СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

BRDF – Bidirectional Reflectance Distribution Function (двулучевая функция отражательной способности)

MSAA – Multi-Sample Anti-Aliasing (множественная выборка сглаживания)

ω_o – направление наблюдения в BRDF

ω_i – направление падающего света в BRDF

ω_g – геометрическая нормаль поверхности

ω_n – мезонормаль

ω_m – микронормаль

NDF – normal distribution function (функция распределения нормалей)

SDF – slope distribution function (функция распределения уклонов)

PBR – physically-based rendering (физически достоверная отрисовка)

$\langle .,. \rangle$ – оператор ограниченного снизу нулем скалярного произведения

$H(x)$ – функция Хевисайда

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Алиасинг – нежелательные эффекты при отрисовке трехмерной сцены на дисплей устройства, состоящий из сетки дискретных пикселей

Оффлайн отрисовка – отрисовка, в которой нет ограничений по времени отрисовки одного кадра, в противоположность отрисовки в реальном времени

Прямая отрисовка – метод отрисовки освещения, в котором каждый объект рисуется с учетом освещения

Отложенная отрисовка – метод отрисовки освещения, в котором расчет освещения происходит после сбора информации о кадре в промежуточный буфер

GBuffer – набор текстур, создаваемый для отрисовки освещения в технике отложенной отрисовки

Альфа-композитинг – наложение изображения на другое изображение с учетом прозрачности

Физически достоверная отрисовка – отрисовка, подчиняющаяся закону сохранения энергии

Мезонормаль – средняя нормаль с карты нормалей по фрагменту поверхности

ВВЕДЕНИЕ

Отрисовка поверхности океанов и других водоемов в солнечную погоду представляет собой достаточно сложную задачу в компьютерной графике. Методы симуляции эффектов, создаваемых светом на поверхности воды в реальности, разнятся от более приближенного к реальности представления воды как гладкой поверхности с большим количеством высокочастотных геометрических деталей, до более стилизованного представления воды как блестящей поверхности.

Одной из основных проблем при отрисовки отражений и отблесков на поверхности воды является алиасинг. Алиасингу в той или иной мере подвержены практически все способы отрисовки отражений и отблесков. Это связано с тем, что отрисовка трехмерных объектов и визуальных эффектов производится на экран из дискретных пикселей, что может создавать множество различных проблем, ухудшающих визуальное качество кадра. [1]

В графике реального времени нормой является взятие выборки из одной точки сцены для отрисовки одного пикселя. Очевидно, что при этом результат отрисовки пикселя может сильно зависеть от того, куда попадает эта точка. Наивное решение проблемы с увеличением размера выборки ведет к экспоненциальному росту вычислительной сложности, а значит часто неприемлемо для графики реального времени. Несмотря на значительное количество работ и существующих методов сглаживания, универсального метода не существует. Различные способы работают лучше для различных ситуаций, что может быть связано с природой трехмерной сцены, вычислительными ресурсами платформы и разрешением дисплея. [2] Также, некоторые методы сглаживания могут быть несовместимы с используемыми техниками отрисовки, например, широко использовавшийся ранее метод MSAA несовместим без особых модификаций с повсеместно используемой в современных игр техникой отложенной отрисовки. [3]

В случае представляющих интерес для данной работы методов отрисовки отражений и отблесков на поверхности воды, сложность

представляют гладкие поверхности с большим количеством высокочастотных геометрических деталей, индуцированных как реальной геометрией модели, так и картами нормалей [4]. В ситуациях, когда эти детали занимают менее одного пикселя на дисплее, их отрисовка может создавать крайне нежелательное мерцание. Поверхность воды в дополнение к большому количеству деталей часто быстро меняется со временем, создавая еще большую сложность при отрисовке. [5]

Работ по отрисовке и сглаживанию отражений достаточно большое количество, например, работы Поди и других [5,6], предложившие алгоритмы временного и пространственного сглаживания отражений конкретно для поверхности воды, или работы Токуйоши и Капланяна [4,7], предложившие способ сглаживания геометрических деталей любых гладких поверхностей, подходы из которого использовались во многих последующих работах в области. Все эти работы будут подробнее рассмотрены в работе.

В последнее время отрисовка нестандартных материалов, таких как блестящие поверхности, многослойные материалы и материалы с анизотропной структурой становится не только предметом научных исследований, но и точкой интереса в игровой индустрии, особенно при разработке игр, которые стремятся к максимальному реализму в изображении тканей, волос и блестящих автомобильных красок. Вследствие этого методы отрисовки и сглаживания блестящих материалов, таких как поверхность океана в солнечную погоду, сильно развивались и совершенствовались.

Отрисовка блестящих материалов – достаточно сложная задача. Даже без ограничений по времени отрисовки одного кадра, накладываемых игровыми движками, однозначного решения этой проблемы не существует. Для оффлайн отрисовки блестящих поверхностей алгоритмы развивались от правдоподобных, но не достоверных физически, например, в работах Джейкоба и других [8] и Яна и других [9], до физически достоверных, но требующих больших затрат памяти на хранение сложных объемных структур, например, в работах Вэнга и других [10,11]. Последняя работа авторов хоть и

близко подходит к тому, чтобы быть использованной в графике реального времени, но, тем не менее, для большинства реальных игр ее эффективность по памяти и вычислительному времени неудовлетворительна. В графике реального времени одной из первых стала работа Зирра и Капланяна [12], предложившая быстрый способ процедурной генерации отблесков, не выполняющий закон сохранения энергии. И, наконец, работа Шермейна и других [13,14] предложили физически достоверный способ процедурной отрисовки отблесков в реальном времени. Однако даже этот способ неидеален, так как он подвержен алиасингу и неточностям в расчетах освещения в пикселе, только частично устраненным в работе [14]. Этот способ недавно был внедрен с некоторыми модификациями в экспериментальную систему материалов Substrate популярного игрового движка Unreal Engine 5.3 [15,16].

В данной работе будут рассмотрены методы отрисовки отражений и отблесков на поверхности воды, определены их достоинства и недостатки, предложены пути устранения некоторых из этих недостатков, реализована улучшенная модификация одного из методов и проанализированы результаты этого улучшения на физическую достоверность, качество отрисовки и вычислительную эффективность.

1 ЛИТЕРАТУРНЫЙ ОБЗОР

1.1 Фильтрация карт нормалей

1.1.1 LEAN Mapping

Один из ключевых способов сглаживания поверхности воды при отрисовке рек, морей, океанов и других больших открытых поверхностей – это сглаживание (или фильтрация) геометрических деталей, задаваемых, обычно, некоторыми текстурами (картами высот, нормалей или смещения [17]). Традиционные способы фильтрации текстур, такие как трилинейная или анизотропная фильтрация, приводят к излишнему сглаживанию деталей на удалении, так как MIP-цепочка усредняет детали поверхности (рисунок 1).

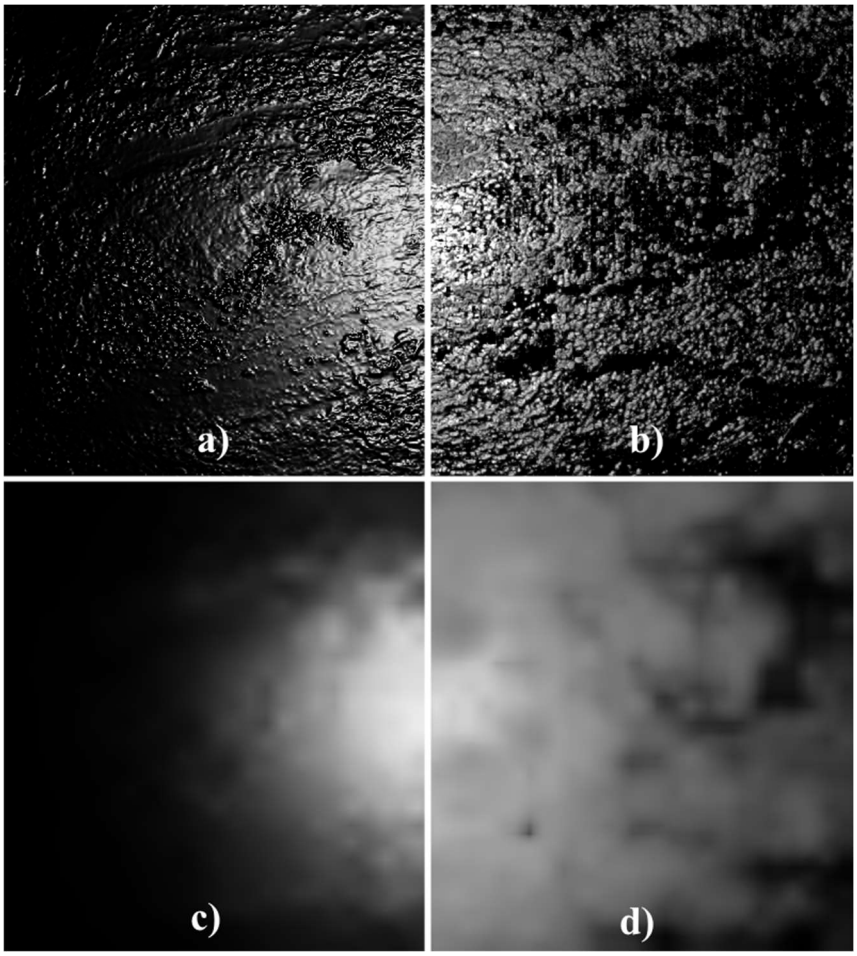

Рисунок 1 – Трилинейная и LEADR фильтрация карт высот (аналогично для карт нормалей): a) трилинейная, mip = 0, b) LEADR, mip = 0, c) трилинейная, mip = 5, d) LEADR, mip = 5 [18]

Марк Олано и Дэн Бейкер предложили в своей статье [19] метод фильтрации карт нормалей Linear Efficient Antialiased Normal (LEAN)

Mapping, сохраняющий детали. Идея этого метода состоит в рассмотрении деталей, индуцированных картой нормалей в общем для всех деталей тангенциальном пространстве полигональной поверхности. При таком рассмотрении распределение нормалей можно хранить в линейно фильтруемом формате (в виде двух текстур, хранящих первый и второй моменты нормалей), совместимом с MIP и анизотропной фильтрациями. Тогда сложение соседних распределений становится тривиальным (рисунок 2), при этом сложение средних дает центр отражения, а сложение вторых моментов позволяет вычислить матрицу ковариации, которая определяет размер и форму отражения.

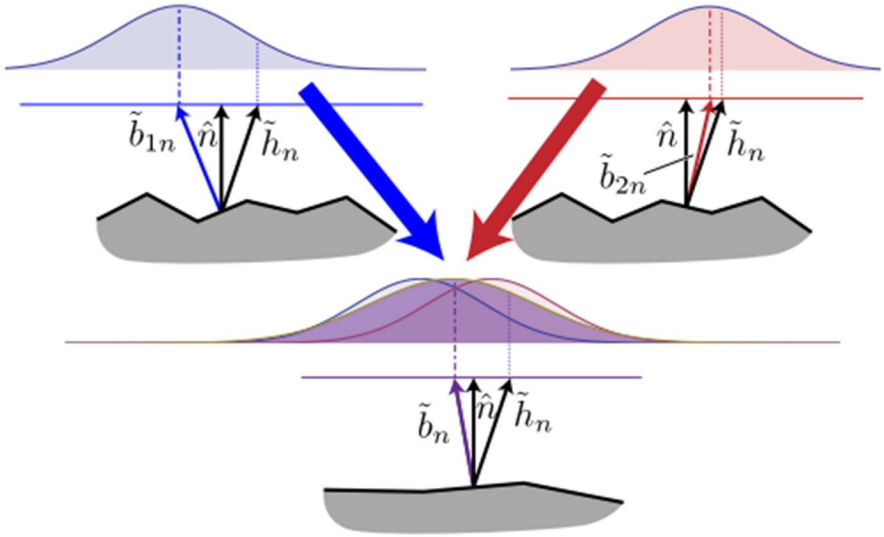

Рисунок 2 – Иллюстрация сложения распределений нормалей соседних выпуклостей, когда они заданы в единой системе координат [19]

В общем тангенциальном пространстве геометрическая нормаль поверхности берется за нулевую, а нормали из карты нормалей представляются двумя проекциями их на геометрическую поверхность. Эти проекции являются первым моментом B. Вторым моментом M являются квадраты этих проекций и произведение одной на другую. Для этих двух текстур генерируются MIP-цепочки и на их основе для каждого уровня детализации восстанавливаются средняя нормаль $\mathrm{b_n}$ и матрица ковариации

$$\Sigma = \begin{bmatrix} M.x - B.x * B.x & M.z - B.x * B.y \\ M.z - B.x * B.y & M.y - B.y * B.y \end{bmatrix}. \quad (1)$$

Полученные величины b_n из текстуры, содержащей первый момент, и Σ из выражения (1) используются для расчета распределения нормалей Бекманна

$$D(h_n) = \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac{1}{2}(h_n - b_n)^T \Sigma^{-1}(h_n - b_n)}. \quad (2)$$

NDF из выражения (2) затем используется в BRDF при расчете освещения.

Результаты работы этого метода представлены на рисунке 3. Здесь можно увидеть, как детали поверхности океана становятся частью микроструктуры поверхности на большом расстоянии и производят меньше зеркальных отражений, а значит не приводят к алиасингу.

Несмотря на простоту и вычислительную эффективность этого метода, он обладает рядом недостатков. Главный недостаток LEAN mapping для использования в современных PBR движках заключается в том, что этот метод фильтрует карту нормалей в предположении что реальная геометрическая поверхность, покрываемая пикселем плоская (то есть имеет не меняющийся или почти не меняющийся вектор нормали). Это накладывает ряд ограничений не только на геометрическую плотность полигонов на модели, но и на техники, которые можно использовать в сочетании с этим методом. В частности, тесселяция с использованием карты смещения и различные геометрические анимации несовместимы с LEAN mapping. Также в этом методе не учитывается вес проекции микрограней на пиксель при наблюдении под углом. Следующий рассмотренный способ фильтрации нормалей исправляет эти недостатки.

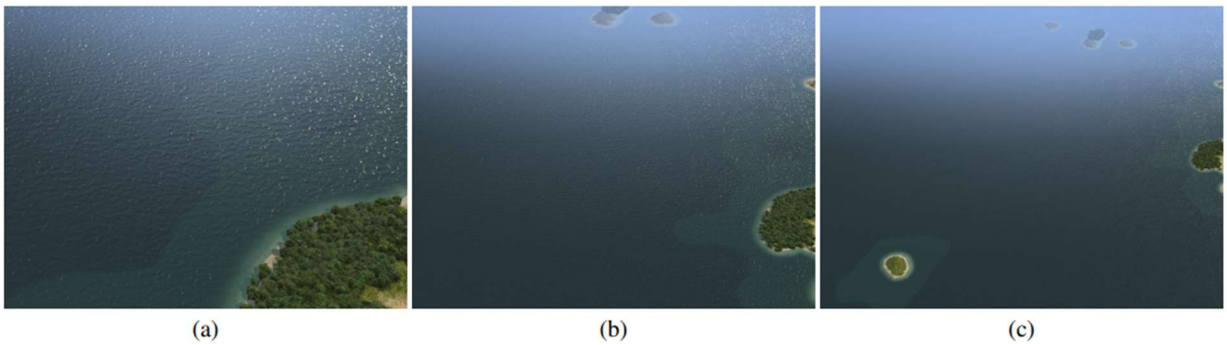

Рисунок 3 – Поверхность океана с использованием LEAN Mapping [19] на близком (a), среднем (b) и далеком (c) расстояниях от камеры

1.1.2 LEADR Mapping

Джонатан Дюпуй и другие в своей статье [20] предложили улучшенную версию метода LEAN Mapping из [19], Linear Efficient Antialiased Displacement and Reflectance (LEADR) Mapping. Этот метод использует такую же схему хранения и линейной фильтрации карт нормалей, как и LEAN Mapping, но использует физически обоснованную BRDF, включающую в себя функцию затемнения и экранирования Смита [21] для нецентрированных распределений Бекманна, которые получаются при наблюдении гауссовой поверхности под углом вследствие экранирования, затенения и изменения проекции площадей микрограней (рисунок 4).

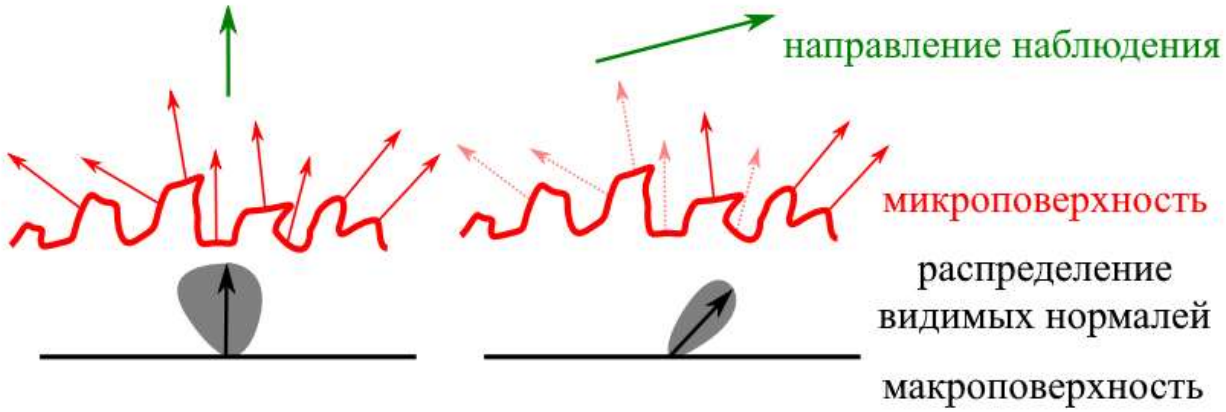

Рисунок 4 – Зависимость распределения видимых нормалей от угла наблюдения из-за экранирования и изменения проекций площадей микрограней [20]

Кроме того, этот метод лучше поддерживает анимации и деформации поверхности (рисунок 4), а также поддерживает освещение картой окружения.

