

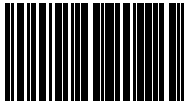
**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS**

**Разработка алгоритма отрисовки световых столбов в реальном времени на основе
соответствующих физических процессов**

Обучающийся / Student Карманович Дмитрий Евгеньевич
Факультет/институт/кластер/ Faculty/Institute/Cluster школа разработки видеоигр
Группа/Group J4221
Направление подготовки/ Subject area 09.04.03 Прикладная информатика
Образовательная программа / Educational program Технологии разработки
компьютерных игр 2022
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Магистр
Руководитель ВКР/ Thesis supervisor Загарских Александр Сергеевич, кандидат
технических наук, Университет ИТМО, школа разработки видеоигр, доцент
(квалификационная категория "доцент практики")

Обучающийся/Student


Документ подписан	
Карманович Дмитрий Евгеньевич	
28.05.2024	

(эл. подпись/ signature)

Карманович
Дмитрий
Евгеньевич

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Загарских Александр Сергеевич	
28.05.2024	

(эл. подпись/ signature)

Загарских
Александр
Сергеевич

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Карманович Дмитрий Евгеньевич

Факультет/институт/кластер/ Faculty/Institute/Cluster школа разработки видеоигр

Группа/Group J4221

Направление подготовки/ Subject area 09.04.03 Прикладная информатика

Образовательная программа / Educational program Технологии разработки компьютерных игр 2022

Язык реализации ОП / Language of the educational program Русский

Квалификация/ Degree level Магистр

Тема ВКР/ Thesis topic Разработка алгоритма отрисовки световых столбов в реальном времени на основе соответствующих физических процессов

Руководитель ВКР/ Thesis supervisor Загарских Александр Сергеевич, кандидат технических наук, Университет ИТМО, школа разработки видеоигр, доцент (квалификационная категория "доцент практики")

Характеристика темы ВКР / Description of thesis subject (topic)

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not

Тема в области прикладных исследований / Subject of applied research: да / yes

Основные вопросы, подлежащие разработке / Key issues to be analyzed

Одной из возможных задач, решаемых в компьютерной графике, является симуляция различных природных явлений, в том числе оптических. Для достоверного изображения отдельно уделяется внимание на физические процессы, лежащие в основе того или иного явления. Например, существуют методы статической отрисовки радуги: на основе волновых функций с учетом параметров среды рассеяния или диаграмм Ли – рассчитанных данных об отражении волн разной длины в том или ином направлении в каплях разного радиуса. Представлены также алгоритмы реализации и других атмосферных явлений: например, миражей, гало, глюрии, северного сияния.

Упомянутые работы наглядно иллюстрируют разнообразные подходы к реализации известных атмосферных оптических явлений. С целью обеспечения реалистичного вида реализованных явлений авторы обращают внимание на физику процессов при тех или иных явлениях. Одним из атмосферных явлений, к которому прослеживается интерес, являются световые столбы. Подобие этого явления встречается в ряде компьютерных игр, однако, данные столбы физически расположены над их источником, в то время как в природе световой столб формируется в пространстве между источником света и наблюдателем. Следовательно, в отличие от реального атмосферного явления, окружение не влияет на интенсивность столба. Таким образом, отрисовка реалистичных световых

столбов может органично дополнить сцены, например, в арктическом или научно-фантастическом сеттингах, которые являются часто встречаемыми в игровой индустрии и кинематографе.

Существует ряд исследований, где приводится описание природы световых столбов. Эта информация принята в качестве исходных данных в этой работе.

Исходя из изложенного, сформированы цель и задачи работы.

Цель: создание алгоритма динамической отрисовки световых столбов с учетом окружения и его реализация с помощью одного из доступных инструментов разработки.

В задачи работы входят следующие пункты:

- изучить процессы, лежащие в основе световых столбов;
- сформировать обобщенный алгоритм отрисовки явления;
- выбрать среду разработки;
- реализовать сформированный алгоритм, предварительно по необходимости скорректировав алгоритм с учётом специфики выбранной среды;
- оценить производительность алгоритма. В случае нескольких вариантов каких-либо этапов алгоритма провести их сравнительный анализ.

Рекомендуемые материалы:

Vincent Noel, Helene Chepfe Study of Planar Ice Crystal Orientations in Ice Clouds from Scanning Polarization Lidar Observations // Journal of Applied Meteorology. – 2005. – №5(44). – с. 653-664.

Anatoli Borovoi, Natalia Kustova Display of ice crystal flutter in atmospheric light pillars // Geophysical Research Letters. – 2009. – №4(36). – 5 с.

Anatoli Borovoi, Victor Galileiskii, Alexander Morozov, Ariel Cohen Detection of ice crystal particles preferably oriented in the atmosphere by use of the specular component of scattered light // OPTICS EXPRESS. – 2008. - №11(16). – 9 с.

Родионов С. А. Основы оптики. Конспект лекций. – СПб: СПб ГИТМО (ТУ), 2000. – 167 с.

Frank D. Luna Introduction to 3D Game Programming with DirectX 11 // Mercury Learning and Information. - 2012. - 754 с.

Tomas Akenine-Moller, Eric Haines, Naty Hoffman, Angelo Pesce, Michal Iwanicki, Sebastien Hillaire Real-Time Rendering // CRC Press. - 2018. - 1199 с.

Форма представления материалов ВКР / Format(s) of thesis materials:

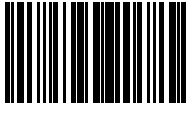
Отчет, презентация, программный код

Дата выдачи задания / Assignment issued on: 22.02.2024

Срок представления готовой ВКР / Deadline for final edition of the thesis 24.05.2024

СОГЛАСОВАНО / AGREED:

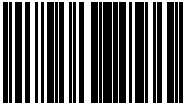
Руководитель ВКР/
Thesis supervisor

Документ подписан	
Загарских Александр Сергеевич	
01.05.2024	

(эл. подпись)

Загарских
Александр
Сергеевич


Задание принял к
исполнению/ Objectives
assumed BY

Документ подписан	
Карманович Дмитрий Евгеньевич	
13.05.2024	

(эл. подпись)

Карманович
Дмитрий
Евгеньевич

Руководитель ОП/ Head
of educational program

Документ подписан	
Карсаков Андрей Сергеевич	
22.05.2024	

(эл. подпись)

Карсаков
Андрей
Сергеевич

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
SUMMARY OF A GRADUATION THESIS**

Обучающийся / Student Карманович Дмитрий Евгеньевич
Факультет/институт/кластер/ Faculty/Institute/Cluster школа разработки видеоигр
Группа/Group J4221
Направление подготовки/ Subject area 09.04.03 Прикладная информатика
Образовательная программа / Educational program Технологии разработки компьютерных игр 2022
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Магистр
Тема ВКР/ Thesis topic Разработка алгоритма отрисовки световых столбов в реальном времени на основе соответствующих физических процессов
Руководитель ВКР/ Thesis supervisor Загарских Александр Сергеевич, кандидат технических наук, Университет ИТМО, школа разработки видеоигр, доцент (квалификационная категория "доцент практики")

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
DESCRIPTION OF THE GRADUATION THESIS**

Цель исследования / Research goal

Сформировать и реализовать алгоритм отрисовки достоверного изображения световых столбов в реальном времени

Задачи, решаемые в ВКР / Research tasks


Изучить процессы, лежащих в основе световых столбов; Сформировать обобщенный алгоритм отрисовки явления; Выбрать среду разработки; Реализовать сформированный алгоритм, предварительно уточнив расчеты с учётом специфики выбранной среды по необходимости; Оценить производительность алгоритма. В случае нескольких вариантов каких-либо этапов алгоритма провести их сравнительный анализ.

Краткая характеристика полученных результатов / Short summary of results/findings

На основе полученных в ходе изучения природы световых столбов данных был сформирован обобщенный алгоритм отрисовки световых столбов. Помимо учета глубины объектов на сцене при построении хода лучей алгоритм также учитывает следующие факторы: френелевское отражение, атмосферное поглощение, распределение кристаллов льда в среде, влияние ширины столба на его яркость. Данный алгоритм был реализован с помощью ПО DirectX 11, конечный результат был изучен с точки зрения производительности и достоверности воспроизведения явления. Кроме того, был произведен сравнительный анализ подходов к воспроизведению световых столбов. Анализ подтвердил возможность отрисовки столбов в реальном времени, отмечена визуальная схожесть результата с реальным изображением явления. Среди рассмотренных подходов к

изображению световых столбов предлагаемый алгоритм наиболее точно воспроизводит указанное явление.

Обучающийся/Student

Документ подписан	
Карманович Дмитрий Евгеньевич	
28.05.2024	

(эл. подпись/ signature)

Карманович
Дмитрий
Евгеньевич

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Загарских Александр Сергеевич	
28.05.2024	

(эл. подпись/ signature)

Загарских
Александр
Сергеевич

(Фамилия И.О./ name
and surname)

Содержание

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ.....	9
ВВЕДЕНИЕ.....	10
1 ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ОТРИСОВКИ ПРИРОДНЫХ ОПТИЧЕСКИХ ЯВЛЕНИЙ	12
2 МОДЕЛИРОВАНИЕ СВЕТОВОГО СТОЛБА	19
2.1 Физические процессы в основе световых столбов	20
2.2 Алгоритм отрисовки световых столбов.....	23
3 РЕАЛИЗАЦИЯ АЛГОРИТМА ОТРИСОВКИ СВЕТОВЫХ СТОЛБОВ	41
3.1 Реализация точечного источника света	31
3.1.1 Класс PointLight	43
3.1.2 Формирование теневых карт.....	44
3.2 Формирование световых столбов	45
3.2.1 Формирование входных данных.....	46
3.2.2 Шейдеры световых столбов	46
3.3 Визуальная оценка результата	52
4 АНАЛИЗ РЕЗУЛЬТАТОВ РАЗРАБОТКИ.....	54
4.1 Тестирование производительности алгоритма	54
4.1.1 Количество отрисовываемых точечных источников света	55
4.1.2 Разрешение cube shadow map.....	56
4.2 Сравнение полученного результата с другими подходами к отрисовке световых столбов.....	57
4.2.1 Двумерный столб.....	57
4.2.2 Volumetric-столб	59
4.2.3 Сравнение подходов.....	59
4.3 Дальнейшая работа	62
4.3.1 Оптимизация отрисовки множества столбов.....	62

4.3.2 Уточнение атмосферного поглощения	63
4.3.3 Уточнение влияния ширины столба на его яркость.....	63
ЗАКЛЮЧЕНИЕ	64
СПИСОК ЛИТЕРАТУРЫ	67
ПРИЛОЖЕНИЕ А Блок-схема алгоритма отрисовки световых столбов ...	71

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

CSM – cube shadow mapping

DPSM – dual-paraboloid shadow mapping

FPS – frames per second

HDRP – high-definition render pipeline

HLSL – high-level shader language

NDC – normalized device coordinates

ПО – программное обеспечение

СЛАУ – система линейных алгебраических уравнений

СКО – среднеквадратическое отклонение

ВВЕДЕНИЕ

Одной из возможных задач, решаемых в компьютерной графике, является симуляция различных природных явлений, в том числе оптических. Для достоверного изображения отдельно уделяется внимание на физические процессы, лежащие в основе того или иного явления. Например, существуют методы статической отрисовки радуги: на основе фазовых функций с учетом параметров среды рассеяния или диаграмм Ли – рассчитанных данных об отражении волн разной длины в том или ином направлении в каплях разного радиуса [1-3]. Представлены также алгоритмы реализации и других атмосферных явлений: например, миражей [4, 9], гало [2, 5], глории [2], северного сияния [6, 7].

Упомянутые работы наглядно иллюстрируют разнообразные подходы к реализации известных атмосферных оптических явлений. С целью обеспечения реалистичного вида реализованных явлений авторы обращают внимание на физику процессов при тех или иных явлениях. Также, практика показывает, что симуляция физических процессов зачастую приводит к увеличению времени формирования конечного изображения, вследствие чего возникает необходимость оптимизации алгоритмов или использования подхода предварительной визуализации (пре-рендеринга). Такая заинтересованность в реалистичной генерации атмосферных явлений отчасти объясняется запросом на эффектные визуальные приемы в кинематографе, фотографии, игровой индустрии.

Еще одним атмосферным явлением, к которому проявляется интерес с художественной точки зрения, являются световые столбы. Подобие этого явления встречается в ряде компьютерных игр: например, *Sea of Thieves*, *No Man's Sky*, *Dead Space*, *Genshin Impact* и других. Однако, данные столбы чаще всего представлены текстурой и физически расположены над их источником, в то время как в природе световой столб формируется в

пространстве между источником света и наблюдателем. Следовательно, в указанных примерах, в отличие от реального атмосферного явления, окружение не влияет на интенсивность столба. Таким образом, отрисовка реалистичных световых столбов может органично дополнить сцены, например, в арктическом или научно-фантастическом сеттингах, которые являются часто встречаемыми в игровой индустрии и кинематографе. Таким образом, данная работа ставит перед собой следующие цель и задачи.

Цель: разработка алгоритма динамической отрисовки световых столбов с учетом окружения и его реализация с помощью одного из доступных инструментов разработки.

В задачи работы входят следующие пункты:

- изучить процессы, лежащие в основе световых столбов;
- сформировать обобщенный алгоритм отрисовки явления;
- выбрать среду разработки;
- реализовать сформированный алгоритм, предварительно по необходимости скорректировав алгоритм с учётом специфики выбранной среды;
- оценить производительность алгоритма. В случае нескольких вариантов каких-либо этапов алгоритма провести их сравнительный анализ.

Согласно изложенным пунктам, в данной работе приводится описание явления световых столбов с физической точки зрения, поэтапно описывается алгоритм отрисовки световых столбов в реальном времени, основанный на геометрическом анализе системы источник-наблюдатель-среда с учетом физических процессов, лежащих в основе этого атмосферного явления. Кроме того, в работе приведена его непосредственная реализация в виде шейдера, производительность которой оценена по заранее составленной методике.

1 ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ОТРИСОВКИ ПРИРОДНЫХ ОПТИЧЕСКИХ ЯВЛЕНИЙ

В этой главе рассмотрены имеющиеся в открытом доступе методы отрисовки различных оптических явлений. В основу каждого из них легли соответствующие тому или иному природному явлению физические процессы.

