Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Моделирование облаков»

Студент ИУ7-54Б (Группа) _____ (Подпись, дата) Булдаков М. Ю. (И. О. Фамилия)

Руководитель курсовой работы _____ (Подпись, дата) Кузнецова О. В. (И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениями.

—

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие сокращения и обозначения.

—

ВВЕДЕНИЕ

Компьютерная графика представляет собой совокупность методов и способов преобразования информации в графическое представление при помощи ЭВМ. Без компьютерной графики не обходится ни одна современная программа. В течении нескольких десятилетий компьютерная графика прошла долгий путь, начиная с базовых алгоритмов, таких как вычерчивание линий и отрезков, до построения виртуальной реальности.

Облака сильно влияют на восприятие изображения, делая его более реалистичным. Поэтому задача визуализации реалистичных облаков чаще всего встречается при разработке компьютерных игр и в кинематографе. Раньше облака в играх рисовались обычными 2D спрайтами, которые всегда повернуты в направлении камеры, но последние годы вычислительные ресурсы компьютеров позволяют рисовать облака, близкие к физически корректным, без заметных потерь в производительности.

Целью данной работы является реализация программного обеспечения, которое предоставляет возможность создавать облачное небо, состоящее из объемных, близких к физически корректным, облаков. При этом созданная программа должна позволить изменять параметры, влияющие на внешний вид неба.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

— изучить явления, происходящие в облаках;

— проанализировать существующие способы описания облаков;

— проанализировать и выбрать способы визуализации облаков;

— проанализировать эффекты, возникающие в облаках, и на их основе разработать модель освещения;

— реализовать выбранные алгоритмы;

— разработать программное обеспечение для отображения облачного неба.

1 Аналитический раздел

1.1 Участвующая среда

Участвующая среда – термин, используемый для описания объемов, заполненных частицами. Этими частицами могут являться: капли воды, кристаллы льда, пыль, молекулы. Проходя через такую среду, свет взаимодействует с частицами и может отражаться, поглощаться или рассеиваться [1].

1.1.1 Закон Бугера-Ламберта-Бера

Закон Бугера-Ламберта-Бера – определяет ослабление пучка света при распространении его в поглощающей среде. Свет, проходя сквозь вещество, подвергается поглощению этим самым веществом. Для дифференциального расстояния dx относительное уменьшение яркости определяется как $k_a(x)dx$, где $k_a(x)$ – коэффициент поглощения среды в точке x. Часть света под воздействием частиц вещества меняет свое направление. Доля рассеянного света определяется как $k_s(x)dx$, где $k_s(x)$ – коэффициент рассеивания. Тогда введем коэффициент затухания $k_t(x) = k_a(x) + k_s(x)$, яркость в точке x находится следующим образом (Закон Бугера-Ламберта-Бера):

$$L(x) = L(x_0)e^{-\int_{x_0}^{x} k_t(u)\,du} = L(x_0)\tau(x_0, x) \tag{1.1}$$

где $L(x)$ – яркость в точке x. Функцию $\tau(x_0, x) = e^{-\int_{x_0}^{x} k_t(u)\,du}$ называют передаточной функцией, она показывает какая доля света останется при прохождении из точки x_0 в точку x [2].

1.1.2 Фазовая функция

Чтобы вывести уравнение переноса (англ. transport equation), необходимо учитывать увеличение яркости из-за эмиссии и рассеивания света в направлении распространения луча. Пространственное распределение рассеянного света моделируется фазовой функцией $p(\vec{\omega}, \vec{\omega}')$. Фазовая функция имеет физическую интерпретацию как интенсивность рассеяния в направлении $\vec{\omega}$, деленная на интенсивность, которая была бы рассеяна в этом направлении, если бы рассеяние было изотропным (т.е. независимым от направления). Фазовые функции в компьютерной графике обычно симметричны относитель-

но направления падения, поэтому их можно параметризовать углом между входящим и исходящим лучами. В качестве фазовой функции для облаков используется функция Хеньи – Гринштейна (1.2):

$$p(\theta) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}} \tag{1.2}$$

где g – варьируемый параметр, причем $-1 \leq g \leq 1$, а θ – угол между входящим и исходящим лучами [3].

1.1.3 Уравнение переноса

Уравнение переноса описывает изменение яркости внутри участвующей среды в точке x:

$$\begin{aligned} \frac{dL(x,\vec{\omega})}{dx} =& k_t(x)J(x,\vec{\omega}) - k_t(x)L(x,\vec{\omega}) = \\ & k_a(x)L_e(x,\vec{\omega}) + \frac{k_s(x)}{4\pi}\int_{4\pi} p(\vec{\omega}',\vec{\omega})L(x,\vec{\omega}')d\omega' - \\ & k_a(x)L(x,\vec{\omega}) - k_s(x)L(x,\vec{\omega}) \end{aligned} \tag{1.3}$$

где $L_e(x,\vec{\omega})$ – яркость выделенного света (эмиссии), а $J(x,\vec{\omega})$ – вносимая яркость. Стоит заметить, что явление эмиссии не относится к облакам, т. к. в облаках не наблюдается люминесценция.

Используя альбедо однократного рассеяния $\Omega = \frac{k_s}{k_t}$, мы можем записать вносимую яркость J, как:

$$J(x,\vec{\omega}) = \underbrace{(1 - \Omega(x))L_e(x,\vec{\omega})}_{J_e(x,\vec{\omega})} + \frac{\Omega(x)}{4\pi}\int_{4\pi} p(\vec{\omega}',\vec{\omega})L(x,\vec{\omega}')d\omega' \tag{1.4}$$

Вносимая яркость описывает вклады яркости в луч $(x,\vec{\omega})$ в точке x внутри среды, которые обуславливаются эмиссией света и рассеянными в рассматриваемом направлении лучами.

