На правах рукописи

Гонахчян Вячеслав Игоревич

Адаптивная стратегия рендеринга динамических трехмерных сцен

Специальность 05.13.11 — математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Москва — 2021

Работа выполнена в Федеральном государственном бюджетном учреждении науки «Институт системного программирования им. В.П. Иванникова Российской академии наук».

Научный руководитель:	**Семенов Виталий Адольфович**, доктор физико-математических наук, профессор
Официальные оппоненты:	**Михайлюк Михаил Васильевич**, доктор физико-математических наук, профессор, заведующий отделом программных средств визуализации Федерального государственного учреждения "Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук"
	Фролов Владимир Александрович, кандидат физико-математических наук, старший научный сотрудник Федерального государственного учреждения "Федеральный исследовательский центр Институт прикладной математики им. М.В. Келдыша Российской академии наук"
Ведущая организация:	Федеральное государственное бюджетное образовательное учреждение высшего образования «Нижегородский государственный архитектурно-строительный университет»

Защита состоится 10 июня 2021 года в 15 часов на заседании диссертационного совета Д 002.087.01 при Федеральном государственном бюджетном учреждении науки «Институт системного программирования им. В.П. Иванникова Российской академии наук», расположенном по адресу: 109004, Москва, ул.Александра Солженицына, 25.

С диссертацией можно ознакомиться в библиотеке и на сайте Федерального государственного бюджетного учреждения науки “Институт системного программирования им. В.П. Иванникова Российской академии наук” http://www.ispras.ru.

Автореферат разослан _____ 2021 г.

Учёный секретарь
диссертационного совета
Д 002.087.01,
к-т физ.-мат. наук Зеленов Сергей Вадимович

Общая характеристика работы

Актуальность исследования

Рендеринг трехмерных сцен является одной из ключевых задач компьютерной графики и широко применяется в таких предметных областях, как научная визуализация, автоматизация проектирования, инженерии и производства (CAD/CAM/CAE), компьютерные игры и анимация, виртуальная и дополненная реальность. В связи с перманентным ростом сложности сцен повышаются требования и к эффективности программных и аппаратных средств рендеринга. Несмотря на обширные исследования в этой области и большое количество опубликованных работ, задачам рендеринга больших динамических сцен уделяется относительно мало внимания. Вместе с тем, класс приложений, оперирующих с подобными сценами, чрезвычайно широк, что определяет актуальность темы исследований. Особую важность тема приобретает в связи со стремительным развитием технологий информационного моделирования зданий и сооружений (BIM), предусматривающих, в частности, интерактивные графические средства динамического моделирования процессов возведения сложных строительных объектов и реализации масштабных инфраструктурных программ.

В работе рассматриваются задачи отображения динамических трехмерных сцен с использованием современных видеокарт (GPU) и графических интерфейсов к ним (OpenGL, DirectX, Vulkan). Видеокарты обеспечивают высокоэффективную поточно-параллельную обработку графических элементов (треугольников, пикселей) для достижения требуемой частоты генерации изображений. Графические интерфейсы позволяют задать описание трехмерной сцены и передать его на видеокарту, установить программы для геометрических преобразований, проецирования треугольников на экранную плоскость, расчета функций освещения, а также сгенерировать итоговые изображения.

Однако при отображении сложных сцен, содержащих большое количество графических элементов и предполагающих использование сложных моделей материалов и освещения, видеокарты часто не справляются с большим объемом вычислений. Подобные ограничения делают невозможными разработку и применение интерактивных графических приложений, предполагающих генерацию и вывод изображений с частотой, соответствующей скорости отклика на пользовательские события.

Одним из традиционных подходов к повышению производительности рендеринга является уменьшение числа растеризуемых графических элементов за счет предварительного анализа и удаления невидимых поверхностей. Распространенные методы, применяемые при отображении динамических сцен, описаны в работах авторов: Coorg, S., Teller, S., Hudson, T., Manocha, D., Bittner, J., Wimmer, M., Mattausch, O. Наиболее перспективными среди них являются методы на основе аппаратных проверок видимости (hardware occlusion queries), которые в ряде случаев позволяют повысить эффективность отображения сцен. Однако регулярное использование аппаратных проверок может приводить к деградации производительности, принимая во внимание дополнительные расходы на подготовку и осуществление

самих проверок. Поэтому необходимыми становятся оценки эффективности применения аппаратных проверок с учетом вычислительных затрат и изменений в сцене.

Для ускорения методов удаления невидимых элементов используются структуры пространственной декомпозиции сцены, такие как окто-деревья и k-d-деревья. Иерархический обход деревьев с отбраковкой невидимых узлов позволяет существенно сократить количество отображаемых объектов. При этом следует учитывать, что затраты на обновление пространственных структур в процессе рендеринга больших динамических сцен могут оказаться неоправданно высокими.

Важными факторами, влияющими на эффективность процесса рендеринга динамических сцен, являются способы составления и отправки командных буферов. Как правило, описание объектов сцены формируется в виде буферов команд рендеринга, которые записываются в основную память один раз и отправляются на видеокарту по шине. Однако при отображении динамических сцен и использовании проверок видимости требуется повторная запись, которая может занимать продолжительное время. При незначительных локальных изменениях на каждом временном шаге моделирования динамической сцены можно сократить затраты на составление и отправку буферов за счет их фрагментации и кэширования в основной памяти. В этом случае требуется обновление только тех буферов, объекты которых подверглись изменениям на текущем временном шаге.

Таким образом, для эффективного рендеринга больших динамических сцен с использованием современных видеокарт перспективным представляется применение комбинированной стратегии, сочетающей в себе следующие методы и техники:

- удаление невидимых объектов с использованием методов пространственной декомпозиции и индексирования;
- аппаратные проверки видимости с отложенной загрузкой результатов;
- фрагментация и кэширование командных буферов с учетом локальных изменений в сцене.

Поскольку данные методы выполняются одновременно в вычислительной системе, важно сбалансированное использование ее ресурсов при конвейерной обработке и передаче графических данных. В частности, должна быть сбалансирована загрузка CPU и GPU процессоров. Вместе с тем, вариативность в количестве и сложности индивидуальных объектов сцены, характере и интенсивности динамики делает подобную балансировку крайне сложной или даже невозможной при фиксировании конкретных методов и техник. Обеспечить высокую производительность вычислительной системы на широком классе задач рендеринга больших динамических сцен представляется возможным в результате адаптивного управления графическим конвейером, предусматривающего выбор и настройку альтернативных базовых методов и техник с учетом доступных ресурсов системы и особенностей отображаемой сцены на конкретном интервале модельного времени в конкретной пространственной области.