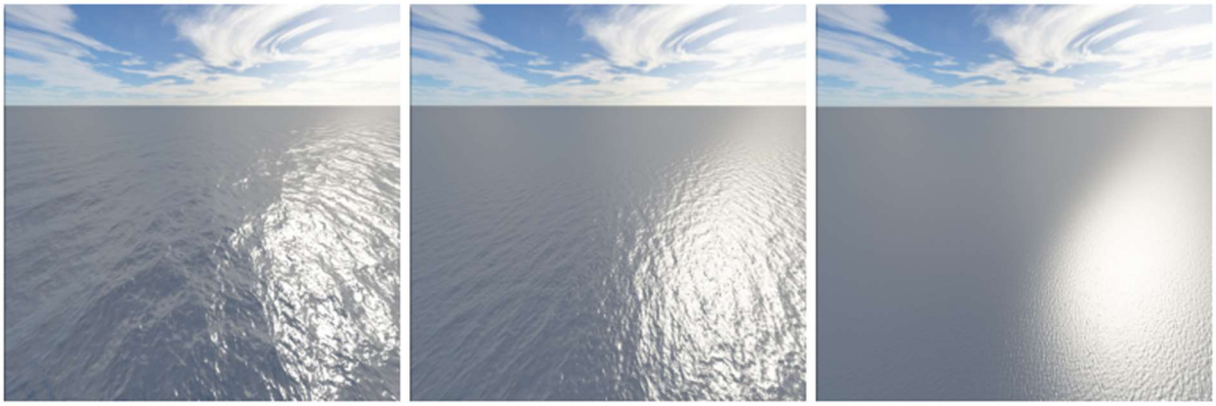

Рисунок 5 – Анимированная поверхность океана, отрисованная с помощью LEADR Mapping в процессе отлета камеры назад, далекие детали без видимых скачков и алиасинга переходят в шероховатость поверхности [20]

Чтобы реализовать освещение от карты окружения, для каждого выбранной точки на склоне рассчитывается вектор нормали и вектор приходящего света, который может образовать отражение в кадре (рисунок 6). После этого карта окружения фильтруется – для каждого точки генерируется свой уровень деталей в зависимости от телесного угла, формируемого отклонениями входящего направления света.

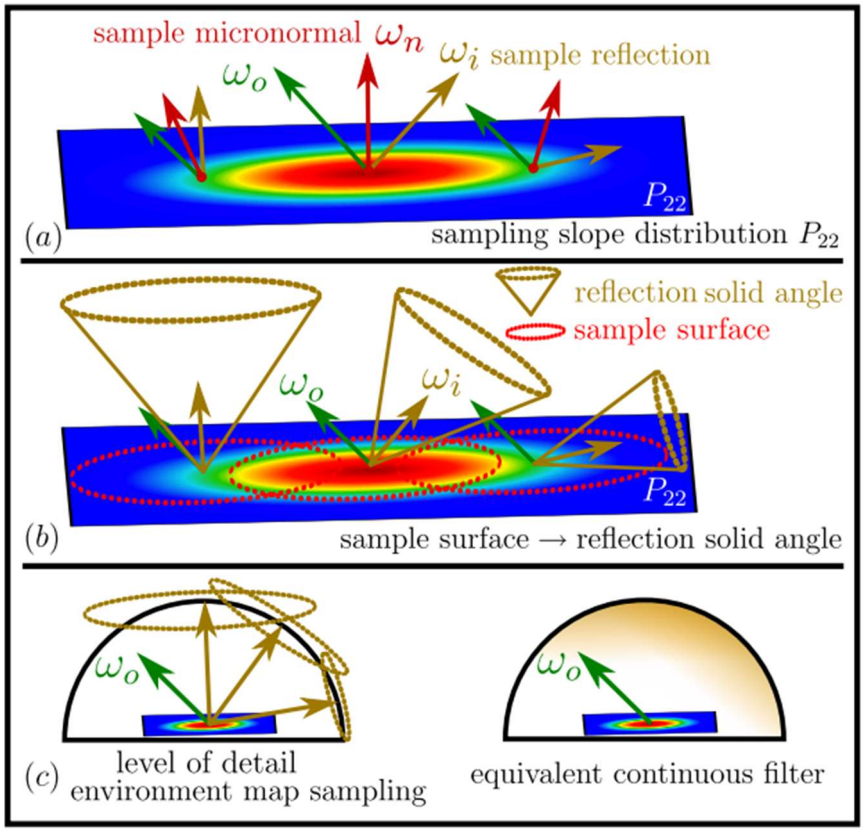

Рисунок 6 – Схема взятия выборки из карты окружения
для расчета освещения [20]
(a) – расчет вектора отражения для каждой микронормали
(b) – область, покрываемая выборкой в системе координат наклона,
конвертируется в телесный угол умножением на якобиан перехода
(c) – для каждого телесного угла берется свой уровень деталей карты

LEADR Mapping предлагает алгоритм, превосходящий LEAN Mapping по точности и физической достоверности, при этом сохраняющей такую же эффективность по памяти и практически такую же эффективность по времени вычисления [20]. Поэтому сейчас использовать LEAN mapping вместо него практически всегда нерационально.

1.1.3 Улучшенная сохраняющая детали фильтрация карт смещения

Лифан Ву и другие предложили в своей статье [22] новый способ фильтрации карт смещения и BRDF, который с большой точностью рассчитывает изменения в отражении, тенях и экранировании при изменении

уровня деталей. В основе метода лежит оптимизированное уменьшение разрешения карты смещения, сохраняющее нормали, и разложение шестимерной BRDF на двумерную функцию от координат на поверхности и четырехмерную функцию направления.

Их метод дает интересные результаты (рисунок 7) и в отличие от LEADR Mapping, корректно обрабатывает поверхности, распределение нормалей в которых отличается от распределения Бекманна (рисунок 8), однако их метод достаточно вычислительно сложен и не применим без изменений для графики реального времени. Однако, модель авторов достаточно эффективна по памяти, поэтому с некоторыми оптимизациями и упрощениями этот метод может стать возможным и в графике реального времени.

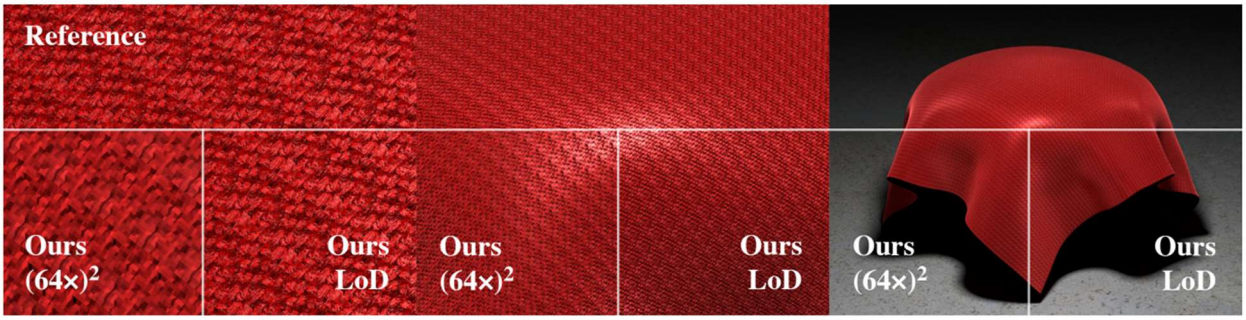

Рисунок 7 – Метод фильтрации [22] карты смещения с негауссовым распределением нормалей на разных уровнях деталей [22]

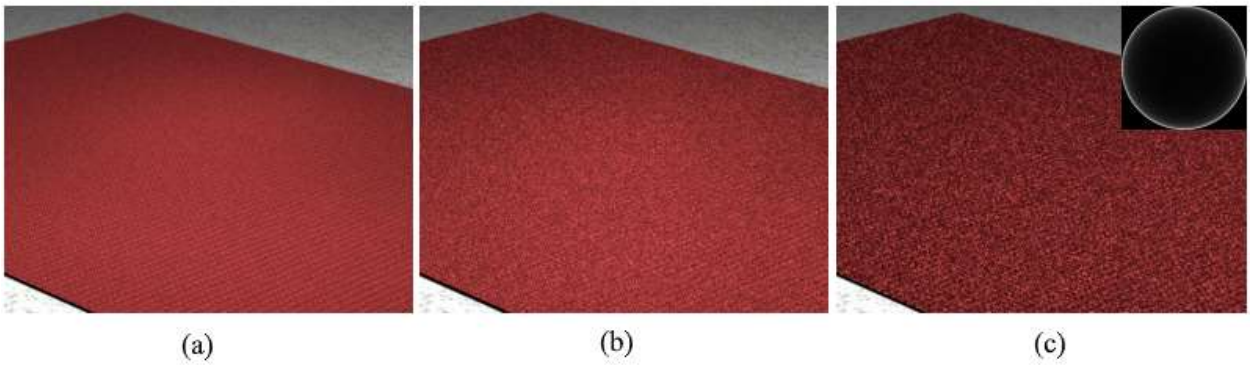

Рисунок 8 – Сравнение методов фильтрации [22] и [20], когда нормали в карте смещения не распределены по Гауссу:
(a) – образец, размер выборки – 1000 точек на пиксель
(b) – метод [22], размер выборки – 35 точек на пиксель
(c) – метод [20], 70 точек на пиксель, видимые артефакты

1.2 Геометрическое сглаживание зеркальных отражений

1.2.1 Зеркальные отражения на поверхности океана

Намо Поди и другие [6] предложили в своей работе метод временного и пространственного сглаживания ярких отражений на быстро меняющихся волнах океана. Их техника временного сглаживания заключается в вычислении пересечения диска источника света и плоскости, задаваемой двумя векторами отражения каждого пикселя последовательных кадров (рисунок 9).

Рисунок 9 – Схема метода временного сглаживания Поди [6]

Для реализации пространственного сглаживания авторы разделяют волны в каждом пикселе на синусоиды большие и меньшие по частоте частоты Найквиста. После чего для волн меньших этой частоты они используют идею из метода фильтрации карт нормалей Олано и Бейкера [19] для эффективного по времени расчета средней интенсивности отраженного света в текущем пикселе с учетом NDF.

Итоговый метод уменьшает временной и пространственный алиасинг для волн в относительной близости от камеры (рисунок 10), однако работает значительно хуже для дальних волн вследствие неточности аппроксимации эллиптической функции Гаусса в функции распределения отражений круговой функцией Гаусса. Этот недостаток авторы предлагают исправить заменой

данной аппроксимации на аппроксимацию несколькими круговыми функциями Гаусса.

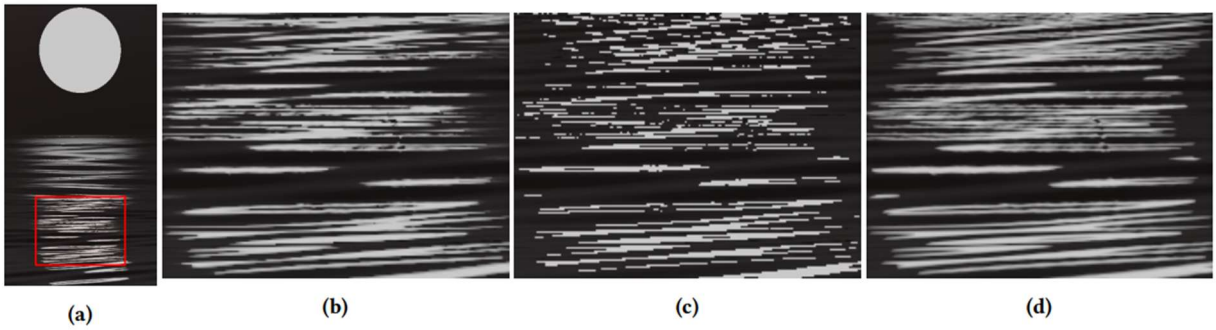

Рисунок 10 – Результат применения метода сглаживания Поди и др. [6]:
(a) – вся сцена, отрисованная с использованием метода
(b) – увеличенные ближние волны, отрисованные с помощью метода
(c) – увеличенные ближние волны, отрисованные без сглаживания
(d) – образец, увеличенные ближние волны, отрисованные в разрешении в 64 раза больше используемого и пониженные до исходного

1.2.2 Геометрическое сглаживания отражений

Токуйоши и Капланян предложили в своей статье [7] улучшение метода сглаживания, предложенного Капланяном и другими в статье [4]. Этот метод позволяет улучшить стабильность отрисовки бликов на гладких поверхностях в случае, когда детали поверхности занимают менее одного дисплейного пикселя. Идея метода заключается в рассмотрении региона, занимающего пиксель экрана и его перевода медианных векторов в нем в так называемую систему координат склона (двумерную систему координат, задающую вектор координатами его точки пересечения с плоскостью, перпендикулярной нормальному вектору и проходящей через его конец [23]). После этого важный для данной ситуации компонент BRDF – NDF фильтруется в этой системе. Для этого интеграл NDF по пикселю заменяется на интеграл по области в системе координат склона (это допустимо в случае, когда справедливо предположение о том, что кривизна поверхности в каждом фрагменте достаточно мала). Поскольку при этом значение NDF перестает зависеть от выбора точки при растеризации, интеграл по площади превращается в площадь фрагмента, которая уже неявно вычисляется при растеризации. Далее задача сводится к

эффективному вычислению отфильтрованного значения конкретной NDF, и авторы приводят способы этого вычисления для функций Бекманна и GGX. Результаты работы метода для функции GGX приведены на рисунке 11.

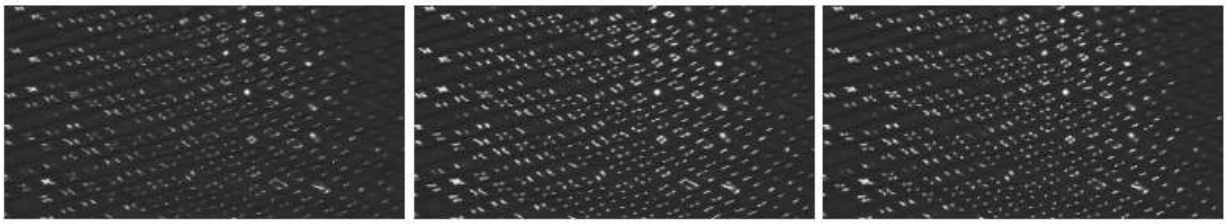

Рисунок 11 – Иллюстрация работы метода сглаживания отражений Капланяна [4]:
(a) – поверхность воды, отрисованная с помощью растеризации с выборкой в 1 точку на пиксель без использования метода
(b) – поверхность воды, отрисованная с помощью растеризации с выборкой в 1 точку на пиксель с использованием метода
(c) – образец, поверхность воды, отрисованная с использованием трассировки лучей

Недостаток этого метода заключается в возникающих артефактах при больших углах наблюдения и вблизи краев моделей из-за ошибок в аппроксимации производных при вычислении якобиана перевода системы координат. Производные для сокращения количества расчетов берутся из встроенных функций в пиксельном шейдере, которые позволяют получить различия произвольных переменных в одном треугольнике внутри квадрата 2 × 2 пикселя (например, ddx/ddy в HLSL [24]). Область интегрирования получается из изменения медианного вектора в этом квадрате. Для уменьшения ошибки аппроксимации авторы ограничивают размеры получившейся области, в таком случае вероятность ложных отражений при наблюдении под большими углами уменьшается [25]. В статье [7] авторы приводят улучшенную версию метода, в которой они не только уменьшают количество артефактов, но и увеличивают эффективность алгоритма. Для исключения ложных отражений при наблюдении под большими углами, они вычисляют производные не в системе координат склона, а в системе координат проекций медианного вектора на плоскость, перпендикулярную

нормали. В таком случае якобиан получается меньшим, чем в изначальном методе, что исключает отражения под большими углами (рисунок 12).

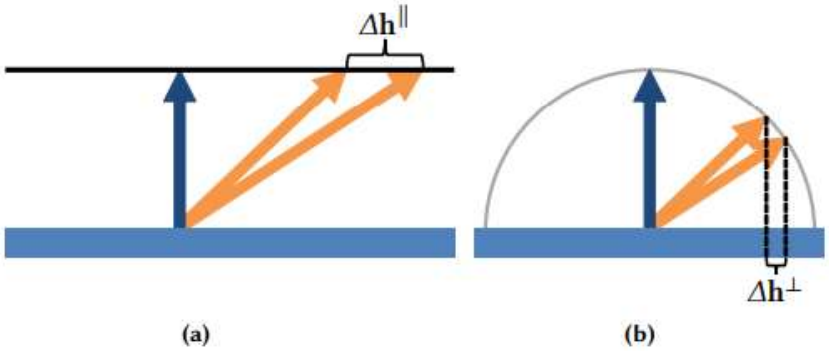

Рисунок 12 – Сравнение вычисления изменения медианного вектора в методе из [4] (a) и [7] (b), можно увидеть, что в (b) изменения при больших углах будут меньше, а значит ошибки будут приводить к меньшему количеству ложных отражений

На рисунке 13 представлены результаты улучшений метода в статье [7]. В статье также приводится версия алгоритма для отложенной отрисовки, когда у шейдера нет прямого доступа к производным медианных векторов. Эта версия использует средние нормали в квадрате пикселей для оценки производных и иногда может быть более применимой для реальных движков даже при использовании прямой отрисовки, поскольку ее производительность не зависит от количества и типа источников света.

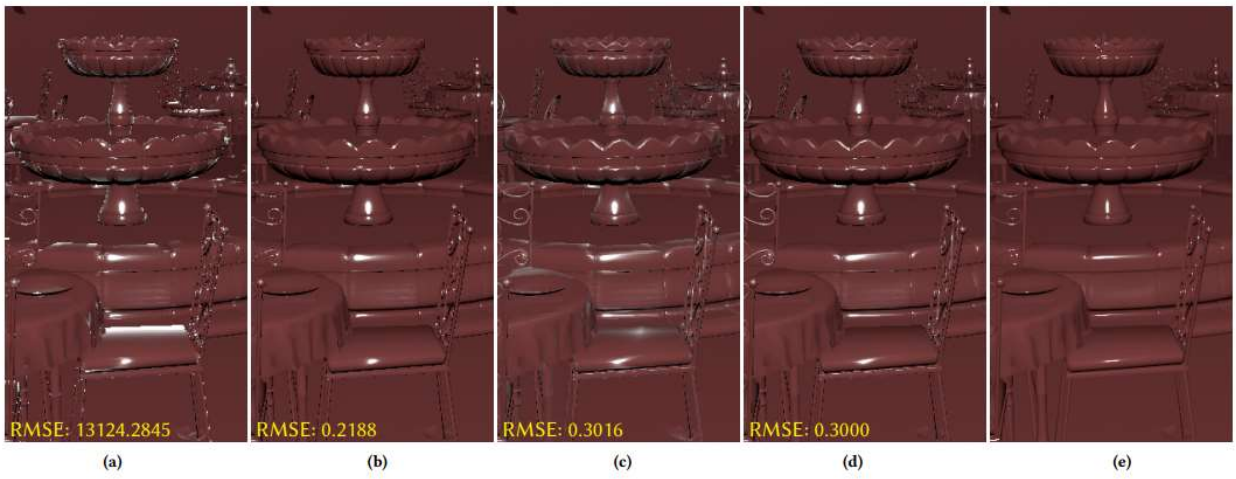

Рисунок 13 – Иллюстрация улучшений, предложенных в статье [7] к методу из [4]:
(a) – сцена, отрисованная с использованием метода из [4] без ограничения размера области интегрирования
(b) – сцена, отрисованная с использованием метода из [7] без ограничения размера области интегрирования
(c) – сцена, отрисованная с использованием метода из [4] с ограничением размера области интегрирования
(d) – сцена, отрисованная с использованием метода из [7] с ограничением размера области интегрирования
(e) – образец, сцена, отрисованная с использованием выборки в 1024 точки на пиксель

Главным недостатком этого метода является излишняя фильтрация и размытие отражений по сравнению с образцом (рисунок 13), однако, как и во всем методах сглаживания, небольшое размытие более приемлемо для человеческого глаза чем алиасинг [26].