Проинтегрировав уравнение переноса (1.3), получим:

$$L(x,\vec{\omega}) = \underbrace{\tau(x_0,x)L(x_0,\vec{\omega})}_{L_{ri}(x,\vec{\omega})} + \underbrace{\int_{x_0}^{x} \tau(u,x)k_t(u)J(u,\vec{\omega})du}_{L_m(x,\vec{\omega})} \tag{1.5}$$

где $L_{ri}(x, \vec{\omega})$ – яркость падающего света, а $L_m(x, \vec{\omega})$ – яркость, обусловленная явлениями внутри среды.

Целью алгоритмов рендеринга является разрешение уравнения (1.5), по крайней мере для точек видимых относительно камеры [2].

1.1.4 Однократное рассеивание

Сложность решения 1.5 заключается в том, что L появляется в обеих частях уравнения (неявно через J). Приближенное решение состоит в том, чтобы учитывать только определенное количество событий рассеяния и рассчитывать затухание луча на промежутках между этими событиями. Считаем вклад яркости за счет явлений в среде равным нулю, тогда:

$$J(x, \vec{\omega}) \approx J_{SS}(x, \vec{\omega}) = J_e(x, \vec{\omega}) + \frac{\Omega(x)}{4\pi} \int_{4\pi} p(\vec{\omega}', \vec{\omega}) L_{ri}(x, \vec{\omega}') d\omega' \qquad (1.6)$$

С учетом 1.6 и 1.5, получаем:

$$L(x, \vec{\omega}) = \tau(x_0, x) L(x_0, \vec{\omega}) + \int_{x_0}^{x} \tau(u, x) k_t(u) J_{SS}(u, \vec{\omega}) du \qquad (1.7)$$

1.1.5 Характеристики участвующей среды

Таким образом, модель участвующей среды, характеризуется: плотностью частиц, коэффициентом поглощения k_a, коэффициентом рассеивания k_s и фазовой функцией [3].

В отличие от поверхностей, чьи геометрические и оптические свойства можно рассматривать отдельно, определение геометрии и оптических свойств участвующих сред тесно связаны. Если коэффициент затухания или плотность частиц среды даны непосредственно как функция от координат, то эта функция определяет и геометрию [2].

1.2 Способы представления облаков

Методы представления должны определять пространственное распределение плотности, форму и поверхность облаков на сцене.

Для представления облаков существуют следующие методы:

— система частиц;

— иерархическое разделение пространства;

— неявное представление;

— объемы, ограниченные поверхностью (англ. Surface-Bounded Volumes).

1.2.1 Система частиц

Система частиц – используемый в компьютерной графике способ представления объектов, не имеющих четких геометрических границ [4]. Облако представляется набором частиц с заранее заданным объемом. Обычно используются сферические частицы с радиальной функцией плотности. Очевидно, что если облако будет состоять из большого количества частиц малого объема, то это приведет к огромным вычислительным затратам. Можно оптимизировать этот метод используя метасферы (от англ. meta-balls). Метасфера определяется своим центром, радиусом и плотностью в центре. Значение плотности в любой точке пространства определяется с помощью функции распределения плотностей от расстояния до центра метасферы [5].

Система частиц представляет собой легкое объемное представление облаков и напрямую поддерживает многие методы облачного моделирования. Сферические частицы хорошо подходят для кучевых или плотных облаков, но менее уместны для слоистых, тонких и перистых облаков.

1.2.2 Иерархическое разделение пространства

Иерархическое разделение пространства создается над регулярной сеткой. При этом существуют следующие способы построения иерархии:

— Воксельные октодеревья. Каждый узел в дереве октантов делит пространство на восемь новых октантов. При этом можно хранить только внешние воксели, которые образуют поверхность облака [3];

— Двоичное разбиение пространства (ДРП). ДРП, например, в виде kd-деревьев, рекурсивно делит пространство пополам, концентрируясь на областях с высокой геометрической детализацией и удаляя пустые области [3];

— BVH (англ. Bounding Volume Hierarchies). BVH окружает геометрические объекты сцены оболочками и выстраивает иерархию в виде древовидной

структуры данных. Такой подход можно использовать для структурирования системы частиц, описанной в 1.2.1.

В общем случае эти методы позволяют представлять все типы облаков. Но в силу того, что они используют регулярную сетку, их прямое использование неэффективно для сцен большого масштаба, ведь тогда регулярная сетка занимает большой объем памяти, причем большинство ячеек сетки могут быть пустыми. Чаще всего в этом случае, сетку вкладывают в сферическую оболочку или используют трассировку лучей, которая учитывает искривление поверхности Земли, таким образом получается существенно снизить затраты по памяти, т. к. снижается размер видимой части сетки без вреда для восприятия сцены (облака плавно скрываются за горизонтом) [3].

1.2.3 Неявное представление

Распространенным способом моделирования и представления поля плотности облака является использование процедурных методов. В то время как общая форма обычно задается простыми геометрическими объектами, такими как сферы или эллипсоиды, внутренняя структура с высоким разрешением моделируется процедурно [3]. Существуют различные способы неявного представления облаков. Например, в [6] использовалась комбинация полигональной сетки и процедурно сгенерированных текстур. Сетка применяется для описания внешней границы облака в низком разрешении. Процедурно сгенерированные 3D-текстуры добавляют деталей к этой границе, при этом среда внутри сетки считается однородной.

Другой подход заключается в использовании двух 3D-текстур, одна из которых состоит из низкочастотного шума и служит для формирования общей формы облака, а другая используется для создания мелких деталей на поверхности. Сами текстуры состоят из шумов Перлина и Ворлея с разными частотами и амплитудами. Форма и положение облака задаются с помощью погодной карты и функции разрежения поля плотности в зависимости от высоты. Также форма и положение облака могут задаваться аналитически с помощью геометрических поверхностей [1; 7].

1.2.4 Объемы, ограниченные поверхностью

Облачный объем представляется окружающей его оболочкой, обычно заданной с помощью полигональной сетки. Поскольку информация о внутренней структуре отсутствует, обычно предполагается однородный объем. Такой подход является компактным и быстрым способом представления облаков, однако он применим только в том случае, если можно предположить быстрое насыщение луча, входящего в объем. Т. о. этот метод применим для плотных облаков, но не подходит для представления тонких или слоистых [3].

