УДК 004.92 ISSN 1995-5499
DOI: https://doi.org/10.17308/sait/1995-5499/2022/2/93-109
Поступила в редакцию 30.11.2021
Подписана в печать 29.06.2022

ПРОЦЕДУРНАЯ ГЕНЕРАЦИЯ ОБЪЕМНЫХ ОБЛАКОВ В РЕАЛЬНОМ ВРЕМЕНИ

© 2022 А. С. Сырых✉, С. Н. Медведев

Воронежский государственный университет
Университетская пл., 1, 394018 Воронеж, Российская Федерация

Аннотация. В статье представлен разработанный алгоритм генерации реалистичных трехмерных облаков в реальном времени. Приведен подробный анализ существующих решений по генерации облаков. Исходя из анализа, для дальнейшей работы была выбрана базовая технология Nubis. В статье описывается теоретический материал, необходимый для реализации алгоритма. Разработанный алгоритм подробно описывается, и для каждого его этапа демонстрируются результаты работы на основе реализованного программного обеспечения. Важной частью статьи является подробное описание вычислительного эксперимента по настройке параметров алгоритма. На основе анализа полученных результатов делаются выводы о работе алгоритма в реальном времени и предлагаются необходимые настройки параметров для качественной работы алгоритма.
Ключевые слова: процедурная генерация, марширующие лучи, шум Перлина, шум Уорли, шум Курла.

ВВЕДЕНИЕ

Исследование и реализация алгоритмов генерации реалистичных объемных облаков является актуальной задачей в сфере компьютерной графики. Облака используются в отраслях игровой и киноиндустрии для придания дополнительной эмоциональной окраски и реалистичности сцены. Правильно подобранные погодные условия и облачное покрытие могут придать абсолютно иное видение сцены, передать более глубокие эмоции и переживания.

Одним из самых важных аспектов в видеоиграх является формирование кадра за малый промежуток времени, то есть за реальное время. Считается, что для рендеринга кадра в реальном времени необходимо 16 мс.

Таким образом, ставится задача реализации алгоритма построения объемных облаков за реальное время, то есть за время, которое человеческий глаз не успеет заметить.

Алгоритм построения облаков должен удовлетворять следующим требованиям:

1. Необходимо иметь возможность управлять размером и областью покрытия облаков в реальном времени.

2. Необходимо имитировать освещение близкое к физически-корректному и изменять его в реальном времени.

3. Необходимо поддерживать некоторый спектр облаков [1] (слоистые, дождевые, кучево-дождевые) в любое время суток.

4. Необходимо иметь возможность плавного перехода между временами суток, а также от одних погодных условий к другим в реальном времени.

На рис. 1 показан пример возможных сгенерированных облаков.

Рис. 1. Возможная генерация облаков
[Fig. 1. Possible cloud generation]

✉ Сырых Александр Сергеевич
e-mail: zsyryhz@mail.ru

Здесь можно наблюдать разреженное облачное небо. Каждое облако имеет мелкие детали и завихрения. Также виден расчет освещения, о чем свидетельствует разность в цвете между левым нижним и правым верхним углами. Такую генерацию можно считать приемлемой.

1. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Для рендеринга облаков существует множество решений. Однако глобально можно выделить две большие группы: с применением объемной генерации и без применения объемной генерации.

1.1. Технологии рендеринга облаков без применения объемной генерации

При рендеринге облаков без применения объемной генерации выделяют два метода: расстановка двумерных изображений облаков в игровом мире и запекание текстуры облаков в Skybox.

При использовании двухмерных изображений облаков, то есть фотографий, происходит их наложение на прямоугольник. Далее этот прямоугольник копируется, и полученные прямоугольники расставляются в игровом мире так, как задумал разработчик. Однако этот метод содержит недостатки:

1) разработчик должен обладать обширной библиотекой изображений правильно отформатированных облаков;

2) при движении игрока облака начинают «рассыпаться» и перестают быть реалистичными.

Проблема, связанная с хранением обширной библиотеки изображений, остается актуальной до сих пор. Для ее решения, например, вместо множества изображений используют несколько фотографий, которые расставляются под разными углами.

Проблему, связанную с нереалистичностью при движении, решают согласованными поворотами прямоугольников, на которые наложены текстуры облаков, по мере движения игрока по игровому миру.

Такой подход облачного рендеринга является малореалистичным, поскольку облака не могут менять свою форму по мере своего движения, нет возможности рассчитывать освещение, а также будет не просто сменить один тип облаков на другой.

Таким образом, в качестве достоинств метода можно выделить простоту реализации, а в качестве недостатков метода можно выделить малую реалистичность и необходимость наличия обширной библиотеки изображений облаков.

При использовании Skybox игрок постоянно находиться в его центре. Это дает возможность корректно реагировать на повороты игрока.

В качестве преимуществ данного метода можно выделить простоту в реализации и малую вычислительную нагрузку, а к недостаткам можно отнести невозможность динамической смены облачного покрытия и малую реалистичность.

1.2. Технологии рендеринга облаков с применением объемной генерации

При рендеринге облаков с применением объемной генерации облака становятся полноценными объектами игрового мира: их можно перемещать, вращать и масштабировать и накладывать на них различные текстуры. Практически каждая игровая студия разработала свои алгоритмы генерации объемных облаков. Ниже будут рассмотрены некоторые технологии, которые зарекомендовали себя с течением времени.

В игре Reset 2012 года выпуска компании Flying Wild Hog технология рендеринга облаков подразумевает работу с тремя слоями.

Верхним слоем является небо, свободное от каких-либо погодных явлений. В среднем слое содержатся объемные облака, формы которых определяются двумя текстурами шума. Погодные явления, такие как туман, моделируются в нижнем слое. Причем дальность видимости в тумане рассчитывается по экспоненциальному закону. Для расчета освещения облаков динамически генерируется две трехмерные текстуры шума. При их создании используется технология Opacity shadow maps [2].

Результат рендера облаков по данной технологии можно наблюдать на рис. 2.

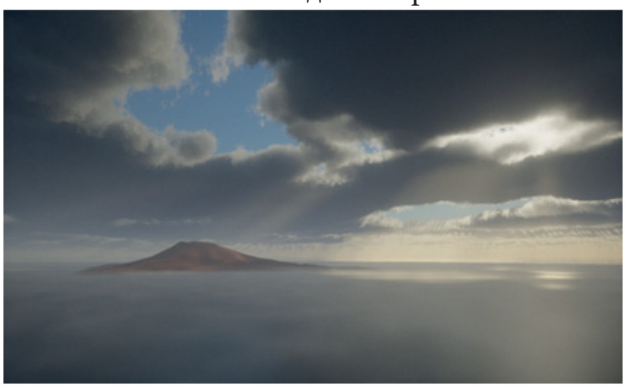

Рис. 2. Объемные облака в игре Reset
[Fig. 2. Volumetric cloud in the Reset game]

Здесь облака имеют визуальный объем с корректно-рассчитанным освещением.