Целью работы является исследование и разработка эффективной стратегии рендеринга динамических трехмерных сцен, сочетающей в себе методы удаления невидимых объектов, отложенные аппаратные проверки видимости, кэширование

командных буферов, а также реализующей адаптивный выбор методов и техник для ускорения процессов рендеринга и обеспечения возможностей разработки перспективных интерактивных графических приложений. Стратегия должна быть применима к широкому классу сцен, объекты которых геометрически представлены треугольными сетками, а их положение и материалы задаются индивидуально в виде соответствующих кусочно-линейных функций, явно зависящих от временного параметра. Предполагается также, что в рассматриваемых, так называемых, псевдо-динамических сценах превалируют локальные изменения, затрагивающие на каждом временном шаге моделирования относительно небольшое число объектов сцены.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. На основе сравнительного анализа существующих методов удаления невидимых поверхностей, техник составления командных буферов и реализации аппаратных проверок видимости выделить наиболее перспективные, применимые к рассматриваемому классу псевдо-динамических трехмерных сцен.
2. Рассмотреть возможные способы ускорения базовых методов и техник на основе структур пространственной декомпозиции и индексирования сцен. Выделить пространственные структуры, допускающие быстрые инкрементальные обновления с учетом локального характера изменений в псевдо-динамических сценах.
3. Разработать и исследовать модель производительности графического конвейера применительно к задачам рендеринга динамических трехмерных сцен, позволяющую оценивать требуемые ресурсы в зависимости от применяемых базовых методов и характеристик отображаемой сцены. В частности, модель должна устанавливать целесообразность применения проверок видимости.
4. Разработать адаптивную стратегию рендеринга динамических трехмерных сцен, реализующую выбор и настройку базовых методов в процессе отображения сцены на основе предложенной модели производительности графического конвейера.
5. Провести серию вычислительных экспериментов с синтетическими и реальными тестовыми наборами сцен для подтверждения релевантности примененной модели производительности графического конвейера и эффективности предложенной адаптивной стратегии.
6. Апробировать разработанную стратегию в составе системы визуального моделирования и планирования индустриальных проектов.

Научная новизна состоит в получении следующих результатов:

1. Предложена модель производительности графического конвейера для однопроходной схемы рендеринга динамических трехмерных сцен. Модель позволяет оценивать требуемые ресурсы (время обработки и передачи графических данных, объем основной и графической памяти) в зависимости от применяемых базовых методов и характеристик отображаемой сцены.

Для получения релевантных оценок на оборудовании пользователя параметры модели калибруются с использованием тестовых наборов.

2. Предложена адаптивная стратегия рендеринга динамических трехмерных сцен, реализующая выбор и настройку базовых методов удаления невидимых объектов, техник отложенных аппаратных проверок видимости и кэширования командных буферов на основе модели производительности графического конвейера непосредственно в процессе отображения сцены. Для ускорения вычислений стратегия предусматривает использование регулярных окто-деревьев в качестве структуры пространственной декомпозиции и индексирования динамической сцены.
3. Предложен метод генерации динамических трехмерных сцен, который позволяет синтезировать семейства сцен с разными характеристиками, определяемыми количеством и сложностью индивидуальных объектов, пространственной разреженностью сцены, интенсивностью событий и их пространственно-временной когерентностью. Сгенерированные синтетические сцены могут использоваться для организации репрезентативных наборов тестов и тестирования методов и программных средств компьютерной графики.

Практическая ценность полученных результатов заключается в возможности применения предложенной стратегии и методов рендеринга для эффективного отображения широкого класса динамических трехмерных сцен, а также для разработки интерактивных графических приложений в таких областях как научная визуализация, автоматизация проектирования, инженерии и производства (CAD/CAM/CAE), компьютерные игры и анимация, виртуальная и дополненная реальность.

Разработанная стратегия и методы программно реализованы в составе системы визуального пространственно-временного (4D) моделирования и планирования индустриальных проектов, обеспечивая прирост производительности при отображении больших динамических сцен в 2–9 раз. В настоящее время данная система успешно применяется в более чем трех сотнях ведущих индустриальных компаний в тридцати шести странах мира, в том числе, и в Российской Федерации.

Методология и методы исследования

Результаты диссертации были получены с использованием подходов и методов вычислительной геометрии, компьютерной графики, теории нелинейной оптимизации.

Основные положения, выносимые на защиту:

1. Модель производительности графического конвейера для однопроходной схемы рендеринга динамических трехмерных сцен.
2. Адаптивная стратегия рендеринга динамических трехмерных сцен.
3. Метод генерации динамических трехмерных сцен.

Апробация работы

Основные результаты работы докладывались на следующих конференциях:

- 27-я Международная конференция по компьютерной графике, обработке изображений и машинному зрению, системам визуализации и виртуального окружения. (GraphiCon 2017) (24–28 сентября 2017 года, г. Пермь).
- 12-я Международная конференция по компьютерной графике, научной визуализации, машинному зрению и обработке изображений (CGVCVIP 2018) (18–20 июля 2018 года, г. Мадрид).
- 13-я Международная конференция по компьютерной графике, научной визуализации, машинному зрению и обработке изображений (CGVCVIP 2019) (16–18 июля 2019 года, г. Порту).
- 29-я Международная конференция по компьютерной графике и машинному зрению (GraphiCon 2019) (23–26 сентября 2019 года, г. Брянск).

Публикации

По теме диссертации опубликовано 7 работ [1–7], в том числе 2 статьи [1, 2] в реферируемых научных журналах из списка изданий, рекомендованных ВАК РФ. Работа [3] входит в международные системы цитирования Web of Science и Scopus. Работы [4; 5] индексированы в Scopus. В работах [1; 2; 5–7] все научные результаты принадлежат автору. В работе [3] автор разработал и описал метод составления и использования командных буферов в сочетании со структурами пространственного индексирования сцены. В работе [4] автор разработал и описал метод генерации динамических сцен.

Личный вклад автора

Все представленные в работе результаты получены лично автором.

Объем и структура работы

Представленная работа состоит из введения, пяти глав основного содержания, заключения и списка используемых источников, включающего 94 работы. Общий объем работы составляет 150 страниц, включая 42 рисунка и 34 таблицы.

Содержание работы

Во введении обоснована актуальность диссертационной работы, сформулирована цель и аргументирована научная новизна исследований, показана практическая значимость полученных результатов.