1.2.5 Выбор способа представления

В последнее время, крупные компании [1; 7], выпускающие игры, используют для представления облаков процедурную генерацию, основанную на двух 3D-текстурах, описанную в пункте 1.2.3. Такой выбор обусловлен огромной экономией памяти по сравнению с другими методами. Использование памяти для визуализации всего неба ограничено стоимостью двух 3D-текстур и одной 2D-текстуры (погодной карты), т. е. составляет примерно 20 МБ [8]. В то же время для хранения облачного неба в виде системы частиц или с помощью иерархии на регулярной сетке, требуется в десятки раз больше памяти.

1.3 Анализ методов визуализации облаков

Рассмотрим следующие методы рендеринга облаков:

— срез объема (англ. Volume slicing);

— разбрызгивание текстур (англ. Splatting);

— Ray Marching.

1.3.1 Срез объема

Срез объемов – это простой метод рендеринга регулярных сеток. Срезы объема производятся перпендикулярно каждой из главных осей или линии взгляда, и результирующая информация для каждого среза представляется в виде 2D-текстуры. Рендеринг выполняется путем проецирования текстур и их смешивания в буфере кадра [9]. Нарезка объема плоскостями, ориентированными под углом, равным половине угла между направлением освещения

и направлением взгляда, называется нарезкой по половинному углу (англ. Half-Angle Slicing). Такой подход позволяет совместить освещение и визуализацию объема в одном процессе путем однократной итерации по всем срезам. Во время этого единственного прохода объема поддерживаются и итеративно обновляются два буфера: один для накопления ослабления яркости в направлении распространения света, а другой для накопления яркости для наблюдателя. Из-за однократного прохождения через объем схема освещения ограничивается либо прямым, либо обратным рассеянием [3].

Чтобы избежать появления артефактов, рекомендуется использовать множество срезов с малым шагом, что может привести к снижению производительности рендеринга. Передача сложной геометрии объема также может потребовать большого количества срезов. Следовательно, методы, основанные на срезах, предпочтительны для объемов с мягкими или размытыми границами, однако их применимость ограничена в случае, когда объем имеет резкие границы [3].

1.3.2 Разбрызгивание текстур

Разбрызгивание стало распространенным методом рендеринга систем частиц. Частицы, которые обычно определяются как независимые от вращения, могут быть визуализированы с помощью текстурированного четырехугольника, представляющего проекцию частицы на плоскость, также называемую «пятном» или «отпечатком» (от англ. splat и footprint соответственно). Частицы визуализируются в обратном порядке, применяя смешивание текстур в буфере кадра [3; 10].

Частицы представляют некоторый сферический, рассеивающий объем, а не четкий геометрический объект, поэтому этот метод подходит для визуализации облаков с мягкими, пушистыми формами. Облака с четкой геометрией поверхности, такие как кучевые облака, не могут быть воспроизведены реалистично.

1.3.3 Ray Marching

Ray Marching бросает лучи в сцену и накапливает объемную плотность через определенные интервалы. Для визуализации облаков, нужно учесть освещение, это можно сделать либо применив модель освещения во время

трассировки, либо путем извлечения значений из предварительно вычисленной структуры данных освещения [1; 3; 7].

Ray Marching позволяет достичь более реалистичного рендеринга облаков, поскольку он учитывает сложные внутренние структуры объема и свойства облака. Он также способен обрабатывать более сложные геометрические формы облаков, такие как кучевые облака.

1.3.4 Выбор метода визуализации

Для визуализации неявно представленных облаков применяется метод Ray Marching, поскольку поле плотности в таких облаках задается текстурой, а не системой частиц или регулярной сеткой. Хоть данный алгоритм медленнее других, рассмотренных выше, он позволяет учесть некоторые важные физические явления, происходящие в облаках и получить кадр более высокого качества.

1.4 Модель освещения

Для реалистичной визуализации облаков, модель освещения должна аппроксимировать следующие эффекты: множественное рассеяние и направленное освещение в облаках, «серебряное обрамление» (эффект возникающий, когда мы смотрим на солнце через облака), и темные границы облака, когда мы смотрим на облако в направлении от солнца.

Как было описано в пункте 1.1, с фотоном попавшим в облако, может произойти три события:

1) он может поглотиться частицей воды или другой частицей, не относящейся к облаку, например пылью;

2) в ходе отражений, он выходит из облака и движется в сторону наблюдателя. Назовем это внутренним рассеянием;

3) в ходе отражений, он выходит из облака и движется в сторону противоположную от наблюдателя. Назовем это внешним рассеянием.

Закон Бера (1.1) является стандартным методом для аппроксимации вероятности каждого из этих трех явлений. Т. к. облака являются неоднородной средой, мы должны накапливать оптическую плотность вдоль луча. Тогда,

приближенно, считаем энергию накопленную лучом:

$$BL = e^{-d} \tag{1.8}$$

где d – накопленная оптическая плотность.

Для моделирования явления «серебряного обрамления», используются фазовые функции, одной из которых, является функция Хеньи-Гринштейна (1.2).

Закон Бера – это модель поглощения, а это означает, что он описывает то, как световая энергия поглощается при распространении вглубь облака. Но такая модель не учитывает важный световой эффект, который связан с внутренним рассеянием на обращенных к солнцу сторонах облаков. Этот эффект проявляется в виде темных краев на облаках, когда направление луча обзора приближается к направлению луча света. Этот эффект наиболее заметен в круглых, плотных областях облаков, настолько, что складки между каждой выпуклостью кажутся ярче, чем сама выпуклость, которая находится ближе к солнцу. Эти результаты кажутся полной противоположностью тому, что моделирует закон Бера. Напомним, что большее количество света рассеивается вперед, вдоль исходного направления светового луча из-за прямого рассеяния. Должна существовать относительно большая оптическая толщина, чтобы фотон мог повернуться на 180 градусов. Пути около границы облака не проходят через достаточно большую оптическую толщину, чтобы полностью обернуть заметную часть фотонов. Пути, которые имеют оптическую толщину, достаточную для поворота фотона на 180 градусов, почти всегда находятся внутри облака, поэтому закон Бера заглушит этот вклад до того, как частица света покинет облако. Щели и трещины являются исключением, они создают «окно», через которое фотоны могут покинуть облако по пути с малой плотностью, делая расщелины ярче, чем окружающие их выпуклости. Для учета этого эффекта в закон Бера вносят поправку:

$$PSE = 1 - e^{-2d} \tag{1.9}$$

Приближено, считаем световую энергию, накопленную лучом, таким образом:

$$LE = 2 * BL * PSE \tag{1.10}$$

где LE - накопленная энергия света.