К достоинствам метода можно отнести визуальный объем облаков и корректный расчет их освещения.

К недостаткам можно отнести сложность в реализации и динамический пересчет нескольких текстур шума, что может сказаться на производительности.

В серии игр Red Dead Redemption 2010 и 2018 годов выпуска компании Rockstar Games с помощью RGB текстуры и текстуры шума генерируется погодная карта размером равной всему игровому миру. Погодная карта учитывает погодные условия и время суток, и задает параметры облачного покрова. А с помощью функции градиента [3] облакам придается объем: они становятся слоистыми и кучевыми. Пример данного рендера облаков можно наблюдать на рис. 3.

Рис. 3. Объемные облака в серии игр Red Dead Redemption
[Fig. 3. Volumetric cloud in the Red Dead Redemption game series]

Здесь видна возможность управления формой облаков, благодаря карте погоды. Также видны мелкие завихрения и «пушистость» облаков.

В качестве достоинств технологии можно отметить получение практически любой формы облаков, а также наличие мелких деталей на облаках, такие как завихрения. Также к достоинствам можно отнести близкий к физически-корректному расчет освещения.

В качестве недостатков данной технологии можно выделить недостаток информации из-за ограниченного доступа к технологии.

В игре The Witness 2016 года компании Thekla Inc в основе технологии рендеринга облаков лежат пересекающиеся плоскости (рис. 4). В каждой плоскости лежит та текстура облаков, которая лучше всего подходит для стилистики игры в данный момент.

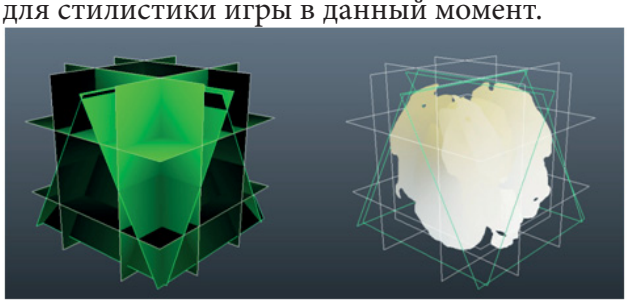

Рис. 4. Пересекающиеся плоскости
[Fig. 4. Intersecting planes]

Здесь слева можно наблюдать пересекающиеся плоскости содержащие альфа-канал, а справа — получившееся облако с отсеченным альфа-каналом.

Облака визуализируются в два прохода [4]. В первом проходе используется альфа-канал для создания карты покрытия облаков, а во втором — технология альфа-перекрытия для их смешивания. Объемные облака в игре The Witness можно наблюдать на рис. 5.

Здесь можно наблюдать сформированные объемные облака с рассчитанным освещением.

В качестве достоинства метода можно выделить стилизованность под любую игру. В качестве недостатка можно выделить необходимость некоторой библиотеки правильно отформатированных изображений облаков и недостаточную реалистичность.

В игровом движке Frostbite компании EA существует возможность создавать облака

Рис. 5. Объемные облака в игре The Witness
[Fig. 5. Volumetric cloud in The Witness game]

«из коробки» путем пары кликов. При рендиренге используется две трехмерные текстуры шума (шум Перлина и шум Уорли) и алгоритм марширующих лучей для расчета плотности облаков и их освещенности. Результат рендеринга облаков можно наблюдать на рис. 6.

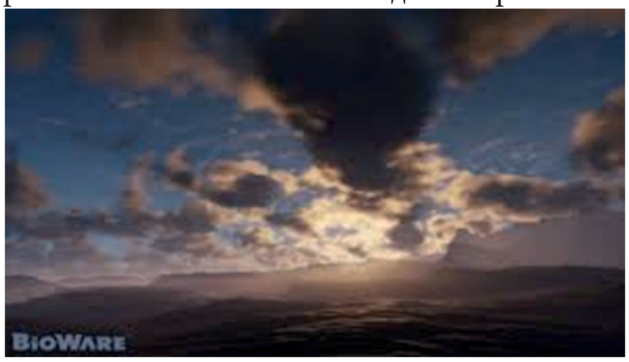

Рис. 6. Объемные облака
в игровом движке Frostbite
[Fig. 6. Volumetric cloud
in the Frostbite game engine]

К достоинству метода можно отнести простоту в использовании при создании некоторого проекта на Frostbie, а к недостаткам можно отнести недостаток информации из-за закрытости технологии.

В игре Sea of Thieves 2018 года компании Rare облака — это полноценные объемные игровые объекты, состоящие из полигональных сеток. Рендеринг облаков происходит в три прохода. Для быстрого рендеринга облаков [5] в первом проходе используется отрисовка полигональных облаков во внеэкранный буфер, размер которого уменьшен до четверти экранного размера. Здесь происходит небольшое размытие по Гауссу глубины облаков. Для придания облакам пушистости происходит альфа-смешивание. Оно сэмплирует подготовленные данные вне экранного буфера и объединяет со всеми остальными объектами. Для придания облакам небольших завихрений и искажений используется карта шумов. Для придания облакам освещения во втором проходе смешивается цвет неба с цветом облака. И, наконец, в финальном третьем проходе накладывается небольшой низко атмосферный туман, дальность видимости которого рассчитывается по экспоненциальному закону.

Заметим, что сгенерированные облака могут принимать любую форму. Результат можно наблюдать на рис. 7.

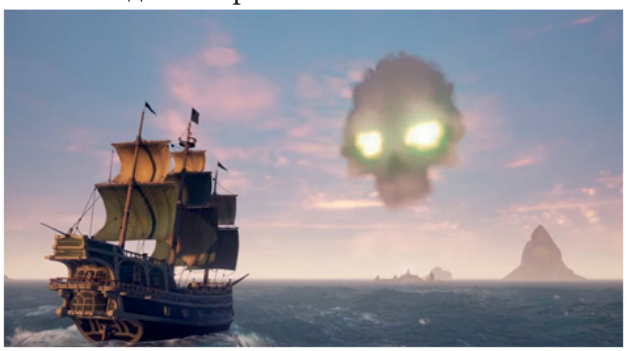

Рис. 7. Облако в форме черепа
[Fig. 7. Cloud in the shape of a skull]

В качестве достоинств данного метода можно выделить стилизованность облаков под любой предмет, а также их динамическую смену.