1.3 Отрисовка и геометрическое сглаживание отблесков

1.3.1 Отрисовка отблесков в реальном времени

Океан в солнечную погоду можно описать как сверкающую поверхность [14]. Сверкающие поверхности сложно отрисовать, особенно в реальном времени, поскольку они содержат огромное количество высокочастотных деталей, производящих многочисленные резкие блики на свету. Самый распространенный способ представления сверкающих деталей – карты нормалей высокого разрешения [27], но этот подход не эффективен по памяти

и плохо подходит для создания вариаций. Сложность заключается не только в создании модели представления отблесков, эффективной по памяти и времени вычислений, но и в фильтрации этой модели для избегания алиасинга, проявляющегося в пространственно и временно нестабильной отрисовке отблесков. В последние годы было предложено несколько моделей для эффективного представления отблесков. Тобиас Зирр и Антон Капланян в своей статье [12] предложили один из первых алгоритмов, пригодных для отрисовки отблесков в реальном времени. Они использовали процедурную генерацию отрисовки отблесков, не включающую хранение отражающих частиц в четырехмерной структуре, как это использовалось в предшествующих методах, а для фильтрации отблесков в фрагменте использовали стандартную MIP-иерархию. Их метод позволяет отрисовывать блестящие поверхности (рисунок 14), однако большой недостаток его в том, что их BRDF не является физически обоснованной, поскольку не выполняет закон сохранения энергии. Это происходит из-за того NDF фрагмента поверхности использует аппроксимацию биномиального распределения для оценки доли отражающих микрограней. Из-за этого факта их BRDF не сходится к BRDF Кука-Торренса [28] на больших расстояниях, как того требует физика.

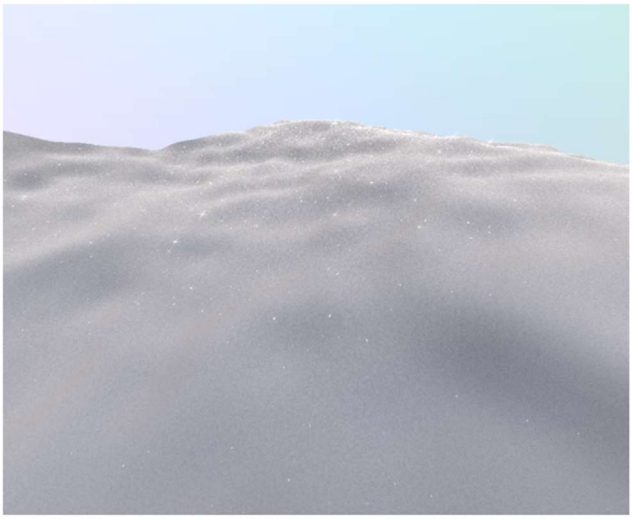

Рисунок 14 – Блестящая снежная поверхность, отрисованная методом [12].

Другой процедурный метод был предложен Вэнгом и другими [11]. В нем авторы предлагают BRDF, основанную на BRDF из работы [10], совместимую с освещением фильтрованной картой окружения. Эта модель не такая быстрая, как [12], поскольку использует схожие четырехмерные структуры для представления отблесков, как и алгоритмы для отрисовки не в реальном времени. Тем не менее она применима для реального времени благодаря разделению алгоритма на три уровня точности аппроксимации в зависимости от размера деталей, производящих отблески, а также модификации модели микрограней для совместимости ее с фильтрацией. Модель является физически достоверной и лучше воспроизводит образец, отрисованный не в реальном времени, чем алгоритм [12] (рисунок 15). Главный недостаток предложенного авторами алгоритма в большом требовании к памяти из-за необходимости хранения четырехмерных структур.

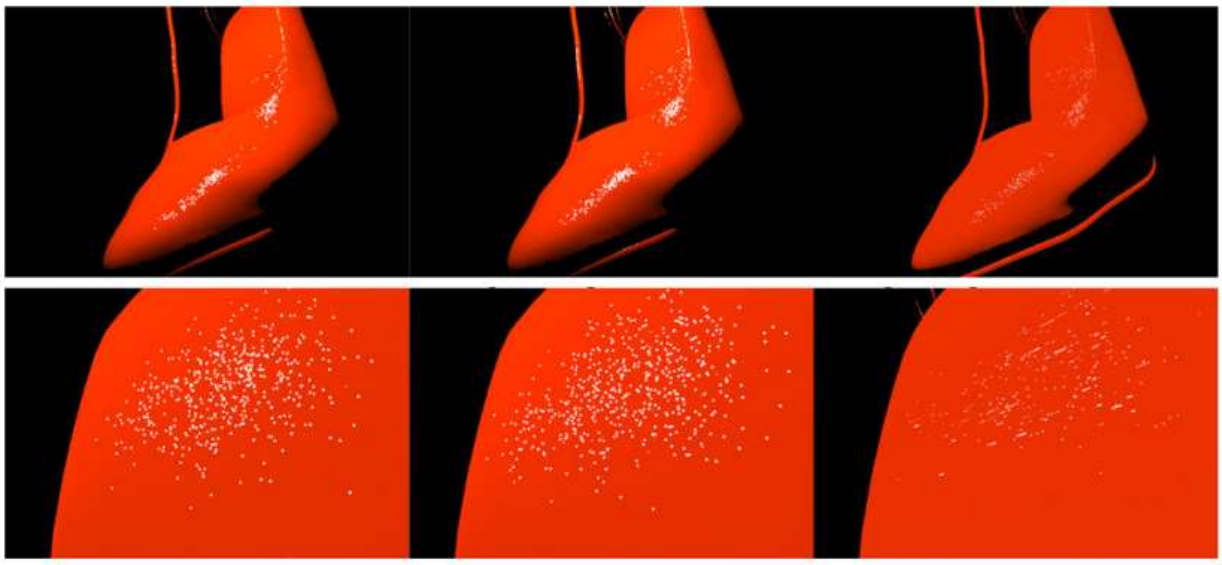

Рисунок 15 – Сравнение алгоритмов [11] (a), образца, отрисованного не в реальном времени (b) и [12] (c) [11]

Очень перспективный алгоритм был предложен Шермейном и другими [13]. В этой работе предложена соблюдающая закон сохранения энергии BRDF с процедурной генерацией отблесков, которая использует одномерные таблицы для генерации SDF. Физическая достоверность их BRDF гарантирует,

что при большой плотности микрограней, она сходится к стандартной гладкой BRDF Кука-Торренса [28] (рисунок 16). Это выгодно отличает алгоритм от предшествующих и позволяет без видимых переходов и артефактов отрисовывать блестящие материалы на любом уровне деталей, а также позволяет дизайнерам не производить правки гаммы и других параметров на этапе после отрисовки сцены.

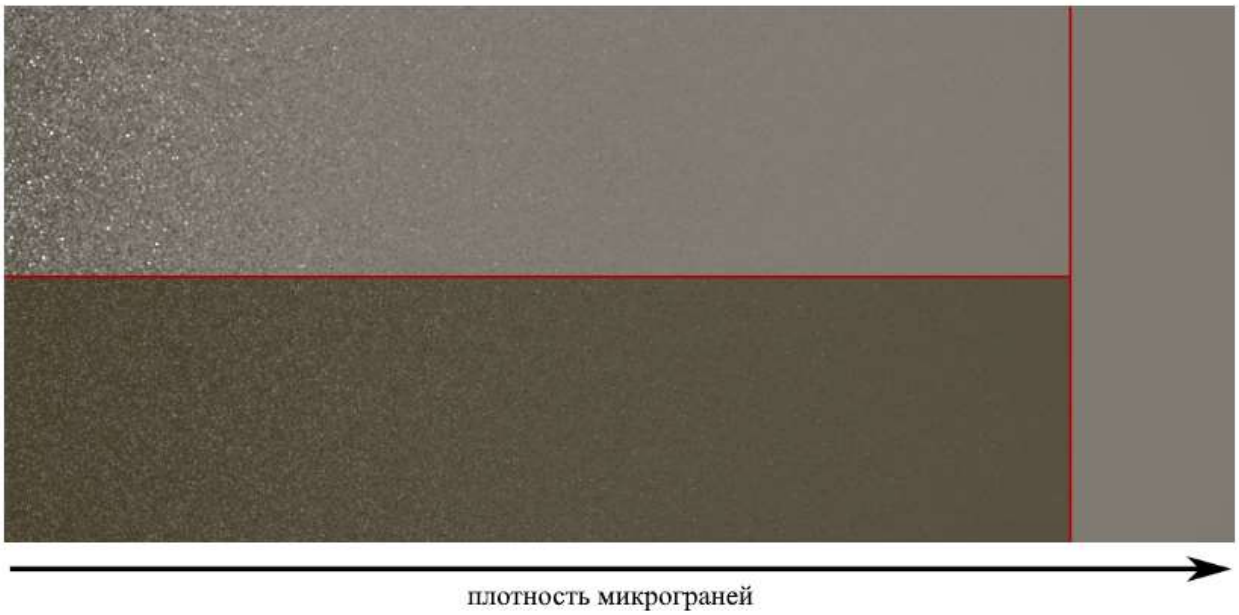

Рисунок 16 – Сравнение BRDF [13] (сверху) и [12] (снизу) при увеличении плотности микрограней, справа приведена BRDF Кука-Торренса [28] [13]

BRDF в работе использует NDF для фрагмента поверхности, задающуюся как взвешенная сумма высокочастотных NDF, хранимых в MIP-иерархии. На высоких уровнях деталей NDF задают всего несколько дискретных частиц, характеризующихся резкими пиками в распределении. С уменьшением уровня деталей частиц становится все больше, пока все распределение не сходится к стандартному распределению Бекманна на последнем уровне. NDF в иерархии генерируются прямо во время отрисовки перемножением двух случайных распределений одномерных распределений, заданных заранее (рисунок 17). После перемножения полученные NDF поворачиваются на случайный угол для каждого уровня MIP-иерархии, что позволяет избежать артефактов повторения при отрисовке, а затем

растягиваются, что позволяет использовать эти распределения для материалов разной шероховатости. Правильное нормирование этих NDF гарантирует выполнение закона сохранения энергии.

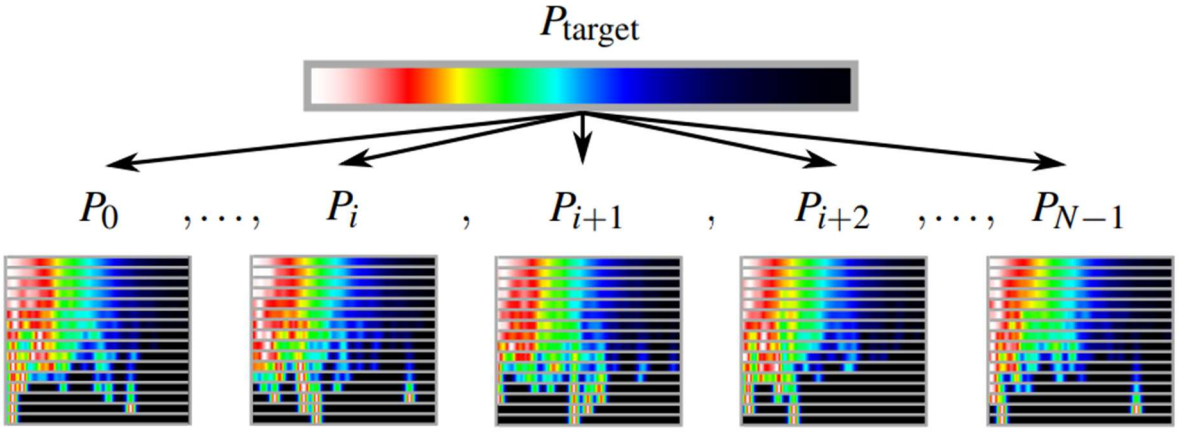

Рисунок 17 – Заданные заранее MIP-иерархии одномерных распределений нормалей, сходящихся к одному и тому же распределению Бекманна [13]

Этот метод производительный, эффективный по памяти и физически достоверный, однако обладает рядом недостатков. В отличие от описанного выше метода [11], этот метод не поддерживает освещение картой окружения. Далее, в качестве функции затемнения и экранирования BRDF авторов использует функцию затемнения V-бороздок и использование более совершенной и часто используемой функции затенения Смита нарушает физическую достоверность модели. Также несмотря на то, что вместо распределения Бекманна, можно использовать некоторые другие распределения, часто используемое в движках распределение GGX не поддерживается. Наконец, анизотропные SDF, которые позволяет задавать модель могут быть только параллельными осям, а также несмотря на то, что NDF получаются фильтрацией по фрагменту поверхности, что должно минимизировать геометрический алиасинг, он может проявляться на не плоских поверхностях, где кривизна, например, индуцирована картой нормалей. Последние два недостатка были рассмотрены авторами следующей работы, которая будет рассмотрена далее.

1.3.2 Геометрическое сглаживания отблесков с фильтрацией карты нормалей

Шермейн и другие [14] предложили в своей работе метод сглаживания отблесков, отрисованных с помощью BRDF из [13]. Когда при отрисовке с помощью этой BRDF поверхность с отблесками имеет кривизну (индуцированную, к примеру, картой нормалей), появляется эффект алиасинга отблесков, который проявляется в пропадании некоторых отблесков на некоторых кадрах. Этот эффект связан с тем, что на выпуклой поверхности вектора наблюдения и падения света могут сильно меняться даже в пределах одного пикселя. При этом на плоской поверхности алиасинг не проявляется, поскольку BRDF из [13] получается фильтрацией каждого фрагмента.

Идея метода сглаживания схожа с методом сглаживания отражений, описанным выше. Интегрирование BRDF так же, как в [13], осуществляется в системе координат склона. Предполагается что единственный не постоянный компонент BRDF в пределах фрагмента – это NDF, полученная из SDF. При использовании нормированного ядра свертки отфильтрованная SDF (и NDF) также будут нормированными, а значит BRDF останется физически обоснованной. Результаты сглаживания можно видеть на рисунке 18.

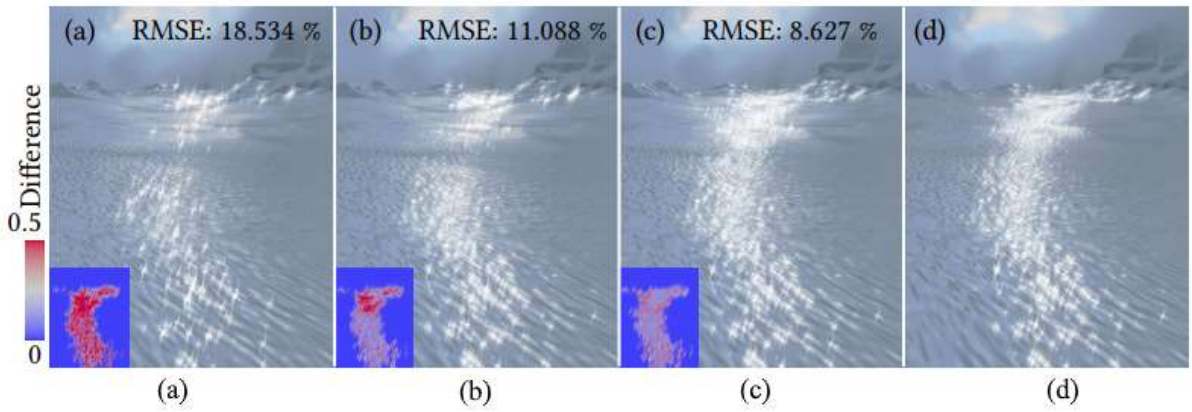

Рисунок 18 – Результат работы метода сглаживания отблесков [14]:
(a) – сцена, отрисованная с использованием метода из [13] без сглаживания
(b) – сцена, отрисованная с использованием метода из [13] со сглаживанием из [14]
(c) – сцена, отрисованная с использованием метода из [13] со сглаживанием из [14] и фильтрацией карты нормалей из [19]

(d) – образец, сцена, отрисованная с использованием выборки размером в 1024 точек на пиксель

Даже в после сглаживания отблески вдали от камеры начинают пропадать по сравнению с образцом, поэтому авторы используют LEAN Mapping [19] для фильтрации карты нормалей поверхности, что возможно благодаря тому, что SDF в [13] сходится к гауссову распределению с уменьшением уровня деталей. Для того чтобы сделать BRDF совместимой с LEAN Mapping авторы добавляют в нее параметр корреляции уклонов, тогда как в [13] можно было изменять лишь стандартное отклонение SDF в x и y направлениях. Без этого параметра LEAN Mapping некорректно фильтрует анизотропные детали карт нормалей (рисунок 19).

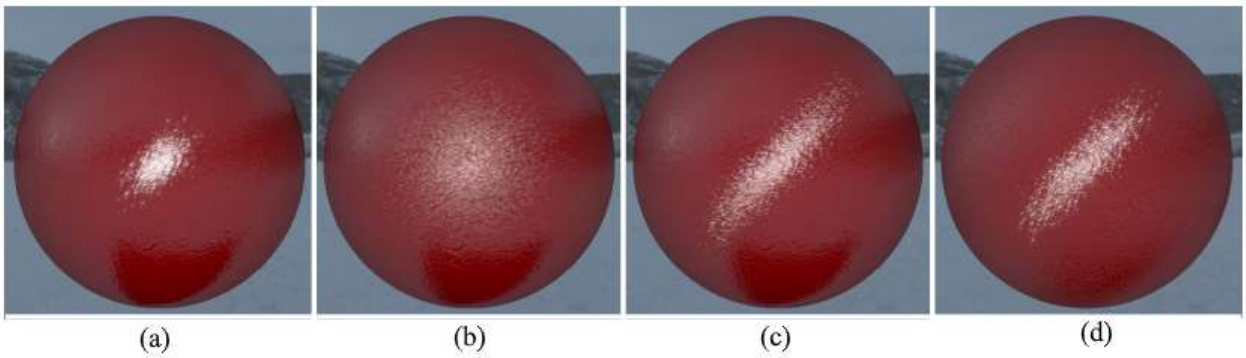

Рисунок 19 – Значение фактора корреляции в LEAN Mapping при фильтрации поверхности с анизотропными деталями [14]
(a) – без LEAN Mapping, отсутствует часть отблесков вдали от центра блика
(b) – LEAN Mapping без фактора корреляции (или фактором корреляции равным 0), анизотропия потеряна
(c) – LEAN Mapping с ненулевым фактором корреляции, анизотропия сохранена
(d) – образец, отрисованный с выборкой в 1024 точки на пиксель

Одним из главных недостатков рассмотренного метода является использование метода LEAN Mapping, а не более совершенного LEADR Mapping, что связано с тем, что BRDF из [13] не работает с функцией затенения и экранирования Смита [21]. Вместо этого для затенения и экранирования она использует модель V-бороздок из работы Роберта Кука и Кеннета Торренса [28], которую в свою очередь нельзя использовать в LEADR

Mapping, поскольку функции затенения и экранирования V-бороздок для нецентрированных распределений не было получено.