В работе «Efficient Rendering of Atmospheric Phenomena» [2] описана отрисовка сразу нескольких явлений: радуги, гало и глории. В основе каждого из воспроизводимых явлений заложено моделирование множественных отражений в сферических частицах (Ми-рассеяние) с учетом геометрических характеристик рассеивающей среды (в контексте рассматриваемых явлений в качестве такой среды рассматриваются воздух при дожде, лед и облака разных типов: *cirrus* и *cumulus*). Авторы работы выполняют отрисовку посредством расчета для заданной оптической системы фазовой функции – функции, описывающей угловое распределение рассеянной частицей энергии. В рамках работы представлено угловое распределение рассеянного света для разных сред, наглядно иллюстрирующее условия наблюдения явлений.

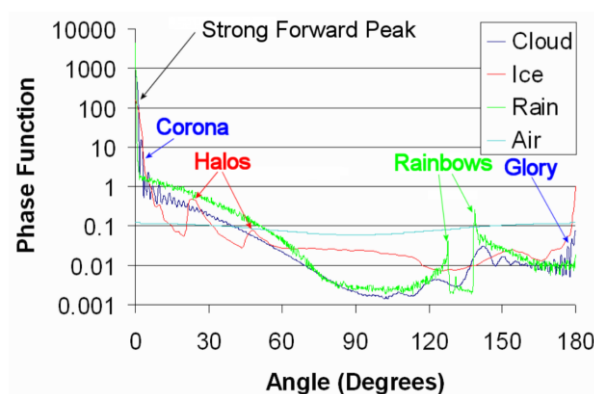


Рисунок 1 – Фазовые функции для разных оптических сред для зеленого света

Предлагаемый алгоритм просчитывает отражения (фазовые функции) в объеме системы источник-среда-наблюдатель для разных волн и таким образом, исходя из параметров системы, отрисовываются те или иные оптические явления. Ниже приведено пошаговое описание предлагаемого алгоритма:

- анализ атмосферного рассеяния. На этом этапе формируется небо;
- volume rendering: анализ объема среды, расчет рассеяния, атмосферное поглощение излучения;
- учет вклада освещения от неба и солнца.

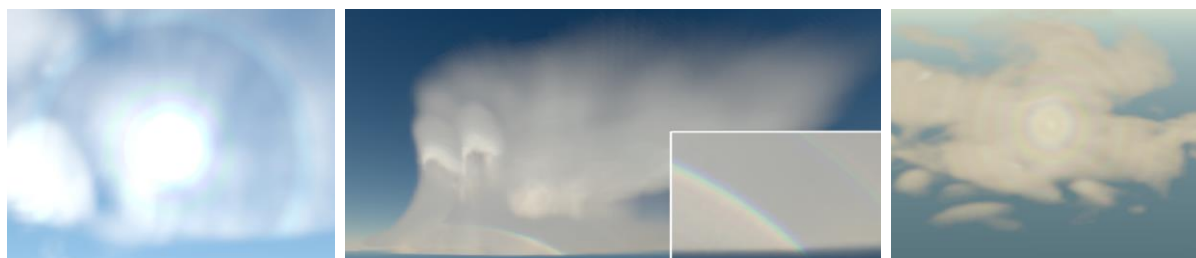


Рисунок 2 – Гало, радуга и gloria, полученные методом объемного расчета Ми-рассеяния

Для выполнения расчетов использовались следующие средства и параметры: процессор Intel Xeon, видеокарта Nvidia GeForce FX 6800, семплирование объема – 256, разрешение изображения – 768x768 пикселей. При этом отрисовка изображений занимает до четырех секунд.

Схожий подход приведен в работе «Physically-Based Simulation of Rainbows» [3]. В ней, как и в предыдущей работе, представлена отрисовка радуги посредством вычисления волновых функций, однако в данном случае реализована трассировка лучей через частицы (капли): волновой фронт разбивается по сетке, где в каждой вершине находится рассчитываемый далее по 33 длинам волн луч. На выходе получается массив волновых функций, готовый к дальнейшей обработке.

Примечательно, что алгоритм кроме хода лучей в капле рассчитывает интерференцию, что делает конечный результат более достоверным.

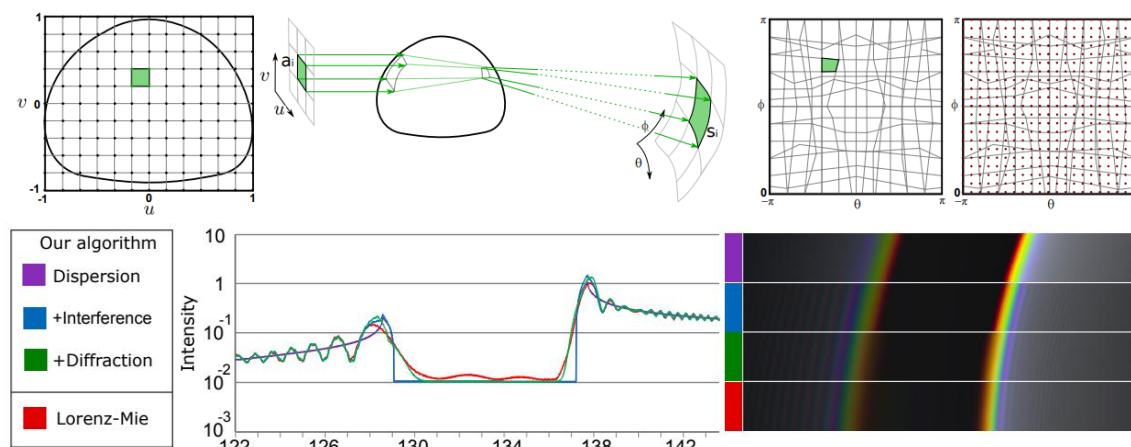


Рисунок 3 – Дискретизация расчета хода лучей в капле (сверху) и конечный результат с учетом интерференции (снизу)

Приведенный в работе алгоритм в противовес своей физической точности имеет низкую производительность: для отрисовки изображения с анализом хода лучей по 33 длинам волн требуется около 350 минут.

В рассмотренных выше работах были представлены алгоритмы отрисовки радуги на основе расчета фазовых функций. Однако, существует другой метод отрисовки радуги, представленный Nvidia [4]. В рамках приведенного алгоритма предлагается использовать диаграммы Ли – по сути, это данные об отраженном в капле свете, зашифрованные в текстуру, где текстурные координаты – радиус капли и угол рассеяния, а цвет пикселя на их пересечении соответствует рассеиваемому в данном направлении свету. Хоть алгоритм и предполагает использование Quad-шейдера, что подразумевает расчеты в каждом из пикселей изображения, его производительность достаточна для выполнения отрисовки радуги в реальном времени.

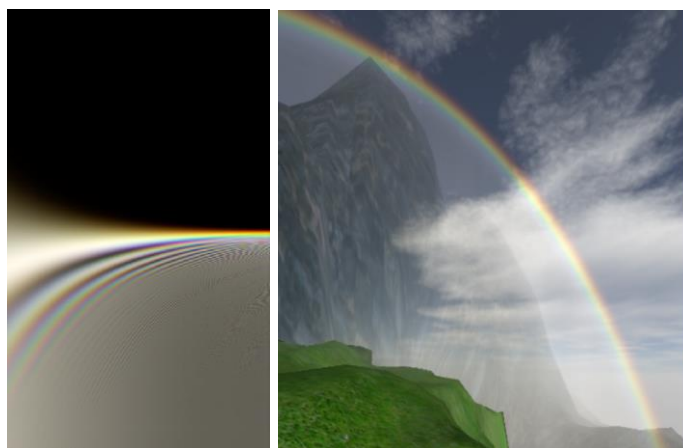


Рисунок 4 – Диаграмма Ли (слева) и полученное изображение (справа)

Миражи также являются объектом изучения в контексте физически достоверных алгоритмов отрисовки природных явлений. В работе «Rendering Ghost Ships and Other Phenomena in Arctic Atmospheres» приведен алгоритм, симулирующий миражи разных видов [5]. В силу того, что миражи возникают из-за дугообразного распространения света в среде из-за ее изменяющегося показателя преломления, отрисовка этого явления не может быть осуществлена методом трассировки лучей, поэтому авторы предлагают алгоритм на основе принципа Ферма – фундаментального постулата геометрической оптики, согласно которому свет распространяется по траектории наименьшей оптической длины пути, описываемой уравнением

$$L = nl, \quad (1)$$

где n – показатель преломления среды, l – геометрическая длина пути. Вместе с созданием сцены с достоверными геометрическими размерами объектов (в том числе Солнца и Земли) и использованием соответствующих метеорологических данных алгоритм численно решает дифференциальное уравнение на основе принципа Ферма, которое в конечном итоге позволяет получить изображение в рассматриваемом направлении на сцене. По данным, приведенным в работе, отрисовка

изображения с разрешением 400x200 пикселей на основе процессора Intel Pentium IV занимает 2-2,5 минуты.

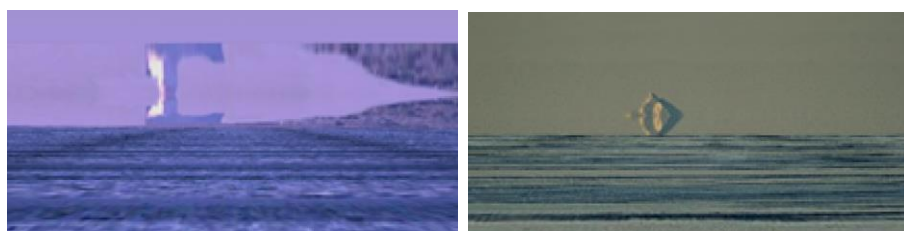


Рисунок 5 – Примеры полученных изображений миражей с помощью алгоритма на основе принципа Ферма

Другим примером физически достоверного подхода отрисовки природных оптических явлений является работа «Physically Based Rendering of Ice Crystal Halos», где представлен алгоритм отрисовки гало [6]. Идея алгоритма заключается в прямом расчете хода лучей через кристаллы льда разных топологий посредством оценки функции распределения двунаправленного отражения и фазовой функции, которые несут в себе информацию об оптических свойствах среды. По результатам измерений производительности работы алгоритма с использованием процессора AMD Ryzen 5 1600 отрисовка изображения явления с разрешением 1280x720 занимает от 1,5 до 4 часов в зависимости от настроек системы.

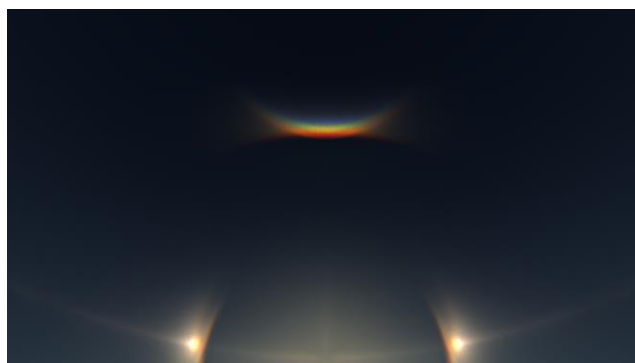


Рисунок 6 – Полученное с помощью объемного расчета рассеяния изображение гало

Существует также метод отрисовки северного сияния на основе физики явления: такой алгоритм представлен в работе «Interactive Volume Rendering Aurora on the GPU» [7]. Для изображения явления выполнено моделирование слоев атмосферы Земли, при котором посредством трассировки лучей просчитывается ход луча сквозь эти слои и вычисляется интенсивность свечения в данном направлении, основываясь на зависимости энергии излучения северного сияния от высоты взаимодействия со слоем атмосферы. В качестве данных о форме сияния принимаются профили северного сияния – текстуры с кривыми, отображающими форму атмосферного явления. Алгоритм показал довольно высокую производительность: с использованием видеокарты ATI Radeon HD 4830 было получено 23 FPS, с помощью Nvidia GeForce 8800M GTS – 38 FPS, а в случае Nvidia GeForce GTX 280 – 60 FPS. Таким образом, в рамках данной работы была обеспечена возможность использования алгоритма в реальном времени. Следует отметить, что в приведенной работе изображения явления статичны, так как профили сияния не изменяются.

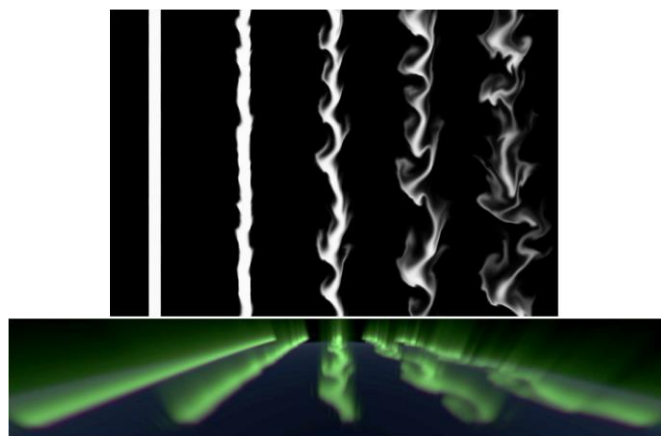


Рисунок 7 – Полученные изображения северного сияния (снизу) и соответствующие им профили (сверху)

Для отрисовки северного сияния также уделяется внимание изучению формы явления, что представлено в работе «Modeling of aurora borealis

using the observed data» [8]. Основное внимание в работе уделено формированию профилей сияния в зависимости от геопозиции наблюдателя на основе характеристик магнитосферы Земли с последующей отрисовкой сияния. В результате вычисление профилей сияния занимает порядка 7,5 секунд при использовании Intel Core 2 Quad, а для отрисовки сияния в зависимости от качества изображения требуется 10-50 минут. Таким образом, в данном случае использование алгоритма в реальном времени нецелесообразно.