К недостатку можно отнести высокие вычислительные затраты из-за использования полигональных сеток, поскольку каждое облако может содержать более тысячи примитивов.

Большой вклад в развитие рендеринга объемных облаков внесла Нидерландская компания Guerilla Games [6]. После разработки игры Horizon Zero Dawn 2017 года выпуска разработчики внедрили в свой движок Decima систему процедурной генерации объемных облаков — Nubis. Система Nubis создает облака в несколько этапов [7]. На первом этапе система определяет, где и какие облака нужно сгенерировать с помощью специальной текстуры, которая называется картой погоды. На втором этапе с помощью несколь-

ких 3D текстур шумов [8] облакам придаются формы и мелкие детали. Наконец, на третьем этапе рассчитывается рассеивание и поглощение света облаками. После этого облака отрисовываются шейдером. Сгенерированные таким способом облака можно наблюдать на рис. 8.

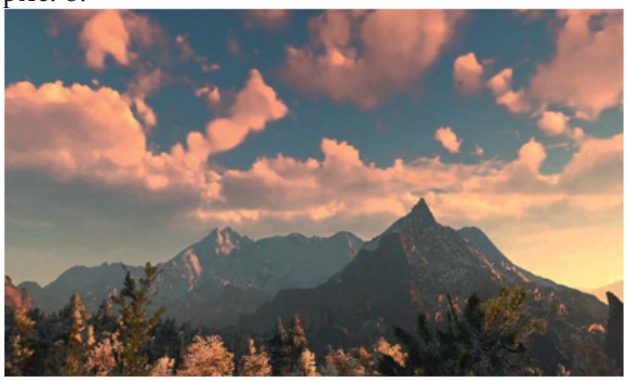

Рис. 8. Объемные облака, сгенерированные системой Nubis
[Fig. 8. Volumetric clouds generated by the Nubis system]

Здесь видны облака с рассчитанным освещением. Также видна некоторая пушистость облаков и заволакивание неба, что можно наблюдать на горизонте за горами.

К достоинствам технологии можно отнести динамическую смену облачного покрытия, расчет освещения близкого к физически-корректному. Также к достоинствам можно отнести открытость этой технологии.

К недостатку можно отнести затраты памяти из-за наличия нескольких 3D текстур шумов.

1.3. Анализ рассмотренных решений

Технологии и подходы без применения объемной генерации не подходят под требования, указанные в постановке задаче, таким образом, уберем их из дальнейшего рассмотрения.

В табл. 1 представлен сравнительный анализ технологий рендеринга объемных облаков в реальном времени. Она позволяет наглядно увидеть преимущества и недостатки каждого из подходов.

На основе анализа табл. 1 можно сделать следующие выводы.

Алгоритм построения облаков с помощью пересекающихся плоскостей, который был применен в игре The Witness, был убран из рассмотрения в связи с ограничениями, которые были поставлены в постановке задачи. А именно, хотя при данном подходе всегда получаются красивые результаты, которые подойдут под любую стилистику игры, но управлять такими облаками достаточно

Таблица 1. Технологии рендеринга объемных облаков
[Table 1. Volumetric cloud rendering technologies]

	Возможность смены облачного покрытия	Возможность управления погодными условиями	Возможность управления формой облака	Наличие документации	Возможность управления освещением	Большая ресурсоемкость
The Witness	–	–	+	+	+	+
Reset	+	+	+	+	+	+
Sea of Theives	–	+	+	+	+	+
Red Dead Redemption	+	+	+	–	+	–
Frostbite	+	+	+	–	+	–
Nubis	+	+	+	+	+	–

сложно. Поскольку на пересекающихся плоскостях находятся отформатированные изображения, то нельзя изменять тип облачного покрытия и грамотно управлять освещением облаков. Также необходимо хранить некоторую библиотеку текстур облаков.

Также была убрана из рассмотрения технология генерации облаков игры Reset. Несмотря на использование 3D текстуры генерации облаков управлять облачным покрытием и освещением все так же тяжело.

Техника реализации объемных облаков в игре Sea of Thieves не подходит из-за использования полигональных сеток и стилизованности под конкретную игру. Вызывать трудности будет реализация плавного перетекания одного облака в другое. Однако таким способом можно получить облака любой формы.

В серии игр Red Dead Redemption из-за ограниченного доступа к технологии кроме визуального сравнения тяжело дать техническое сравнение метода с другими.

Облачные системы игрового движка Frostbite и технологии Nubis очень похожи, но технология рендеринга во Frostbite не так сильно документирована, как в Nubis.

Облачная система Nubis использует не только 3D текстуры шумов, которые задают тип и вид облака, но и карту погоды. Эта текстура, как и в серии игр Red Dead Redemption, определяет тип и вид облака, которые необходимы в данной местности и которые подходят под погодные условия.

Таким образом, в качестве базовой технологии рендеринга, на которой будет основываться разработанный алгоритм, был выбран подход генерации облаков Nubis, поскольку он имеет следующие преимущества:

1) возможность рассчитывать освещение облаков близкое к физически-корректному;

2) возможность управлять покрытием и видом облаков с использованием карты погоды;

3) возможность поддерживать широкий спектр облаков с использованием карты погоды;

4) возможность плавно сменять облачное покрытие;

5) возможность добиваться реальной формы и поведения облаков в реальном времени с использованием 3D текстур шумов.

Более того, система генерации облаков Nubis имеет обширную документацию, что позволяет экспериментировать при реализации данной системы.

Далее будет рассмотрен необходимый материал для реализации алгоритма генерации объемных облаков. После этого подробно будет расписан сам алгоритм и проанализированы результаты его работы.

2. ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Дальнейшие пункты расставлены в необходимом для понимания алгоритма порядке. Сначала будет рассмотрена технология рендеринга облаков, далее будет рассмотрена карта погоды, а после будет описан расчет освещения облаков.

2.1. Марширующие лучи

Для рендеринга облаков будет использоваться технология марширующих лучей (ray marcing).

В отличие от классического рендеринга в компьютерной графике и от трассировки лучей [9] марширующий луч имеет дело не только с поверхностью объекта, а еще и с внутренней его частью. То есть луч делится внутри примитива на некоторое число разбиений и начинает делать, например, некоторую выборку цвета. В связи с этим, одной из основных задач технологии марширующих лучей является вычисление рассеянного света внутри объекта.