В **первой главе** приведен обзор основных методов, которые используются для реализации эффективного рендеринга на GPU. В данной работе рассматривается выполнение рендеринга с использованием аппаратных возможностей современных видеокарт. Видеокарты состоят из потоковых процессоров, выполняющих поточно-параллельную обработку команд (SIMD), разделяемой памяти и общей памяти, блоков вывода для объединения результатов и записи в кадровый буфер. В результате работы графического конвейера по полигональному представлению трехмерной сцены создается двухмерное изображение.

В разделе 1.1 рассматриваются возможности современных программных интерфейсов рендеринга (OpenGL, DirectX, Metal, Vulkan).

В разделе 1.2 приведен обзор методов удаления невидимых поверхностей. Распространенные варианты этих методов описаны в работах авторов: Airey, J.M.,

Teller, S., Luebke, D., Greene, N., Zhang, H., Cohen-Or, D., Funkhouser, T., Wonka, P., Fuchs, H., Durand, F., Boreskov, A. В методах с использованием потенциально видимых множеств объектов, порталов, расширенных проекций (Potentially visible sets, Portals and mirrors, Extended projections) выполняется предварительный анализ видимости элементов из разных пространственных ячеек. При изменении положения камеры и смене занимаемой ячейки уточняется множество видимых элементов. Данные методы эффективны для статических сцен, в которых статус видимости объектов не меняется в зависимости от происходящих в сцене событий и может быть вычислен заранее.

Метод иерархического буфера глубины (Hierarchical z-buffer), метод исключения на основе бинарного пространственного разбиения (BSP tree culling) отбирают объекты сцены, видимые из заданного положения камеры, с помощью специальных пространственных структур данных. Проверки видимости, выполняясь на центральном процессоре (CPU), позволяют составить буфер команд только для видимых объектов и, тем самым, уменьшить нагрузку на видеокарту. Ограниченные вычислительные ресурсы CPU являются наиболее узким местом для подобных реализаций, обычно проявляющимся при отображении больших сцен.

Методы, описанные в работах авторов Coorg, S., Teller, S., Hudson, T., Manocha, D., выполняют проверки видимости с учетом имеющихся в сцене больших объектов-преград (large occluders). На этапе предобработки подобные преграды, если присутствуют в сцене, выявляются. На этапе отображения определяется статус перекрытий объектов, для чего они сортируются в порядке удаленности от камеры. Методы имеют отмеченные ограничения, а анализ видимости объектов с учетом множественных преград (occluder fusion) может требовать значительных вычислительных ресурсов.

В разделе 1.3 рассматриваются методы на основе оценки времени выполнения рендеринга.

В разделе 1.4 рассматриваются методы пространственного индексирования объектов сцены. Для этого используются окто-деревья, k-d-деревья, иерархии BVH (Bounding Volume Hierarchies). В качестве ограничивающих объемов применяются ориентированные вдоль координатных осей параллелепипеды AABB (Axis Aligned Bounding Box), произвольно ориентированные параллелепипеды OBB (Oriented Bounding Box), а также многогранники с фиксированным количеством ориентаций граней k-DOP. Структуры декомпозиции сцены и объектов, выполняющие функции пространственных индексов, позволяют существенно сократить объем вычислений за счет быстрого вынесения отрицательного вердикта относительно видимости объектов, что достигается за счет недорогих, иерархически организованных проверок видимости ограничивающих их пространственных ячеек или объемов.

Важным достоинством методов пространственной декомпозиции и индексирования является применимость к динамическим сценам, хотя в этом случае необходимо эффективно обновлять соответствующие пространственные структуры при наступлении соответствующих изменений в сцене. Возможные способы решения этой проблемы предложены в работах авторов Sudarsky, O., Gotsman, C., Semenov, V., Zolotov, V. Метод временного ограничивающего объема (Temporal Bounding

Volume) вычисляет результирующий параллелепипед, ограничивающий объект с учетом всех его положений на траектории движения, и подготавливает необходимые структуры для эффективного рендеринга сцены с удалением невидимых элементов. Метод успешно применяется для сцен с детерминированным характером динамики, однако имеет ограничения для сцен, в которых события происходят случайно или моделируются специальными условиями.

В разделе 1.5 приведены выводы, которые позволили определить дальнейшее направление исследований. Для динамических сцен наиболее целесообразным является использование однопроходной схемы рендеринга, которая позволяет сократить время записи командных буферов и ресурсы, затрачиваемые на преобразование вершин. Составление командного буфера проводится на CPU, чтобы можно было использовать аппаратные проверки видимости совместно со структурой пространственного разбиения. Для отображения динамических сцен с интерактивной частотой кадров необходимо своевременно и эффективно проводить проверки видимости. Метод выполнения проверок видимости относительно буфера глубины GPU процессора на основе расширения NV_Occlusion_Query справляется с целевым семейством сцен, поддерживается в современных программных интерфейсах рендеринга.

Во **второй главе** предложена **модель производительности графического конвейера**, которая расширяет существующие модели, описанные в работах Funkhouser T., Wimmer M. Новизна заключается в том, что учитываются следующие аспекты рендеринга:

- затраты на запись и отправку командных буферов, в том числе с использованием рассматриваемых техник;
- затраты на отправку, выполнение, получение результатов аппаратных проверок видимости.

В разделе 2.1 приведено описание исследуемого графического конвейера (рис. 1).

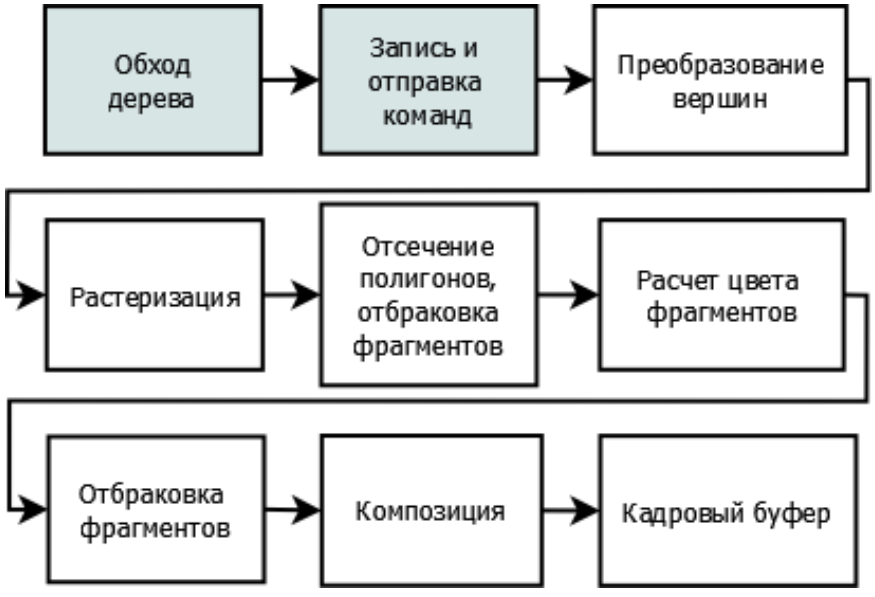

Рис. 1 — Исследуемая модель графического конвейера. Первые два этапа выполняются на CPU, остальные — на GPU.