В итоге, полностью наша модель освещения описывается следующим образом:

$$E = 2 * e^{-d} * (1 - 2e^{-2d}) * \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\cos\theta)^{3/2}} \tag{1.11}$$

где g – некоторая константа в функции Хеньи-Гринштейна, а θ – угол между лучами взгляда и света.

1.4.1 Расчет освещения

Очевидный способ найти количество падающего света – измерить переносимую световую энергию внутри облака между рассматриваемой точкой и солнцем. Однако на освещенность любой точки облака сильно влияет освещенность в областях вокруг нее в направлении источника света, т. е. нам необходимо рассмотреть конус с вершиной в рассматриваемой точке. Внутри конуса мы выбираем случайным образом, некоторое количество точек, в которых рассчитываем плотность. Суммируя плотности в точках, мы получаем плотность накопленную в конусе, далее используем ее, чтобы вычислить световую энергию в рассматриваемой точке.

Для более грубого расчета освещения, можно накапливать плотность не внутри конуса, а в точках на луче, который начинается в освещаемой точке и направлен к источнику света.

1.5 Виды шумов, применяемых в процедурной генерации облаков

1.5.1 Шум Ворлея

Для генерации шума Ворлея необходимо создать регулярную сетку некоторого размера и разместить в центре каждой ячейки точку. Далее используя хеш-функцию, мы сдвигаем каждую точку в пределах ячейки. Теперь, чтобы рассчитать шум для любого пикселя, нужно определить в какую ячейку он попадает и какие ячейки ему смежны. Далее проходя по всем ячейкам, мы определяем расстояние между нашим пикселем и смещенными точками, значение шума Вороного – минимальное из расстояний, а инвертировав шум Вороного, получаем искомый шум Ворлея. Недостаток – этот шум не скла-

дывается воедино, т. е. при сложении текстур созданных с помощью шума, получаются резкие стыки, чтобы это исправить, необходимо брать смещение точки по модулю [11].

1.5.2 Шум Перлина

Шум Перлина – это градиентный шум, состоящий из набора псевдослучайных единичных векторов (направлений градиента), расположенных в определенных точках пространства и интерполированных функцией сглаживания между этими точками. Беря по модулю псевдослучайные числа, добиваемся того, что текстуры созданные с помощью шума, складываются воедино.

Вывод

В данном разделе был проведен анализ способов хранения поля плотности облаков, методов визуализации и модели освещения, которые возможно использовать для решения поставленных задач. В итоге была выбрана связка из неявного представления облаков с алгоритмом Ray Marching, так как такой подход позволяет достичь высокой реалистичности, а также точности построенного изображения.

2 Конструкторский раздел

В данном разделе представлены требования к программному обеспечению, а также схемы алгоритмов, выбранных для решения поставленной задачи.

2.1 Требования к программному обеспечению

Программа должна предоставлять доступ к функционалу:

— задать направление взгляда камеры;

— изменение положения источника света;

— конфигурация облачного неба с помощью загрузки погодной карты;

— варьирование параметров облачного неба.

К программе предъявляются следующие требования:

— время отклика программы должно быть менее 50 мс [7];

— программа должна корректно реагировать на любые действия пользователя.

2.2 Разработка алгоритмов

Алгоритм Ray Marching применяется для визуализации облачного неба.

Для повышения эффективности алгоритма Ray Marching необходимо снизить количество шагов вне облака. Для этого применяются объемлющие оболочки, в данном случае, такими оболочками будут являться сферы. Когда луч пересекает оболочку, необходимо сделать N шагов, на каждом из которых вычисляется плотность в точке пространства и ее освещенность, используя эти данные итеративно вычисляется значение цвета пикселя.

Как показано на рисунке 2.1, атмосфера будет моделироваться с помощью двух концентрических сфер, между которыми и будет происходить генерация облаков.