В данной работе было решено устранить этот недостаток, путем получения функции затенения и экранирования V-бороздок для нецентрированных распределений и применения более точных весов проекции при расчете освещения при помощи метода LEADR Mapping.

1.3.3 Обзор имплементации алгоритма отрисовки отблесков в Unreal Engine 5.3

Недавно в популярный движок Unreal Engine разработчики движка добавили возможность отрисовывать блестящие материалы. Алгоритм отрисовки отблесков в Unreal Engine 5.3 основан на статье Шермейна и других [14]. Возможность отрисовки отблесков является частью новой концепции материалов в движке, называемой Substrate [29], пример отрисовки отблесков в движке приведен на рисунке 20. Единственный параметр, предоставленный разработчикам материалов (помимо параметризации текстурных координат) – плотность отблесков. На его основе этого параметра определяется также и размер отблеска.

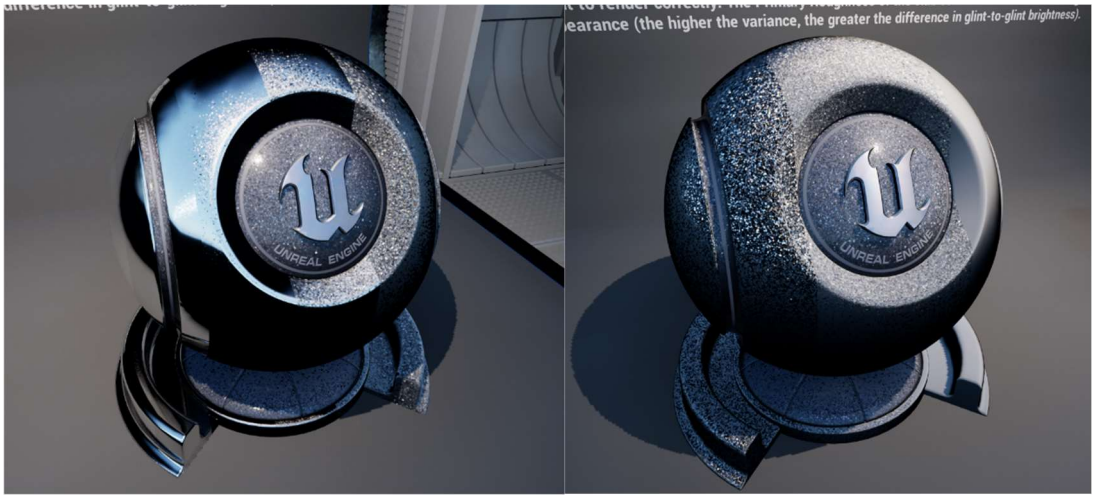

Рисунок 20. Примеры материалов, использующих блестящую BRDF в Unreal Engine 5.3

В исходной статье для упрощения используется множитель Френеля равный единице, что делает результат отрисовки не совсем физически

достоверным, но для данной модели основное значение имеет функция распределения нормалей, а в множителе Френеля нет никаких особенностей. Однако при имплементации в реальный движок это уже имеет большее значение и отсутствие этого множителя в имплементации в UE5 уменьшает реализм (особенно поверхностей, обладающих кривизной).

В коде имплементации UE5 присутствует множитель затенения и экранирования, однако он учитывается только если в функцию передается специальный аргумент. Без модификации исходного кода движка этот аргумент нельзя изменить и поэтому при отрисовке отблесков множитель затенения и экранирования также принимается равным единице.

Наконец, фильтрация карты нормалей из работы также не используется, так как Unreal Engine использует отложенную отрисовку и нормали к моменту отрисовки отблесков уже обработаны и помещены в GBuffer. Фильтрация нормалей в Unreal Engine происходит с помощью стандартных анизотропной или трилинейной фильтрации без специальных манипуляций с MIP-цепочкой, что приводит к потере деталей и излишнему сглаживанию объектов на расстоянии [19,20].

Подводя итог, имплементация отрисовки отблесков в Unreal Engine 5.3 не является физически достоверной и не выполняет закон сохранения энергии, однако просто наличие способа отрисовки таких материалов в движке – это большой шаг и это делает проблему отрисовки отблесков еще более актуальной.

2 ПОСТРОЕНИЕ ТЕОРЕТИЧЕСКОЙ МОДЕЛИ

2.1 Профили микроповерхностей и функции затенения и экранирования

Физически достоверная BRDF состоит из трех множителей, учитывающих разные аспекты рассеивания и отражения света, падающего на поверхность с направления ω_i и наблюдаемого с направления ω_o (рисунок 21). Эти множители – это функция распределения нормалей NDF, множитель Френеля и обычно объединяемые в одну функции экранирования и затенения [30,31].

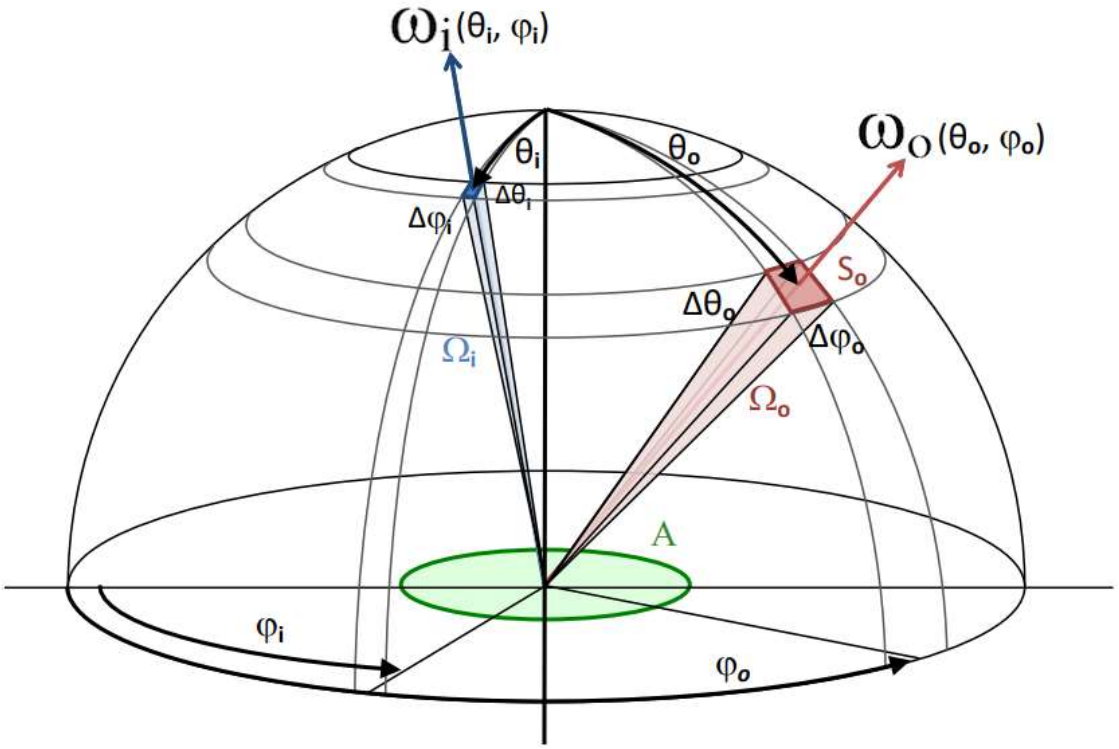

Рисунок 21 – Система координат и углы используемые для описания модели отражения и рассеивания при построении BRDF [30]

Функция экранирования – важный компонент физически достоверной BRDF, нормализующий функцию и обеспечивающий выполнение закона сохранения энергии [32]. Геометрический смысл такой функции в том, чтобы статистически определить какая часть микрограней с данной нормалью перекрыта от наблюдателя другими гранями и выполнить условие сохранение

видимой проекции площади фрагмента [33,34]. Чаще всего конкретный вид функции выводится исходя из этого предположения.

Однако вид функции также зависит от профиля микроповерхности, т. е. набора предположений о расположении и организации микрограней на поверхности. Без этих предположений количество функций, удовлетворяющих условиям нормировки бесконечно большое [34].

В ограничениях графики реального времени BRDF обычно моделирует только одно отражение света. Поэтому в дополнение к экранированию вводится также множитель затенения, поскольку луч, не дошедший до микрограни из-за перекрытия другой необходимо отбросить. Наличие этого множителя позволяет избежать создания лишней энергии, однако означает что даже физически достоверные BRDF $\rho(\omega_o, \omega_i)$ не подчиняются строгому условию нормировки (3) (так называемому White Furnace Test [34]):

$$\int_{\Omega} \rho(\omega_o, \omega_i) \left| \omega_g \cdot \omega_i \right| d\omega_i \neq 1 . \quad (3)$$

Поиск модели BRDF, которая способна моделировать несколько отражений в реальном времени является важным объектом многих исследований [35–37], но в данный момент в графике реального времени подобные модели практически не используются.

Две самые часто используемые модели и функции затенения и экранирования для них – модель V-бороздок [28] и модель Смита [21]. Рассмотрим их подробнее, прежде чем перейти к постановке задачи данной работы.

2.1.1 Функция затенения и экранирования V-бороздок

Модель V-бороздок из работы Кука и Торренса [28] является одной из самых простых моделей для решения задачи затенения и экранирования. В ней предполагается, что любую микроповерхность можно представить в виде взвешенного среднего примитивных микроповерхностей, в которых микрограни на поверхности организованы в так называемые V-бороздки

(стороны бороздок имеют симметричные относительно общей нормали поверхности нормали ω_m и ω'_m) (рисунок 22).

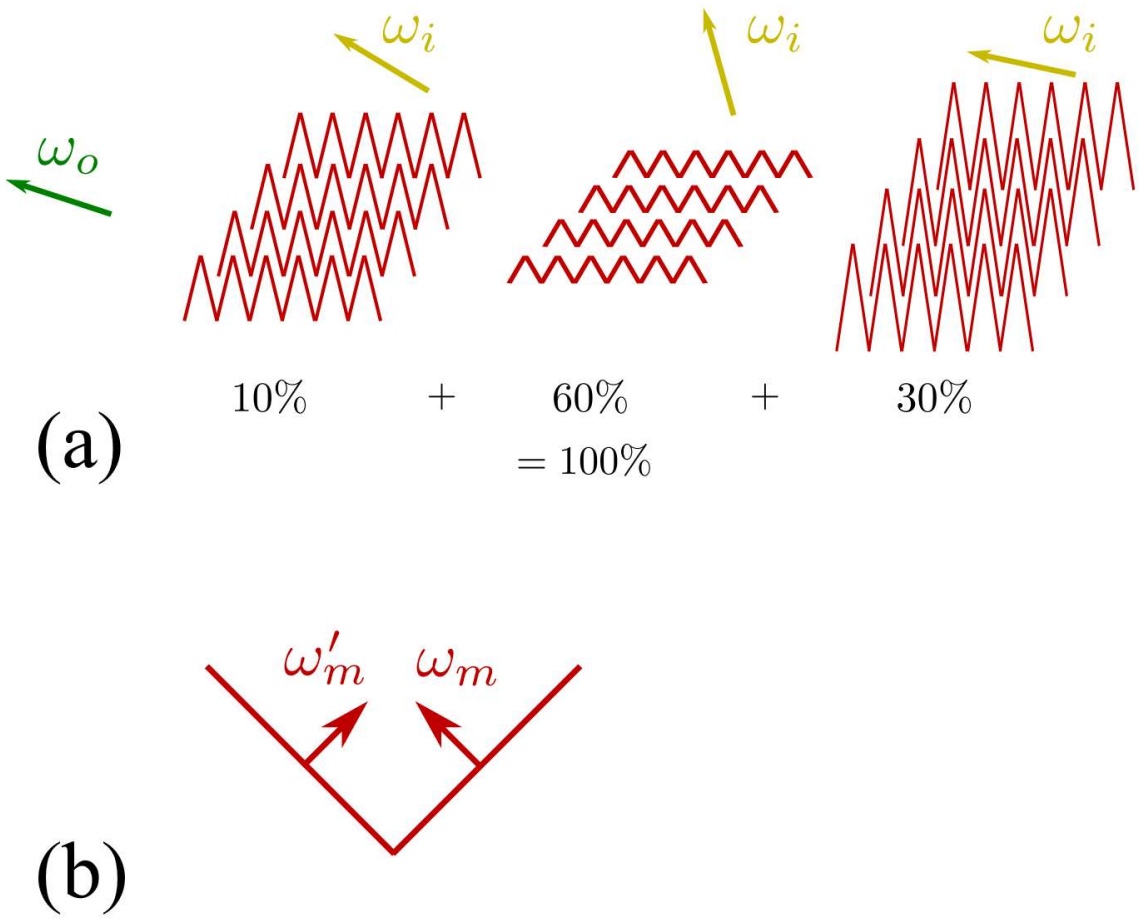

Рисунок 22 – (a) – модель V-бороздок моделирует рассеивание на различных микроповерхностях и смешивает результаты,
(b) – каждая V-бороздка имеет две симметричные нормали ω_m и ω'_m [34]

В этом случае для отдельного компонента поверхности и для выбранного направления возможны две ситуации: либо в этом направлении видны обе нормали и экранирования не происходит, либо видна только одна, причем микрограни с этой нормалью частично перекрыты (рисунок 23)

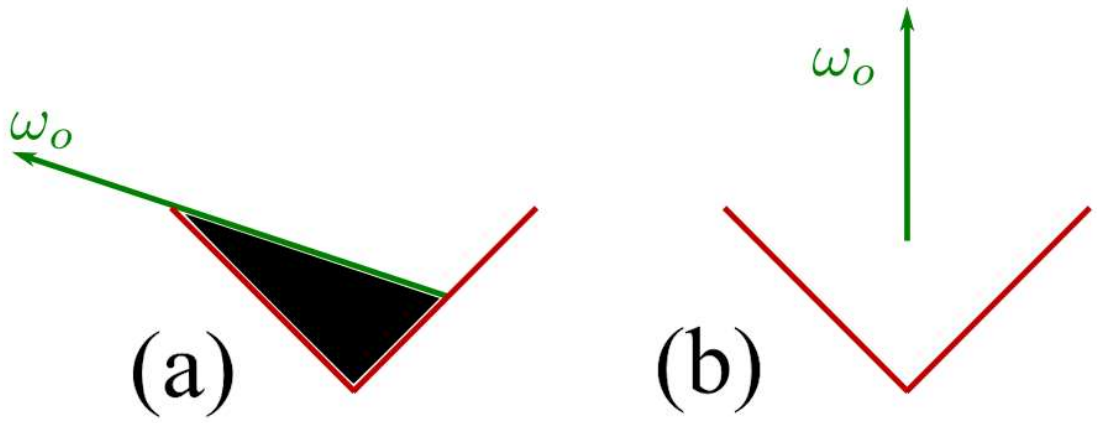

Рисунок 23 – (a) – ситуация, когда видна только одна из сторон бороздки, при этом она частично экранирована,
(b) – ситуация, когда видны обе стороны бороздки, экранирования не происходит [34]

Точный вид функции экранирования нормали ω_m для данной модели получен в работе [28]

$$G_1(\omega_o, \omega_m) = \min\left(1{,}2\frac{(\omega_m \cdot \omega_g)(\omega_o \cdot \omega_g)}{\langle \omega_o, \omega_m \rangle}\right). \quad (4)$$

Оператор $\min(1, -)$ позволяет учесть случай, когда обе нормали видны, и, соответственно, экранирования не происходит.

2.1.2 Функция затенения и экранирования Смита

Широко распространенной и популярной в современных движках моделью для затенения и экранирования является модель Смита из работы [21]. Основным преимуществом ее по сравнению с моделью V-бороздок является более приближенное к реальности поведение освещения на шероховатых поверхностях при приближении углов наблюдения к 90 градусам по сравнению с нормалью поверхности.

Предположение, которое лежит в основе профиля микроповерхности в этой модели – отсутствие автокорреляции направления нормали в точках на микроповерхности. Это предположение не выполняется в реальности (рисунок 24), однако эмпирические наблюдения показывают, что картина экранирования такой поверхности близко к реальной. Связано это с тем, что в

реальной поверхности в целом выполняется свойство поверхности Смита о том, что ориентация нормали – это локальное свойство поверхности, а перекрытие ее другими гранями происходит на неком достаточно большом расстоянии и эти два свойства не коррелируют друг с другом [34].

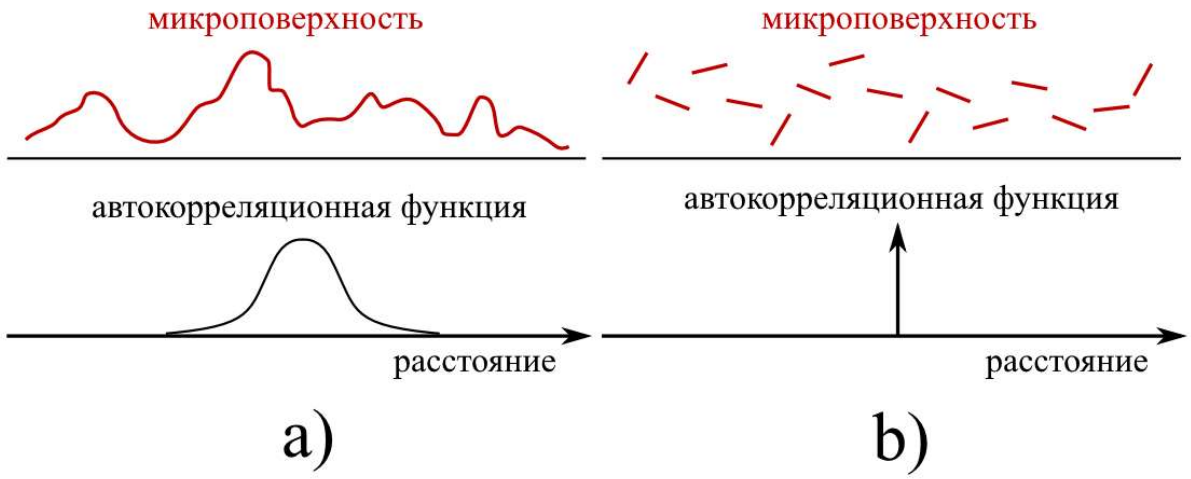

Рисунок 24 – a) профиль и автокорреляционная функция реальной микроповерхности,
b) профиль и автокорреляционная функция микроповерхности Смита [34]

Поэтому функцию экранирования можно разделить на локальную и дальнюю части, где локальная часть – простое отбрасывание нормалей микрограней, повернутых задней частью

$$G_1^{local} = H(\omega_o \cdot \omega_m). \quad (5)$$

Дальняя часть чаще всего используется в следующем виде

$$G_1^{dist} = \frac{1}{1 + \Lambda(\omega_o)}, \quad (6)$$

где $\Lambda(\omega_o)$ – интеграл по пространству уклонов микрограней. [23,34] Перемножение частей (5) и (6) дает общую функцию экранирования.