В разделе 2.2 рассматривается прямая однопроходная схема рендеринга динамических трехмерных сцен. В рамках этой схемы записывается одна команда рендеринга для каждого объекта сцены, командный буфер составляется на центральном процессоре, осуществляется один проход по графическим элементам для генерации кадрового буфера. В данной работе рассматриваются способы реализации процессов рендеринга, выполняющие шаги:

1. Изменение видимости, положений, материалов объектов сцены при изменении анимационного времени t_{anim}.
2. Получение результатов аппаратных проверок видимости.
3. Ожидание готовности командного буфера.
4. Обход структуры пространственного разбиения и выполнение отсечений.
5. Запись команд рендеринга для отдельных объектов сцены.
6. Запись ссылок на вспомогательные буфера в главный командный буфер.
7. Запись буфера для выполнения аппаратных проверок видимости.
8. Отправка командного буфера на выполнение.

Шаги 1, 3, 8 выполняются для всех способов. Запись команд рендеринга может проводиться напрямую в главный командный буфер (шаг 5) или может выполняться запись командных буферов для групп объектов сцены (шаг 6). Для ускорения рендеринга может использоваться структура пространственного разбиения H (шаг 4), могут выполняться аппаратные проверки видимости (шаги 2, 7). Условное выполнение этих шагов задается параметрами h, g, q в формулах ниже.

В разделе 2.3 задается описание динамической сцены.

В разделе 2.4 выводятся формулы для оценки потребляемой памяти при выполнении рендеринга.

В разделе 2.5 выводятся формулы для оценки **времени выполнения рассматриваемых способов реализации процессов рендеринга**. Время выполнения рендеринга на центральном процессоре выражается с помощью формулы:

$$T_{CPU}\big(S, S_{vis}, H, H_{vis}, H_{upd}, H_q, h, g, q, HW\big) =$$
$$T_{update}(S, HW) + [h = 1]T_{update}(H, HW) +$$
$$[h = 0]T_{fc}(S, HW) + [h = 1](T_{traverse}(H, HW) + T_{fc}(H, HW)) +$$
$$[g = 0]T_{buf}(S_{vis}, HW) + [g = 1](T_{buf}\big(H_{upd}, HW\big) + T_{\mathrm{sec\_}buf}(H_{vis}, HW)) +$$
$$[q = 1](T_{buf}\big(H_q, HW\big) + T_{recv\_query}\big(H_q, HW\big)),$$

где h, g, q — параметры, которые задают способ рендеринга,

$[h = 1] = \begin{cases} 1, \text{если } h = 1 \\ 0, \text{иначе} \end{cases}$ — скобка Айверсона,

индексом vis помечены видимые объекты (узлы), upd — обновленные на текущем шаге анимации,

T_{update} — время обновления S (вычисление матриц, запись новых значений) или H (добавление объектов) при изменении анимационного времени,

$T_{traverse}, T_{fc}$ — время обхода структуры пространственного разбиения, выполнения отсечений,

$T_{buf}, T_{\text{sec\_buf}}$ — время записи, отправки командных буферов,

T_{recv_query} — время получения результатов проверок видимости.

Время выполнения рендеринга на GPU процессоре выражается с помощью формулы:

$$T_{GPU}(S, H_q, q, HW) = \max\left(T_{state}(S, HW), T_{vert}(S, HW), T_{frag}(S, Vp, HW)\right) + [q = 1]\max\left(T_{state}(H_q, HW), T_{vert}(H_q, HW), T_{frag}(H_q, Vp, HW), T_{exec\_query}(H_q, HW)\right),$$

где $T_{state}, T_{vert}, T_{frag}, T_{exec\_query}$ — время работы рассматриваемых этапов конвейера GPU.

Итоговое время рендеринга выражается с помощью формулы:

$$T(S, S_{vis}, H, H_{vis}, H_{upd}, H_q, h, g, q, HW) = \begin{cases} T_{CPU}(S, S_{vis}, H, H_{vis}, H_{upd}, H_q, h, g, q, HW) + T_{GPU}(S_{vis}, H_q, q, HW), \text{один главн. коман. буфер} \\ \max\left(T_{CPU}(S, S_{vis}, H, H_{vis}, H_{upd}, H_q, h, g, q, HW), T_{GPU}(S_{vis}, H_q, q, HW)\right), \text{иначе.} \end{cases}$$

Эта формула задает время рендеринга для двух сценариев. В первом сценарии используется один главный командный буфер (буфер, который отправляется на GPU). При этом запись команд возможна только тогда, когда чтение буфера завершено на GPU процессоре, а работа на CPU и GPU выполняется последовательно. Во втором сценарии используется несколько главных командных буферов и выполняется параллельная работа над разными буферами.

В **третьей главе** предложена **адаптивная стратегия рендеринга** динамических трехмерных сцен, а также описываются вспомогательные методы и техники.

В разделе 3.1 предложена **техника составления командных буферов** с использованием окто-дерева. Создается окто-дерево и заполняется объектами сцены. Для каждого непустого узла создается буфер с командами рендеринга. При изменении свойств объектов динамической сцены (положение, видимость, материал) выполняется перезапись командных буферов только для затронутых узлов. Использование этой техники позволяет сократить время записи командных буферов, когда происходят локальные изменения в процессе анимации динамической сцены.

Выведем условие того, что памяти достаточно для хранения окто-дерева:

$$|H|\left(M_{node} + \max\left(M_{\text{sec\_buf}}, numInstPerNode\ M_{cmd}\right)\right) \leq availMem,$$

где $numInstPerNode$ — максимальное количество объектов в узле дерева,

M_{node} — требуемая память для хранения узла,

$M_{cmd}, M_{\text{sec\_buf}}$ — требуемая память для хранения команды, вспомогательного буфера.

Таким образом, чтобы уложиться в заданный объем доступной памяти, нужно задать начальное значение *numInstPerNode.* Затем изменять его и проверять это условие до тех пор, пока не будет выбрано подходящее значение *numInstPerNode.*

В разделе 3.2 предложен **метод оценки количества аппаратных проверок видимости** для эффективного выполнения рендеринга.