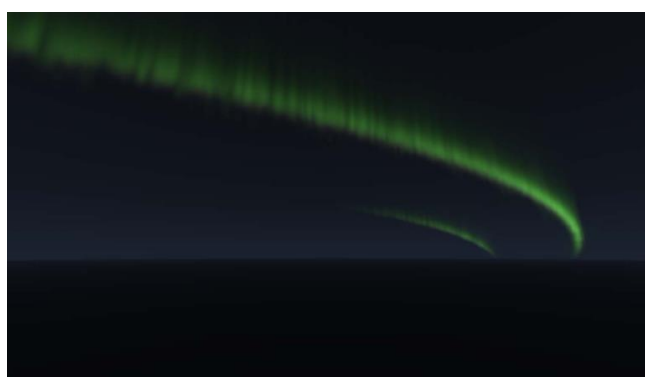


Рисунок 8 – Полученное изображение северного сияния по вычисленному профилю

Работа «Visual simulation of heat shimmering and mirage» представляет метод воспроизведения миражей и искажения при нагретом воздухе на основе трассировки лучей и метода решеточных уравнений – численного метода гидродинамики, при котором жидкость (в данном случае – разогретый воздух) рассматривается как система некоторого количества отдельных частиц, которые двигаются и взаимодействуют друг с другом [9]. По данным, приведенным в исследовании, при использовании видеокарты Nvidia GeForce 6800 Ultra удалось добиться 5,7 FPS, что в данном случае исключает возможность использования алгоритма в реальном времени.



Рисунок 9 – демонстрация результата работы алгоритма (слева – виртуальная сцена без эффектов, справа – сцена после применения алгоритма)

Рассмотренные работы наглядно демонстрируют разнообразие изучаемых явлений, а также подходов к их физически достоверному воспроизведению. Часть из данных работ акцентирует свое внимание на детализации физических процессов, зачастую жертвуя производительностью алгоритма, другие же представляют более упрощенные модели явлений при достаточной производительности с возможностью реализовать отрисовку необходимых изображений в реальном времени в том или ином виде. Тем не менее, так или иначе в основе работ лежит воспроизведение физические процессов, присущих конкретному явлению – во многом это позволяет получить удовлетворительный результат.

Таким образом, при поставленной цели по реализации алгоритма отрисовки световых столбов в реальном времени необходимо разобраться в природе данного явления и создать физико-математическую модель процессов. Данный этап работы подробно описан в следующей главе.

2 МОДЕЛИРОВАНИЕ СВЕТОВОГО СТОЛБА

В данной главе рассмотрена природа световых столбов. На основе полученной информации создан и описан поэтапный математический алгоритм отрисовки явления.

2.1 Физические процессы в основе световых столбов

По сути световой столб - мнимое изображение источника света, получаемое при отражении света от ледяных частиц в атмосфере. Явление наблюдается в случае горизонтальной ориентации плоских гексагональных частиц льда в пространстве, которые в такой конфигурации с точки зрения наблюдателя выполняют роль зеркала. Схематическое изображение формирования световых столбов приведено на рисунке 10.

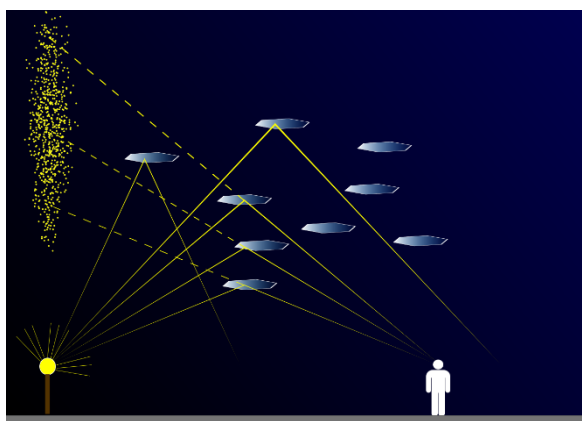


Рисунок 10 – Схематическое описание механизма формирования светового столба [10]

В свою очередь содержание ледяных частиц той или иной конфигурации зависит от метеорологических условий: влажность воздуха и температура окружающей среды напрямую влияют на процесс кристаллизации водяных частиц. Данные зависимости были изучены еще в 1930-х годах японским исследователем Укичиро Накайей [11]. На основе его работ был проведен ряд исследований, расширивших данные о формировании, морфологии, динамике различных частиц льда.

В контексте моделирования световых столбов следует обратить внимание на исследования, в которых приводится информация о динамике плоских ледяных частиц. В работе «Study of Ice Crystal Orientation in Cirrus

Clouds Based on Satellite Polarized Radiance Measurements» представлены результаты спутниковых исследований перьевых (cirrus) облаков, которые образуются в верхних слоях атмосферы на высоте более пяти тысяч метров и сформированы в том числе из плоских гексагональных частиц льда [12]. Авторы утверждают, что около 20% этих плоских частиц имеют горизонтальную пространственную ориентацию. Также, авторы исследовали пространственное отклонение пластин от горизонтального положения: по их данным в 80% рассмотренных облаков (всего было изучено 31 облако перьевого типа) отклонение от горизонтального положения составило менее пяти градусов. Кроме того, в работе приводится анализ распределения отклонения по двух гипотетическим моделям: распределения Гаусса и квадратического (хи-квадрат) распределения с максимумом в значении аргумента 0 (отсутствует отклонение, плоский кристалл ориентирован строго горизонтально). Приведенные данные о характере отклонения плоских ледяных частиц следует учесть при моделировании явления световых столбов.

Также, приведены исследования динамики плоских кристаллов льда и физики световых столбов с учетом полученных данных о позиционировании частиц льда [13, 14]. Кроме того, исследуется интенсивность светового столба в зависимости не только от угла отклонения частиц льда, но и от азимутального и зенитного углов наблюдения этого явления. В ходе исследования был определен максимальный угол отклонения от горизонтального положения: он составил пять градусов, а также рассмотрены два варианта распределения отклонения частиц льда: равномерное и нормальное.

В дополнение к приведенным данным следует упомянуть формулы Френеля, описывающие распределение энергии между отраженным и преломленным лучами при падении света на плоскую границу двух сред (рисунок 11) [15]. Рассматриваются перпендикулярная и параллельная

относительно плоскости падения составляющие вектора электрической напряженности \vec{E} .

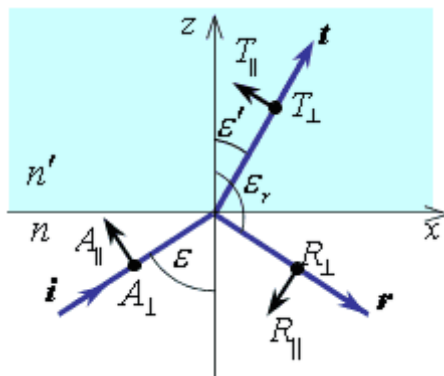


Рисунок 11 – Отражение и преломление света на границе двух сред [15]

Для падающей волны справедливы следующие определения коэффициентов отражения ρ_{\parallel} и ρ_{\perp} [15]:

$$\begin{cases} \rho_{\parallel} = \frac{tg^2(\varepsilon - \varepsilon')}{tg^2(\varepsilon + \varepsilon')} \\ \rho_{\perp} = \frac{\sin^2(\varepsilon - \varepsilon')}{\sin^2(\varepsilon + \varepsilon')} \end{cases}, \quad (2)$$

где ε – угол падения, ε' – угол преломления (см. рисунок 11).

Коэффициент отражения ρ обозначает долю энергии, которая отразилась от границы раздела сред при падении на нее света. Фактически коэффициент отражения зависит только от угла падения, так как угол преломления связан с углом падения законом Снеллиуса [15]:

$$n \sin \varepsilon = n' \sin \varepsilon', \quad (3)$$

где n и n' – показатели преломления двух сред.

В компьютерной графике для вычисления коэффициента отражения используется аппроксимация Шлика [16]. Как и в случае формул Френеля коэффициент отражения зависит от угла падения:

$$\rho(\varepsilon) = \rho_0 + (1 - \rho_0)(1 - \cos \varepsilon)^5, \quad (4)$$

где $\rho_0 = \left(\frac{n-n'}{n+n'}\right)^2$ – коэффициент отражения в случае нормального падения.

Кроме того, перед началом работы над алгоритмом следует также упомянуть и учитывать далее атмосферное поглощение излучения, которое можно оценить по закону Ламберта-Бугера-Бера:

$$a(l) = \exp(-kl), \quad (5)$$

где k – эмпирически заданный показатель поглощения, а l – толщина среды.

Также, следует учитывать зависимость яркости столба от его ширины: чем больше ширина, тем количественно больше частиц отражают свет от источника и тем ярче столб в рассматриваемом направлении.

Приведенных данных достаточно, чтобы выполнить моделирование световых столбов. Алгоритм моделирования изложен далее.

2.2 Алгоритм отрисовки световых столбов

Цель описываемого алгоритма: формирование в реальном времени изображения световых столбов, соответствующего текущему кадру, и его дальнейшее совмещение с кадром.

В качестве исходных данных для работы алгоритма необходимы: transform-матрица зрителя T_V , проекционная матрица камеры M_{proj} , view-матрица камеры M_{view} , данные об источнике света (его цвет и мировая позиция $S(S_x, S_y, S_z)$). Далее приведены этапы работы предлагаемого алгоритма. Расчет производится для каждого пикселя кадра.

Шаг 1: получение вектора взгляда \vec{v} , соответствующего данному пикселю.

Вычисление пиксельных координат $A_p(x_p, y_p)$ некоторой вершины в мировом пространстве $A(x, y, z)$ выполняется следующим образом:

1. Вычисляются NDC-координаты (Normalized Device Coordinates) вершины:

$$A_{NDC} = A \cdot T_V \cdot M_{view} \cdot M_{proj} \quad (6)$$

2. NDC-координаты переводятся в пиксельные с учетом ширины (width) и высоты (height) дисплея:

$$\begin{cases} x_p = (x_{NDC} + 1) \frac{width}{2} \\ y_p = (y_{NDC} + 1) \frac{height}{2} \end{cases} \quad (7)$$

Соответственно, для вычисления мировых координат точки, которой соответствует рассматриваемый пиксель, нужно выполнить обратные преобразования:

3. Пиксельные координаты переводятся в NDC-координаты:

$$\begin{cases} x_{NDC} = \frac{2x_p}{width} - 1 \\ y_{NDC} = \frac{2y_p}{height} - 1 \end{cases} \quad (8)$$

4. По полученным NDC-координатам вычисляется мировая позиция соответствующей им точки:

$$A = A_{NDC} \cdot M_{proj}^{-1} \cdot M_{view}^{-1} \cdot T_V^{-1} \quad (9)$$

5. Для оптимизации вычислений вектор взгляда определяется как разность между мировыми позициями точки и зрителя в ортонормированном виде:

$$\overrightarrow{VA}(A_x - V_x, A_y - V_y, A_z - V_z) \quad (10)$$

$$\vec{v} = \frac{\overrightarrow{VA}}{|\overrightarrow{VA}|} \quad (11)$$

Шаг 2: расчет минимального угла отклонения δ .

Согласно геометрическим построениям, изображающим ход лучей в световых столбах, точка отражения R_0 , в которой наблюдается минимальный угол отклонения, находится следующим образом:

1. Находится точка O - точка пересечения луча, испущенного из положения наблюдателя в направлении вектора зрения \vec{v} , с высотой, на которой располагается источник света ($z = S_z$).
2. Искомая точка отражения является точкой пересечения луча, испущенного из положения наблюдателя в направлении вектора зрения \vec{v} , с плоскостью, перпендикулярной отрезку SO и пересекающей его по середине.

Для иллюстрации этого шага на рисунке 12 приведена боковая проекция хода лучей, где S – положение источника, V – положение зрителя, O – совпадающая по высоте с источником точка на линии взгляда:

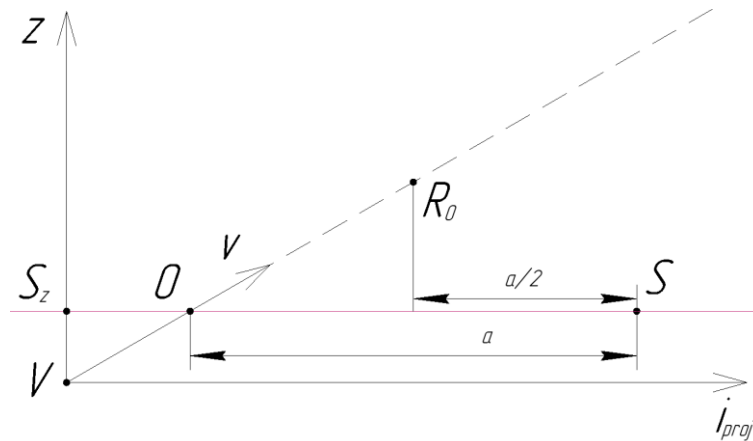


Рисунок 12 – Боковая проекция хода лучей

Аналогичный результат получается при решении данной задачи с помощью функционального анализа, однако этот метод в алгоритме не используется в силу его меньшей рациональности в сравнении с геометрическим подходом.

В аналитическом виде решение задачи нахождения точки наименьшего отклонения кристалла от горизонтального положения выглядит следующим образом:

1. Определяется параметр $shift$ такой, что:

$$S_z = V_z + v_z \cdot shift \quad (12)$$

$$shift = \frac{S_z - V_z}{v_z} \quad (13)$$

2. Определяются координаты точки R_0 по половине расстояния между точками O и S :

$$\begin{cases} R_x = \frac{S_x + V_x + v_x \cdot shift}{2} \\ R_y = \frac{S_y + V_y + v_y \cdot shift}{2} \\ R_z = V_z + v_z \frac{R_x - V_x}{v_x} \end{cases} \quad (14)$$

3. Определяется вектор отражения $\vec{r}(r_x; r_y; r_z)$, исходя из координат точек R_0 и S :

$$\begin{cases} r_x = S_x - R_{0x} \\ r_y = S_y - R_{0y} \\ r_z = S_z - R_{0z} \end{cases} \quad (15)$$

4. Определяется вектор нормали $\vec{n}(n_x; n_y; n_z)$ к поверхности кристалла. Так как в случае отражения углы падения и отражения равны, нормаль можно найти с помощью биссектрисы угла, образованного векторами \vec{r} и $-\vec{v}$. Биссектриса угла, образованного двумя некоторыми векторами определяется как векторная сумма соответствующих ортонормированных векторов:

$$\begin{cases} n_x = \frac{v_x}{\sqrt{v_x^2 + v_y^2 + v_z^2}} - \frac{r_x}{\sqrt{r_x^2 + r_y^2 + r_z^2}} \\ n_y = \frac{v_y}{\sqrt{v_x^2 + v_y^2 + v_z^2}} - \frac{r_y}{\sqrt{r_x^2 + r_y^2 + r_z^2}} \\ n_z = \frac{v_z}{\sqrt{v_x^2 + v_y^2 + v_z^2}} - \frac{r_z}{\sqrt{r_x^2 + r_y^2 + r_z^2}} \end{cases} \quad (16)$$

5. Определяется минимальный угол отклонения δ_{min} кристалла от горизонтального положения. По сути это угол между найденной ранее нормалью и осью Oz:

$$\delta_{min} = \arccos\left(\frac{n_z}{\sqrt{n_x^2 + n_y^2 + n_z^2}}\right) \quad (17)$$

Исходя из полученного угла, определяются дальнейшие действия:

- $\delta_{min} > 5^\circ$: в рассматриваемом направлении отсутствуют горизонтально ориентированные кристаллы льда, вносящие вклад в образование светового столба от данного источника света. Дальнейшие вычисления не требуются, рассматриваемый пиксель считается «темным»;
- $\delta_{min} \leq 5^\circ$: в рассматриваемом направлении присутствуют горизонтально ориентированные кристаллы льда, вносящие вклад в образование светового столба от данного источника света, при этом их отклонение от горизонтального положения находится на отрезке $[\delta_{min}; 5^\circ]$. Требуется определение интенсивности столба в этом направлении наблюдения.