Общую схему марширующих лучей можно наблюдать на рис. 9.

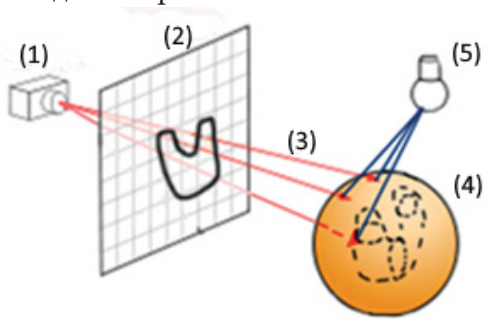

Рис. 9. Общая схема марширующих лучей
[Fig. 9. General scheme of ray marching]

Здесь (1) — камера, (2) — область изображения, (3) — луч, выпущенный из камеры, (4) — трехмерный примитив, а черная объемная поверхность внутри — необходимая трехмерная поверхность для рендеринга, (5) — источник освещения.

2.2. Карта погоды

Карта погоды контролирует процесс создания облаков. Она указывает, какие облака будут созданы и какие погодные условия должны быть. Карта погоды — это 2D текстура, которая стандартно имеет разрешение 512×512 пикселей. Каждый канал этой текстуры хранит информацию об облаках: красный канал используется для указания первичных мест появления облаков; зеленый канал указывает на информацию о типе облака; синий канал погодной карты описывает количество дождевых капель в облаках.

Карта погоды может быть создана заранее (художниками) или же сгенерирована автоматически с помощью шумов. В качестве основных шумов, которые определяют форму облака, были выбраны шум Перлина [10], шум Уорли [11], шум Курла [12], а также шум Перлина-Уорли [13], который является смешиванием шума Перлина и Уорли соответственно.

Так же если ввести некоторый параметр $coverage \in [0,1]$, то можно будет управлять глобальным покрытием облаков на карте. Для этого необходимо вычесть параметр *coverage* из красного канала карты погоды, ограничив при необходимости получившееся значение в диапазоне $[0,1]$.

Пример карты погоды можно наблюдать на рис. 10.

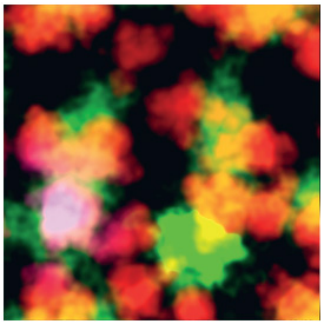

Рис. 10. Карта погоды
[Fig. 10. Weather map]

2.3. Освещение облаков

Классическая модель освещения Фонга [14] не подходит для расчета освещения облаков, поскольку облака не имеют четкой формы и расчет зеркальной составляющей невозможен. От данной модели используются только фоновая и рассеянная компоненты. Эти составляющие цвета рассчитываются для каждой точки, выбранной с помощью маршрирующих лучей.

Чтобы найти рассеянную компоненту света, необходимо рассчитать, сколько его проходит в конкретную точку. Однако облака состоят из мельчайших частиц, и при расчете приходится учитывать отражение от каждой частицы. Такой подход может значительно снизить производительность. Иллюстрация такого подхода продемонстрирована на рис. 11. Здесь луч выпущен от источника света к наблюдателю, и после этого происходит расчет пути луча, отражающегося от множества частиц.

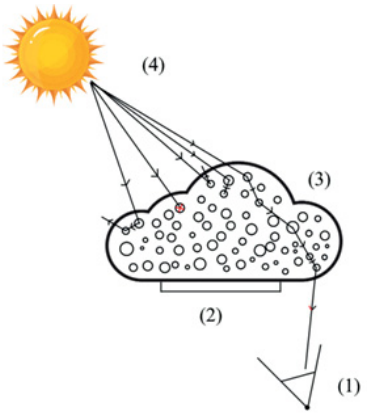

Рис. 11. Корректный расчет освещения
[Fig. 11. Correcting lighting calculation]

Чтобы существенно снизить нагрузку и не потерять качество освещения, используют аппроксимированный подход, идея которого отражена на рис. 12.

Его идея заключается в том, что выпускаются лучи от наблюдателя во все допустимые стороны (в каждый пиксель на экране). Далее на каждом луче из каждой точки его разбиения выпускается вспомогательный луч к источнику освещения. На этом вспомогательном луче также берется некоторое число точек разбиения, и в каждой из них вычисля-

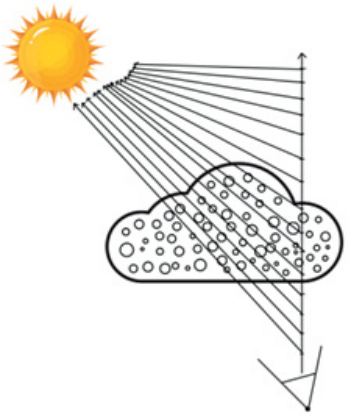

Рис. 12. Аппроксимированный подход к вычислению освещения [Fig. 12. Approximate approach to lighting calculation]

ется оптическая плотность согласно формуле (1).

$$d_{нов} = d_{ст} + x_r, \quad (1)$$

где $d_{нов}$ — оптическая плотность на текущем шаге,

$d_{ст}$ — накопленная оптическая плотность для конкретной точки на предыдущем шаге,

x_r — значение красного канала текстуры шума для конкретной точки.

В дальнейшем данное значение оптической плотности будет использоваться для расчета энергии светового луча, необходимой для расчета освещения.

2.4. Расчет энергии луча

Введем следующие обозначения:

d — плотность, которую проходит луч от текущей точки в облаке к источнику освещения,

p — множитель поглощения света дождевыми облаками,

$BackSM \in [-1;1]$ — множитель обратного распространения света,

$FwdSM \in [-1;1]$ — множитель прямого распространения света,

θ — угол в градусах между солнечным лучом и лучом, выпущенным от камеры,

E — энергия луча.

Lerp — функция линейной интерполяции, которая первым параметром принимает левую границу отрезка, вторым параметром — правую границу отрезка, а третьим параметром — параметр смешивания, который находится в диапазоне от 0 до 1.

Для расчета энергии луча используются четыре основных компонента:

1. Закон Бера [15], который служит для придания базового освещения облаков (формула 2).

$$E = e^{-d}. \quad (2)$$

2. Фазовая функция Хеньи — Гринштейна, HG, [16] для придания эффекта серебряной подложки облаков.

3. Функция рассеяния наружу [17] (эффект сахарной пудры), которая служит для придания выразительности «краям» облаков.