Выделяются следующие стадии метода:

1. Отбор ограниченного количества $N_{queries}$ узлов дерева для выполнения проверок видимости. Постепенное накопление данных о видимости графических элементов и узлов дерева.
2. Вывод количества проверок видимости $N_{queries}^{efficient}$ для эффективного рендеринга на основе накопленных данных.
3. Отбор $N_{queries}^{efficient}$ узлов дерева для выполнения проверок видимости до тех пор, пока это выгодно (выполнение проверок позволяет сократить время рендеринга) и количество изменившихся в процессе анимации объектов незначительно.

Во время первой стадии производится обход дерева в ширину для отбора узлов согласно критерию отбора:

$$t_{\text{sec\_buf}}(HW) + \max\left(t_{state}(HW), NumVerts(node)\hat{t}_{vert}, NumFrags(node, Vp)\hat{t}_{frag}\right) \geq \text{C}\left(t_{buf}(HW) + t_{exec\_query}(HW) + t_{recv\_query}(HW)\right),$$

где $\hat{t}_{vert}$ — среднее время преобразования вершины для всех материалов сцены,

$\hat{t}_{frag}$ — среднее время расчета цвета фрагмента для всех материалов сцены,

C — некоторая константа больше единицы.

Проверяется видимость отобранных узлов и объектов, содержащихся в них. Собираются данные, задающие зависимость количества невидимых элементов (узлов дерева, объектов, вершин, фрагментов) в сцене от количества проверок видимости.

Вычислим количество проверок видимости для эффективного рендеринга с помощью предложенной модели производительности графического конвейера. Сначала рассмотрим сценарий, когда этапы рендеринга на CPU и GPU выполняются последовательно. Как правило, самым долгим этапом конвейера GPU является преобразование вершин или расчет цвета фрагментов. Тогда время рассматриваемых этапов определяется по формуле:

$$T(x) = c(x)t_c + s(x)t_s + e(x)t_e,$$

где x — количество проверок видимости узлов структуры пространственного разбиения,

$c(x)$ — количество записываемых команд рендеринга,

$s(x)$ — количество отправляемых командных буферов,

$e(x)$ — количество графических элементов на лимитирующем этапе конвейера (вершин или фрагментов),

t_c — время записи одной команды рендеринга,

t_s — время отправки вспомогательного командного буфера,

t_e — время выполнения лимитирующего этапа конвейера (преобразования вершины, расчета цвета фрагмента).

Накопленные данные о видимости узлов дерева позволяют определить функции $c(x)$, $s(x)$, $e(x)$. Количество невидимых узлов, объектов, вершин (фрагментов)

монотонно растет с увеличением количества проверок видимости. Зачастую рост происходит быстрее при малых x, а затем останавливается при больших x. Приближенное описание таких данных задается с помощью логарифмической зависимости $c_1 + c_2 \ln(1 + x)$, где c_1 и c_2 — коэффициенты, которые вычисляются методом наименьших квадратов.

Накопив достаточное количество данных о видимости узлов дерева, определим коэффициенты регрессии и подставим в функции:

$$c(x) = \alpha_n(o_0 - o_1 - o_2 \ln(1 + x)) + x,$$
$$s(x) = (n_0 + 1 - s_1 - s_2 \ln(1 + x)),$$
$$e(x) = (e_0 + px - e_1 - e_2 \ln(1 + x)),$$

где α_n — средняя доля перезаписываемых командных буферов узлов дерева,

o_0 — текущее количество объектов в сцене,

n_0 — количество непустых узлов окто-дерева,

e_0 — количество вершин (фрагментов) при выполнении рендеринга сцены,

p — количество вершин (фрагментов), приходящихся на один узел окто-дерева,

$o_1, o_2, s_1, s_2, e_1, e_2$ — коэффициенты регрессии.

Определение минимума $T(x)$ вместе с ограничениями на количество проверок видимости $0 \le x \le n_0$ является задачей нелинейного программирования. Функция $T(x)$ является непрерывной и дифференцируемой на множестве ограничений, поэтому минимум является либо стационарной точкой, либо лежит на границе множества ограничений. Для нахождения стационарной точки решим уравнение $T'(x) = 0$ и получим:

$$x_1^* = \frac{\alpha_n o_2 t_c + s_2 t_s + e_2 t_e}{p t_e + t_c} - 1.$$

В случае выхода значения стационарной точки за пределы отрезка $[0, n_0]$ берется ближайшая точка на границе. Сравнив $T(0)$, $T(n_0)$, $T(x_1^*)$, найдем минимальное время выполнения рендеринга сцены и соответствующее количество проверок видимости x.

Рассмотрим сценарий с параллельной работой с буферами на CPU и GPU. Тогда время рендеринга определяется по формуле: $T(x) = \max(T_{CPU}(x), T_{GPU}(x))$. Нужно определить минимум двух функций:

$$T_{CPU}(x) = c(x)t_c + s(x)t_s,$$
$$T_{GPU}(x) = e(x)t_e.$$

Уравнение $T'_{CPU}(x) = 0$ имеет решение: $x_2^* = \frac{\alpha_n o_2 t_c + s_2 t_s}{t_c} - 1$. Сравнив $T_{CPU}(0)$, $T_{CPU}(n_0)$, $T_{CPU}(x_2^*)$, найдем минимальное время выполнения рендеринга на CPU и соответствующее значение x_2. Уравнение $T'_{GPU}(x) = 0$ имеет решение: $x_3^* = \frac{e_2}{p} - 1$. Сравнив $T_{GPU}(0), T_{GPU}(n_0), T_{GPU}(x_3^*)$, найдем минимальное время выполнения рендеринга на GPU и соответствующее значение x_3. В случае пересечения $T_{CPU}(x)$ и $T_{GPU}(x)$ необходимо также использовать точки пересечения в качестве граничных точек. Найдем корни функции $T_{CPU}(x) - T_{GPU}(x)$ с помощью метода деления отрезка пополам, взяв в качестве начальных отрезков $[0, x_4^*]$, $[x_4^*, n_0]$, где

$x_4^* = \frac{e_2 t_e - \alpha_n o_2 t_c - s_2 t_s}{p t_e - t_c} - 1$. Если функции пересекаются, то получим корни $x_{\mathrm{r}1}$, $x_{\mathrm{r}2}$. После этого сравним $T(0)$, $T(n_0)$, $T(x_2)$, $T(x_3)$, а также значения $T(x_{\mathrm{r}1})$, $T(x_{\mathrm{r}2})$ в случае пересечения $T_{CPU}(x)$ и $T_{GPU}(x)$ на соответствующих отрезках, и возьмем количество проверок видимости x, которое соответствует минимуму $T(x)$.

