTER(CLUSTER n, VIEW v, TIME t, RESOLUTION r)
{
        IF (VALUE_OF(BEHAVIOR_FUNCTION(n), t) EQUAL 0)
                RETURN
        ELSE IF (IS_OUTSIDE_FRUSTUM(BOUNDING_BOX(n), v))
                RETURN
        ELSE IF (VALUE_OF(DELTA_FUNCTION(n), t) / DISTANCE(n, v) < r)
        {
                IF (VALUE_OF(BEHAVIOR_FUNCTION(n), t) >= 0.5)
                        RENDER(GEOMETRY(n), v))
                ELSE
                        RETURN
        }
        ELSE
        {
```

```
                SET_OF_CLUSTER children = CHILDREN_NODES(n)
                FOR_EACH (CLUSTER child IN children)
                        DISPLAY_CLUSTER(child, v, t, r)
        }
}
```

8. Приложение 2. Псевдокод алгоритма кластеризации псевдо-динамических объектов

```
PROCEDURE GENERATE_HDLOD(SCENE scene, INTEGER levels, HDLOD tree)
{
        SET_OF_CLUSTER active = NULL, next = NULL
        FOR_EACH (OBJECT object IN OBJECTS(scene))
        {
                CLUSTER cluster = FORM_CLUSTER(object)
                 ADD_TO(cluster, tree)
                 ADD_TO(cluster, active)
         }
        FOR_EACH (INTEGER step = 1 TO levels)
        {
                REAL epsilon, gamma, w
                 COMPUTE_LEVEL_THRESHOLDS(scene, levels, step, epsilon, gamma, w)
                WHILE (NOT_EMPTY(active))
                {
                        CLUSTER representative = SELECT_REPRESENTATIVE(active)
                        SET_OF_CLUSTER neighbors = FIND_NEIGHBORS(active,
                                representative, gamma, w)
                         IF (IS_EMPTY(neighbors))
                        {
                                ADD_TO(representative, next)
                                REMOVE_FROM(representative, active)
                        }
                        ELSE
                        {
                            SET_OF_CLUSTER children
                            ADD_TO(representative, children)
                            FOR_EACH(CLUSTER neighbor IN neighbors)
                                    ADD_TO(neighbor, children)
                           CLUSTER cluster = CREATE_CLUSTER(children)
                           SIMPLIFY (cluster, epsilon)
                           ADD_TO(cluster, tree)
                           ADD_TO(cluster, next)
                           REMOVE_FROM(representative, active)
                           FOR_EACH (CLUSTER neighbor IN neighbors)
                                    REMOVE_FROM(neighbor, active)
                        }
                 COPY(next, active)
                 EMPTY(next)
        }
}
```

СПИСОК ЛИТЕРАТУРЫ

1. *Luebke D. et al.* Level of Detail for 3D Graphics. Morgan Kaufmann Publishers Inc. San Francisco. 2003. 432 p.
2. *Garland M., Heckbert P.S.* Surface Simplification Using Quadric Error Metrics // SIGGRAPH '97 Proceedings of the 24th annual conference on Computer graphics and interactive techniques. 1997. P. 209–216.
3. *Erikson C. et al.* HLODs for Faster Display of Large Static and Dynamic Environments // I3D '01 Proceedings of the 2001 symposium on Interactive 3D graphics. 2001. P. 111–120.
4. *Lilley S., Cozzi P.* Cesium 3D Tiles. Beyond 2D Tiling. FOSS4G-NA Presentation, 2016, https://cesium.com/presentations/files/FOSS4GNA2016/3DTiles.pdf.
5. *Toledo L. et al.* Hierarchical Level of Detail for Varied Animated Crowds // The Visual Computer. 2014. V. 30. № 6–8. P. 949–961.
6. *Cohen-Or D., Chrysanthou Y.L., Silva C.T., Durand F.* A Survey of Visibility for Walkthrough Applications // IEEE Transactions on Visualization and Computer Graphics. 2003. V. 9. № 3. P. 412–431.
7. *Bittner J. et al.* Coherent Hierarchical Culling: Hardware Occlusion Queries Made Useful // Computer Graphics Forum. 2004. V. 23. № 3. P. 615–624.
8. *Guthe M. et al.* Near Optimal Hierarchical Culling: Performance Driven Use of Hardware Occlusion Queries // Eurographics Symposium on Rendering. 2006. P. 207–214.
9. *Mattausch O. et al.* CHC++: Coherent Hierarchical Culling Revisited // Computer Graphics Forum. 2008. V. 27. P. 221–230.
10. *Xu D., Tian Y.* A Comprehensive Survey of Clustering Algorithms // Annals of Data Science. 2015. V. 2. P. 165–193.
11. *Samet H.* Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann Publishers Inc. San Francisco, 2006. 1024 p.
12. *Morozov S. et al.* Indexing of Hierarchically Organized Spatial-Temporal Data Using Dynamic Regular Octrees // Perspectives of System Informatics, Lecture Notes in Computer Science. 2018. V. 10742. P. 276–290.
13. *Lorach T.* Approaching Zero Driver Overhead. SIGGRAPH 2014 Presentation, 2014, https://on-demand.gputechconf.com/siggraph/2015/presentation/SIG1512-Tristan-Lorach.pdf
14. *Bennett J., Carter M.* Performance Gains Achieved Through Modern OpenGL in the Siemens DirectModel Rendering Engine. GTC Presentation, 2015, http://on-demand.gputechconf.com/gtc/2015/presentation/S5387-Jeremy-Bennett.pdf
15. *Gonakhchyan V.* Efficient Command Buffer Recording for Accelerated Rendering of Large 3D Scenes // Proceedings of the 12th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, 2018. P. 397–402.