Шаг 3: определение граничных характеристик светового столба.

Как уже было указано ранее, плоские кристаллы льда в процессе падения отклоняются от горизонтального положения на угол до пяти градусов. Учитывая это обстоятельство, необходимо вычислить границы светового столба, на которых наблюдаются кристаллы с максимальным смещением от горизонтального положения. Эта информация будет необходима далее для следующих расчетов:

1. Расчет доли частиц, участвующих в образовании столба в данном направлении. Частицы, имеющие максимальное отклонение от горизонтального положения, будут наиболее удаленными от источника и таким образом они создают некоторый объем, в котором формируется столб. В лучшем случае в рассматриваемом направлении внутри этого объема не будет посторонних объектов, тогда будет учтен вклад от каждого кристалла в этом направлении. Если же внутри этого объема будет находиться некоторый объект, то собой он будет заслонять часть участвующих в образовании светового столба кристаллов и интенсивность столба в этом направлении сократится. Именно для учета таких случаев необходимо вычислить пространственные границы светового столба.
2. Расчет интегрального коэффициента отражения в данном направлении. Коэффициент отражения зависит от угла падения: чем ближе падающий луч к нормали, тем меньшая часть исходной энергии отражается от границы раздела сред. В то же время чем дальше от наблюдателя точка, в которой свет отражается от источника в зрителя, тем меньше угол падения и тем меньше коэффициент отражения. Для вычисления интегрального коэффициента необходимо вычислить углы падения на границе объема светового столба. Препятствия также вносят вклад в интегральный коэффициент отражения, что учитывается аналогичным предыдущему пункту образом.

Задача, решаемая в этом пункте, сводится к нахождению расстояний до границ объема светового столба в рассматриваемом направлении. Для этого необходимо определить два единичных минус-вектора нормалей \vec{n}'

на границах объема ($\vec{n}' = -\vec{n}$). Условие граничного отражения обеспечивается следующим равенством:

$$\vec{n}'^k = 5^\circ, \Rightarrow \frac{n'_z}{|n|} = \frac{n'_z}{1} = n'_z = \cos 5^\circ \quad (18)$$

ким образом, определена координата z искомых нормалей. Для нахождения координат x и y единичных нормалей рассмотрим треугольник VRS (рисунок 13), где V – положение зрителя, R – точка отражения луча, S – положение источника света.

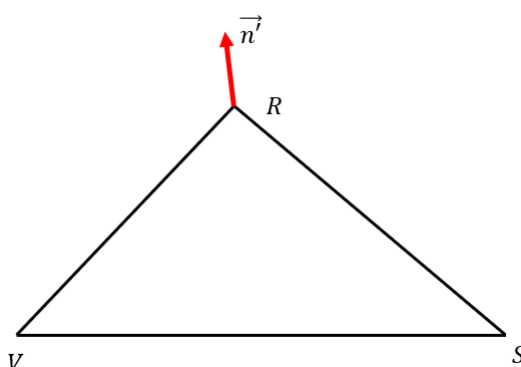


Рисунок 13 – Источник света, зритель, отражение луча и нормаль к отражающей поверхности

Точки V , R и S лежат в одной плоскости, следовательно, векторы, сонаправленные сторонам этого треугольника, компланарны.

Также, векторы \overrightarrow{SR} и \overrightarrow{RV} сонаправлены падающему и отраженному лучам соответственно. Тогда вектор нормали \vec{n}' , векторы \overrightarrow{SR} и \overrightarrow{RV} компланарны, так как луч падающий, луч отраженный и нормаль к отражающей поверхности лежат в одной плоскости [15].

Таким образом, вектор нормали \vec{n}' , векторы \overrightarrow{VR} и \overrightarrow{VS} компланарны. Появляется возможность однозначно определить оставшиеся координаты векторов нормалей, сформировав при известных \overrightarrow{VR} и положении наблюдателя уравнение плоскости следующего вида:

$$Ax + By + Cz + D = 0, \quad (19)$$

где A, B, C и D – коэффициенты плоскости.

Для получения уравнения плоскости по двум неколлинеарным векторам \vec{v} и \vec{w} и точке $M(x_0, y_0, z_0)$ необходимо использовать следующее выражение:

$$\begin{vmatrix} x - x_0 & v_x & w_x \\ y - y_0 & v_y & w_y \\ z - z_0 & v_z & w_z \end{vmatrix} = 0 \quad (20)$$

Используя положение наблюдателя и векторы \overrightarrow{VR} и \vec{n} , получим следующие значения коэффициентов плоскости:

$$\begin{cases} A = VR_y n_z - VR_z n_y \\ B = -VR_x n_z + VR_z n_x \\ C = VR_x n_y - VR_y n_x \\ D = -V_x A - V_y B - V_z C \end{cases} \quad (21)$$

Добавив к этому выражению условие ортонормированности векторов нормалей, получим следующую систему уравнений с двумя неизвестными:

$$\begin{cases} An'_x + Bn'_y + Cn'_z + D = 0 \\ \sqrt{n'^2_x + n'^2_y + n'^2_z} = 1 \end{cases} \quad (22)$$

Решив данную систему, получим следующее:

$$\begin{cases} n'_x = \frac{-B \cos 5^\circ - C n'_z}{A} \\ n'_y = \frac{-2BC \cos 5^\circ + \sqrt{(2BC \cos 5^\circ)^2 - 4(A^2 + C^2)(B^2 \cos^2 5^\circ - A^2 \cos^2 5^\circ)}}{2(A^2 + C^2)} \end{cases} \quad (23)$$

или

$$\begin{cases} n'_x = \frac{-B \cos 5^\circ - C n'_z}{A} \\ n'_y = \frac{-2BC \cos 5^\circ - \sqrt{(2BC \cos 5^\circ)^2 - 4(A^2 + C^2)(B^2 \cos^2 5^\circ - A^2 \cos^2 5^\circ)}}{2(A^2 + C^2)} \end{cases} \quad (24)$$

Таким образом, вычислены два вектора нормалей. Для краткости эти векторы записаны в виде $\vec{n}'_1(n'_{1x}, n'_{1y}, n'_{1z})$ и $\vec{n}'_2(n'_{2x}, n'_{2y}, n'_{2z})$.

Теперь необходимо найти единичные векторы \vec{r}_1 и \vec{r}_2 падающих на кристаллы лучей, для которых будут справедливы найденные нормали. Для этого можно воспользоваться выражением для координат вектора нормали из предыдущего пункта:

$$\begin{cases} r_{1x} = \frac{v_x}{|v|} - n'_{1x} \\ r_{1y} = \frac{v_y}{|v|} - n'_{1y} \\ r_{1z} = \frac{v_z}{|v|} - n'_{1z} \end{cases} \quad \text{или} \quad \begin{cases} r_{2x} = \frac{v_x}{|v|} - n'_{2x} \\ r_{2y} = \frac{v_y}{|v|} - n'_{2y} \\ r_{2z} = \frac{v_z}{|v|} - n'_{2z} \end{cases} \quad (25)$$

На данном этапе есть возможность получить точки отражения лучей для каждого из двух случаев. По точкам положений зрителя и источника света, а также по векторам падающего и отраженного лучей формируются две функции от трех переменных, графики которых представляют из себя прямые. Графики будут параллельны соответствующим им векторам, а также будут проходить через соответствующие им точки. Каноническое уравнение прямой, проходящей через некоторую точку $O(O_x, O_y, O_z)$ и параллельной некоторому вектору $\vec{a}(a_x, a_y, a_z)$, будет иметь вид:

$$\frac{x - O_x}{a_x} = \frac{y - O_y}{a_y} = \frac{z - O_z}{a_z} \quad (26)$$

Тогда уравнения прямых, соответствующих падающему и отраженному лучам, будут иметь вид:

$$\frac{x - V_x}{v_x} = \frac{y - V_y}{v_y} = \frac{z - V_z}{v_z} \quad \text{и} \quad \frac{x - S_x}{r_x} = \frac{y - S_y}{r_y} = \frac{z - S_z}{r_z} \quad (27)$$

Так как эти прямые пересекаются, они будут иметь общую точку R_0 – она и является искомой точкой, в которой происходит отражение. Ниже канонические уравнения прямых приведены для точки R_0 и переведены в параметрический вид, где параметры t и k отображают количество длин векторов, которые необходимо отложить от точек V и S соответственно до точки отражения:

$$\begin{cases} R_{0x} = V_x + tv_x \\ R_{0y} = V_y + tv_y \\ R_{0z} = V_z + tv_z \end{cases} \quad \text{и} \quad \begin{cases} R_{0x} = S_x + kr_x \\ R_{0y} = S_y + kr_y \\ R_{0z} = S_z + kr_z \end{cases} \quad (28)$$

Для нахождения точки отражения достаточно найти одну из неизвестных. Итак, приравняв соответствующие выражения, получим следующую систему линейных алгебраических уравнений (СЛАУ) с двумя неизвестными:

$$\begin{cases} V_x + tv_x = S_x + kr_x \\ V_y + tv_y = S_y + kr_y \\ V_z + tv_z = S_z + kr_z \end{cases} \quad (29)$$

Далее выразим параметр t относительно первого уравнения СЛАУ:

$$t = \frac{S_x + kr_x - V_x}{v_x} \quad (30)$$

На данном этапе выражение выше можно подставить как во второе уравнение СЛАУ, так и в третье. Однако, при подстановке следует учесть, чтобы выбранное и подставляемое уравнения не были линейно зависимыми. Алгоритм действий в этом случае можно представить следующим образом: если $\frac{V_y}{V_z} = \frac{v_y}{v_z}$ и $\frac{S_y}{S_z} = \frac{r_y}{r_z}$, то выполняется подстановка полученного выражения во второе уравнение СЛАУ. В ином случае подстановка выполняется в третье уравнение.

Таким образом, если выражение подставлено во второе уравнение, параметр k будет определен следующим образом:

$$k = \frac{S_y - V_y + \frac{v_y}{v_x}(V_x - S_x)}{\frac{r_x v_y}{v_x} - r_y} \quad (31)$$

Если же выражение было подставлено в третье уравнение СЛАУ, то получим следующее:

$$k = \frac{S_z - V_z + \frac{v_z}{v_x}(V_x - S_x)}{\frac{r_x v_z}{v_x} - r_z} \quad (32)$$

В результате точка отражения R_0 определяется следующим образом:

$$\begin{cases} R_{0x} = S_x + kr_x \\ R_{0y} = S_y + kr_y \\ R_{0z} = S_z + kr_z \end{cases} \quad (33)$$

После нахождения ближней (R_{0n}) и дальней (R_{0f}) точек отражения для двух нормалей можно определить первичные границы расчетов для дальнейших шагов. Для этого сравниваются расстояния от наблюдателя до указанных точек, расстояние до точки R_{min} и соответствующее значение z_0 в z -буфере. Таким образом, возможны четыре случая:

1. $z_0 \leq VR_{0n}$: объект находится перед световым столбом. В таком случае в данном направлении интенсивность светового столба равно нулю.
2. $VR_{0n} < z_0 \leq VR_{min}$: объект находится в первой половине объема светового столба в данном направлении – до точки R_{min} . В этом случае расчет интенсивности проводится для кристаллов от ближней границы до объекта.
3. $VR_{min} < z_0 \leq VR_{0f}$: объект находится во второй половине объема светового столба в данном направлении. В этом случае расчет интенсивности проводится для кристаллов от ближней границы до точки минимального отклонения кристаллов от горизонтального положения R_{min} и от точки R_{min} до объекта в данном направлении.
4. $VR_{0f} \leq z_0$: объект находится за световым столбом. В этом случае интенсивность столба в этом направлении рассчитывается по всему его объему.

Шаг 4: определение доли кристаллов, вносящих вклад в образование столба.

На данном этапе в границах, обозначенных в предыдущем пункте, выполняется расчет доли кристаллов льда, участвующих в образовании

светового столба, с учетом буфера глубины соответствующего источника света. Буфер глубины источника света необходим для учета препятствий для света между источником и точкой отражения, так как они влияют на итоговую интенсивность столба. На рисунке 14 схематически изображены описанные случаи. Зритель и источник света расположены в точках пространства V и S соответственно, синими линиями изображены границы светового столба.

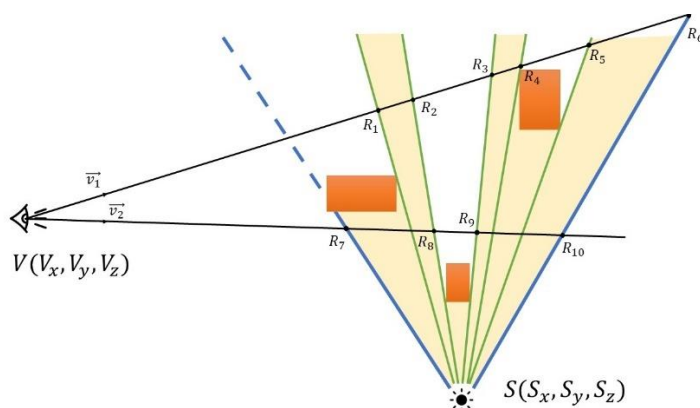


Рисунок 14 – Влияние объектов на интенсивность светового столба

Задачей данного шага является определение с некоторой погрешностью точек перехода между неосвещенным и освещенным пространствами (точки R_1, R_2, \dots, R_{10} на Рисунке 5). Для этого с некоторым шагом по направлению взгляда проверяется освещенность точки внутри объема столба посредством буфера глубины соответствующего точечного источника света.