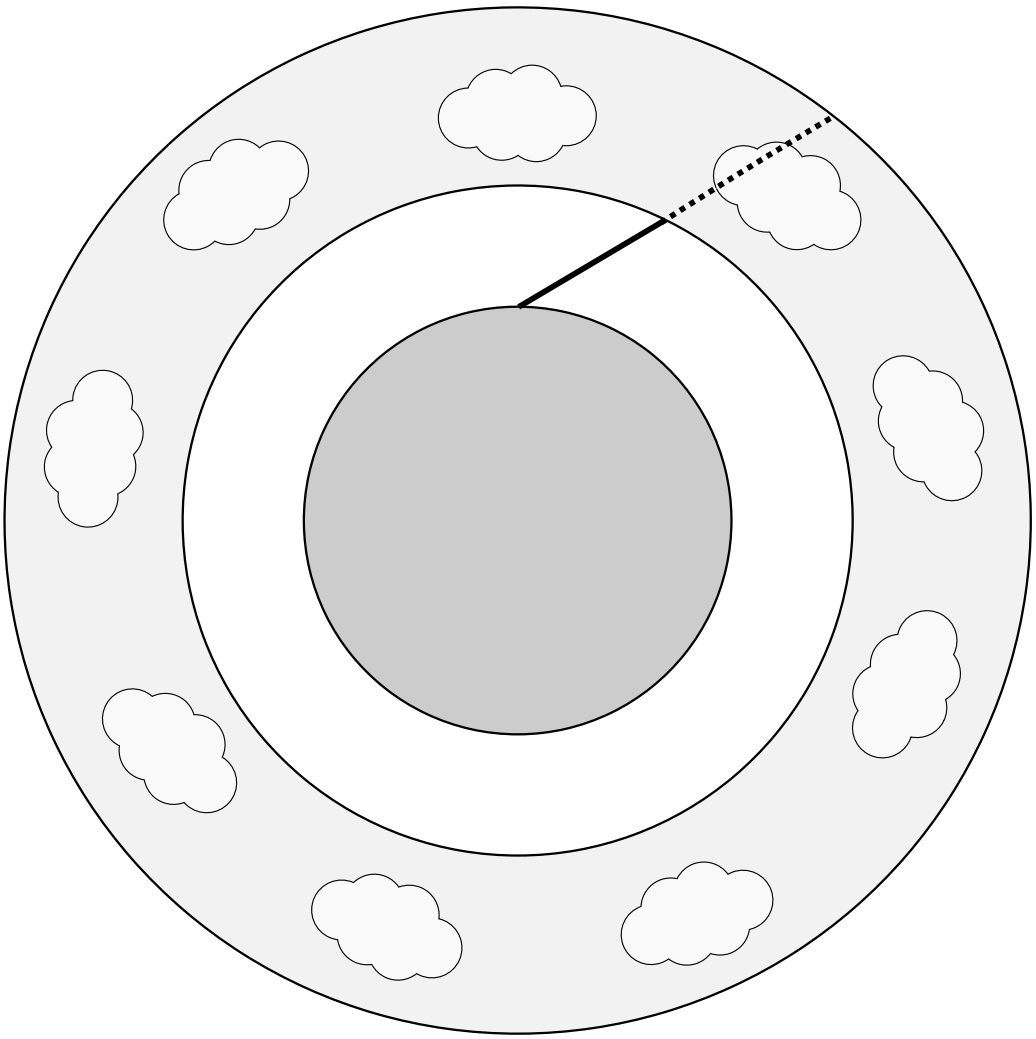

Рисунок 2.1 – Схема атмосферы и положения наблюдателя

2.2.1 Пересечение луча со сферой

Уравнение луча запишем следующим образом:

$$P = S + t\vec{D}, t \geq 0 \tag{2.1}$$

где S - точка, откуда луч испускается, а $\vec{D}$ - направление луча.

Пусть сфера задается своим центром C и радиусом r. Если луч пересекает сферу, тогда точка P - лежит на поверхности, запишем это следующим образом:

$$\|P - C\| = r \tag{2.2}$$

Перепишем (2.2), используя скалярное произведение:

$$\sqrt{\langle P - C, P - C \rangle} = r \tag{2.3}$$

Подставим в (2.3) уравнение луча (2.1):

$$\sqrt{\langle S + t\vec{D} - C, S + t\vec{D} - C\rangle} = r \tag{2.4}$$

Обозначим $S - C = \vec{SC}$ и раскроем скалярное произведение, возведя обе части уравнения (2.4) в квадрат:

$$t^2\langle\vec{D}, \vec{D}\rangle + 2t\langle\vec{SC}, \vec{D}\rangle + \langle\vec{SC}, \vec{SC}\rangle - r^2 = 0 \tag{2.5}$$

Решая квадратное уравнение (2.5) находим точки пересечения луча с объемлющей оболочкой.

2.2.2 Вычисление плотности облаков в атмосфере

Как было описано в пункте 1.2.3, для представления поля плотности облаков будем хранить две объемные текстуры. Первая (основная) текстура отвечает за низкочастотный шум, имеет размер $128 \times 128 \times 128$. Вторая (вспомогательная) текстура отвечает за высокочастотный шум, имеет размер $32 \times 32 \times 32$. В таблице 2.1 показаны какие шумы используются для формирования текстур.

Таблица 2.1 – Таблица шумов.

Текстура	Шумы			
Основная	П-В (НЧ)	В (НЧ)	В (СЧ)	В (ВЧ)
Вспомогательная	В (НЧ)	В (СЧ)	В (ВЧ)	-

Примечание: П-В – шум Перлина-Ворлея, В – шум Ворлея, НЧ – низкая частота, СЧ – средняя частота, ВЧ – высокая частота.

В итоге, чтобы рассчитать плотность в некоторой точке, необходимо получить срезы шумов из текстуры и составить из них fBM (англ. fractal Brownian motion). FBM представляет собой сумму ряда октав шума, каждая из которых имеет более высокую частоту и более низкую амплитуду.

Основная текстура применяется для формирования общей грубой формы облака и может использоваться для определения факта попадания луча в облако. Вспомогательная текстура используется для добавления деталей на поверхности облака.

Для возможности формировать облачное небо используются погодные карты. Погодная карта является двумерной текстурой, и состоит из трех кана-

лов RGB, где в R канале хранится процент покрытия неба облаками, G хранит тип облаков, B канал может быть использован для хранения вероятности возникновения дождя, но в данном алгоритме он не будет использоваться.

Тип облаков будем задавать с помощью функции, которая рассеивает облако в зависимости от его высоты.

В итоге плотность в данной точке вычисляется с помощью базовой плотности, полученной из текстуры, затухания, полученного из функции типа облака и R канала погодной карты.

2.2.3 Общий алгоритм построения изображения

Общий алгоритм построения изображения показан на рисунке 2.2.