2.2 Постановка задачи и анализ существующих подходов к решению аналогичных задач

Поскольку для сохранения физической достоверности BRDF Шермейна [13,14] не может использовать функцию Смита [21] для затенения и

экранирования, задача состоит в том, чтобы адаптировать профиль микроповерхности V-бороздок и соответствующую ему функцию затенения и экранирования для нецентрированного распределения Бекманна, которое используется в LEADR mapping [20].

В работе, которая представила метод LEADR Mapping [20], функция Смита для распределения Бекманна получается, как функция для распределения с произвольным средним. Для получения такой функции, авторы произвели сдвиг параметров функции Смита на средний наклон поверхности вдоль направления падающего света. Аналогичный подход можно использовать и в данном случае, но поскольку V-бороздки в отличие от микрограней в модели Смита имеют жесткие связи между собой, необходимо рассмотреть, как будет реально расположена в пространстве подобная структура.

2.3. Модель V-бороздок для нецентрированных распределений

Для распространения модели V-бороздок на случай распределения с произвольным средним можно повернуть профиль микроповерхности на средний наклон поверхности задаваемых картой смещения вдоль направления падающего света. Этот прием задает несколько маловероятную модель макроповерхности в целом (рисунок 25), однако вполне интуитивно понятен при рассмотрении отдельного пикселя с отдельным распределением.

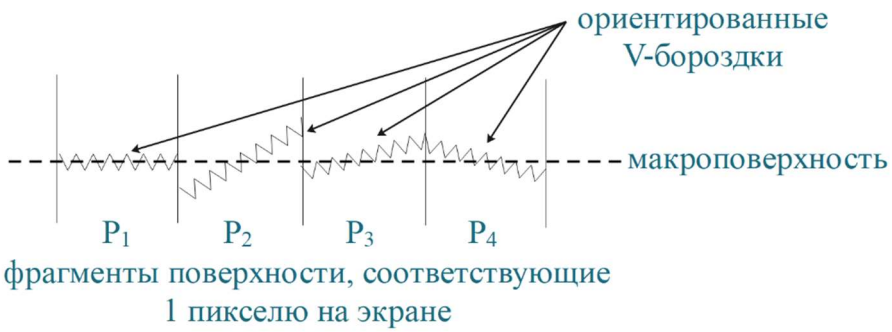

Рисунок 25 – Модель V-бороздок со смещенным средним, в силу построения модели в профиле микроповерхности присутствуют разрывы между соседними пикселями P_1, P_2, P_3, P_4

В самом деле, для конкретно выделенного пикселя средняя нормаль (мезонормаль) поверхности будет повернута относительно геометрической нормали поверхности, поэтому естественно смоделировать эту ситуацию, повернув модель профиля микроповерхности на тот же угол (рисунок 26). Мезонормаль поверхности ω_n имеет наклон θ по сравнению с геометрической нормалью ω_g. Моделируется затенение и экранирование V-бороздок, повернутых на тот же самый угол θ относительно ω_g.

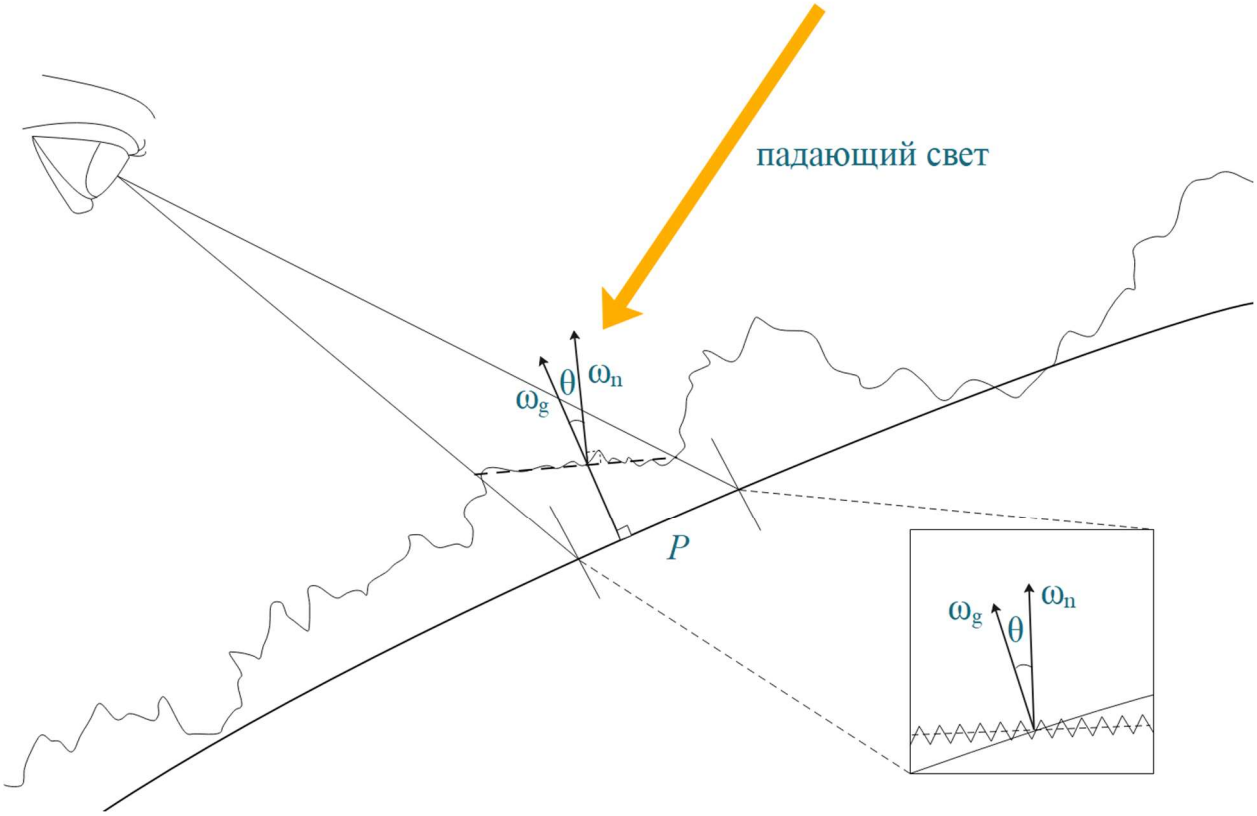

Рисунок 26 – Общая схема, демонстрирующая идею метода

Рассмотрим возникающую при этом ситуацию более детально в плоскости, задаваемой векторами падающего света и направлением наблюдения (рисунок 27).

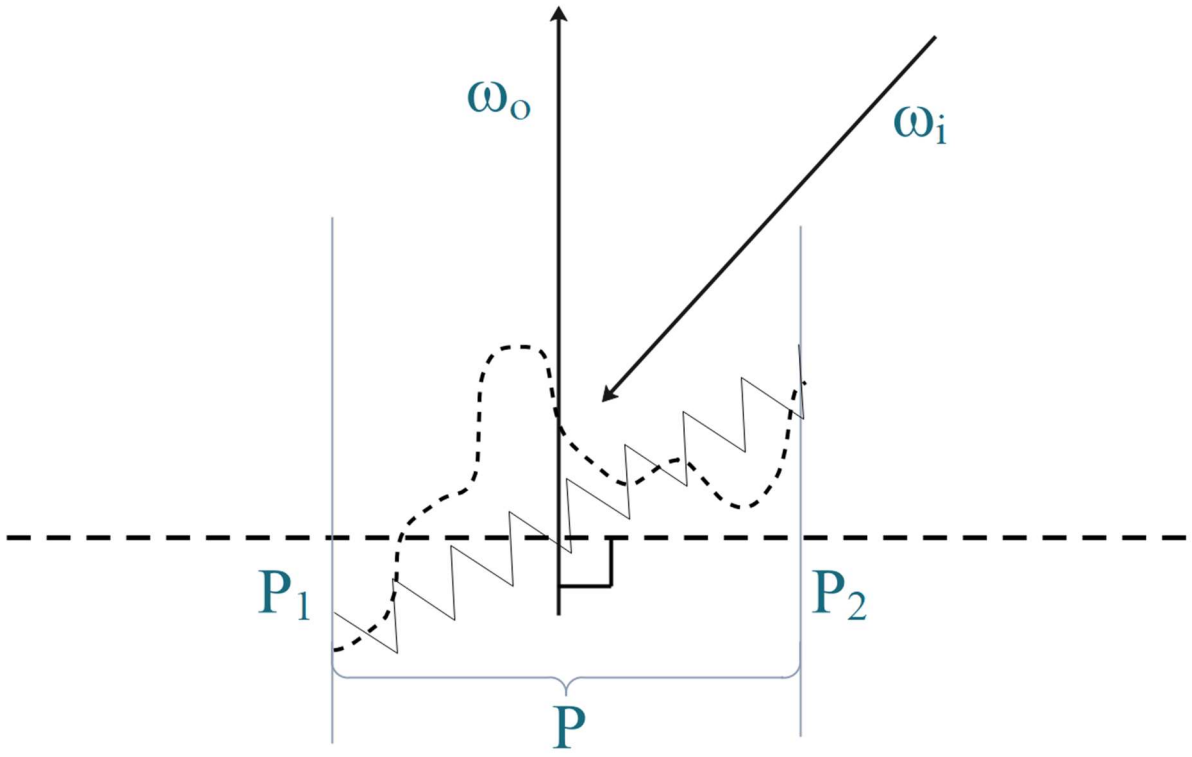

Рисунок 27 – Фрагмент поверхности P, соответствующий одному пикселю на экране в плоскости, образованной падающим светом ω_i и направлением наблюдения ω_o. Крупным пунктиром обозначена ось, перпендикулярная направлению наблюдения, мелким – детали на карте смещений

Будем считать, что поверхность, задаваемая картой смещения непрерывна (в случае поверхности воды это разумное предположение) и задается на отрезке P функцией f(x). Формальное определение среднего наклона такой поверхности (f′ – производная функции)

$$\langle f' \rangle = \frac{1}{|P|} \int_P f'(x)dx = \frac{f(P_2) - f(P_1)}{|P|}. \quad (7)$$

Из выражения (7) следует, что средний наклон поверхности определяется смещением поверхности в начале и в конце отрезка P. Это подводит к важному наблюдению – ситуации, когда обе стороны бороздки повернуты к наблюдателю задней стороной не может произойти, так как точки начала и конца поверхности лежат на соответствующих им лучам границ фрагмента. Это наблюдение еще будет рассмотрено в обсуждении корректности модели.

С другой стороны, лучи падающего света, очевидно, могут попасть в заднюю сторону поверхности, даже если они находятся в верхней полусфере относительно геометрической поверхности, лежащей в основе. В этом состоит качественное отличие новой модели от стандартной модели V-бороздок для центрированных распределений. Этот случай нужно рассматривать как полное затенение (рисунок 28).

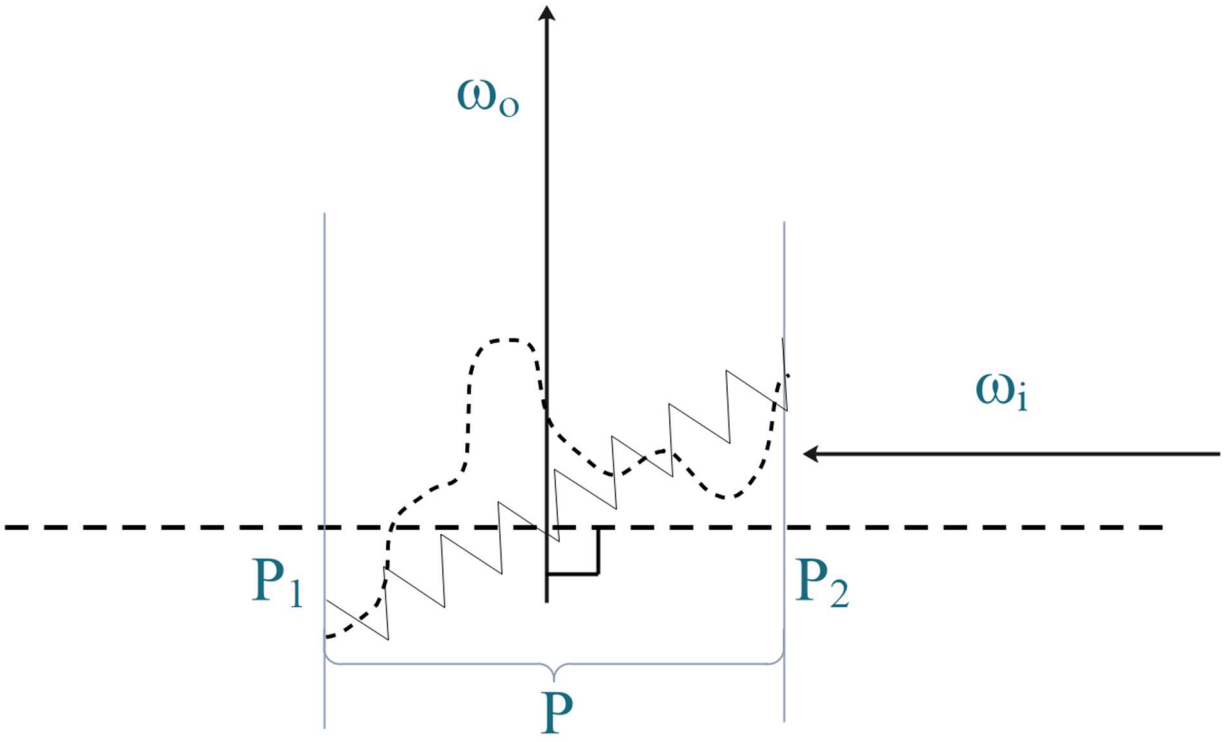

Рисунок 28 – Ситуация полного затенения (все поверхности V-бороздок повернуты к лучу света задней частью)

Выведем в явном виде множитель экранирования для примитивной микроповерхности из повернутых V-бороздок. Пусть $M(\theta)$ – матрица поворота на угол θ – средний наклон микроповерхности по сравнению с макроповерхностью. Тогда $M(\theta)\omega_m$ и $M(\theta)\omega'_m$ – нормали сторон повернутой V-бороздки. Тогда распределение нормалей имеет вид

$$D(\omega) = \frac{1}{2}\frac{\delta_{M(\theta)\omega_m}(\omega)}{(M(\theta)\omega_m)\cdot\omega_g} + \frac{1}{2}\frac{\delta_{M(\theta)\omega'_m}(\omega)}{(M(\theta)\omega'_m)\cdot\omega_g}, \quad (8)$$

где $\delta(x)$ – дельта-функция Дирака, ω_g – нормаль геометрической макроповерхности.

Далее по отношению к распределению (8), использовалось свойство сохранения видимой проекции поверхности для получения множителя экранирования $G_1(\omega_o, \omega)$, как предложено в работе [34]

$$\cos\theta_o = \int_\Omega G_1(\omega_o, \omega)\langle\omega_o, \omega\rangle\, D(\omega)d\omega =$$

$$= \frac{1}{2} G_1(\omega_o, \omega_m) \frac{\langle\omega_o, M(\theta)\omega_m\rangle}{(M(\theta)\omega_m)\cdot\omega_g} + \frac{1}{2} G_1(\omega_o, \omega'_m) \frac{\langle\omega_o, M(\theta)\omega'_m\rangle}{(M(\theta)\omega'_m)\cdot\omega_g}, \quad (9)$$

где θ_o – угол между нормалью геометрической макроповерхности и направлением наблюдения.

Для множителя экранирования также как в ситуации центрированных распределений (4) возможны ситуации, когда $G_1(\omega_o, \omega_m) = G_1(\omega_o, \omega'_m) = 1$, если обе нормали видны, или какая-то из нормалей не видна, тогда $G_1(\omega_o, \omega'_m) = 0$ (для простоты примем что ω_m – нормаль, угол которой с направлением наблюдения меньше), тогда из выражения (9) следует

$$\cos\theta_o = \frac{1}{2} G_1(\omega_o, \omega_m) \frac{\langle\omega_o, M(\theta)\omega_m\rangle}{(M(\theta)\omega_m)\cdot\omega_g},$$

$$G_1(\omega_o, \omega_m) = 2\frac{\cos\theta_o\left((M(\theta)\omega_m)\cdot\omega_g\right)}{\langle\omega_o, M(\theta)\omega_m\rangle} = 2\frac{(\omega_o\cdot\omega_g)\left((M(\theta)\omega_m)\cdot\omega_g\right)}{\langle\omega_o, M(\theta)\omega_m\rangle}. \quad (10)$$

Комбинация случая, когда $G_1(\omega_o, \omega_m) = G_1(\omega_o, \omega'_m) = 1$, и случая в выражении (10) дает общую формулу для множителя экранирования

$$G_1(\omega_o, \omega_m) = \min\left(1, 2\frac{(\omega_o\cdot\omega_g)\left((M(\theta)\omega_m)\cdot\omega_g\right)}{\langle\omega_o, M(\theta)\omega_m\rangle}\right). \quad (11)$$

Таким образом, множитель экранирования (11) аналогичен стандартной модели с точностью до поворота векторов.

Множитель затенения будет во многом похож на множитель экранирования, но должен учитывать случай на рисунке 28 о котором говорилось выше. Обозначим его $G'_1(\omega_i, \omega_m)$. Математически необходимо чтобы G'_1 обращался в 0, когда угол между ω_g и ω_i был больше по модулю, чем $\theta' = \frac{\pi}{2} - |\theta|$ и был разного знака с θ (рисунок 29).