Определение искомых точек осуществляется по следующему алгоритму:

1. Проверяется освещенность точки R_{0n} . Если расстояние от источника света до точки R_{0n} меньше, чем соответствующее значение из буфера глубины, то точка считается освещенной и

записывается в список граничных точек. Иначе – перед точкой есть препятствие и источник ее не освещает.

2. Происходит аналогичная предыдущему пункту оценка точек вдоль направления взгляда с некоторым шагом. Если состояние освещенности в сравнении с прошлой точкой меняется, то точка записывается в список граничных точек.
3. В случае, если на точку минимального отклонения кристаллов R_0 приходится освещенный участок пространства, то эта точка добавляется в список дважды между точками, ограничивающими этот освещенный участок.
4. Поиск точек завершается, если очередная точка находится дальше выявленной ранее дальней границы.

Таким образом, по завершению поиска имеется список из четного количества точек. Например, для вектора взгляда \vec{v}_1 на Рисунке 5 будет сформирован список из точек $R_1, R_2, R_3, R_4, R_5, R_6$, между парами точек которого и будет считаться доля участвующих в образовании светового столба кристаллов.

Исходя из рассмотренных исследований динамики плоских кристаллов льда, было принято, что отклонение кристаллов от горизонтального положения подчиняется распределению Гаусса. Для дальнейшего анализа используется нормированная функция Гаусса, так как в случае участия в образовании светового столба горизонтально ориентированных кристаллов со всеми возможными углами отклонения $\delta \in [0; 5^\circ]$ будет наблюдаться максимум интенсивности столба, а интегральное значение гауссовой функции на этом интервале будет равно единице:

$$N(\delta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-0,5\left(\frac{\delta-a}{\sigma}\right)^2} \quad (34)$$

Необходимо определить параметры распределения Гаусса:

1. Математическое ожидание a – наиболее вероятное отклонение кристалла от горизонтального положения, $a = 0$.
2. Среднеквадратическое отклонение (СКО) σ . Согласно рассмотренному выше исследованию горизонтально позиционированные плоские кристаллы льда не отклоняются на угол более пяти градусов. В то же время для распределения Гаусса справедливо «правило трех сигм»: почти все (99,7%) значения, которые только может принять случайная величина, лежат на отрезке $[a - 3\sigma; a + 3\sigma]$. Пренебрегая оставшимися 0,3% значений, примем:

$$3\sigma = 5, \Rightarrow \sigma = \frac{5}{3}.$$

Таким образом, распределение Гаусса для отклонения плоских кристаллов льда от горизонтального положения описывается следующим законом:

$$N(\delta) = \frac{3}{5\sqrt{2\pi}} e^{-0,5\left(\frac{3\delta}{5}\right)^2} \quad (35)$$

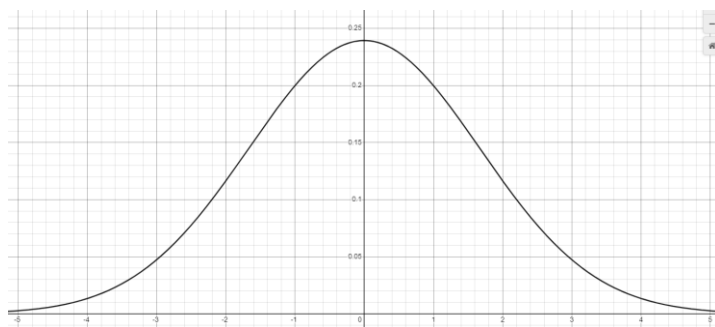


Рисунок 15 – Нормированное распределение Гаусса для плоских кристаллов льда

Следует снова обратить внимание на значения, принимаемые параметром δ на данном этапе: $\delta \in [0; 5^\circ]$. Таким образом, часть значений

функции $N(\delta)$ при $\delta < 0$ не учитывается, чем нарушается нормировка. Так как для расчета используются лишь значения больше нуля из отрезка $\delta \in [-5^\circ; 5^\circ]$, а функция распределения четная, она может быть нормирована путем домножения на два.

Следовательно, для вычисления доли участвующих в образовании столба кристаллов n , отклонение которых лежит между значениями $\delta_1, \delta_2 \in [0; 5^\circ], \delta_1 < \delta_2$, необходимо вычислить следующий интеграл:

$$n = \int_{\delta_1}^{\delta_2} 2N(\delta)d\delta = \frac{6}{5\sqrt{2\pi}} \int_{\delta_1}^{\delta_2} e^{-0,5\left(\frac{3\delta}{5}\right)^2} d\delta \quad (36)$$

Однако, этот интеграл является неберущимся. При использовании геометрического метода вычисления интеграла будут значительные потери или в точности, или в производительности. Для оптимизации этого шага было использовано разложение функции Гаусса в ряд Тейлора. Такое решение обеспечивает приемлемую точность воспроизведения функции Гаусса при значительном упрощении вычисления интеграла [17].

$$N_T(\delta) = \sum_{i=0}^n \frac{N^{(n)}(a)}{n!} (\delta - a)^n \quad (37)$$

Для имеющейся функции Гаусса был сформирован ряд Тейлора девятой степени в точке $\delta = 2,5^\circ$. Таким образом выведен ряд Тейлора следующего вида:

$$\begin{aligned} N_T(\delta) = & 0,0777 - 0,07(\delta - 2,5) + 0,0175(\delta - 2,5)^2 + 0,00315(\delta - 2,5)^3 - 0,0023(\delta - 2,5)^4 + \\ & + 0,00018(\delta - 2,5)^5 + 0,00011(\delta - 2,5)^6 - 0,000024(\delta - 2,5)^7 - 2,27 \cdot 10^{-6}(\delta - 2,5)^8 + \\ & + 1,12 \cdot 10^{-6}(\delta - 2,5)^9 \end{aligned} \quad (38)$$

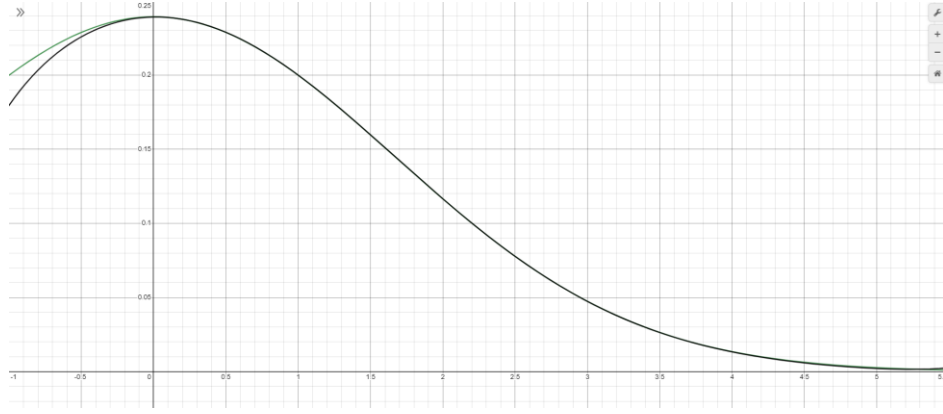


Рисунок 16 – Распределение Гаусса (зеленый график) и вычисленный для него ряд Тейлора (черный график)

При этом доля кристаллов n , участвующих в образовании светового столба, вычисляется следующим образом:

$$n = \int_{\delta_1}^{\delta_2} 2N_T(\delta)d\delta = 2 \left(\frac{9 \cdot 10^5 \delta^{10} - 2,5 \cdot 10^7 \delta^9 + 27,5 \cdot 10^7 \delta^8 - 1,5 \cdot 10^9 \delta^7 + 3,9 \cdot 10^9 \delta^6 - 17,8 \cdot 10^8 \delta^5}{8 \cdot 10^{12}} + \right. \\ \left. + \frac{11,3 \cdot 10^9 \delta^4 - 126 \cdot 10^9 \delta^3 + 5,9 \cdot 10^9 \delta^2 + 19,3 \cdot 10^{11} \delta}{8 \cdot 10^{12}} \right) \Big|_{\delta_1}^{\delta_2} \quad (39)$$

Значение угла отклонения кристалла δ от горизонтального положения в градусах в некоторой точке отражения точке R определяется следующим образом:

$$\begin{cases} r_x = S_x - R_x \\ r_y = S_y - R_y \\ r_z = S_z - R_z \end{cases} \quad (40)$$

$$\begin{cases} n_x = \frac{v_x}{\sqrt{v_x^2 + v_y^2 + v_z^2}} - \frac{r_x}{\sqrt{r_x^2 + r_y^2 + r_z^2}} \\ n_y = \frac{v_y}{\sqrt{v_x^2 + v_y^2 + v_z^2}} - \frac{r_y}{\sqrt{r_x^2 + r_y^2 + r_z^2}} \\ n_z = \frac{v_z}{\sqrt{v_x^2 + v_y^2 + v_z^2}} - \frac{r_z}{\sqrt{r_x^2 + r_y^2 + r_z^2}} \end{cases} \quad (41)$$

$$\delta = \frac{\arccos \left(\frac{n_z}{\sqrt{n_x^2 + n_y^2 + n_z^2}} \right)}{\pi} 180 \quad (42)$$

Таким образом, при найденных углах отклонения $\delta_1, \delta_2, \dots, \delta_k$, соответствующих вычисленным ранее точкам R_1, R_2, \dots, R_k , искомая доля кристаллов n определяется выражением:

$$n = \sum_{i=1}^k \int_{\delta_{2i-1}}^{\delta_{2i}} 2N_T(\delta) d\delta. \quad (43)$$

Шаг 5: вычисление интегрального коэффициента отражения.

Для вычисления коэффициента отражения используется упомянутая ранее аппроксимация Шлика:

$$\rho(\varepsilon) = \rho_0 + (1 - \rho_0)(1 - \cos\varepsilon)^5.$$

Параметр ρ_0 для границы воздух-лед равен [25]:

$$\rho_0 = \left(\frac{1,31 - 1}{1,31 + 1} \right)^2 = 0,018.$$

В рамках решаемой задачи разность коэффициента преломления для разных длин волн было принято игнорировать вследствие малого влияния этого аспекта на результат. Данная функция интегрируется на отрезках $[\varepsilon_1, \varepsilon_2], [\varepsilon_3, \varepsilon_4], \dots, [\varepsilon_{k-1}, \varepsilon_k]$ где $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_k$ – углы падения на найденных ранее границах объема светового столба в данном направлении. Для отражения в некоторой точке R_0 угол падения рассчитывается следующим образом:

$$\varepsilon = \arccos \left(\frac{-n_{0x}v_x - n_{0y}v_y - n_{0z}v_z}{|n_0||v|} \right). \quad (44)$$

Таким образом, нормированный интегральный коэффициент отражения рассчитывается по формуле:

$$\rho_I = \frac{1}{1,74} \int_{\varepsilon_1}^{\varepsilon_2} (0,018 + 0,982 \cdot (1 - \cos\varepsilon)^5) d\varepsilon, \quad (45)$$

где $\frac{1}{1,74}$ – нормировочный коэффициент.

$$\rho_I = \left(\frac{0,15 \sin(4\varepsilon) + 3,68 \sin(2\varepsilon) - 0,2 \sin^5 \varepsilon + 3,93 \sin^3 \varepsilon - 15,71 \sin \varepsilon + 8,73 \varepsilon}{1,74} \right) \Big|_{\varepsilon_1}^{\varepsilon_2}$$

$$\rho_I = \sum_{i=1}^{\frac{k}{2}} \int_{\varepsilon_{2i-1}}^{\varepsilon_{2i}} (0,018 + 0,982 \cdot (1 - \cos \varepsilon)^5) d\varepsilon$$

Шаг 6: расчет значения рассматриваемого пикселя.

Помимо доли кристаллов и интегрального отражения в расчет принимаются:

1. Атмосферное поглощение излучения столба. Рассчитывается в виде коэффициента a по закону Ламберта-Бугера-Бера.
2. Влияние толщины столба на его яркость. Определяется как условный коэффициент w , принимающий значение от 0 до 1 в зависимости от толщины столба.

Для определения цвета рассматриваемого пикселя необходимо учесть цвет излучения источника и соответствующую интенсивность светового столба, которая рассчитывается следующим образом:

$$I = n \cdot \rho_I \cdot a \cdot w \quad (46)$$

Учитывая цвет излучения (R_s, G_s, B_s) , получим цвет данного пикселя:

$$(R_s, G_s, B_s, I).$$

После того, как данные расчеты были проведены для каждого пикселя и было сформировано изображение столбов в виде текстуры, последняя накладывается на текущий кадр.

Таким образом, был сформирован алгоритм отрисовки на виртуальных сценах световых столбов с учетом описанных в ряде работ физических процессов, лежащих в основе рассматриваемого явления. В итоговом варианте алгоритм состоит из двух этапов: расчета для каждого пикселя и совмещение полученного изображения с изображением сцены. В свою очередь для каждого пикселя выполняются:

- определение вектора взгляда;
- расчет минимального угла отклонения;

- определение граничных характеристик светового столба;
- определение доли кристаллов, участвующих в образовании столба;
- вычисление интегрального коэффициента отражения.

Далее необходимо выполнить программную реализацию приведенного алгоритма на специализированном программном обеспечении, что описано в следующей главе.

3 РЕАЛИЗАЦИЯ АЛГОРИТМА ОТРИСОВКИ СВЕТОВЫХ СТОЛБОВ

В данной главе приведено пошаговое описание реализации алгоритма на DirectX 11 [23] и HLSL (High-Level Shader Language) [24], с последующим визуальным сравнением полученного результата с реальными изображениями световых столбов.

3.1 Реализация точечного источника света

Как было отмечено ранее, световые столбы возникают в атмосфере при наличии наземных точечных источников света. Для реализации алгоритма был написан класс PointLight, главной задачей которого в контексте текущей работы является формирование соответствующих теневых карт для последующего анализа окружения.