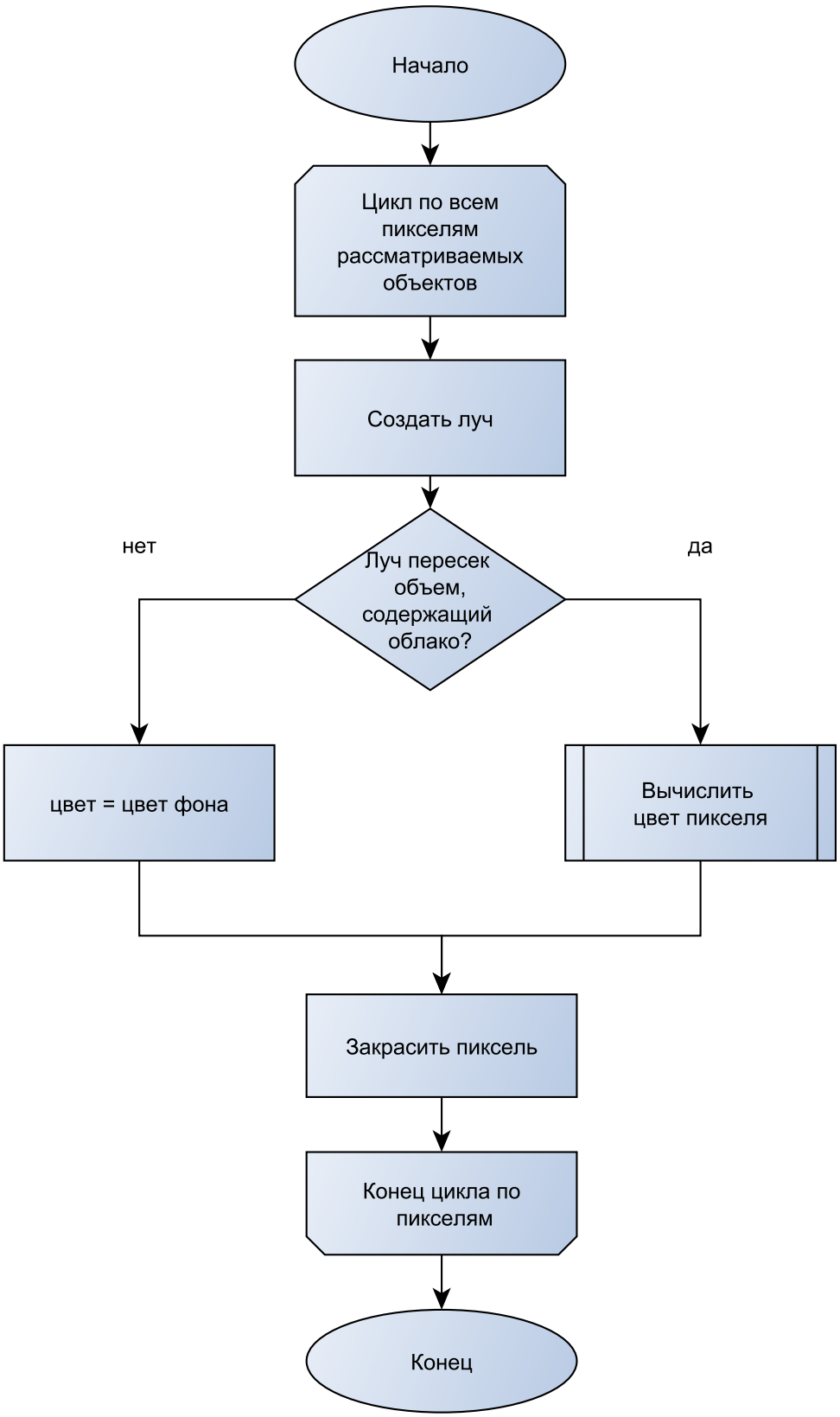

Рисунок 2.2 – Схема визуализации облаков с помощью алгоритма Ray Marching

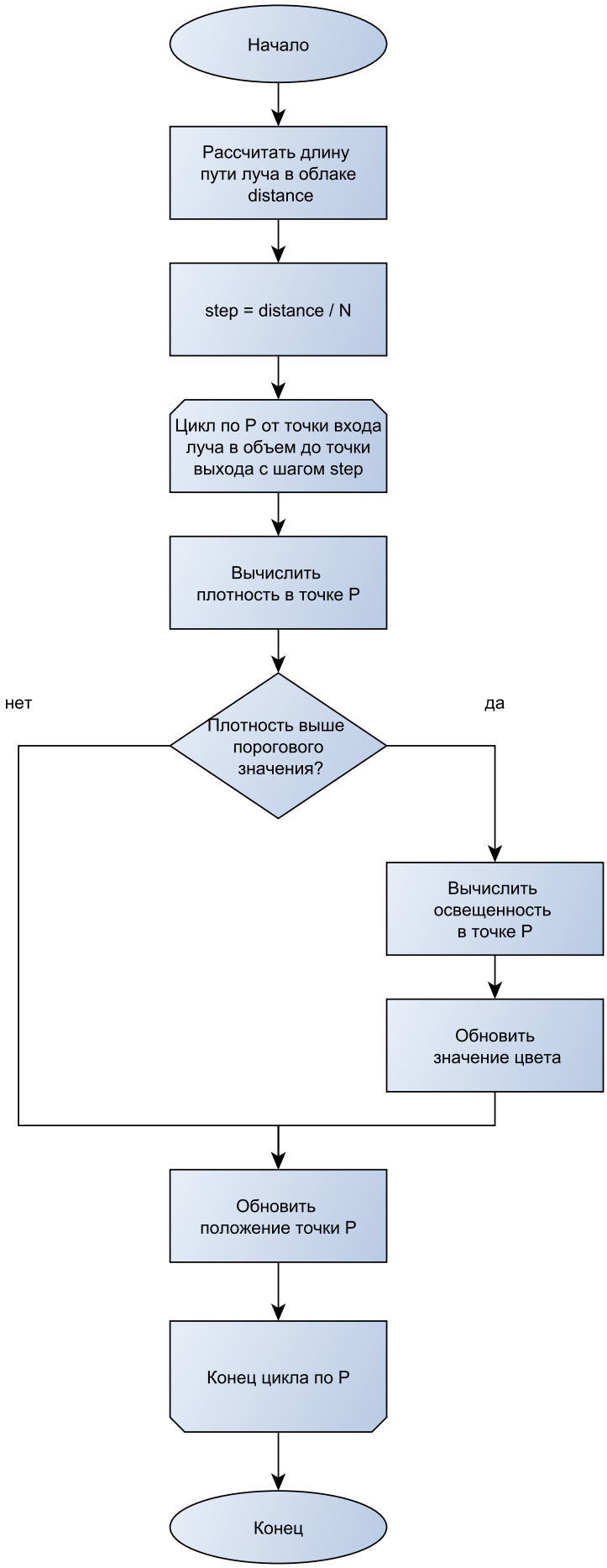

Рисунок 2.3 – Схема расчета цвета пикселя

2.3 Выбор используемых типов и структур данных

В данной работе используются следующие типы и структуры данных:

1) источник света – задается вектором направления и интенсивностью;
2) облака – задаются с помощью объемных текстур, погодной карты, функций рассеивания;
3) текстура – задается с помощью двумерных и трехмерных массивов, состоящих из цветов;
4) цвет – хранит три или четыре составляющие RGB или RGBA модели цвета соответственно;
5) математические абстракции:
   - точка – хранит координаты x, y, z;
   - вектор – хранит направление по x, y, z.