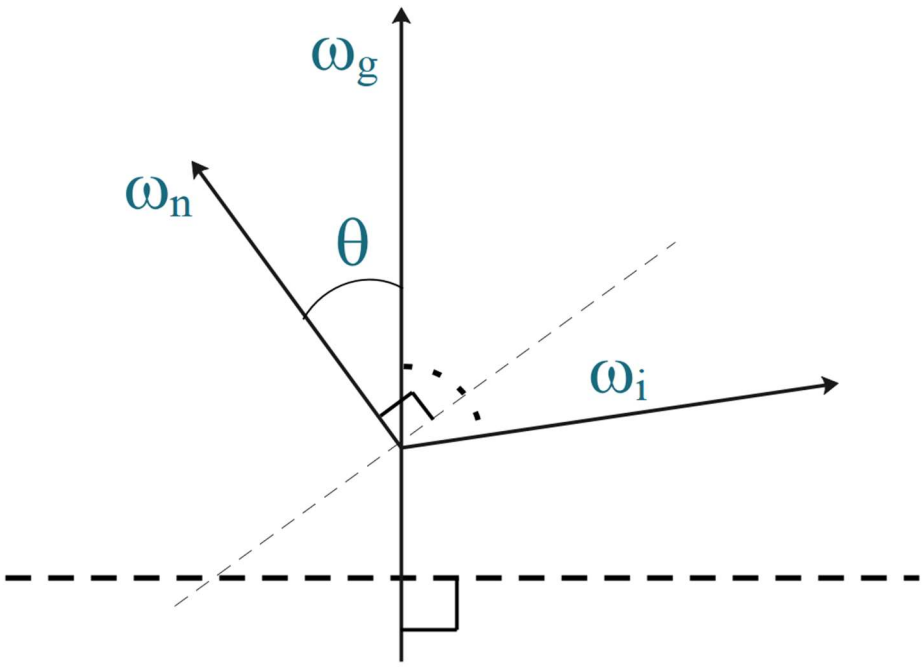

Рисунок 29 – Геометрическая картина углов в ситуации полного затенения

Это равносильно тому, что проекция ω_i на направление ω_n отрицательна, тогда

$$G'_1(\omega_i, \omega_m) = H(\omega_i \cdot \omega_n) \cdot \min\left(1, 2\frac{(\omega_i \cdot \omega_g)\left((M(\theta)\omega_m) \cdot \omega_g\right)}{\langle \omega_i, M(\theta)\omega_m \rangle}\right). \qquad (12)$$

Итак, после объединения выражений (11) и (12), общая функция затенения и экранирования нормали ω_m при среднем наклоне поверхности θ:

$$G_2(\omega_o, \omega_i, \theta, \omega_m) = H\big(\omega_i \cdot \omega_n(\theta)\big) \cdot \min\left(1, 2\frac{\big(\omega_i \cdot \omega_g\big)\Big(\big(M(\theta)\omega_m\big) \cdot \omega_g\Big)}{\langle \omega_i, M(\theta)\omega_m\rangle}\right) \cdot$$

$$\cdot \min\left(1, 2\frac{\big(\omega_o \cdot \omega_g\big)\Big(\big(M(\theta)\omega_m\big) \cdot \omega_g\Big)}{\langle \omega_o, M(\theta)\omega_m\rangle}\right). \qquad (13)$$

Записью $\omega_n(\theta)$ отмечено, что ω_n и θ связаны между собой.

2.4 Обсуждение корректности функции

Проверим, что распределение видимых нормалей поверхности, индуцированное множителем экранирования $G_1(\omega_o, \omega_m)$ удовлетворяет условию нормировки

$$\int_\Omega D_{\omega_o}(\omega_m)\mathrm{d}\omega_m = 1\,. \qquad (14)$$

Используем для выражение из работы [34] и подставим в него $G_1(\omega_o, \omega_m)$ из (11):

$$D_{\omega_o}(\omega_m) = \frac{G_1(\omega_o, \omega_m)\langle \omega_o, M(\theta)\omega_m\rangle D(\omega_m)}{\cos\theta_o} =$$

$$= \frac{\min\left(1, 2\frac{\big(\omega_o \cdot \omega_g\big)\Big(\big(M(\theta)\omega_m\big) \cdot \omega_g\Big)}{\langle \omega_o, M(\theta)\omega_m\rangle}\right)\langle \omega_o, M(\theta)\omega_m\rangle D(\omega_m)}{\cos\theta_o}. \qquad (15)$$

Рассмотрим случаи в операторе $\min(1, -)$ выражения (15) отдельно. Начнем со случая, когда минимален второй компонент, тогда

$$D_{\omega_o}(\omega_m) = \frac{2\big(\omega_o \cdot \omega_g\big)\Big(\big(M(\theta)\omega_m\big) \cdot \omega_g\Big)\langle \omega_o, M(\theta)\omega_m\rangle D(\omega_m)}{\langle \omega_o, M(\theta)\omega_m\rangle \cos\theta_o} =$$

$$= 2H(\omega_o \cdot M(\theta)\omega_m)\Big(\big(M(\theta)\omega_m\big) \cdot \omega_g\Big) D(\omega_m). \qquad (16)$$

Подставим полученное выражение (16) в условие нормировки (14) и получим:

$$2\int_{\Omega} \mathrm{H}(\omega_o \cdot \mathrm{M}(\theta)\omega_m)\left((\mathrm{M}(\theta)\omega_m)\cdot\omega_g\right)\mathrm{D}(\omega_m)\mathrm{d}\omega_m\,. \qquad (17)$$

Здесь, аналогично случаю центрированного распределения [34], можно заметить, что второй компонент $\min(1,-)$ минимален, когда направление ω_o имеет достаточно близкий к прямому угол с мезонормалью ω_n и иметь такие углы оно может с обеих сторон от нее. При этом ω_n отвечает средней нормали в распределении. Поэтому $\mathrm{H}(\omega_o \cdot \mathrm{M}(\theta)\omega_m)$ делит нормированное распределение (18)

$$\int_{\Omega}\left((\mathrm{M}(\theta)\omega_m)\cdot\omega_g\right)\mathrm{D}(\omega_m)\mathrm{d}\omega_m = 1 \qquad (18)$$

ровно пополам (с одной стороны от ω_n значение аргумента положительно, а с другой – отрицательно). Поэтому из выражения (17):

$$2\int_{\Omega} \mathrm{H}(\omega_o \cdot \mathrm{M}(\theta)\omega_m)\left((\mathrm{M}(\theta)\omega_m)\cdot\omega_g\right)\mathrm{D}(\omega_m)\mathrm{d}\omega_m = 1. \qquad (19)$$

Из выражения (19) следует, что условие нормировки выполнено и полученная функция экранирования корректна.

Проверить корректность функции затенения достаточно сложно, так как она в силу определения должна терять энергию, однако можно убедиться, что она не добавляет энергию. Это достаточно очевидно следует из того, что она отличается от только что проверенной функции экранирования (с точностью до замены ω_o на ω_i) лишь функцией Хевисайда, которая принимает значения 0 и 1 и никак не может увеличить значение при интегрировании.

Таким образом, вся функция затенения и экранирования (13) физически корректна.

3 РЕАЛИЗАЦИЯ АЛГОРИТМА

3.1 Оптимизация функции экранирования и затенения V-бороздок для нецентрированных распределений

Модифицированные функции затенения и экранирования требуют поворота векторов на средний наклон поверхности в тангенциальном пространстве:

$$G_1(\omega_o, \omega_m) = \min\left(1, 2\frac{(\omega_o \cdot \omega_g)\left((M(\theta)\omega_m) \cdot \omega_g\right)}{\langle \omega_o, M(\theta)\omega_m \rangle}\right), \quad (20)$$

$$G_1(\omega_i, \omega_m) = H(\omega_i \cdot \omega_n) \cdot \min\left(1, 2\frac{(\omega_i \cdot \omega_g)\left((M(\theta)\omega_m) \cdot \omega_g\right)}{\langle \omega_i, M(\theta)\omega_m \rangle}\right). \quad (21)$$

Для получения среднего значения наклона в (20) и (21) воспользуемся результатом работы LEADR Mapping. Для получения множителя затенения и экранирования Смита авторы работы воспользовались следующим представлением параметров гауссова распределения нормалей поверхности, спроецированного в направление источника света (для множителя затенения) или наблюдения (для множителя экранирования) $\omega = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$:

$$\mu(\phi) = \cos\phi\, E[x_{\tilde{n}}] + \sin\phi\, E[y_{\tilde{n}}], \quad (22)$$

$$\sigma^2(\phi) = \cos^2\phi\, \sigma_x^2 + \sin^2\phi\, \sigma_y^2 + 2\cos\phi\sin\phi\, c_{xy}. \quad (23)$$

Здесь $\mu(\phi)$ в выражении (22) это среднее, а $\sigma^2(\phi)$ в выражении (23) – среднеквадратичное отклонение.

Для модифицированной функции V-бороздок интерес представляет только среднее $\mu(\phi)$ из выражения (22). Для получения среднего авторы работы раскладывают вектор ω по тангенциальному базису, обозначаемому векторами X, Y, Z (Z – нормаль). Обозначим компоненты разложения

$\omega_x, \omega_y, \omega_z$. Теперь обозначим средний наклон вдоль векторов базиса x_0, y_0. После этого среднее выражается как

$$\mu(\phi) = \omega_x x_0 + \omega_y y_0. \quad (24)$$

Далее рассмотрим то, как применить поворот на полученное в (24) значение к вектору.

Операцию поворота нужно проводить в пиксельном шейдере (т.е. для каждого пикселя на экране). Поворот с помощью умножения на матрицу не совсем рационален для такой цели, так как это требует дополнительных манипуляций для поиска вектора оси поворота, поэтому рассмотрим альтернативный способ рассчитать поворот.

На рисунке 30 представлено две ситуации поворота вектора в плоскости, образованной z и xy компонентами вектора.

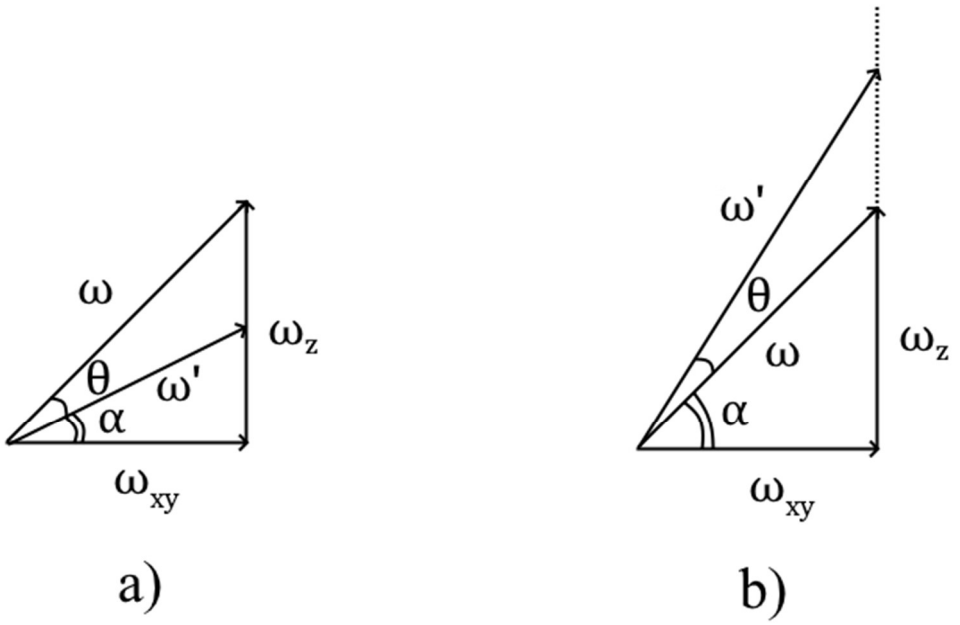

Рисунок 30 – Конфигурация вектора ω и его проекций на тангенциальный базис при повороте на угол θ a) от нормали, b) к нормали

Можно заметить, что поскольку интерес представляет только направление вектора, то можно рассчитать, как должна измениться z-компонента вектора, чтобы направление изменилось на требуемый угол θ.

Обозначим угол нового направления за α. На рисунке 30а представлена ситуация, когда поворот происходит в направлении от нормали. В этом случае можно заметить, что

$$\cos(\alpha + \theta) = \frac{\omega_z}{\omega}, \tag{25}$$

$$\sin(\alpha + \theta) = \frac{\omega_{xy}}{\omega}. \tag{26}$$

После разложения (25) и (26) как синус и косинус суммы получаем

$$\cos\alpha\cos\theta - \sin\alpha\sin\theta = \frac{\omega_z}{\omega}, \tag{27}$$

$$\sin\alpha\cos\theta + \cos\alpha\sin\theta = \frac{\omega_{xy}}{\omega}. \tag{28}$$

Разделим выражения (27) и (28) на $\cos\alpha$

$$\cos\theta - \operatorname{tg}\alpha\sin\theta = \frac{\omega_z}{\omega\cos\alpha}, \tag{29}$$

$$\operatorname{tg}\alpha\cos\theta + \sin\theta = \frac{\omega_{xy}}{\omega\cos\alpha}. \tag{30}$$

Объединим выражения (29) и (30) и выразим $\operatorname{tg}\alpha$

$$\frac{\cos\theta - \operatorname{tg}\alpha\sin\theta}{\omega_z} = \frac{\operatorname{tg}\alpha\cos\theta + \sin\theta}{\omega_{xy}},$$

$$\omega_{xy}\cos\theta - \omega_{xy}\operatorname{tg}\alpha\sin\theta = \omega_z\operatorname{tg}\alpha\cos\theta + \omega_z\sin\theta,$$

$$\operatorname{tg}\alpha\left(\omega_z\cos\theta + \omega_{xy}\sin\theta\right) = \omega_{xy}\cos\theta - \omega_z\sin\theta,$$

$$\operatorname{tg}\alpha = \frac{\omega_{xy}\cos\theta - \omega_z\sin\theta}{\omega_z\cos\theta + \omega_{xy}\sin\theta}. \tag{31}$$

Тогда с учетом (31) новая z-компонента выражается как

$$\omega_z' = \omega_{xy}\operatorname{tg}\alpha = \omega_{xy}\frac{\omega_{xy}\cos\theta - \omega_z\sin\theta}{\omega_z\cos\theta + \omega_{xy}\sin\theta}. \tag{32}$$

В случае, представленном на рисунке 30b, можно выразить новую z-компоненту следующим образом

$$\omega_z' = \omega_{xy}\,\mathrm{tg}(\alpha+\theta) = \omega_{xy}\frac{\sin(\alpha+\theta)}{\cos(\alpha+\theta)} =$$

$$= \omega_{xy}\frac{\sin\alpha\cos\theta+\sin\theta\cos\alpha}{\cos\alpha\cos\theta-\sin\alpha\sin\theta} = \omega_{xy}\frac{\omega_z\cos\theta+\omega_{xy}\sin\theta}{\omega_{xy}\cos\theta-\omega_z\sin\theta}. \qquad (33)$$

Выражения (32) и (33) значительно легче рассчитать в пиксельном шейдере, чем матрицу поворота. После этих расчетов останется только нормализовать полученный вектор ω'.

3.2. Реализация алгоритма в фреймворке на OpenGL 3.3

Для имплементации алгоритма и удобства оценки результатов его работы за основу был взят код из приложения к работе Шермейна [14]. Реализация написана авторами для их фреймворка, выполненного на языке C++ с использованием графического API OpenGL 3.3, библиотеки GLFW для создания окна и ввода-вывода, библиотеки Dear ImGui для отладочного интерфейса, библиотеки stb_image для загрузки текстур, библиотеки assimp для загрузки моделей и библиотеки tinyexr для работы с изображениями в широком динамическом диапазоне. Этот фреймворк подходит для оценки результатов работы алгоритма, однако далее алгоритм также был имплементирован в Unreal Engine 5.3 на более современном графическом API DirectX 12, так как OpenGL 3.3 уже не является актуальным API, особенно после популяризации графических API нового поколения таких как Vulkan и DirectX 12.

После замены функции затенения и экранирования V-бороздок на модификацию для нецентрированных распределений и учет весов проекции по методу из работы [20] результат работы алгоритма на арктической сцене представлен на рисунке 31 (арктическая сцена выбрана для удобства сравнения с результатами из работы [14]).

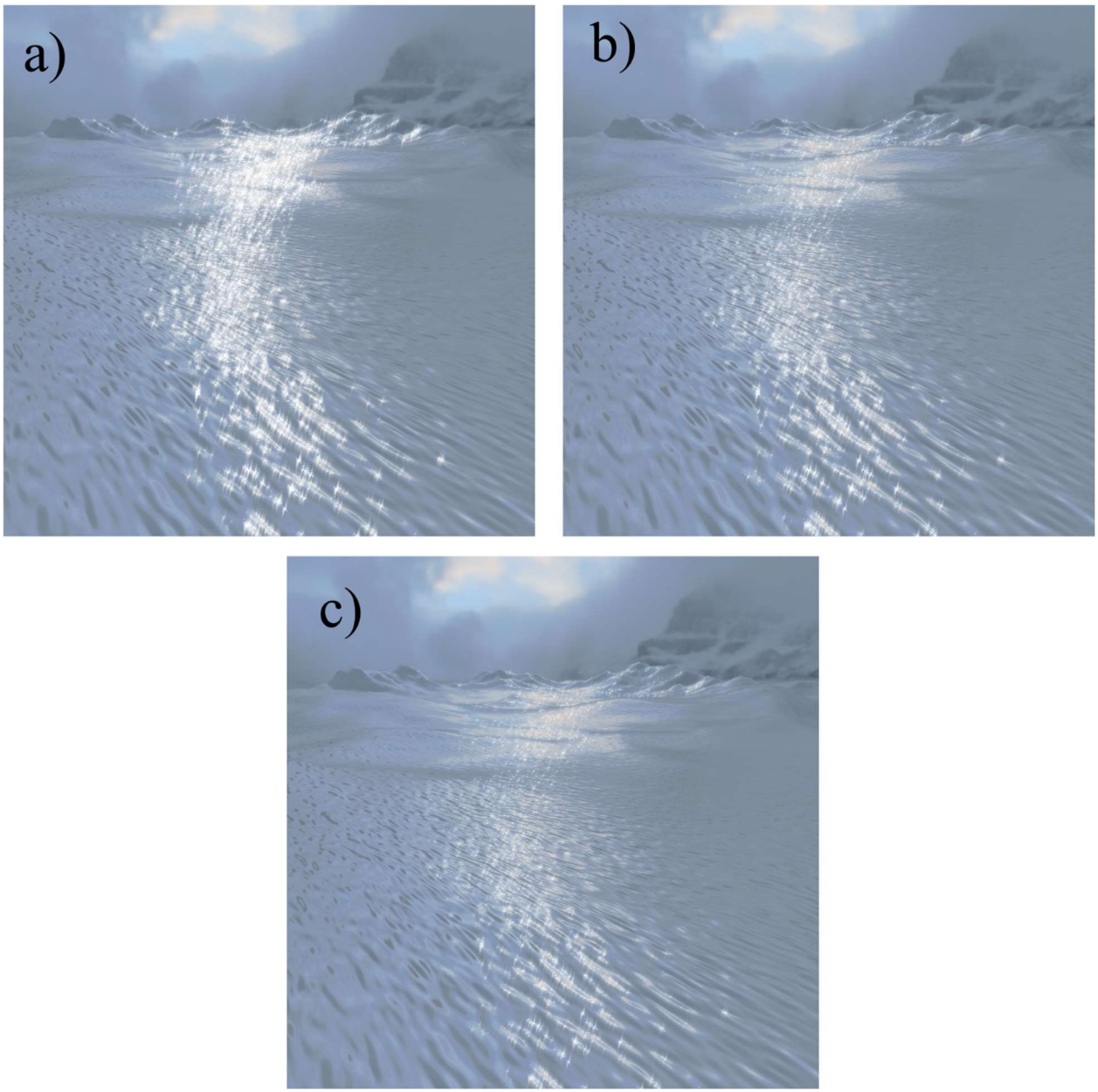

Рисунок 31 – Результат работы a) изначального алгоритма [14], b) модифицированного алгоритма, c) модифицированного алгоритма с учетом корректного множителя Френеля

Как можно увидеть, новый алгоритм отбрасывает большее количество отблесков чем исходный, что ожидаемо, так как модифицированная функция затенения и экранирования зависит не от геометрической нормали, а от мезонормали, которая может достаточно сильно отличаться от нее (и при просмотре под небольшими углами к поверхности это в основном будет приводить к большему экранированию при повороте от наблюдателя или затенению при повороте к наблюдателю). Также можно заметить, что

интенсивность некоторых отблесков уменьшилась вследствие использования более точных весов проекции.