Существует два основных подхода к получению теневых карт точечного источника света:

1. Cube shadow mapping (CSM). При таком подходе из позиции источника света отрисовываются шесть теневых карт при угловом поле 90 градусов и соотношении сторон 1:1, таким образом создается «куб» теневых карт, несущий в себе информацию об окружении.

2. Dual-paraboloid shadow mapping (DPSM). В данном случае формируются только две текстуры, полученные проекцией окружения на параболоид – поверхность второго порядка. Каждый из параболоидов включает в себя проекцию половины окружения [18].

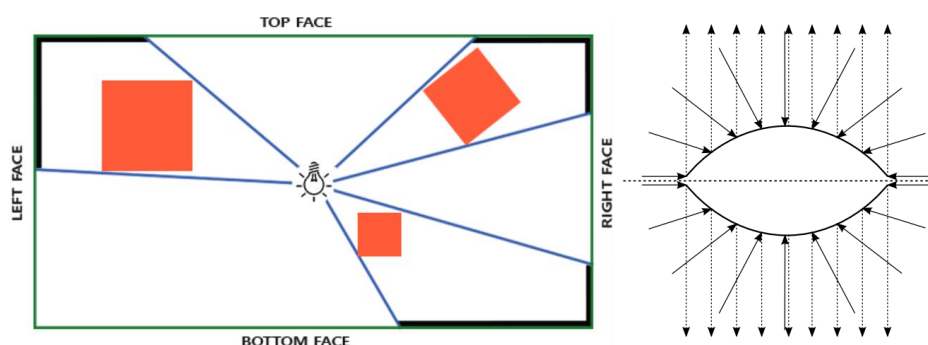


Рисунок 17 – Схематические изображения принципов CSM и DPSM [18]

Следует отметить, что существуют и иные методы формирования теневых карт. Например, теневая карта окружения для точечного источника света может быть записана в текстуру, полученную с помощью проекции на грани тетраэдра [19].

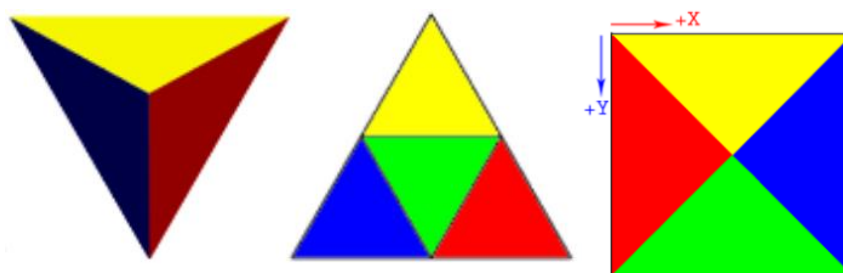


Рисунок 18 – Тетраэдр, его развертка и один из возможных способов формирования теневой карты [19]

Однако, в рамках данной работы, руководствуясь простотой подхода, для формирования теневых карт точечного источника света было принято решение использовать CSM. С учетом этого выбора ниже приведены используемые в алгоритме поля и методы класса PointLight.

3.1.1 Класс PointLight

В основу класса в виде полей легли следующие данные:

- *position* – мировая позиция источника света;
- *color* – цвет испускаемого света;
- *viewMtrcs* – массив из шести видовых матриц для реализации *cube shadow mapping*;
- *projectionMtrx* – проекционная матрица для *cube shadow mapping*;
- *depthTextures* – полученные текстуры глубины;
- *depthStencilViews* – соответствующие текстурам *depth stencil view*;
- *depthShaderRes* – соответствующие текстурам *shader resource view*;
- *samplerState* – семплер для чтения текстур глубины;
- *textureResolution* – разрешение текстур глубины.

Для класса были определены два конструктора:

- Базовый, при котором все приведенные выше параметры задаются по умолчанию;
- Конструктор с задачей позиции и цвета.

Для корректной отрисовки световых столбов в классе также определены три метода:

- *Update* – обновление текстур глубины для динамической отрисовки световых столбов;
- *PrepareResources* – подготовка ресурсов для дальнейшего их использования в шейдерах;
- *DestroyResources* – освобождение памяти, занятой ресурсами, при завершении работы программы.

3.1.2 Формирование теневых карт

Шаг 1: подготовка входных данных.

Для формирования текстур создан шейдер и отдельный графический пайплайн, где производится отрисовка всех объектов на сцене с учетом глубины позиции относительно источника света. В качестве исходных данных в шейдер поступают матрица вида-проекции и model-матрица объекта. Так как в алгоритме используется cube shadow mapping, матрица проекции имеет вид:

```
projectionMtrx = DirectX::SimpleMath::Matrix::CreatePerspectiveFieldOfView(  
    ((float)PI) / 2.0f,  
    1.0f,  
    0.1f,  
    farDistance  
);
```

Здесь параметр *farDistance* – радиус освещаемого объема источником света.

Видовая матрица имеет следующий вид (на примере грани, обращенной к зрителю):

```
viewMtrcs[2] = DirectX::SimpleMath::Matrix::CreateLookAt(  
    position,  
    DirectX::SimpleMath::Vector3(viewerPos.x, position.y, viewerPos.z),  
    DirectX::SimpleMath::Vector3::Up  
);
```

Пайплайн настроен так, что одна из граней cube map всегда «повернута» к зрителю, при этом верхняя и нижняя грани параллельны плоскости Oxy.

Шаг 2: текстуры и их подключение к шейдеру глубины.

Каждый экземпляр класса PointLight обладает шестью указателями на текстуры для cube mapping, объявленными посредством *ID3D11Texture2D**. Так как эти текстуры используются в двух пайплайнах

В разных контекстах, они имеют следующие флаги: *D3D11_BIND_SHADER_RESOURCE*, *D3D11_BIND_DEPTH_STENCIL*. Для формирования текстур глубины они подключаются к пайплайну через соответствующий *DepthStencilView*.

Шаг 3: отрисовка текстур глубины в шейдере.

С учетом входных данных в шейдере глубины обрабатываются вершины геометрии и оценивается расстояние до них:

```
float4 VSMain(float3 input : POSITION0) : SV_POSITION
{
    float4 pos = float4(input, 1.0f);
    float4 modelPos = mul(pos, model);
    float4 output = mul(modelPos, viewProjection);

    return output;
}

float PSMain(float4 input : SV_POSITION) : SV_Depth
{
    float depth = input.z / input.w;
    return depth;
}
```

Шаг 4: передача текстур в шейдер световых столбов.

Для отрисовки светового столба с учетом окружения в шейдер передаются текстуры глубины как *Shader Resource*:

```
res = device->CreateShaderResourceView(depthTextures[2],
&shaderResViewDesc, &(depthShaderRes[2]));
context->PSSetShaderResources(1, 1, &(pointLights.at(0)->
depthShaderRes[2]));
```

3.2 Формирование световых столбов

В этом пункте приведены лишь действия, связанные с непосредственной отрисовкой световых столбов. В связи с этим некоторые детали реализации графического конвейера не приводятся.

3.2.1 Формирование входных данных

Согласно приведенному ранее алгоритму, необходимо подготовить требуемые входные данные для шейдера:

- константный буфер наблюдателя: матрица вида-проекции, соответствующая ей инвертированная матрица, мировая позиция;
- буфер источника света: мировая позиция, цвет, передняя и верхняя матрицы вида-проекции.

В методе *DrawPillars*, который вызывается в основном цикле работы программы, данные буферы формируются последовательно на основе заранее обновленных данных наблюдателя и источника света.

Далее настраиваются параметры работы графического конвейера: устанавливаются заранее скомпилированные соответствующие вершинный и пиксельные шейдеры, подключаются сформированные ранее буферы, текстуры глубины наблюдателя и источника света и соответствующие им семплеры. Кроме того, топология примитивов устанавливается в *D3D_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP*, так как для отрисовки световых столбов используется fullscreen quad – отрисовка по всему полю зрения наблюдателя (с точки зрения пользователя отрисовка происходит по всей площади окна программы).

Наконец, методом *Draw(4, 0)* вызывается шейдер для четырех точек, расположенных в углах окна.

3.2.2 Шейдеры световых столбов

Вызываемые шейдеры, а также используемые константы и функции прописаны в hlsl-файле *PillarsShader.hlsl*. Его основные составляющие перечислены ниже:

1. Константы:

- *CRIT_ANGLE_RAD*, *CRIT_ANGLE_DEG* – значения предельного угла отклонения кристаллов льда в радианах и градусах соответственно. Использование разных единиц измерения обусловлено различием параметров некоторых функций;
- *PI* – число Пи;
- *ABSORPTION_PARAMETER* – коэффициент поглощения света средой;
- *DEPTH_READING_FREQUENCY* – частота анализа окружения внутри объема столба;
- *PILLAR_WIDTH_FACTOR* – условный параметр, отвечающий за зависимость светимости столба от его ширины.

2. Функции:

- *CalculateGaussProt* – приближенное вычисление первообразной функции Гаусса для распределения отклонения частиц льда для заданного угла в градусах;
- *CalculateGaussIntegral* – приближенное вычисление интеграла функции Гаусса для распределения отклонения частиц льда в заданных угловых пределах в градусах;
- *CalculateRefractionProt* – вычисление первообразной аппроксимирующей функции Шлика для заданного угла падения в радианах;
- *CalculateRefractionIntegral* – вычисление интеграла аппроксимирующей функции Шлика в заданных пределах углов падения в радианах.

3. Буферы наблюдателя и источника света.

4. Ресурсы: текстуры глубины поля зрения наблюдателя, передней и верхней граней cube shadow map источника света и соответствующие им семплеры.

Далее описан алгоритм работы шейдеров. В контексте fullscreen quad и световых столбов единственной задачей вершинного шейдера является передача в пиксельный шейдер текстурных координат точки, для которой в данный момент производится вычисление:

```
float4 VSMain(uint vI : SV_VERTEXID) : SV_POSITION
{
    float2 texcoord = float2(vI & 1, vI >> 1);
    return float4((texcoord.x - 0.5f) * 2.0f, -(texcoord.y - 0.5f) *
2.0f, 0.0f, 1.0f);}
```

В пиксельном шейдере реализована основная часть сформулированного в прошлой работе алгоритма. Здесь используются все приведенные выше данные и функции. Этапы работы алгоритма изложены в порядке, соответствующем описанию алгоритма в прошлой главе.

Шаг 1: вычисление вектора наблюдения *viewDir*.

На вход пиксельного шейдера подаются текстурные координаты рассматриваемого пикселя, которые первым шагом переводятся в Normalized Device Coordinates (NDC). Для получения вектора наблюдения используется буфер наблюдателя, а именно: позиция наблюдателя и инвертированная матрица вида-проекции. Посредством матрицы NDC-координаты преобразуются в точку в мировом пространстве, из которой вычитается мировая позиция наблюдателя.

Таким образом получается вектор наблюдения *viewDir*, который для оптимизации вычислений сразу ортонормируется через функцию *normalize()*.

```
float4 worldPoint = mul(NDC, constData.invertedCamViewProjection);
worldPoint /= worldPoint.w;
float3 viewDir = normalize(worldPoint.xyz - constData.viewerPos);
```


Шаг 2: расчет минимального угла отклонения.

На данном этапе вычисляется точка минимального отклонения $R0$ в заданном направлении. Для этого вычисляется параметр $shiftY$ – смещение в длинах вектора наблюдения, при котором наблюдатель окажется на одной высоте с источником света. Далее, относительно этого смещения с учетом буферов наблюдателя и источника вычисляется серединная точка в заданном направлении в горизонтальной плоскости. Эта точка является проекцией точки $R0$ на горизонтальную плоскость, в которой находится источник. Высота точки $R0$ (y-координата) определяется пропорционально смещению данной точки относительно положения источника света в горизонтальной плоскости.

Следующим шагом определяется вектор отражения *reflectionDir0* как нормализованная разность позиции источника и точки $R0$. Наконец, чтобы получить искомый угол, необходимо определить нормаль *normal0* к отражающей поверхности и найти ее отклонение от оси Oy :

```
float3 reflectionDir0 = normalize(pointLightData.lightSourcePosition.xyz -  
R0);  
normal0 = normalize(viewDir - reflectionDir0);
```

Так как нормаль единичная, ее y-координата является косинусом угла между нормалью и осью Oy . На данном моменте предусмотрена первая точка выхода из функции пиксельного шейдера: если угол отклонения больше предельного, то шейдер ничего не изображает на соответствующем пикселе.

```
if (normal0.y < cos(CRIT_ANGLE_RAD)) return float4(0, 0, 0, 0);
```

Шаг 3: определение граничных характеристик светового столба.

Чтобы найти граничные точки в заданном направлении, необходимо найти нормали к поверхностям предельно наклоненных кристаллов в этом направлении. Алгоритм в этом пункте отличается от первоначального: вместо предложенного поиска нормалей через их компланарность с падающим, отраженным векторами нахождение нормалей реализовано через принадлежность вектора к рабочей плоскости.

Рабочая плоскость задается по двум независимым векторам и точке: по вектору наблюдения *viewDir*, нормали *normal0* и позиции наблюдателя. Таким образом, искомые нормали также лежат в рабочей плоскости. Кроме того, искомые нормали единичные. Исходя из этих условий формируется и решается система уравнений и находятся нормали *n1* и *n2*.

Соответствующие найденным нормальям векторы отражения находятся следующим образом:

```
float3 reflectionDir1 = reflect(viewDir, n1);  
float3 reflectionDir2 = reflect(viewDir, n2);
```

Далее граничные точки *R1* и *R2* находятся как точки пересечения прямых, направляющими векторами которых являются вектора *viewDir* и *reflectionDir1* или *reflectionDir2*. Кроме того, из найденных точек определяется ближняя и дальняя точки:

```
float3 Rn = R1, Rf = R1;  
if (length(R1 - constData.viewerPos) > length(R2 - constData.viewerPos))  
    Rn = R2;  
else Rf = R2;
```

Здесь предусмотрена вторая точка выхода. Если точка *Rn* удалена на большее расстояние, чем соответствующее значение в текстуре глубины

наблюдателя, то шейдер возвращает «прозрачный» пиксель (столб закрыт некоторым объектом):

```
float4 camSpacePoint = mul(constData.camViewProjection, float4(Rn.xyz, 0));
float depthValue = camDepthTexture.SampleLevel(camDepthSampler,
float2((NDC.x + 1.0f) / 2, -(NDC.y + 1.0f) / 2), 0).x;
if (depthValue <= camSpacePoint.z / camSpacePoint.w && depthValue != 1.0f)
    return float4(0, 0, 0, 0);
```

Шаг 4: определение доли кристаллов, вносящих вклад в образование столба.