4. Экспоненциальная функция для эффекта дождевого облака, то есть для увеличения поглощения света.

Для расчета энергии луча используются формула (3)

$$E = 2e^{-dp}(1-e^{-2d}) \times \max\{HG(\theta,-BackSM), HG(\theta,FwdSM)\} \quad (3)$$

или (4)

$$E = 2e^{-dp}(1-e^{-2d}) \times lerp(HG(\theta,-BackSM), HG(\theta,FwdSM), blendFactor). \quad (4)$$

Окончательный цвет облаков рассчитывается по формуле (5), которая включает в себя компоненты модели Фонга без зеркального освещения:

$$lightColor = E \cdot (directFactor \cdot directLigh + ambientFactor \cdot ambientLight), \quad (5)$$

где *directFactor* — коэффициент, отвечающий за силу освещения источника света;

directLight — цвет источника освещения;

ambientFactor — коэффициент, отвечающий за силу фонового освещения;

ambientLight — цвет фонового освещения.

Заметим, что цвет фонового освещения, в отличие от модели Фонга, будет вычисляться с помощью линейной интерполяции заданного цвета атмосферы и заданного цвета облаков.

3. АЛГОРИТМ ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ ОБЪЕМНЫХ ОБЛАКОВ В РЕАЛЬНОМ ВРЕМЕНИ

После рассмотрения основных понятий и формул, из которых формируется алгоритм, перейдем к рассмотрению самого алгоритма.

Алгоритм 1. Общий алгоритм рендеринга объемных облаков.

1 шаг. Генерация карты погоды, 2D текстуры шума и двух 3D текстур шума.

Карта погоды генерируется с помощью шума Перлина — Уорли. Также она может быть создана художником заранее.

2D текстура с разрешением 128×128 генерируется с помощью шума Курла и состоит из трех RGB каналов.

Первая 3D текстура с разрешением $128\times128\times128$ генерируется с помощью шума Перлина — Уорли. Она определяет базовую форму облака и состоит из четырех RGBA каналов. Красный канал содержит низкочастотный шум Перлина — Уорли, а следующие каналы состоят из шума Перлина — Уорли с увеличивающейся частотой.

Вторая 3D текстура с разрешением $32\times32\times32$ генерируется с помощью шума Перлина — Уорли. Она состоит из трех RGB каналов. Цветовые каналы содержат шум Перлина — Уорли с увеличивающейся частотой.

Заметим, что при необходимости эти текстуры просто копируется в нужном направлении, чтобы подстроиться под необходимый размер «облачного куба».

2 шаг. Генерация грубой формы облаков при помощи первой 3D текстуры шума и технологии марширующих лучей.

Грубая форма облака получается при помощи первой 3D текстуры. Ее красный канал образует грубую форму облака, а последующие каналы добавляют детали за счет размытия основной формы облака. Срез первой трехмерной текстуры можно наблюдать на рис. 13.

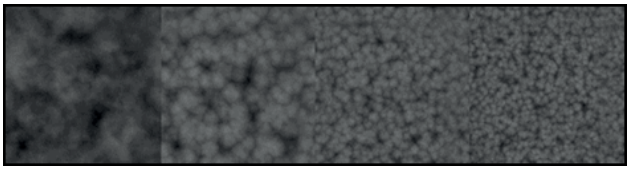

Рис. 13. Срез первой 3D текстуры
[Fig. 13. Slicing the first 3D texture]

Согласно технологии марширующих лучей внутри примитива на луче отмечаются точки в соответствии с некоторым разбиением. На каждой точке этого разбиения происходит проверка наличия облака с помощью карты погоды, и в положительном случае применяется формула (1) для первой 3D текстуры шума.

3 шаг. Создание мелких деталей облаков за счет второй 3D текстуры шума.

Вторая 3D текстура шума придает облаку естественную форму, создавая мелкие детали, что подчеркивает его неоднородность. Срез второй трехмерной текстуры можно наблюдать на рис. 14.

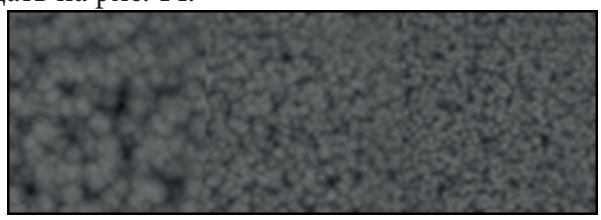

Рис. 14. Срез второй 3D текстуры
[Fig. 14. Slicing the second 3D texture]

Здесь также в каждой точке разбиения луча происходит проверка наличия облака в этом регионе с помощью карты погоды и в положительном случае по технологии марширующих лучей начинается выборка цвета из второй 3D текстуры шума по формуле (1), которая придает облаку мелкие детали.

Опционально на данном шаге можно использовать 2D текстуру шума Курла. Эта текстура придает облаку эффект турбулентности воздуха, создавая мелкие завихрения. Эта текстура изображена на рис. 15.

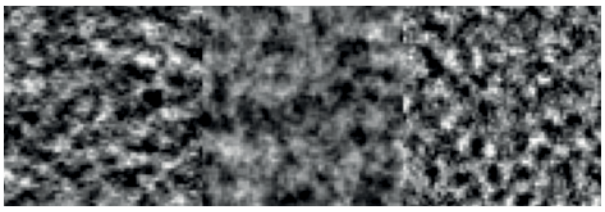

Рис 15. 2D текстура шума
[Fig. 15. 2D noise texture]

Для выборки цвета используются аналогичные действия.

4 шаг. Изменение высоты облака.

На предыдущих шагах алгоритма была сгенерирована основная форма облаков и добавлены мелкие детали для реалистичности.

Однако на данный момент все облака имеют одинаковую высоту. Эта проблема решается на данном шаге алгоритма.

Для придания разного размера облакам используется три изображения-градиента, состоящих из четырех RGBA каналов (рис. 16).