В разделе 3.3 предложена **адаптивная стратегия рендеринга**. С помощью разработанной модели производительности графического конвейера проводится оценка эффективности базовых методов и техник для данного состояния динамической сцены. В зависимости от полученных результатов выбирается один из трех способов реализации процессов рендеринга в рамках однопроходной схемы:

1. Запись командного буфера без кэширования. Отсечение объектов, не попадающих в область видимости камеры.
2. Пространственное индексирование с использованием окто-дерева. Запись и кэширование командных буферов для каждого непустого узла. Отсечение узлов, не попадающих в область видимости камеры. Сбор данных для оценки количества невидимых узлов, объектов, вершин и фрагментов для данного состояния динамической сцены.
3. Пространственное индексирование с использованием окто-дерева. Запись и кэширование командных буферов для каждого непустого узла. Отсечение узлов, не попадающих в область видимости камеры. Выполнение аппаратных проверок видимости.

Рассмотрим критерии применения этих способов рендеринга. Первый способ используется, пока выполняется условие:

$$T_{buf}(S_{vis}, HW) + T_{fc}(S, HW) <$$
$$T_{update}(H, HW) + T_{buf}\big(H_{upd}, HW\big) + T_{\mathrm{sec\_buf}}(H_{vis}, HW) + T_{fc}(H, HW).$$

Для сцен с малым значением α_n и существенным временем записи это условие перестает выполняться и происходит переход ко второму способу рендеринга. Производится создание (обновление) окто-дерева. Также начинается накопление данных о количестве невидимых элементов. Второй способ рендеринга используется до тех пор, пока выполняется условие:

$$T\big(S, S_{vis}, H, H_{vis}, H_{upd}, \emptyset, h = 1, g = 1, q = 0, HW\big) <$$
$$T\big(S, S'_{vis}, H, H'_{vis}, H_{upd}, H_q, h = 1, g = 1, q = 1, HW\big),$$

где $T\big(S, S'_{vis}, H, H'_{vis}, H_{upd}, H_q, h = 1, g = 1, q = 1, HW\big)$ — время рендеринга объектов с выполнением проверок видимости,

S_{vis}, H_{vis} — видимые объекты сцены, узлы окто-дерева (видимость определяется с помощью усеченной пирамиды обзора),

S'_{vis}, H'_{vis} — видимые объекты сцены, узлы окто-дерева (видимость определяется с помощью усеченной пирамиды и аппаратных проверок видимости),

H_q — узлы окто-дерева, для которых выполняются аппаратные проверки видимости.

При неэффективном выполнении проверок видимости способ рендеринга изменяется в соответствии с условиями переходов.

В разделе 3.4 предложен **метод генерации сцен**, который позволяет синтезировать семейства сцен с разными характеристиками, определяемыми количеством и сложностью индивидуальных объектов, пространственной разреженностью сцены, интенсивностью событий и их пространственно-временной когерентностью. Время вывода объектов рассчитывается с помощью кривых, заполняющих пространство. Это позволяет воспроизвести процесс постепенного возведения сложных строительных объектов. В результате работы метода создается файл в формате FBX с полигональным представлением сцены и информацией о видимости объектов в зависимости от времени. На рисунке 2 показаны примеры сцен, полученных с помощью метода.

Рис. 2 — Одно высокое здание (слева). Группа маленьких зданий (центр). Вывод объектов с низкой пространственно-временной когерентностью (справа).

В **четвертой главе** рассмотрены основные программные модули, используемые в реализации модели производительности и стратегии рендеринга. В разделе 4.1 описаны функции разработанной графической библиотеки в составе системы для пространственно-временного моделирования и планирования (рис. 3). Поддерживается вывод на экран динамической сцены, навигация по сцене с использованием мыши, запись проектной анимации в видеофайл.

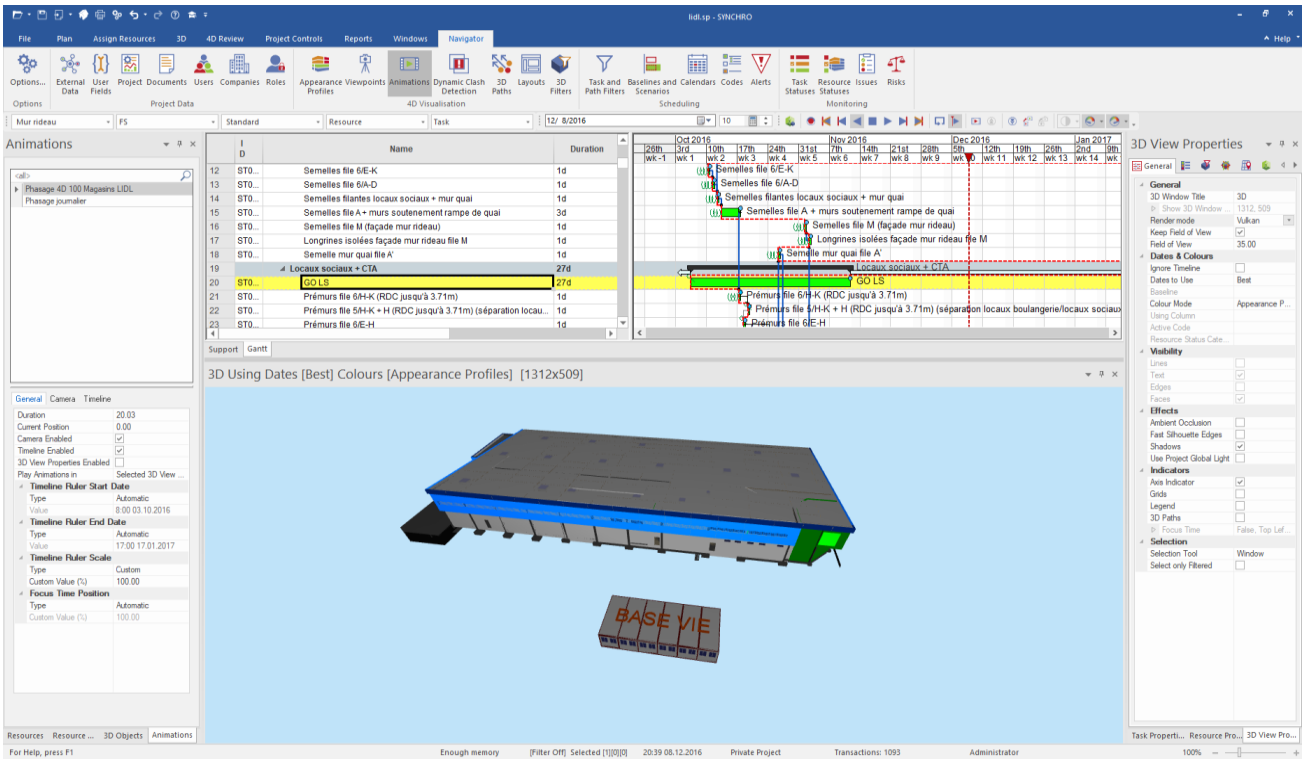

Рис. 3 — Главное окно программы для пространственно-временного моделирования и планирования индустриальных проектов.