Убедимся, что отброшенные отблески не являются следствием алиасинга. Для этого сравним результат с результатом, полученным усреднением выборки из 1024 точек на пиксель (рисунок 32). С помощью сравнения плотности красных областей на тепловой карте (отвечающей отсутствующим отблескам по сравнению с большой выборкой) можно убедиться, что дополнительного алиасинга по сравнению с оригинальным алгоритмом модификация не вызывает.

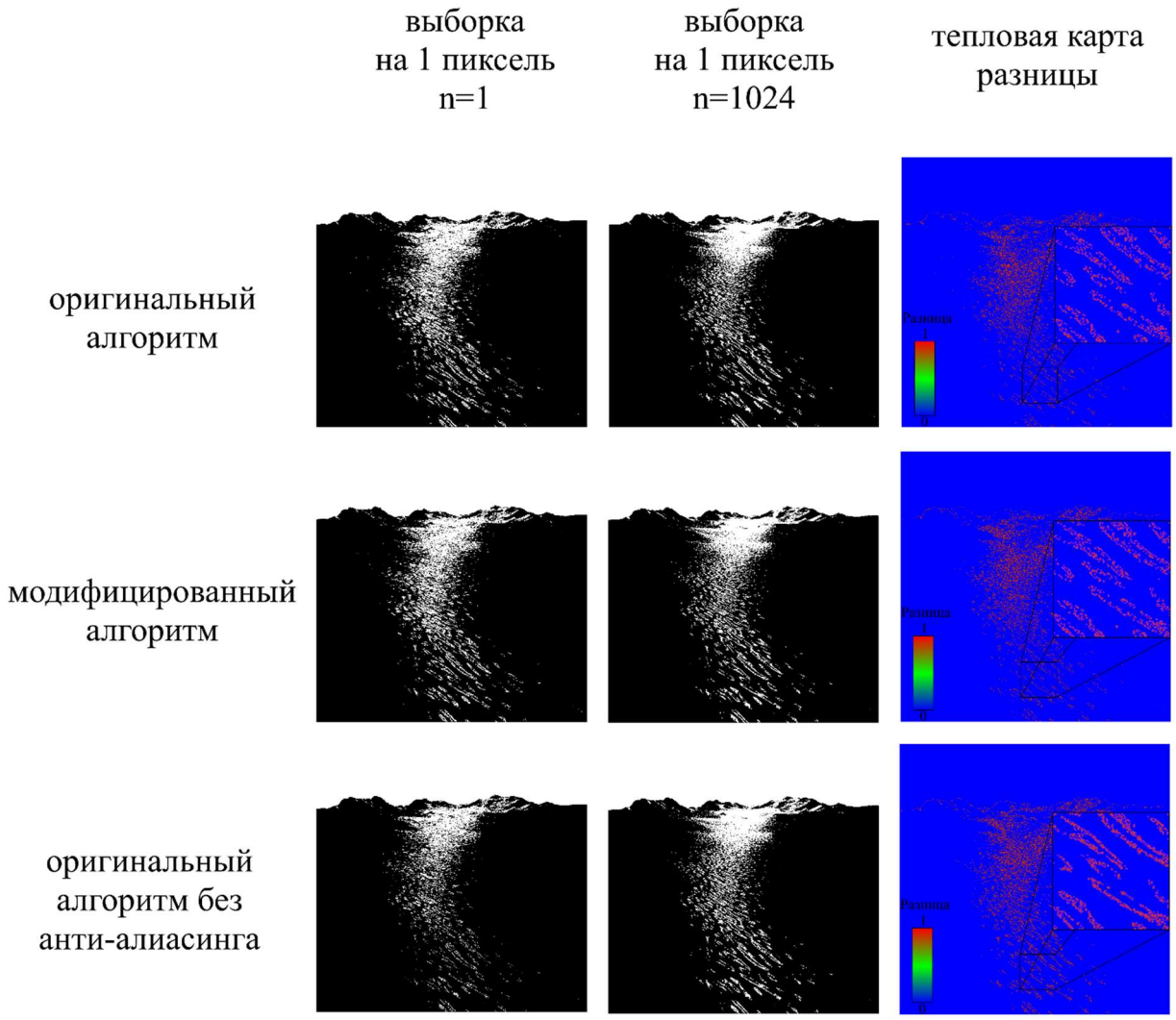

Рисунок 32 – Сравнение алиасинга в оригинальном алгоритме, модифицированном алгоритме и оригинальном алгоритме с отключенным геометрическим анти-алиасингом (только зеркальные отражения, эффект bloom выключен, уровни света приведены к 1 в пикселях, где они больше 0)

На рисунке 31c также приведен результат работы алгоритма с множителем Френеля для льда. Поскольку одна из целей модификации – это улучшение физической достоверности алгоритма, множитель был учтен с применением известной аппроксимации Шлика [38]

$$F = F_0 + (1 - F_0) \cdot (1 - \omega_o \cdot \omega_h)^5, \quad (34)$$

где F_0 – отражение света, падающего на поверхность материала под прямым углом к ней, ω_o – направление наблюдения, ω_o – направление наблюдения, ω_h – так называемый «halfway vector», т. е. среднее между направлением падающего света и направлением наблюдения. В (34) используется этот вектор, а не вектор нормали, так как он соответствует вектору нормали микрограней, учитываемых в расчете зеркальной BRDF.

3.3 Реализация алгоритма в Unreal Engine 5.3

Для реализации модификации был выбран Unreal Engine 5, так как этот движок широко используется в современных графически интенсивных играх, имеет открытый исходный код, а также имеет свою собственную реализацию алгоритма Шермейна и других в своей экспериментальной модели материалов Substrate [16,29]. Было решено придерживаться обычной модели материалов, так как модель Substrate все еще находится в разработке, и схема ее работы может измениться, но отрисовка отблесков в Substrate была использована для сравнения алгоритмов.

С момента выхода Unreal Engine 4 в Unreal Engine используется отложенная отрисовка [39]. Это значит, что непрозрачные материалы при базовом проходе записывают свои атрибуты в набор текстур (называемый GBuffer), и далее при расчете освещения эти атрибуты попиксельно считываются. Проблемой для алгоритма является тот факт, что все нормали, переведенные в единое пространство также записаны в GBuffer (рисунок 33). Соответственно, без большой модификации GBuffer-а и методов фильтрации текстур в движке, LEAN и LEADR mapping в движке не реализовать.

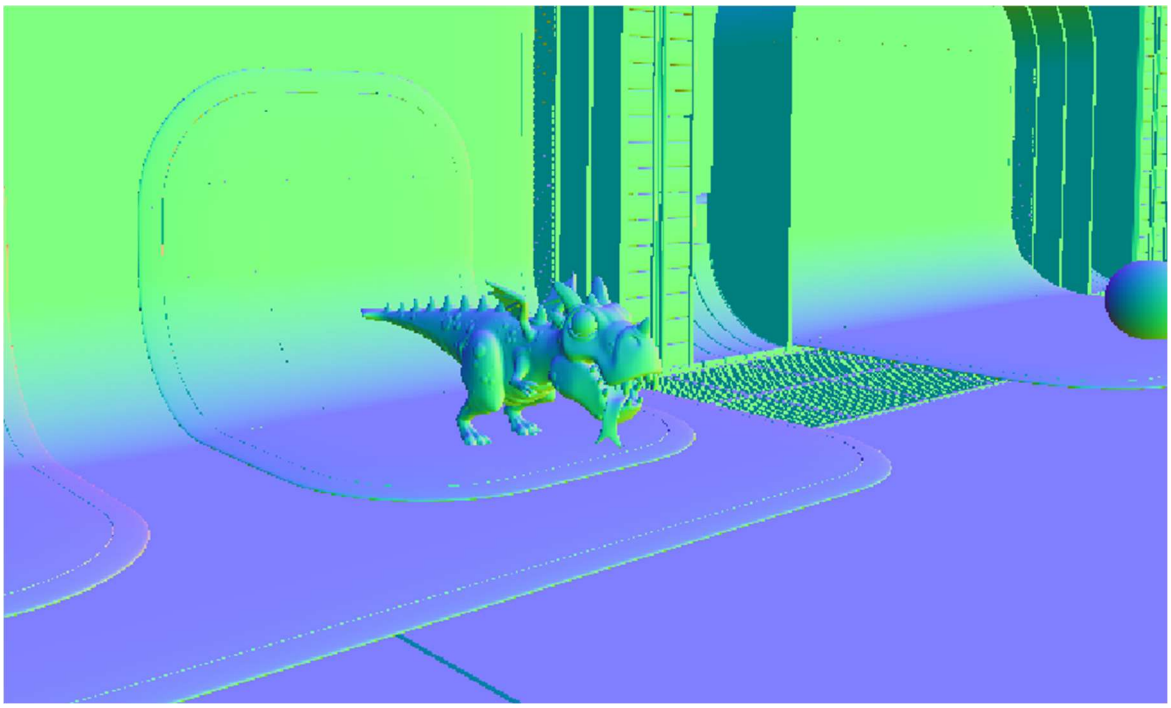

Рисунок 33 – GBufferA, содержащий нормали в мировом пространстве, после базового прохода отрисовки в Unreal 5.3

К счастью, в движке есть проход отрисовки, идеально подходящий для алгоритма – проход отрисовки полупрозрачных объектов. Этот проход использует прямую отрисовку и просчитывает освещение на полупрозрачных объектах, отрисовывая их поверх основной сцены с помощью альфа-смешивания [40]. В этом проходе можно отрисовать поверхность воды, используя полученный модифицированный алгоритм и LEADR Mapping и избежать недостатков существующей реализации в Substrate материалах в Unreal Engine 5.3.

В Unreal Engine 5 полупрозрачные объекты отрисовываются прямым проходом по ним и наложением на отрисованную сцену с помощью альфа-композитинга. Для использования этого прохода в движке была создана новая модель освещения материалов по аналогии с Thin Translucent (для тонких полупрозрачных объектов). Далее был использован код из Substrate модели для расчета NDF отблесков, рассчитан множитель Френеля в аппроксимации Шлика [38], веса проекции по выражению из работы [20] и множитель затенения и экранирования. Диффузная составляющая освещения была оставлена без изменений.

Для. тестов без анимации MIP-цепочки текстур карты нормалей, необходимые для LEAN/LEADR генерируются до старта игры с помощью простых шейдеров. Для тестов с анимацией генерация происходит каждый кадр после того, как генерируется новая карта нормалей.

Результат можно видеть на рисунке 34 вместе с примером отрисовки отблесков в модели Substrate, использующей реализацию алгоритма из работы Шермейна. Можно обратить внимание на то, что у алгоритма из Substrate очень большая дисперсия, которую нельзя поменять встроенными средствами, это создает нереалистичную для поверхности воды картину. Кроме того, вследствие полного отсутствия множителя затенения и экранирования в BRDF, используемой в Substrate, отблески практически равномерно покрывают поверхность и не исчезают на затененных от источника света областях.

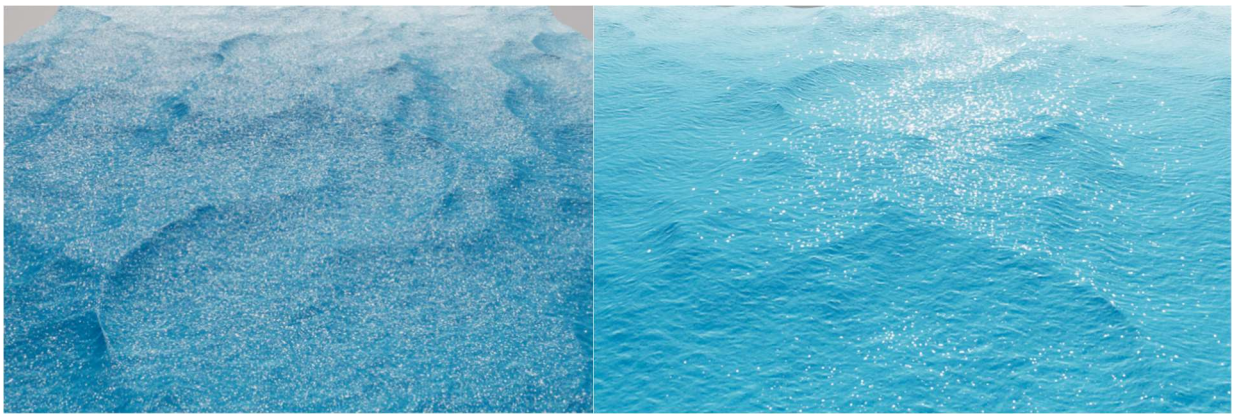

Рисунок 34 – Сравнение работы алгоритма из Substrate (слева) и нового алгоритма (справа)

Отрисовка воды в реальной игре почти всегда требует анимацию волн, а значит динамически меняющуюся карту нормалей. Для анимации волн и оценки вычислительного времени, требуемого для генерации MIP-цепочек LEAN/LEADR Mapping, была использована готовая реализация [41], основанная на работе [42]. Для симуляции волн эта реализация использует быстрое обратное преобразование Фурье для получения карт смещения волн на разных масштабах из меняющегося со временем спектра. Карта большего масштаба используется для смещения точек плоскости в шейдере, а меньшего

масштаба – в качестве карты нормалей для более мелких волн. На основе этой карты нормалей в свою очередь генерируются MIP-цепочки для LEADR Mapping и применяются в шейдере для достоверной фильтрации карт нормалей. На рисунке 35 приведен пример эффекта блестящей воды из реальной жизни и схожий эффект, отрисованный с помощью разработанного метода и симуляции волн.

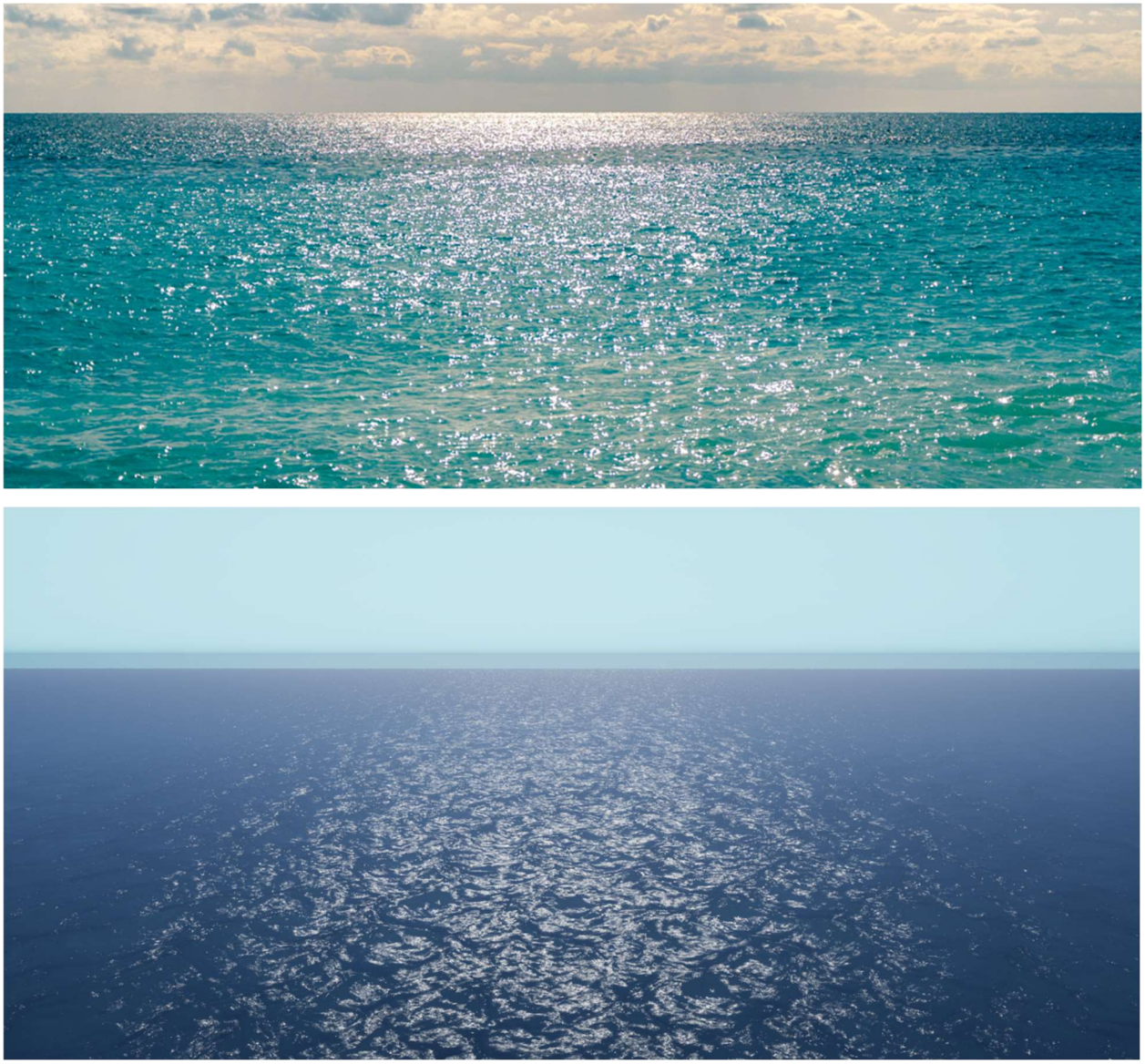

Рисунок 35 – Эффект блестящей морской поверхности в реальности (сверху) [43] и отрисованный новым алгоритмом на симулированной поверхности океана (снизу)

4 АНАЛИЗ РЕЗУЛЬТАТОВ

4.1 Анализ реализации алгоритма в фреймворке на OpenGL 3.3

Экспериментально оценить качество модифицированной модели отрисовки отблесков достаточно сложно. По сравнению с алгоритмом из исходной работы авторов [14] он настолько же физически достоверен. Это связано с тем, что для определения физической достоверности BRDF в графике реального времени используется так называемый weak white furnace test [34], который лишь проверяет что суммарная отраженная энергия не превышает суммарную пришедшую энергию (так как в графике реального времени моделируется только первое отражение, то есть часть энергии теряется). Оба алгоритма аналитически проходят этот тест. Реальная разница между этими алгоритмами заключается в более точных весах проекции, полученных из LEADR Mapping и более вероятной конфигурации V-бороздок в функции затенения и экранирования. Однако это сложно исследовать экспериментально.