Рис. 16. Градиенты изменения высоты и плотности облака
[Fig. 16. Cloud height and density gradient]

Высота облака вычисляется по формуле (6)

$$H = S(HD - y), \quad (6)$$

где $S(.)$ — интерполяционный полином

$$S(x) = \begin{cases} 0, \text{ если } x < 0, \\ 1, \text{ если } x > 1, \\ 3x^2 - 2x^3, \text{ иначе}, \end{cases} \quad (7)$$

y — текущая высота облака в точке,

HD — высотный параметр точки,

$$HD = g_1 \cdot w_1 + g_2 \cdot w_2 + g_3 \cdot w_3, \quad (8)$$

здесь g_i — вектор значений каналов (R, G, B, A) i-го градиента, $i = 1,...,3$,

w_1, w_2, w_3 — значения весов, рассчитанные по формулам (9)–(11).

$$w_1 = 1 - clamp(2 cloudTypeProb), \quad (9)$$

$$w_2 = 1 - 2 \cdot abs(cloudTypeProb - 0.5), \quad (10)$$

$$w_3 = 2 \cdot clamp(cloudTypeProb - 0.5). \quad (11)$$

Здесь параметр *cloudTypeProb* — значение зеленого канала карты погоды в конкретной точке, а *clamp*(.) — функция, которая ограничивает значение в диапазоне $[0,1]$.

5 шаг. Расчет освещения облаков.

На данном шаге для каждой точки на вспомогательном марширующем луче по формулам (3) или (4) рассчитывается итоговая освещенность каждого пикселя.

4. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Предложенный алгоритм генерации объемных облаков был реализован на игровом движке Unity. В качестве среды разработки была выбрана Visual Studio. На языке высокого уровня C# производилась подготовка и передача текстур для дальнейших расчетов. На языке HSLS производился расчет текстур шумов, а на языке ShaderLab производились все основные расчеты.

На рис. 17 представлены окно с параметрами, отвечающими за создание базовой формы облаков.

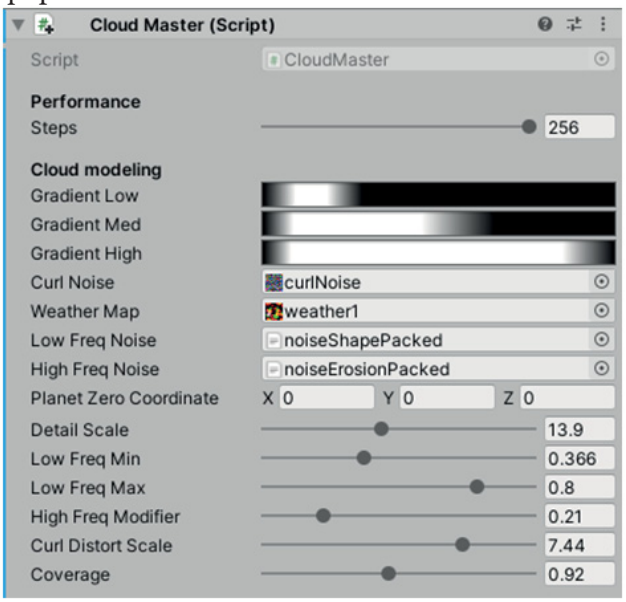

Рис. 17. Окно настройки параметров формы облаков
[Fig. 17. Cloud shape settings window]

Коротко опишем каждый из них.

Steps — количество точек разбиений на первичном луче.

Gradient Low, *Gradient Med*, *Gradient High* — градиенты, которые используются на шаге 4.

Curls Noise, *Low Freq Noise*, *High Freq Noise* — шумы, которые придают облакам форму. Они используются на 2-м и 3-м шагах алгоритма.

Weather Map — карта погоды. Она генерируется с помощью текстур шумов на 1-м шаге алгоритма. Карта погоды определяет места, где облака будут располагаться (2-й шаг алгоритма), служит для изменения типа облака (4-й шаг алгоритма) и, наконец, на 5-м шаге алгоритма она добавляет облакам темные области (эффект дождевого облака).

Detail Scale, *Low Freq Min*, *Low Freq Max*, *High Freq Modifier*, *Curl Distortion Scale* — технические параметры, которые влияют на генерацию текстур шумов на 1-м шаге алгоритма.

Coverage — используется на 2-м шаге алгоритма и определяет, насколько сильно облака затягивают небо.

На рис. 18 предоставлены окна настройки параметров освещения облаков.

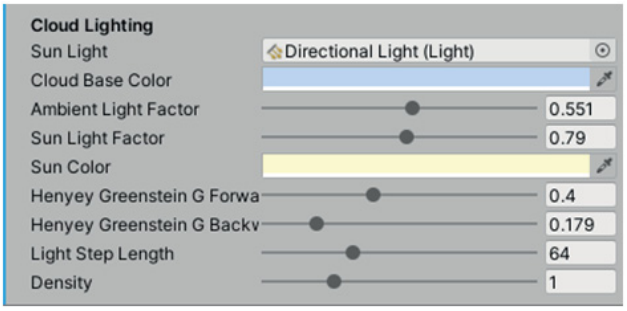

Рис. 18. Окно настройки освещения облаков
[Fig. 18. Cloud lighting settings window]

Также коротко опишем каждый из них.

Sun Light — тип источника освещения.

Cloud Base Color — цвет облаков.

Ambient Light Factor и *Sun Light Factor* — параметры формулы (4).

Sun Color — цвет источника освещения.

Henyey Greenstein G Forward, Henyey Greenstein G Backward — параметры фазовой функции Хеньи — Гринштейна, которые используются в формулах (2)–(3).

Light Step Length — количество разбиений на вторичном (вспомогательном) луче, отвечающем за освещение.

Density — параметр плотности, который используется в законе Бера.

Пример построенных облаков, с заданными выше параметрами, можно наблюдать на рис. 19.

Рис. 19. Пример рендеринга облаков
[Fig. 19. Cloud rendering example]

Здесь облака разного размера, с мелкими деталями и с добавлением эффекта турбулентности воздуха. Полученный результат с такими параметрами можно считать приемлемым.

Однако если параметр *Steps*, отвечающий за количество разбиений на луче, уменьшить в два раза (то есть он станет равным 125), то появятся визуальные искажения. Это можно наблюдать на рис. 20 и 21. На рис. 20 и 21 а) представлены фрагменты генерации облака с параметром $steps = 250$, а на рис. 20 и 21 б) представлены те же фрагменты облака с параметром $steps = 125$.

а) *б)*

Рис. 20. Сравнение регионов
а) steps=250, б) steps=125
[Fig. 20. Comparison of regions
a) step = 250, b) steps = 125]

а) *б)*

Рис. 21. Сравнение регионов
а) steps=250, б) steps=125
[Fig. 21. Comparison of regions
a) step = 250, b) steps = 125]

Заметим, что на рис. 20 и 21 б) присутствуют визуальные искажения, которые выражаются в пикселизации, а именно в хаотичном задании цветов пикселям.

Если изменять параметры LowFreqMin и LowFreqMax, отвечающие за генерацию текстур шумов, то можно изменять структуру облака. Это можно наблюдать на рис. 22.