В разделах 4.2 и 4.3 описаны программные модули, реализующие модель производительности графического конвейера и адаптивную стратегию рендеринга.

В **пятой главе** приводятся результаты вычислительных экспериментов для подтверждения эффективности разработанных методов и модели производительности графического конвейера.

В разделе 5.1 рассматриваются реальные и синтетические тестовые наборы сцен. Индустриальные сцены показаны на рисунке 4.

В разделе 5.2 проведено тестирование эффективности методов пространственного индексирования. Показано, что наиболее целесообразным при отображении динамических сцен является использование окто-дерева.

В разделе 5.3 приведен анализ эффективности аппаратных проверок видимости. Показано, что большинство невидимых объектов можно вычислить с помощью ограниченного количества проверок видимости. Это достигается благодаря иерархическому выполнению проверок видимости при обходе дерева в ширину.

В разделе 5.4 приведен анализ производительности составления и отправки командных буферов. Показано, что для сокращения затрат при записи буферов нужно кэшировать вспомогательные буфера с командами рендеринга.

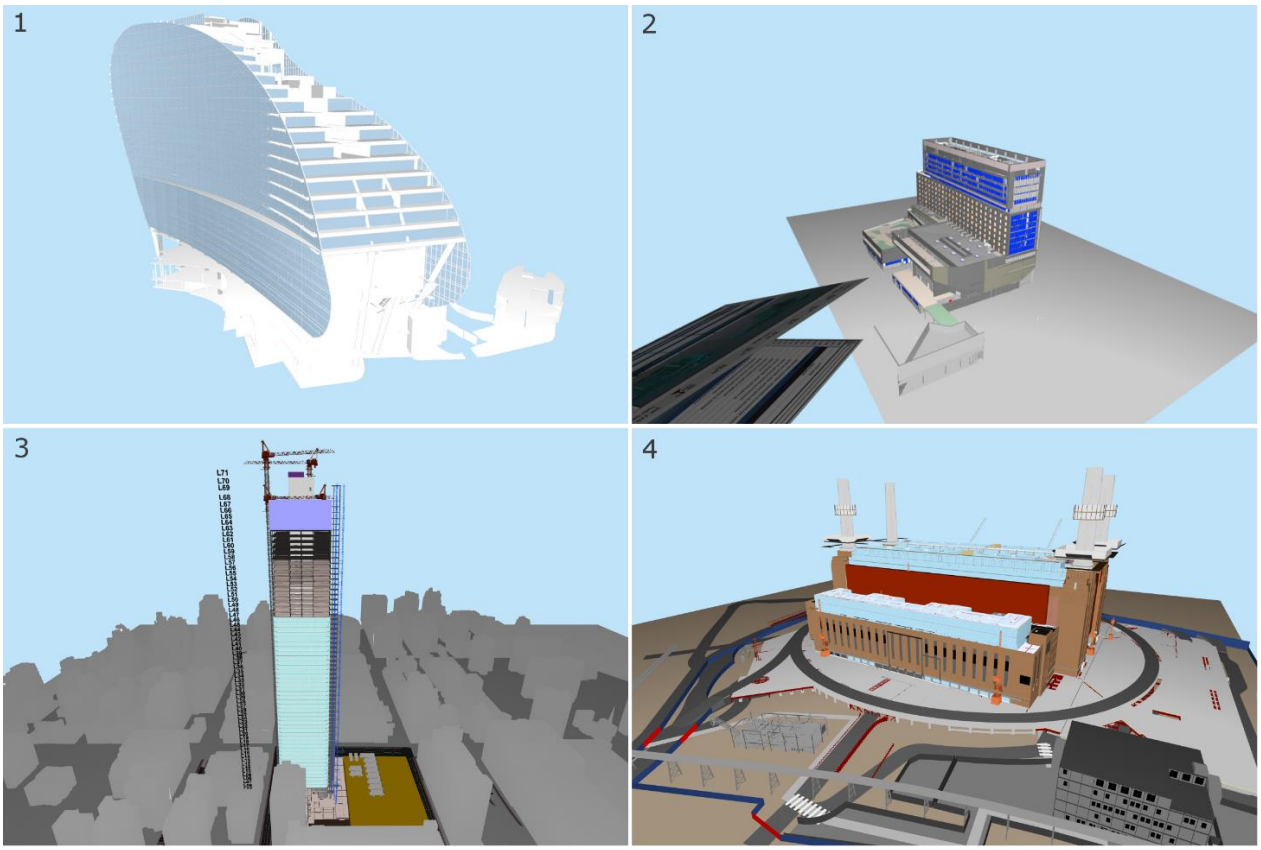

Рис. 4 — Индустриальные динамические сцены 1–4.

В разделе 5.5 проведено тестирование модели производительности графического конвейера. Использовались синтетические и реальные наборы данных. В рамках тестирования проводилось измерение среднего времени рендеринга за несколько кадров и вычисление времени по выведенной формуле, посчитано отклонение от измеренного времени. Результаты тестирования показали, что предложенную модель производительности можно применять для получения релевантных оценок времени рендеринга при использовании разнообразных сцен и способов рендеринга.

В разделе 5.6 приведен анализ производительности адаптивной стратегии рендеринга. Тестирование проводится на динамических сценах 1–4. В процессе анимации сцен происходит добавление новых объектов, удаление временных объектов, изменение цвета объектов. Тестирование заключается в отображении сцены с фиксированным положением камеры, проигрывании анимации и измерении времени рендеринга с применением одного из трех способов реализации рендеринга:

- Frustum Culling. Выполняется обход окто-дерева с отсечением узлов, не попадающих в камеру. Проводится запись команд в главный командный буфер без кэширования.
- Occlusion queries and FC. Выполняются аппаратные проверки видимости относительно буфера глубины на GPU процессоре для всех узлов окто-дерева. Также применяется разработанная техника записи команд с использованием кэширования и фрагментации буферов.
- Предложенная адаптивная стратегия.