Вывод

В данном разделе были представлены требования к разрабатываемому программному обеспечению и разработана схема разрабатываемого алгоритма. Так же, были описаны структуры данных, которые будут использоваться при реализации программного обеспечения.

3 Технологический раздел

В данном разделе будут представлены средства разработки программного обеспечения, детали реализации и процесс сборки разрабатываемого программного обеспечения.

3.1 Выбор средств реализации

В качестве языка программирования для разработки программного обеспечения был выбран язык Python версии 3 [12]. Данный выбор обусловлен тем, что данный язык предоставляет весь функционал требуемый для решения поставленной задачи.

Визуализации производится с помощью библиотеки ModernGL [13]. Данная библиотека дает возможность производить вычисления параллельно на видеокарте, что позволяет ускорить работу программы в разы.

3.2 Реализация алгоритмов

На листингах **??** – **??**, приведенных ниже, описаны основные алгоритмы используемые для визуализации облаков на языке шейдеров GLSL.

Листинг 3.1 – Реализация функции определяющей плотность в точке

```
float cloudGetHeight(vec3 position, vec2 cloudMinMax){
    return (position.y - cloudMinMax.x) / (cloudMinMax.y -
       cloudMinMax.x);
}

float saturate(float height){
    height -= 0.4;
    height *= 100;
    float v = 2 * 2 * 2 / (height * height  + 2 * 2);
    v *= 100;

    return clamp(v, 0, 1);
}

float remap(float value, float minValue, float maxValue, float
   newMinValue, float newMaxValue)
{
    return newMinValue+(value-minValue) /
        (maxValue-minValue)*(newMaxValue-newMinValue);
}

float cloudSampleDensity(vec3 position, vec2 cloudMinMax)
{
    position.xz+=vec2(0.2)*u_time;
    float base = texture(u_lfNoise, position / 48).r;
    float height = cloudGetHeight(position, cloudMinMax);

    float coverage = texture(u_weatherMap, position.xz / 480).r;
    float coff = saturate(height);

    base *= coff;

    float baseCloudWithCoverage = remap(base, 1-coverage, 1, 0,
       1);

    baseCloudWithCoverage *= coverage;

    float hfFBM = texture(u_hfNoise, position / 48).r;
    float hfNoiseModifier = mix(hfFBM, 1 - hfFBM, clamp(height *
       10, 0, 1));
```

```
37
38      float finalCloud = remap(baseCloudWithCoverage,
            hfNoiseModifier * 0.2, 1, 0, 1);
39
40
41      return max(finalCloud, 0);
42  }
```

Листинг 3.2 – Реализация алгоритма Ray Marching

```
const vec3 EXTINCTION_MULT = vec3(0.8, 0.8, 1.0);
const float CLOUD_LIGHT_MULTIPLIER = 50.0;

vec4 mainMarching(vec3 ro, vec3 viewDir, vec3 sunDir, vec3
   sunColor, vec3 ambientColor)
{
    vec2 t = rsi(ro, viewDir, minCloud);
    vec3 position = ro + viewDir * t.y;

    vec2 t2 = rsi(ro, viewDir, maxCloud);
    vec3 position2 = ro + viewDir * t2.y;

    float avrStep = (maxCloud - minCloud) / 64;

    vec2 cloudMinMax;
    cloudMinMax.x = position.y;
    cloudMinMax.y = position2.y;

    vec3 iPos = position;

    float mu = dot(viewDir, sunDir);

    vec3 transmittance = vec3(1);
    vec3 scattering = vec3(0);

    vec3 sunLight = sunColor * CLOUD_LIGHT_MULTIPLIER;
    vec3 ambient = vec3(AMBIENT_STRENGTH * sunColor) * u_ambient;

    for (int i = 0; i < 128; ++i){
        if (length(iPos) > maxCloud) break;

        float density = cloudSampleDensity(iPos, cloudMinMax);

        if (density > 0.01){
            vec3 luminance = ambient + sunLight *
               calculateLightEnergy(iPos, sunDir, mu,
               cloudMinMax);
            vec3 ttransmittance = exp(-density * avrStep *
               EXTINCTION_MULT * u_attenuation);
            vec3 integScatt = density * (luminance - luminance *
```

```
                ttransmittance) / density;
37
38              scattering += transmittance * integScatt;
39              transmittance *= ttransmittance;
40          if (length(transmittance) <= 0.01) {
41                  transmittance = vec3(0.0);
42                  break;
43              }
44          }
45          iPos += viewDir * avrStep;
46      }
47
48      transmittance = saturate3(transmittance);
49      vec3 color = ambientColor.xyz * transmittance + scattering;
50
51      return vec4(color, 1);
52  }
```