Рис. 22. Результат изменения параметров LowFreqMin и LowFreqMax
[Fig. 22. Result of changing the parameters LowFreqMin and LowFreqMax]

Теперь перейдем к рассмотрению изменения параметров освещения облаков.

На рис. 23 можно наблюдать изменения количества разбиения на луче до 10. В результате, получаются тёмные облака. На рис. 23 также представлено изменение цвета облаков.

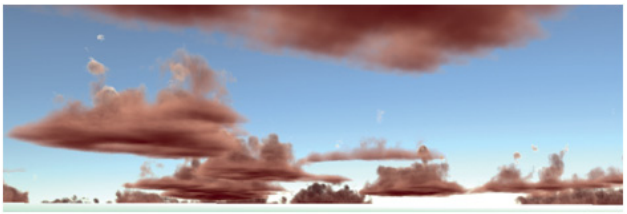

Рис. 23. Уменьшение количество разбиений на вспомогательном луче
[Fig. 23. Reducing the number of splits on the auxiliary ray]

5. АНАЛИЗ ВРЕМЕНИ РАБОТЫ АЛГОРИТМА

В данном пункте производится оценка времени работы алгоритма при заданных параметрах на основе реализованного программного обеспечения.

Важным этапом алгоритма является генерация различных текстур с помощью шумов. Эта трудоемкая по вычислениям процедура не является целью данного исследования, то есть не ставится задача оптимизации известных или разработки новых процедур генерации шумов. Заметим, что такие вычисление происходят только на первом шаге работы алгоритма. Это позволяет устройствам, которые не в состоянии быстро сгенерировать их самостоятельно, подгружать заранее сгенерированные на более мощных вычислительных устройствах текстуры из памяти.

Итак, выделим две цели эксперимента:

1) оценка времени работы первого шага алгоритма (генерация шумов) без настройки параметров генерации шумов;

2) оценка времени работы алгоритма в зависимости от его параметров. Были выделены три основных параметра, существенно влияющих на время работы алгоритма: количество разбиений на первичном луче; параметр *coverage*, отвечающий за покрытие неба облаками; количество разбиений на вторичном луче, отвечающем за освещение.

Эксперимент проводился на устройстве со следующими характеристиками: процессор — AMD Ryzen 5 1600 с тактовой частотой 3600 МГц и количеством ядер 6, ОЗУ — 16 ГБ, видеокарта – Nvidia RTX 2060.

В экспериментах для каждого случая производилось по 50 запусков алгоритма. В табл. 2–6 указанно среднее время.

В табл. 2 представлены результаты по времени работы первого шага алгоритма, на котором генерируются все текстуры шумов.

Таблица 2. Время работы первого шага алгоритма
[Table 2. Running time of the first step of the algorithm]

Текстура шума	Время рендеринга, мс
Карта погоды	3
2D текстура шума	1
Первая 3D текстура шума	8
Вторая 3D текстура шума	6
Итоговое время работы (суммарное)	18
Итоговое время работы 1-го шага алгоритма (реальное)	18,3

Таким образом, получаем, что генерация шумов занимает в среднем 18 мс.

Далее перейдем к оценке времени работы алгоритма без учета первого его шага.

В табл. 3 представлено время работы алгоритма в зависимости от количества разбиений первичного луча. При этом два других параметра были заданы: *coverage* — 0,5, количество разбиений вторичного луча — 64.

Таблица 3. Время работы алгоритма с различным количеством разбиений на луче [Table 3. Running time of the algorithm with a different number of partition on the ray]

Количество разбиений	Время рендеринга, мс
50	0,2
100	0,4
150	1
200	1,6
250	2

На рис. 24 данная зависимость представлена графически.

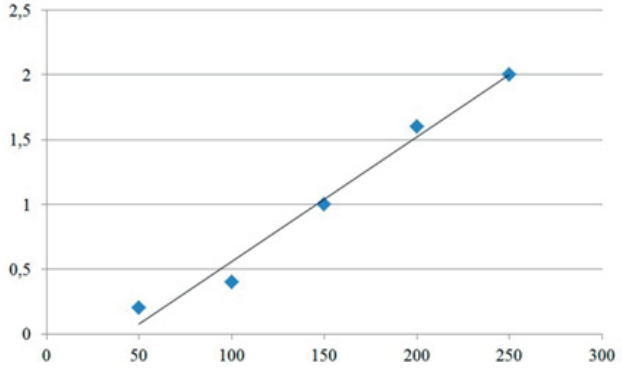

Рис. 24. График зависимости времени работа алгоритма от количества разбиений на первичном луче [Fig. 24.Graph of the time dependence of the algorithm on the number of partitions on the primary ray]

Можно отметить, что зависимость близка к линейной. Регрессионный анализ показал зависимость $y = 0.0096x - 0.4$ с коэффициентом детерминации 97,9 %.

В табл. 4 представлено время работы алгоритма в зависимости от параметра *coverage*. При этом остальные два параметра заданы следующим образом: количество разбиений первичного луча — 250, количество разбиений вторичного луча — 64.

Таблица 4. Время работы алгоритма в зависимости от параметра coverage [Table 4. Running time of the algorithm depending on the coverage parameters]

Параметр coverage	Время рендеринга, мс
0,1	1,3
0,2	1,6
0,5	2
0,7	2,2
1,0	2,6

На рис. 25 данная зависимость представлена графически.

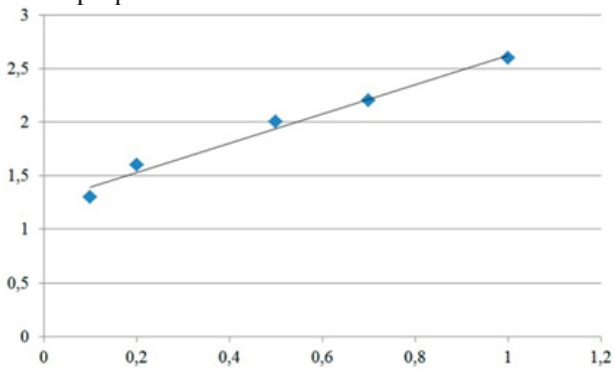

Рис. 25. График зависимости времени работа алгоритма от параметра процента покрытия облаками неба [Fig. 25. Graph of the dependence of the time the algorithm works on the parameter of the percentage of sky cloud coverage]

Заметим, что здесь также прослеживается линейная зависимость. Регрессионный анализ показал зависимость $y = 1.37037x + 1.25481$ с коэффициентом детерминации 98,2 %.