Гораздо легче оценить производительность алгоритма. Эффективность модифицированного алгоритма по памяти такая же как у исходного, так как он не использует новых буферов или текстур. Поэтому проанализируем эффективность по времени. Для того чтобы измерить время интересующего основного прохода отрисовки было использовано расширение KHR_debug для OpenGL, чтобы отметить время начала и конца прохода, используя функции glPushDebugGroup и glPopDebugGroup.

С помощью Nvidia NSight на системе с GPU RTX 3070 и при разрешении 2560 на 1440 пикселей была измерена производительность оригинального и модифицированного алгоритмов. Результаты приведены в таблице 1. Время приведено для всего прохода отрисовки сцены.

Таблица 1 – Производительность оригинального и модифицированного алгоритмов на фреймворке из работы Шермейна [14]

	С LEAN/LEADR Mapping, мс	Без LEAN/LEADR Mapping, мс
Оригинальный алгоритм	3.40	2.91
Модифицированный алгоритм	3.38	–

Как можно заметить, по сравнению с оригинальным алгоритмом при включенной фильтрации карт нормалей LEAN Mapping, сложность алгоритма с LEADR Mapping практически не изменяется, то есть его можно использовать вместо оригинального и получить более физически достоверный результат.

4.2. Анализ реализация алгоритма в Unreal Engine 5.3

В Unreal Engine 5.3 было проведено сравнение результатов работы алгоритма по сравнению с алгоритмом, реализованном в системе материалов Substrate. Качественные различия между этими алгоритмами частично уже обсуждались выше, модель Substrate на данный момент не предоставляет пользователю многие параметры настройки модели, такие как среднеквадратичные отклонения распределения по осям и фактор корреляции. Скорее всего, это временные ограничения модели, так как возможность задать эти параметры присутствует в исходном коде движка. Также в имплементации отсутствует множитель затенения и экранирования, а множитель Френеля так же, как и в работе Шермейна принят равным 1. Тем не менее эта имплементация работоспособная и, если физическая достоверность не принципиальна, позволяет, например, создавать достаточно реалистично выглядящие блестящие краски.

Для измерения времени, затрачиваемого на отрисовку был использован внутренний инструмент Unreal Engine – stat gpu. Время отрисовки замерялось как время, добавляемое к времени отрисовки в соответствующем проходе

(«Translucency» для полупрозрачных материалов, «Lights» для непрозрачных материалов) по сравнению с отрисовкой аналогичного материала без отблесков. Замеры были проведены на собранной игре, чтобы избежать возможных проблем, привносимых редактором движка.

Результаты измерений производительности двух алгоритмов представлены в таблице 2 (для Substrate были взяты два материала – непрозрачный и полупрозрачный).

Таблица 2 – Сравнение производительности оригинального и модифицированного алгоритмов в одинаковой сцене в Unreal Engine 5.3

Измерение	Время отрисовки отблесков в Unreal Engine, мс
Substrate модель, непрозрачный материал	1.02
Substrate модель, полупрозрачный материал	1.15
Модифицированный алгоритм, полупрозрачный материал	1.09

Результаты измерений вычислительной эффективности алгоритма в Unreal Engine 5 еще раз подтверждают, что алгоритм не привносит значимой дополнительной сложности при отрисовке по сравнению с полупрозрачным материалом, отрисованным прямой отрисовкой, при этом являясь более точным в расчете яркости отблесков и их затенении. Однако, в силу того что он рисуется после остальной сцены, в случае непрозрачных поверхностей это приведет к бесполезной отрисовке игрового мира за поверхностью. Это значит, что в случае непрозрачных поверхностей метод не эффективен. В случае же прозрачных поверхностей, таких как вода, два этих метода имеют очень схожую эффективность. Значения около одной миллисекунды для отрисовки одного эффекта достаточно большие для использования в играх, где

бюджет кадра достигает максимум 33.3 мс, но если этот эффект важен для графики в игре, то его можно использовать без сильных компромиссов.

Далее было измерено время отрисовки двух алгоритмов в Unreal Engine в зависимости от доли экрана, которую занимает блестящая геометрия. Смысл этого измерения в том, чтобы оценить, насколько реально использовать отрисовку отблесков в ситуациях, когда такой геометрии разное количество в сцене. Геометрия для простоты представляла собой плоскость, закрывавшую части экрана и находившуюся на одном расстоянии от него. Результаты приведены в таблице 3 и на рисунке 36.

Таблица 3 – Сравнение роста вычислительной стоимости алгоритмов в зависимости от доли экрана, занимаемого блестящей геометрией

Процент экрана, занимаемый блестящей геометрией	Substrate, непрозрачный материал, мс	Substrate, полупрозрачный материал, мс	Алгоритм, полупрозрачный материал, мс
0	0.00	0.00	0.00
25	0.65	0.73	0.34
50	1.23	1.41	0.66
75	1.77	2.13	1.22
100	2.30	3.05	1.57

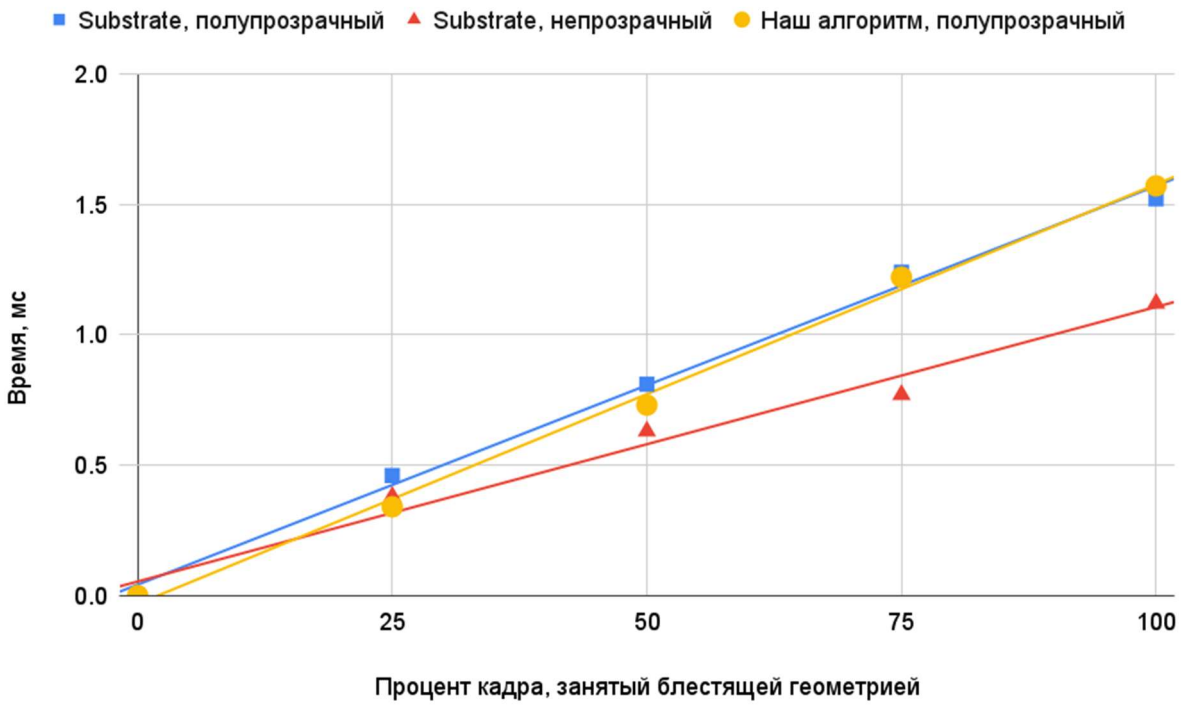

Рисунок 36 – Сравнение производительности алгоритма с алгоритмом модели Substrate в зависимости от процента экрана, занятого геометрией с блестящим материалом

Из этих результатов аналогично видно, что алгоритм дает схожую производительность с алгоритмом Substrate для полупрозрачных материалов и проигрывает в производительности непрозрачным материалам в Substrate. В случае использования одного из алгоритмов для отрисовки поверхности водоемов можно заметить, что при нахождении их на фоне игрового мира процент экрана, который они будут занимать достаточно ограничен, поэтому и вычислительная стоимость будет ограничена, что может позволить использовать такой подход в реальных играх на современных графических процессорах. В случае же когда камера может приблизиться к поверхности воды, вычислительная стоимость может быстро выйти из бюджета кадра. Кроме того, вблизи эффект начинает выглядеть достаточно неубедительно для поверхности воды, так как отблески при геометрическом размере распределения отблеска больше экранного пикселя имеют форму круга (Рисунок 37), когда отражения на поверхности воды вблизи в реальной жизни имеют неправильную форму, повторяющую формы волн.

Рисунок 37 – Эффект, создаваемый алгоритмом вблизи камеры

Наконец, была измерена вычислительная стоимость получения MIP-цепочек карт нормалей, генерируемых в реальном времени используемой симуляцией волн. Поскольку карты нормалей, используемые в симуляции, имеют достаточно небольшое разрешение в 256 на 256 пикселей, генерация занимает всего 0.04 мс – совсем небольшое время в сравнении с затрачиваемым на последующую отрисовку отблесков.

ЗАКЛЮЧЕНИЕ

Реализованная модификация алгоритма отрисовки отблесков дает более физически достоверный результат с более совершенным по сравнению с работой Шермейна [14] методом фильтрации карт нормалей LEADR Mapping. Благодаря использованию этого метода BRDF использует более точные веса проекции для распределения нормалей в пикселе в дополнение к более вероятной модели затенения и экранирования для деталей поверхности, индуцированных картой нормалей.

Вычислительная эффективность модификации по сравнению с оригинальным алгоритмом изменилась незначительно, а эффективность по памяти осталась прежней. Это дает возможность использовать алгоритм вместо оригинального без падения производительности в играх, вычислительный бюджет которых позволяет производить отрисовку отблесков оригинальным алгоритмом.

Полученный алгоритм может быть использован не только для отрисовки поверхности воды, но и для более точной отрисовки других блестящих поверхностей с индуцированными картой нормалей деталями, таких как блестящие ткани, краски и снег.

Алгоритм, как и оригинальный сохраняет ограничение в необходимости гауссова распределения нормалей, индуцированного картой нормалей. В случае невыполнения этого условия получившаяся поверхность может выглядеть чересчур шероховатой или чересчур блестящей. Поиск и применение метода фильтрации карт нормалей в реальном времени, сохраняющего детали и не требующего выполнения этого условия может быть дальнейшей перспективой улучшения алгоритма.

Результаты данной работы в виде исходных кодов, демонстрационных проектов Unreal Engine и инструкций по сборке опубликованы в репозитории на сервисе GitHub по ссылке: https://github.com/YakovDavis/WaterGlints.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Thoman P. Diving into Anti-Aliasing [Electronic resource] // Beyond3D. 2014. URL: https://www.beyond3d.com/content/articles/122 (accessed: 10.01.2023).
2. Temporal AA and the quest for the Holy Trail [Electronic resource] // The Code Corsair. 2022. URL: https://www.elopezr.com/temporal-aa-and-the-quest-for-the-holy-trail/ (accessed: 10.01.2023).
3. Banterle F. et al. Multisample anti-aliasing in deferred rendering // In Proc. Eurographics 2020 - Short Papers. 2020.
4. Kaplanyan A.S. et al. Filtering distributions of normals for shading antialiasing // High-Performance Graphics - ACM SIGGRAPH / Eurographics Symposium Proceedings, HPG . Association for Computing Machinery, 2016. Vol. 2016-June. P. 151–162.
5. Podee N. et al. Temporal and spatial anti-aliasing for rendering reflection on a water surface // ACM SIGGRAPH 2019 Posters, SIGGRAPH 2019. Association for Computing Machinery, Inc, 2019.
6. Podee N. et al. Temporal and spatial anti-aliasing for rendering reflections on water waves // Comput Vis Media (Beijing). Tsinghua University, 2021. Vol. 7, № 2. P. 201–215.
7. Tokuyoshi Y., Kaplanyan A.S. Improved geometric specular antialiasing // Proceedings - I3D 2019: ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. Association for Computing Machinery, Inc, 2019.
8. Jakob W. et al. Discrete Stochastic Microfacet Models // ACM Transactions on Graphics (Proceedings of SIGGRAPH). 2014. Vol. 33, № 4. P. 115:1–115:10.
9. Yan L.-Q. et al. Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces // ACM Transactions on Graphics (Proceedings of SIGGRAPH). 2014. Vol. 33, № 4. P. 116:1–116:9.

10. Wang B., Wang L., Holzschuch N. Fast Global Illumination with Discrete Stochastic Microfacets Using a Filterable Model // Computer Graphics Forum. Blackwell Publishing Ltd, 2018. Vol. 37, № 7. P. 55–64.
11. Wang B., Deng H., Holzschuch N. Real-Time Glints Rendering With Pre-Filtered Discrete Stochastic Microfacets // Computer Graphics Forum. Blackwell Publishing Ltd, 2020. Vol. 39, № 6. P. 144–154.
12. Zirr T., Kaplanyan A.S. Real-time rendering of procedural multiscale materials // In Proc. 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, I3D 2016. Association for Computing Machinery, Inc, 2016. P. 139–148.
13. Chermain X. et al. Procedural Physically based BRDF for Real-Time Rendering of Glints // Computer Graphics Forum. 2020. Vol. 39, № 7. P. 243–253.
14. Chermain X. et al. Real-Time Geometric Glint Anti-Aliasing with Normal Map Filtering // In Proc. ACM on Computer Graphics and Interactive Techniques. Association for Computing Machinery (ACM), 2021. Vol. 4, № 1. P. 1–16.
15. Substrate Materials Overview [Electronic resource] // Unreal Engine 5.3 Documentation. 2023.
16. Hillaire S., de Rousiers C. Authoring Materials That Matters - Substrate in Unreal Engine 5 // Advances in Real-Time Rendering in Games, SIGGRAPH 2023. 2023.
17. Yuksel C., Lefebvre S., Tarini M. Rethinking Texture Mapping // Computer Graphics Forum. Blackwell Publishing Ltd, 2019. Vol. 38, № 2. P. 535–551.
18. Neyret F. LEAN/LEADR LOD of specular [Electronic resource] // Shadertoy. URL: https://www.shadertoy.com/view/wlXXRn (accessed: 17.02.2024).
19. Olano M., Baker D. LEAN Mapping // In Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM Press, 2010. P. 181–188.
20. Dupuy J. et al. Linear Efficient Antialiased Displacement and Reflectance Mapping // ACM Trans. Graph. 2013. Vol. 32, № 6.

21. Smith B. Geometrical Shadowing of a Random Rough Surface // IEEE Trans Antennas Propag. 1967. Vol. 15, № 5. P. 668–671.
22. Wu L. et al. Accurate appearance preserving prefiltering for rendering displacement-mapped surfaces // ACM Trans Graph. Association for Computing Machinery, 2019. Vol. 38, № 4.
23. Reed N. Slope Space in BRDF Theory [Electronic resource] // Nathan Reed. 2021. URL: https://www.reedbeta.com/blog/slope-space-in-brdf-theory/ (accessed: 10.01.2023).
24. Luna F. Introduction to 3D Game Programming with DirectX 12. Mercury Learning and Information, 2016.
25. Tao C. et al. Real-Time Antialiased Area Lighting Using Multi-Scale Linearly Transformed Cosines // Eurographics Proceedings. 2021.
26. Maule M. et al. Transparency and Anti-Aliasing Techniques for Real-Time Rendering // 25th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials. 2012. P. 50–59.
27. Chermain X., Claux F., Mérillou S. A microfacet-based BRDF for the accurate and efficient rendering of high-definition specular normal maps // Visual Computer. Springer, 2020. Vol. 36, № 2. P. 267–277.
28. Cook R.L., Torrance K.E. A Reflectance Model for Computer Graphics // ACM Trans Graph. 1982. Vol. 1, № 1. P. 7–24.
29. Oztalay M. The Future of Materials in Unreal Engine // Game Developers Conference. 2023.
30. Simonot L. et al. Modeling, measuring, and using BRDFs: significant French contributions // J. Opt. Soc. Am. A. Optica Publishing Group, 2019. Vol. 36, № 11. P. C40–C50.
31. Sztrajman A. et al. Neural BRDF Representation and Importance Sampling // Computer Graphics Forum. John Wiley and Sons Inc, 2021. Vol. 40, № 6. P. 332–346.

32. Wittmann B. et al. BRDF and gloss computation of polyurethane coatings from roughness measurements: Modelling and experimental validation // Prog Org Coat. Elsevier B.V., 2021. Vol. 156.
33. Sun L. Calculating shadowing and masking on a rough surface // Heliyon. 2023. Vol. 9, № 5. P. e15928.
34. Heitz E. Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs // Journal of Computer Graphics Techniques. 2014. Vol. 3, № 2.
35. Chermain X., Claux F., Mérillou S. Glint Rendering based on a Multiple-Scattering Patch BRDF // Computer Graphics Forum. 2019. Vol. 38, № 4. P. 27–37.
36. d'Eon E. An analytic BRDF for materials with spherical Lambertian scatterers // Computer Graphics Forum. 2021. Vol. 40, № 4. P. 153–161.
37. Cui Y. et al. Multiple-bounce Smith Microfacet BRDFs using the Invariance Principle. 2023.
38. Schlick C. An Inexpensive BRDF Model for Physically-based Rendering // Computer Graphics Forum. 1994. Vol. 13, № 3. P. 233–246.
39. Rendering Overview [Electronic resource] // Unreal Engine 4 Documentation. URL: https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Overview/ (accessed: 19.02.2024).
40. Kostas Anagnostou. How Unreal Renders a frame part 3 [Electronic resource] // Interplay of Light. 2017. URL: https://interplayoflight.wordpress.com/2017/10/25/how-unreal-renders-a-frame-part-3/ (accessed: 19.02.2024).
41. DeathreyCG. Ocean Simulation [Electronic resource] // Unreal Engine Community Tutorials. 2023.
42. Tessendorf J. Simulating Ocean Water // SIG-GRAPH'99 Course Note. 2001.
43. Kit Suman. Tranquillity of Nice [Electronic resource]. 2023. URL: https://www.kitsuman.com/nice (accessed: 22.04.2024).