На данном шаге производится обход точек внутри объема столба с некоторым шагом. По данным в текстурах глубины источника вычисляются интервалы, состоящие из освещенных точек. Для каждого интервала вычисляется доля интересующих кристаллов, доля отражения света, атмосферное поглощение, которые формируют яркость столба в данном направлении, и ширина интервала. По завершению обхода суммарная яркость корректируется исходя из ширины столба и полученная яркость подается как множитель к цвету пикселя на выходе:

```
float3 step = (Rf - Rn) / DEPTH_READING_FREQUENCY;
float3 currentPoint = Rn, startPoint = Rn;
float3 currentVector = currentPoint - pointLightData.lightSourcePosition.xyz;
bool isDepthTestPassed = true;
float intensity = 0, L = 0;
for (int i = 0; i <= DEPTH_READING_FREQUENCY; i++)
{
    currentPoint = Rn + step * i;
    currentVector = normalize(currentPoint - pointLightData.lightSourcePosition.xyz);
    float depthValue;
    <...>
    if (depthValue <= lightSpacePos.z / lightSpacePos.w && depthValue != 1.0f)
    {
        if (isDepthTestPassed)
        {
            isDepthTestPassed = false;
            n1 = normalize(viewDir + normalize(startPoint -
pointLightData.lightSourcePosition.xyz));
```

```

        n2 = normalize(viewDir + currentVector);
        float e1 = acos(n1.x * viewDir.x + n1.y * viewDir.y + n1.z * viewDir.z);
        float e2 = acos(n2.x * viewDir.x + n2.y * viewDir.y + n2.z * viewDir.z);
        float3 nR0 = R0 - startPoint;
        float3 nf = currentPoint - startPoint;
        float N;
        if (length(nR0) < length(nf) && nf.x / nR0.x >= 0)

            N = abs(CalculateGaussIntegral(acos(n1.y) / PI * 180.0f, acos(normal0.y) / PI
* 180.0f))
                + abs(CalculateGaussIntegral(acos(n2.y) / PI * 180.0f, acos(normal0.y) /
PI * 180.0f));
        else
            N = abs(CalculateGaussIntegral(acos(n1.y) / PI * 180.0f, acos(n2.y) / PI *
180.0f));

        intensity = intensity + N * abs(CalculateRefractionIntegral(e1, e2))
            * pow(2.71f, -ABSORPTION_PARAMETER * length((startPoint + currentPoint) /
PILLAR_WIDTH_FACTOR - constData.viewerPos));
        L += length(currentPoint - startPoint);
    }
}
else
{
    if (!isDepthTestPassed)
    {
        isDepthTestPassed = true;
        startPoint = currentPoint;
    }
}
intensity *= clamp(L / PILLAR_WIDTH_FACTOR, 0, 1);
return float4(pointLightData.lightColor.xyz * intensity, 1.0f);

```

В приложении А представлена блок-схема шейдера, в котором реализован алгоритм.

3.3 Визуальная оценка результата

Ниже приведено сравнение полученного изображения с реальными фотографиями световых столбов. Субъективное сходство вида столбов считается достигнутым.



Рисунок 19 – Сравнение полученного изображения (слева) с реальными снимками (справа)

Далее приведена демонстрация влияния окружения на изображение столба. Для наглядности был создан симуляция тест-объекта, частично перекрывающего ход лучей света. Слева направо: сокрытие столба за объектом на сцене, влияние нахождения тест-объекта в объеме столба на его яркость (в ближней части объема, в середине объема и в дальней части объема). Очевидно влияние положения тест-объекта на его яркость. Для большей читаемости яркость изображений была увеличена.

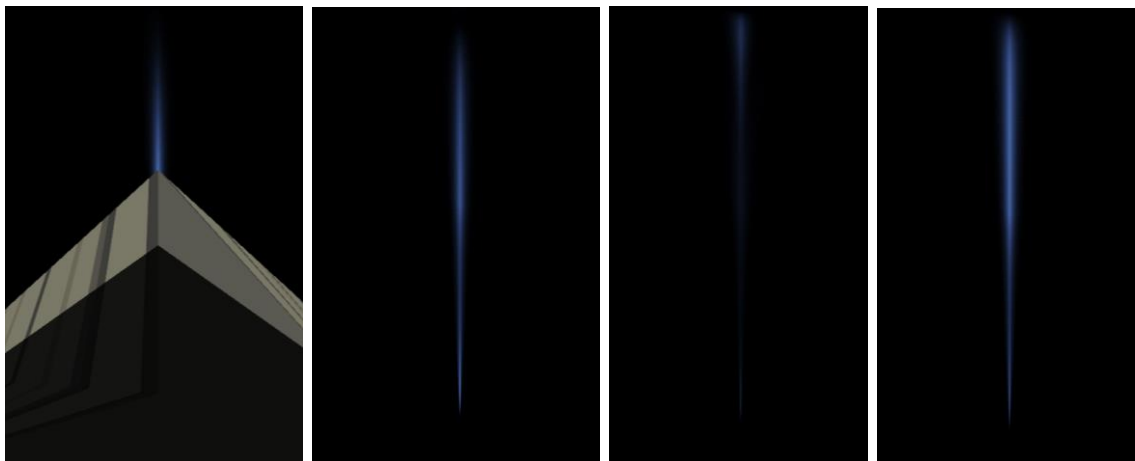


Рисунок 20 – Демонстрация влияния окружения на отрисовку столба

Таким образом, выполненная практическая реализация сформулированного ранее алгоритма фактически обеспечивает влияние

окружения на яркость светового столба. Продемонстрированы основные свойства алгоритма, приведены сравнительные материалы как разных состояний столба, так и отрисованного и реального изображения атмосферного явления.

Сравнение результатов с реальными изображениями явления показало значительное визуальное сходство, что означает, что задача по реализации алгоритма отрисовки световых столбов выполнена. С другой стороны, открытым остается вопрос о возможности использования предложенного алгоритма в реальном времени. Для этого необходимо сформировать и провести тесты производительности алгоритма, чему и посвящена следующая глава.

4 АНАЛИЗ РЕЗУЛЬТАТОВ РАЗРАБОТКИ

В рамках данной главы описан ряд исследований вариаций предложенного алгоритма, на основе полученных данных произведена оценка производительности, а также выполнен сравнительный анализ. Также, предполагается проведение сравнительного анализа полученного результата с существующими аналогами, изображающими рассматриваемое атмосферное явление.

4.1 Тестирование производительности алгоритма

Было проведено тестирование производительности предложенного алгоритма при разных параметрах и условиях. При исследовании зависимости производительности от изменения одного параметра остальные настройки оставались без изменений. Исходные графические параметры указаны в таблице ниже.

Таблица 1 – Значение графических параметров по умолчанию

Параметр	Значение
Размер дисплея	1920x1080 пикселей
Количество чтений cube shadow map за один проход	100 раз
Качество cube shadow map	1024x1024 пикселей

Тестирование производилось на устройстве, оснащённом следующими комплектующими:

- видеокарта Nvidia GeForce RTX 3050 (Laptop), 4 ГБ видеопамати;
- процессор Intel Core i5-11400H, 2,7 ГГц;
- 16 ГБ оперативной памяти.

4.1.1 Количество отрисовываемых точечных источников света

В рамках данного теста измерялась производительность алгоритма при отрисовке двух и более световых столбов. Для этого был реализован функционал, обеспечивающий поддержку отрисовки столбов от множества точечных источников света. Для получения значения производительности измерялось время отрисовки каждого из 5000 кадров, а затем вычислялось среднее время отрисовки кадра t_{cp} в миллисекундах. Предлагается следующая система оценки производительности:

- $t_{cp} \leq 17$ мс: хорошо;
- $17 \text{ мс} < t_{cp} \leq 40$ мс: удовлетворительно;
- $t_{cp} > 40$ мс: неудовлетворительно.

Результаты измерений приведены ниже.

Таблица 2 – зависимость производительности от количества столбов

Количество источников	Время отрисовки кадра, мс	FPS
1	14,0	71
2	16,4	60
5	35,0	28
10	69,8	14
20	125,4	8

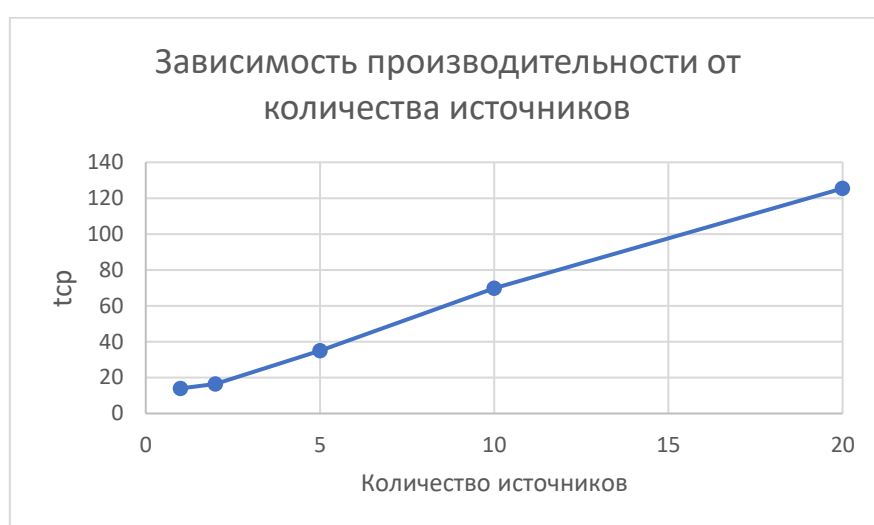


Рисунок 21 – Зависимость производительности от количества источников

Таким образом, в условиях проведения тестирования хорошую производительность показали только отрисовки одного и двух столбов. Отрисовка до пяти столбов обеспечивает производительность удовлетворительного уровня и выше. Далее значение FPS падает до неудовлетворительного уровня.

4.1.2 Разрешение cube shadow map

В рамках данного теста измерялась производительность алгоритма в зависимости от разрешения карт теней, которые отрисовываются для

каждого точечного источника света. Критерии и методика измерения аналогичны предыдущему тесту.

Таблица 3 – Зависимость производительности от разрешения CSM

Разрешение	Время отрисовки кадра, мс	<i>FPS</i>
512x512	9,5	105
1024x1024	14,0	71
2048x2048	20,9	47
4096x4096	59,3	16

Изменение разрешения CSM значительно влияет на производительность, так как карты обновляются каждый кадр. Это влияние возможно устранить, если окружение статично, - в этом случае текстуры CSM будут переиспользоваться и отрисовывать их каждый кадр не потребуются. В противном случае потребуется выбирать разрешение текстур в зависимости от мощности устройства. В данном случае разрешение 1024x1024 является оптимальным.

4.2 Сравнение полученного результата с другими подходами к отрисовке световых столбов

В этом пункте рассмотрены некоторые подходы к отрисовке световых столбов и главным образом проведено сравнение с предлагаемым решением.

4.2.1 Двумерный столб

Смысл данного подхода заключается в следующем: на сцену помещается текстура, которая всегда поворачивается к зрителю. Это очень малозатратный способ изобразить световой столб, однако у обозначенного

метода есть некоторые недостатки в сравнении с предлагаемым решением. Рассмотрим этот подход с точки зрения реализма.

Во-первых, так как изображение столба занимает конкретное положение на сцене, оно может быть закрыто некоторым объектом, помещенным перед ним. Реальный столб, как и столб в предлагаемом решении, также может быть закрыт другими объектами, но здесь следует отметить следующий момент: чем ближе объект к «оси» столба, тем больше кристаллов льда внесут вклад в формирование изображения, и столб перед объектом будет по-прежнему виден, хоть и с меньшей интенсивностью. В случае обычной текстуры, помещенной на сцене, любое положение объекта между столбом и зрителем приведет к сокрытию части столба. Средствами среды разработки Unity была реализована демонстрационная версия светового столба, сформированного указанной техникой.



Рисунок 22 – Демонстрация сокрытия столба объектом (слева) и сравнение с решением (справа)

Во-вторых, окружение никак не влияет на интенсивность столба: положение объекта не изменяет вид явления. К тому же, при помещении объекта «внутри» столба будет четко прослеживаться граница между ними.

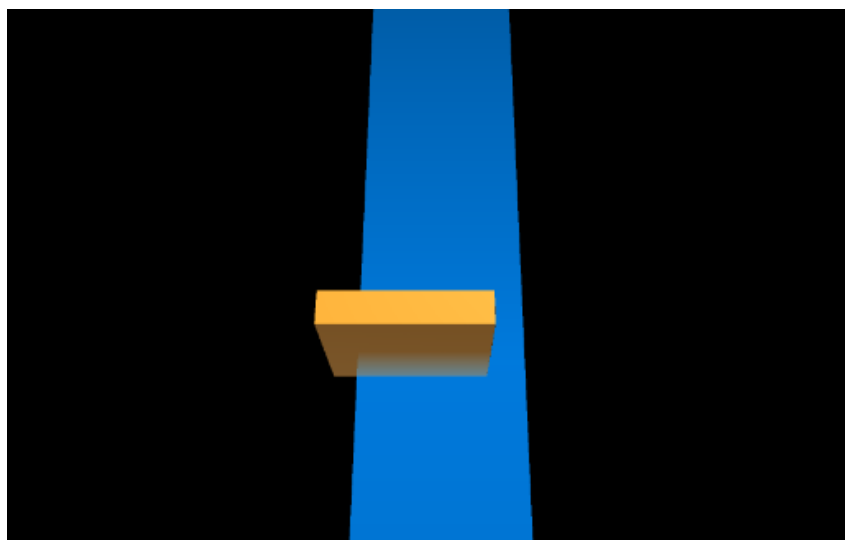


Рисунок 23 – Демонстрация видимой границы между столбом и объектом, размещенным над ним

4.2.2 Volumetric-столб

Еще одним способом отрисовки световых столбов является реализация столба рассеяния от направленного источника света посредством техники Volumetric Lighting. В отличие от предыдущего способа данный метод способен в некоторой степени принимать в расчет окружение, но, так как в данном случае по сути симулируется рассеяние света находящимися в окружающей среде частицами (эффект Тиндаля), поведение столба сильно отличается от реального. Как и в предыдущем случае, средствами среды разработки Unity, а именно с помощью HDRP (High Definition Render Pipeline) и Volumetric light, была реализована демонстрационная версия светового столба, сформированного указанной техникой [26, 27].