Методы отбраковки объектов с использованием усеченной пирамиды видимости (view frustum) и аппаратных проверок видимости (occlusion queries) широко используются в индустрии. Сравнение позволяет показать целесообразность применения различных способов рендеринга в рамках стратегии в зависимости от состояния сцены и производительности вычислительной системы. Результаты тестирования приведены на рисунке 5.

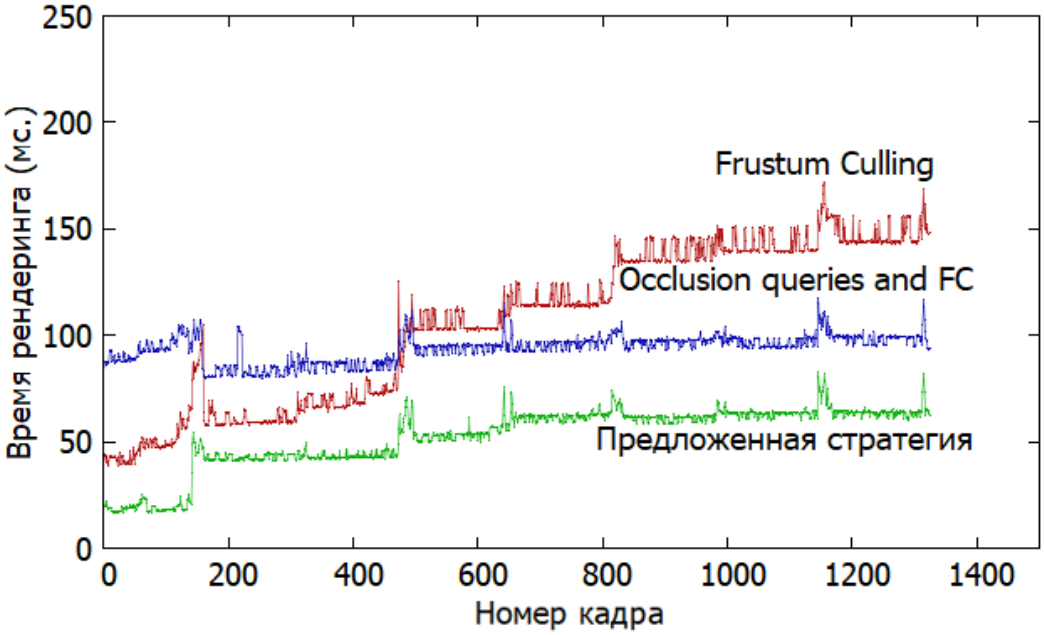

Рис. 5 — Сравнение времени рендеринга с использованием рассматриваемых способов при отображении сцены 4 и одновременном проигрывании анимации.

В среднем за счет адаптивного управления графическим конвейером, предусматривающего выбор и настройку альтернативных базовых методов и техник, достигается прирост производительности в 1.3–2.5 раза на рассмотренных тестовых сценах.

В разделе 5.7 приводится сравнение производительности разработанной и текущей коммерческой графических библиотек, которые используются в составе системы визуального пространственно-временного (4D) моделирования и планирования индустриальных проектов. В таблице 1 приведено сравнение времен проигрывания проектной анимации. Разработанные методы позволяют получить прирост производительности в 2–9 раз при работе с относительно большими динамическими сценами за счет фрагментации и обновления только затронутых узлов дерева и их командных буферов.

Таблица 1 — Время работы при проигрывании проектных анимаций тестовых сцен с использованием графических библиотек

Способ / Сцена	Текущая графич. библиотека, сек.	Разработанная графич. библиотека, сек.
1	108	69
2	119	30
3	381	160
4	735	84

В заключении подводятся итоги проведенного исследования.

Основные результаты

Основные результаты диссертационной работы состоят в следующем:

1. Предложена модель производительности графического конвейера для однопроходной схемы рендеринга динамических трехмерных сцен. Модель позволяет оценивать требуемые ресурсы (время обработки и передачи графических данных, объем основной и графической памяти) в зависимости от применяемых базовых методов и характеристик отображаемой сцены. Для получения релевантных оценок на оборудовании пользователя параметры модели калибруются с использованием тестовых наборов.
2. Предложена адаптивная стратегия рендеринга динамических трехмерных сцен, реализующая выбор и настройку базовых методов удаления невидимых объектов, техник отложенных аппаратных проверок видимости и кэширования командных буферов на основе модели производительности графического конвейера непосредственно в процессе отображения сцены. Для ускорения вычислений стратегия предусматривает использование регулярных окто-деревьев в качестве структуры пространственной декомпозиции и индексирования динамической сцены.
3. Предложен метод генерации динамических трехмерных сцен, который позволяет синтезировать семейства сцен с разными характеристиками, определяемыми количеством и сложностью индивидуальных объектов, пространственной разреженностью сцены, интенсивностью событий и их пространственно-временной когерентностью. Сгенерированные синтетические сцены могут использоваться для организации репрезентативных наборов тестов и тестирования методов и программных средств компьютерной графики.
4. Проведено исследование и обоснование разработанной адаптивной стратегии рендеринга псевдо-динамических трехмерных сцен. В частности, показана ее высокая эффективность на разнообразных синтетических и реальных индустриальных сценах.

Публикации автора по теме диссертации

1. Гонахчян В.И. Модель производительности графического конвейера для однопроходной схемы рендеринга динамических трехмерных сцен // Труды института системного программирования РАН. 2020. №4 (32). стр. 53–72.
2. Гонахчян В.И. Алгоритм удаления невидимых поверхностей на основе программных проверок видимости // Труды института системного программирования РАН. 2018. №2 (30). стр. 81–98.
3. 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. Vol. 46. pp. 223–232.
4. Gonakhchyan V., Tarlapan O., Semenov V. Generating dynamic 3D scenes for rendering benchmarks // MCCSIS 2019: 13th Multi Conference on Computer Science and Information Systems. 2019. pp. 485–488.
5. Gonakhchyan V.I. Efficient command buffer recording for accelerated rendering of large 3d scenes // MCCSIS 2018: 12th Multi Conference on Computer Science and Information Systems. 2018. pp. 397–402.
6. Gonakhchyan V.I. Comparison of hierarchies for occlusion culling based on occlusion queries // Proceedings of the GraphiCon 2017 conference on Computer Graphics and Vision. 2017. pp. 32–36.
7. Гонахчян В.И. Адаптивный метод рендеринга динамических трехмерных сцен // Международная конференция ГрафиКон-2019 по компьютерной графике и машинному зрению. 2019. стр. 32–36.