Листинг 3.3 – Реализация модели освещения

```
float HenyeyGreenstein(float g, float mu) {
    float gg = g * g;
    return (1.0 / (4.0 * PI))  * ((1.0 - gg) / pow(1.0 + gg -
       2.0 * g * mu, 1.5));
}

float cloudSampleDirectDensity(vec3 position, vec3 sunDir, vec2
   cloudMinMax)
{
    float avrStep=(cloudMinMax.y - cloudMinMax.x)*0.01;
    float sumDensity=0.0;

    for(int i=0;i<4;i++)
    {
        float step=avrStep;

        if (i==3)
            step=step*6.0;

        position+=sunDir*step;
        float density=cloudSampleDensity(position,
           cloudMinMax)*step;
        sumDensity+=density;
    }
    return sumDensity;
}

vec3 calculateLightEnergy(vec3 position, vec3 sunDir, float mu,
   vec2 cloudMinMax) {

    float density = cloudSampleDirectDensity(position, sunDir,
       cloudMinMax)* u_attenuation2;
    vec3 beersLaw = exp(-density * EXTINCTION_MULT * a) *
       HenyeyGreenstein(u_eccentrisy2, mu);
    vec3 powder = 1.0 - exp(-density * 2.0 * EXTINCTION_MULT);

    return beersLaw * mix(2.0 * powder, vec3(1.0), remap(mu,
       -1.0, 1.0, 0.0, 1.0));
}
```

3.3 Вывод

В данном разделе были представлены средства разработки программного обеспечения и детали реализации. В итоге был получен рендер следующего кадра:

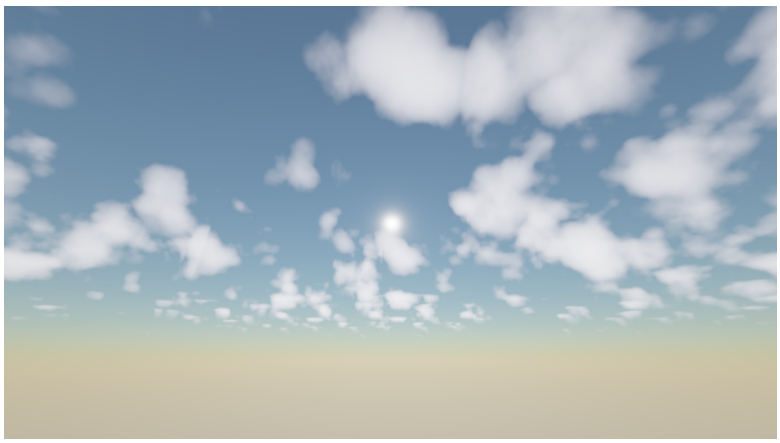

Рисунок 3.1 – Кадр, облачного неба, созданный с помощью описанных алгоритмов

4 Исследовательский раздел

ЗАКЛЮЧЕНИЕ

В процессе выполнения поставленных задач были рассмотрены различные алгоритмы визуализации облаков и способы их представления. Изучены явления, происходящие в облаках, и на их основе разработана модель освещения.

Подробно были изучены алгоритм Ray Marching и неявное представление облаков. Было спроектировано программное обеспечение, позволяющее визуализировать облачное небо.

На основе проделанной работы было разработано программное обеспечение, которое строит кадр, содержащий облачное небо, приемлемого качества. В дальнейшем данное программное обеспечение можно модифицировать, добавляя интерфейс и подбирая параметры облаков.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Physically Based Sky, Atmosphere and Cloud Rendering in Frostbite. — 2016. — Режим доступа: https://media.contentapi.ea.com/content/dam/eacom/frostbite/files/s2016-pbs-frostbite-sky-clouds-new.pdf (дата обращения: 16.07.2023).
2. A Survey on Participating Media Rendering Techniques. — 2005. — Режим доступа: https://www.researchgate.net/publication/47407939_A_Survey_on_Participating_Media_Rendering_Techniques (дата обращения: 23.07.2023).
3. A Survey of Cloud Lighting and Rendering Techniques. — 2019. — Режим доступа: https://dspace5.zcu.cz/bitstream/11025/1082/1/Hufnagel.pdf (дата обращения: 18.07.2023).
4. КОМПОНЕНТЫ И КЛАССИФИКАЦИЯ ЧАСТИЦ И ИХ СИСТЕМ В КОМПЬЮТЕРНОЙ ГРАФИКЕ. — 2016. — Режим доступа: https://www.elibrary.ru/item.asp?id=26847523 (дата обращения: 13.07.2023).
5. Display of Clouds Taking into Account Multiple Anisotropic Scattering and Sky Light. — 1996. — Режим доступа: https://www.researchgate.net/publication/220720838_Display_of_Clouds_Taking_into_Account_Multiple_Anisotropic_Scattering_and_Sky_Light (дата обращения: 27.07.2023).
6. Interactive multiple anisotropic scattering in clouds. — 2008. — Режим доступа: https://inria.hal.science/inria-00333007/document (дата обращения: 28.07.2023).
7. The Real-Time volumetric cloudscape of Horizon Zero Dawn. — 2015. — Режим доступа: http://killzone.dl.playstation.net/killzone/horizonzerodawn/presentations/Siggraph15_Schneider_Real-Time_Volumetric_Cloudscapes_of_Horizon_Zero_Dawn.pdf (дата обращения: 16.07.2023).
8. The Current State of the Art in Real-Time Cloud Rendering With Raymarching. — 2020. — Режим доступа: https://www.researchgate.net/publication/343404421_The_Current_State_of_the_Art_in_

Real - Time _ Cloud _ Rendering _ With _ Raymarching (дата обращения: 30.07.2023).

9. MC Slicing for Volume Rendering Applications. — 2005. — Режим доступа: https : / / www . researchgate . net / publication / 226214561 _ MC _ Slicing _ for _ Volume _ Rendering _ Applications (дата обращения: 20.07.2023).
10. A simple, efficient method for realistic animation of clouds. — 2000. — Режим доступа: https://www.researchgate.net/publication/220721744_A_simple_efficient_method_for_realistic_animation_of_clouds (дата обращения: 13.07.2023).
11. A Cellular Texture Basis Function. — 1996. — Режим доступа: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.412&rep=rep1&type=pdf (дата обращения: 26.07.2023).
12. Python. — 2001. — Режим доступа: https://www.python.org/ (дата обращения: 20.08.2023).
13. ModernGL. — 2022. — Режим доступа: https://moderngl.readthedocs.io/en/latest/index.html (дата обращения: 20.08.2023).