Рисунок 24 – Демонстрация изменения volumetric-столба вместе с окружением

Таким образом, несмотря на физическую проработку, метод изображает другое явление и не отражает в полной мере природу световых столбов: столб является буквально направленным вверх и его может прервать объект. Более того, этот метод обладает тем же недостатком, что и предыдущий вариант: столб может быть загорожен объектом, помещенным между ним и зрителем. В предлагаемой же модели эти моменты учтены.

4.2.3 Сравнение подходов

На основе приведенной информации приведена сравнительная таблица ниже. Два рассмотренных метода, а также разработанное решение были проанализированы по четырем факторам:

- область формирования светового столба;
- влияние окружения на яркость столба;
- влияние окружения на структуру столба;
- производительность метода.

Таблица 4 – Сравнение методов отрисовки световых столбов

	Предлагаемое решение	Двумерный столб	Volumetric-столб
Область формирования	Пространство между источником и зрителем	Плоскость над источником	Объем над источником
Влияние окружения на яркость	Да	Нет	Нет
Изменение структуры с изменением окружения	Да	Нет	Да
Влияние на скорость отрисовки сцены	Значительное	Низкое	Значительное

Тестирование показало, что согласно обозначенным критериям отрисовка по крайней мере двух световых столбов не вызывает критического падения производительности, а отрисовка пяти столбов обеспечивает приемлемый уровень производительности. Качество анализа окружения, выполняемое посредством чтения Cube Shadow Map, также влияет на производительность, однако выбранная конфигурация алгоритма обеспечивает достаточный баланс между качеством анализа окружения и производительностью.

Кроме того, было выполнено сравнение приведенного алгоритма с другими способами отрисовки световых столбов, где были обозначены основные преимущества и недостатки каждого из обозначенных методов. Основываясь на приведенных данных и сравнивая их с характеристиками реальных световых столбов, следует отметить, что предлагаемое решение наиболее точно отражает физику и динамику световых столбов. Тем не менее, несмотря на возможность отрисовки явления в реальном времени, на данном этапе алгоритм не может обеспечить стабильно достаточную

производительность при отрисовке множества (а именно – более десяти) световых столбов, что является недостатком приведенного алгоритма.

4.3 Дальнейшая работа

В ходе разработки алгоритма были выполнены поставленные задачи: удалось получить реалистичные изображения световых столбов, при этом их отрисовка происходит в реальном времени. Тем не менее, алгоритм имеет некоторые недостатки. В этом пункте рассмотрены основные проблемы алгоритма и возможные пути их решения.

4.3.1 Оптимизация отрисовки множества столбов

Тестирование производительности выявило проблему отрисовки множества столбов: при отрисовке более пяти столбов наблюдается довольно низкий уровень FPS. В основе решения данной проблемы лежит два пункта:

1. Оптимизация шейдеров. На данный момент для каждого столба совершается отдельный проход по графическому конвейеру. Имеет смысл реорганизовать работу шейдера, обеспечив возможность отрисовки множества столбов за один проход, передав при этом на вход все необходимые ресурсы.
2. Оптимизация формирования теневых карт. На данный момент решение настроено таким образом, что для каждого источника света теневые карты отрисовываются за шесть проходов по графическому конвейеру. Это не является проблемой, пока количество столбов невелико. Однако, чем больше столбов необходимо отрисовать, тем более объемные будут вычисления. Для сокращения этих вычислений следует обратить внимание на culling-методы, сокращающие количество геометрии, которое

необходимо учитывать при формировании CSM. Так, например, в работе «Fast Shadow Map Rendering for Many-Lights Settings» предлагается использовать voxel-culling, который, по результатам исследования, позволяет добиться значительной оптимизации процесса формирования теневых карт для сцен с множеством источников света [28].

4.3.2 Конкретизация атмосферного поглощения

Показатель атмосферного поглощения зависит от многих параметров оптической среды: состава воздуха, наличия и видов частиц, находящихся в среде, типа поглощаемого излучения. С одной стороны, расчет атмосферного поглощения является комплексным [29], так как нужно определить и учитывать все обозначенные факторы, но с другой стороны учет этого аспекта явления значительно повысит достоверность изображения при известных параметрах среды.

4.3.3 Конкретизация влияние ширины столба на его яркость

На данный момент фактор влияния ширины столба определен эмпирически. Для его уточнения следует рассмотреть столб как систему множества точечных источников света в виде частиц льда в атмосфере. Если произвести типовой энергетический расчет для такой оптической системы и пересчитать энергетические значения яркости в выходные значения пиксельного шейдера, то оценка влияния ширины светового столба на его яркость станет объективной [30].

ЗАКЛЮЧЕНИЕ

В рамках данной работы был разработан алгоритм отрисовки такого атмосферного оптического явления, как световые столбы.

Обзор решений по реалистичной отрисовки различных оптических явлений показал разнообразие подходов к воспроизведению соответствующих изображений. Все они в той или иной степени опираются на физические процессы, лежащие в основе рассматриваемых явлений. На основе проведенного анализа было выявлено перспективное направление работы: создание алгоритма отрисовки физически достоверных световых столбов в реальном времени, что и стало целью этой работы.

Так как было принято решение физически обосновать алгоритм отрисовки, в работе приведен ряд научных данных о природе световых столбов. Эта информация была систематизирована и формализована, за базу алгоритма были приняты:

- динамика частиц льда: статистическое распределение положения частиц в атмосфере;
- законы отражения Френеля;
- атмосферное поглощение излучения;
- суммирование излучения в заданном направлении.

Важно отметить, что алгоритм не предполагает использование классического приема трассировки лучей.

Исходя из принятых положений, сформирован алгоритм отрисовки световых столбов. Он состоит из следующих этапов вычисления для каждого пикселя изображения:

1. Вычисление вектора наблюдения.
2. Расчет минимального угла отклонения.
3. Определение граничных характеристик светового столба.

4. Определение доли кристаллов, вносящих вклад в образование столба.
5. Вычисление интегрального коэффициента отражения.
6. Расчет значения рассматриваемого пикселя.

Полученное изображение смешивается с изображением сцены.

Итоговый результат отрисовки визуально сравнен с реальными изображениями световых столбов. Сходство результата с реальным явлением признано достаточным, таким образом выполнена задача по отрисовке световых столбов на основе их природы.

Для оценки производительности алгоритма проведены тесты и сформулированы критерии оценки результатов. Тестирование имело следующие результаты:

1. В минимальном виде алгоритм обеспечивает отрисовку световых столбов в реальном времени.
2. Существует предел количества одновременно отрисовываемых столбов, равный десяти, при котором наблюдаются значительные потери производительности, что приводит к невозможности оптимальной работы алгоритма в реальном времени при заданной конфигурации системы.
3. В случае одного столба алгоритм, формируя теневые карты высокого разрешения (2048x2048), способен детальнее анализировать окружение и работать с удовлетворительной производительностью.

Таким образом, задача по обеспечению работы алгоритма в реальном времени формально выполнена.

Проведенное сравнение полученного результата с другими подходами воспроизведения световых столбов показало, что, несмотря на сравнительно большую ресурсоемкость, предлагаемый алгоритм

изображает световые столбы с физической точки зрения достовернее, чем другие рассмотренные техники.

Несмотря на оптимальные результаты, представленный алгоритм имеет ряд недочетов:

- значительное падение производительности при отрисовки множества столбов;
- неуточненные данные об атмосферном поглощении излучения столба;
- неуточненные данные о влиянии ширины столба на его яркость в заданном направлении наблюдения.

Данные пункты могут быть обозначены как потенциальные направления для дальнейшей работы.

Разработанное решение опубликовано в репозитории сервиса GitHub и размещено по следующей ссылке: <https://github.com/Fortythird/LightPillars>.

СПИСОК ЛИТЕРАТУРЫ

1. Lee R. L. Mie theory, Airy theory, and the natural rainbow // Applied Optics. – 1998. – Т. 37. – №. 9. – С. 1506-1519.
2. Kirk Riley, David S. Ebert, Martin Kraus, Jerry Tessendorf Charles Hansen Efficient Rendering of Atmospheric Phenomena // Eurographics Symposium on Rendering. – 2004. – 12 с.
3. Iman Sadeghi, Adolfo Muñoz, Philip Laven, Wojciech Jarosz, Francisco Seron, Diego Gutierrez, Henrik Wann Jensen Physically-Based Simulation of Rainbows // ACM Transactions on Graphics. – 2011. – №1(31). – 12 с.
4. Clint Brewer How to Render a Real Rainbow // Nvidia [Электронный ресурс] - http://download.nvidia.com/developer/presentations/GDC_2004/gdc2004_RainbowFogbow.pdf (Дата обращения: 14.04.2024).
5. Diego Gutierrez, Francisco J. Seron, Adolfo Muñoz, Oscar Anson Rendering Ghost Ships and Other Phenomena in Arctic Atmospheres // International Conference in Central Europe on Computer Graphics and Visualization. – 2005. – 4 с.
6. Arthur Firmino Physically Based Rendering of Ice Crystal Halos // University of Victoria. – 2018. – 40 с.
7. Orion Sky Lawlor, Jon Genetti Interactive Volume Rendering Aurora on the GPU // Computer Graphics Forum. – 2020. – №4(39). – 10 с.
8. Tomokazu Ishikawa, Yonghao Yue, Kei Iwasaki, Yoshinori Dobashi, Tomoyuki Nishita Modeling of aurora borealis using the observed data // SCCG '11: Proceedings of the 27th Spring Conference on Computer Graphics. – 2011. – С. 13-16.

9. Zhao Y. et al. Visual simulation of heat shimmering and mirage // IEEE transactions on visualization and computer graphics. – 2006. – Т. 13. – №. 1. – С. 179-189.
10. Схема образования световых столбов при отражении света от горизонтальных поверхностей ледяных кристаллов в воздухе // Wikipedia [Электронный ресурс] – https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BB%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D0%B9_%D1%81%D1%82%D0%BE%D0%BB%D0%B1#/media/%D0%A4%D0%B0%D0%B9%D0%BB:Light_Pillars_Scheme.svg (Дата обращения: 14.04.2024).
11. Nakaya Ukichiro Snow crystals: natural and artificial // Cambridge: Harvard University Press. – 1954. – 510 с.
12. Vincent Noel, Hélène Chepfe Study of Planar Ice Crystal Orientations in Ice Clouds from Scanning Polarization Lidar Observations // Journal of Applied Meteorology. – 2005. – №5(44). – с. 653-664.
13. Anatoli Borovoi, Natalia Kustova Display of ice crystal flutter in atmospheric light pillars // Geophysical Research Letters. – 2009. – №4(36). – 5 с.
14. Anatoli Borovoi, Victor Galileiskii, Alexander Morozov, Ariel Cohen Detection of ice crystal particles preferably oriented in the atmosphere by use of the specular component of scattered light // OPTICS EXPRESS. – 2008. - №11(16). – 9 с.
15. Родионов С. А. Основы оптики. Конспект лекций. – СПб: СПб ГИТМО (ТУ), 2000. – 167 с.
16. Christophe Schlick An Inexpensive BRDF Model for Physically-based Rendering // Computer Graphics Forum. – 1994. – №3(13). – с. 233-246.
17. Письменный Д. Т. Конспект лекций по высшей математике: полный курс – М.: Айрис-пресс, 2009. – 608 с.

18. Jan Navratil, Jozef Kobrtek, Pavel Zemcik A Survey on Methods for Omnidirectional Shadow Rendering // Journal of WSCG. – 2012. - №20. – 8 с.
19. Wolfgang Engel GPU Pro 360 guide to shadows // CRC Press. – 2019. – 230 с.
20. Назаров В. Н., Балашов И. Ф. Энергетическая оценка импульсных лазерных дальномеров // СПбГУ ИТМО. - Электронный учебник.
21. Frank D. Luna Introduction to 3D Game Programming with DirectX 11 // Mercury Learning and Information. - 2012. - 754 с.
22. Tomas Akenine-Moller, Eric Haines, Naty Hoffman, Angelo Pesce, Michal Iwanicki, Sebastien Hillaire Real-Time Rendering // CRC Press. - 2018. - 1199 с.
23. Документация DirectX 11 [Электронный ресурс] – <https://learn.microsoft.com/en-us/windows/win32/direct3d11/atoc-dx-graphics-direct3d-11> (Дата обращения: 14.04.2024)
24. Справочник по HLSL [Электронный ресурс] – <https://learn.microsoft.com/ru-ru/windows/win32/direct3dhls/dx-graphics-hlsl-reference> (Дата обращения: 14.04.2024)
25. Золотарев В. М. Оптические постоянные природных и технических сред : справочник / В. М. Золотарев, В. Н. Морозов, Е. В. Смирнова - Ленинград : Химия, Ленинградское отделение, 1984. - 216 с.: ил.
26. Документация Unity: HDRP [Электронный ресурс] – <https://unity.com/ru/how-to/getting-started-high-definition-render-pipeline-hdrp-games> (Дата обращения: 14.04.2024)
27. Документация Unity: Volumetric lighting [Электронный ресурс] – <https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@13.1/manual/Volumetric-Lighting.html> (Дата обращения: 14.04.2024)

28. Selgrad K. et al. Fast Shadow Map Rendering for Many-Lights Settings // EGSR (EI&I). – 2016. – С. 41-47.
29. Коротаяев В. В., Мусяков В. Л. Энергетический расчет ОЭП / Учебное пособие по курсовому и дипломному проектированию. - СПб: СПбГУ ИТМО, 2006. – 44 с.
30. Тимофеев Ю.М., Васильев А.В. Основы теоретической атмосферной оптики / Учебно-методическое пособие. – СПб: СПбГУ, 2007. – 152 с.

ПРИЛОЖЕНИЕ А

Блок-схема алгоритма отрисовки световых столбов