В табл. 5 представлена зависимость времени работы от количества разбиений на вторичном луче, отвечающем за освещение. При этом количество разбиений на первичном луче равно 250, а параметр $coverage = 0,5$.

На рис. 26 представлен график этой зависимости, которая также близка к линейной. Регрессионный анализ показал зависимость $y = 0.008x + 1.42$ с коэффициентом детерминации 99,5 %.

По итогам проведенного эксперимента можно сделать вывод, что зависимость во всех случаях близка к линейной.

Таблица 5. Время работы алгоритма в зависимости от количества разбиений на вторичном луче
[Table 5. Running time of the algorithm depending on the number of splits on the secondary ray]

Количество разбиений	Время рендериннга, мс
50	1,8
100	2,2
150	2,7
200	3,0
250	3,4

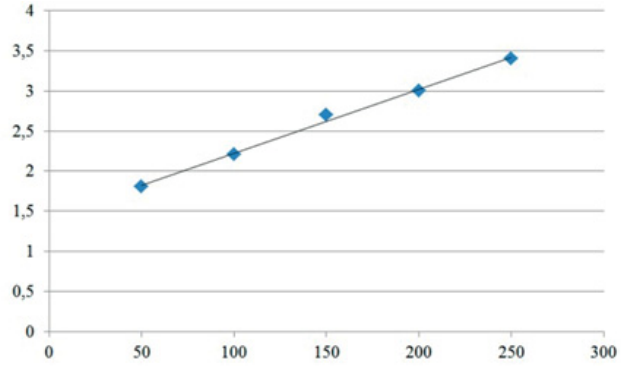

Рис. 26. График зависимости времени работа алгоритма от количества разбиений на вторичном луче
[Fig. 26. Graph of the time dependence of the algorithm on the number of partitions on the secondary ray]

Теперь оценим общее время работы алгоритма 1 при различных значениях выбранных параметров (табл. 6).

Заметим, что действительно на время работы алгоритма больше всего оказывает влияние значение параметра *coverage*. Как было описано ранее, это связано с неоднородностью карты погоды.

Важнейшим требованием к алгоритму является генерация облаков в реальном времени, то есть за время, которое человеческий глаз не сможет заметить. Обычно за это время принимают 16 мс [18]. Заметим, что в табл. 6 облака генерируются в среднем за 1–3 мс при различных параметрах. Таким образом, на рендеринг сцены остаётся около 13–15 мс, что является оптимальным параметром даже

Таблица 6. Время работы алгоритма
[Table 6. Running time of the algorithm]

Количество разбиений первичного луча	Параметр *coverage*	Количество разбиений вторичного луча	Время работы алгоритма, мс
100	0,3	100	0,5
100	0,3	200	0,7
100	0,8	100	0,9
100	0,8	200	1,1
200	0,3	100	0,9
200	0,3	200	1,3
200	0,8	100	2,2
200	0,8	200	2,8

для самых нагруженных сцен [19]. Генерация шумов также остаётся важной частью алгоритма и самой длительной, однако их можно сгенерировать один раз вначале алгоритма, либо же вообще загружать из внешней памяти, заранее сгенерировав их на более мощных вычислительных устройствах.

Исходя из табл. 6, можно сказать, что следующие параметры являются удовлетворительными по времени работы и качеству построенных облаков: количество разбиений на первичном луче — 250, параметр *coverage* — 0,5 (он может варьироваться в процессе работы программы), и количество разбиений на вторичном луче — 64. В среднем с предложенными параметрами алгоритм генерирует облака за 2 мс. Результат работы алгоритма при данных параметрах можно наблюдать в видеофайлах, которые доступны по следующим ссылкам:

https://disk.yandex.ru/i/hb8-UqApJULftA (параметры: 125; 0,5; 128),

https://disk.yandex.ru/i/uzp-GAqm6q-V6g (параметры: 250; 0,5; 64),

https://disk.yandex.ru/i/sGv6DWU-PNTXDw (параметры: 250; 0,8; 128).

ЗАКЛЮЧЕНИЕ

Таким образом, в данной статье продемонстрирован алгоритм генерации трехмер-

ных облаков в реальном времени. Для этого были рассмотрены два вида рендеринга: с использованием объемной генерации и без нее, выделены преимущества и недостатки всех методов. Сразу были убраны из рассмотрения алгоритмы без объемной генерации. В связи с этим детальнее были проанализированы алгоритмы с применением объемной генерации и, в итоге, была выбрана для дальнейшего анализа и базой к реализации технология Nubis, поскольку она больше всех подходила под требования, описанные в постановке задачи.

В статье представлен подход марширующих лучей и математический аппарат для реализации освещения.

На основе теоретического материала был детально описан сам разработанный алгоритм. Для каждого шага продемонстрированы результаты его работы с помощью реализованного программного обеспечения.

И, наконец, был проведен вычислительный эксперимент по оценке времени работы алгоритма по основным параметрам, которые непосредственно влияют на производительность алгоритма. Было выяснено, что для сбалансированной по скорости и качеству работы алгоритму необходимо около 2 мс без учета генерации шумов. В завершении предложены демонстрационные файлы с результатами работы алгоритма.

КОНФЛИКТ ИНТЕРЕСОВ

Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.

СПИСОК ЛИТЕРАТУРЫ

1. Атлас облаков / Д. П. Беспалов, А. М. Девяткин, Ю. А. Довгалюк [и др.]; под ред. Л. К. Сурыгина. – Санкт-Петербург : РИФ «Д'АРТ», 2011. – 252 с.

2. Opacity shadow maps. – Режим доступа: http://crystalspace.ru/download/opacity-shadow-maps.pdf. (дата обращения: 08.01.2022).

3. Development of Red Dead Redemption. – Режим доступа: https://gamerant.com/long-red-dead-redemption-2-development. (дата обращения: 08.01.2022).

4. The Art of the Witness – Clouds. – Режим доступа: http://www.artofluis.com/3dwork/the-art-of-the-witness/clouds/. (дата обращения: 08.01.2022).

5. The technical art of sea of thieves. – Режим доступа: https://dl.acm.org/doi/10.1145/3214745.3214820. (Дата обращения: 08.01. 2022).

6. Horizon Dawn Zero – Режим доступа: https://www.guerrillagames.com/play/horizon. (дата обращения: 08.01.2022).

7. NUBIS: Authoring Real-Time Volumetric Cloudscapes With The Decima Engine. – Режим доступа. https://www.guerrilla-games.com/read/nubis-authoring-real-time-volumetriccloudscapes-with-the-decima-engine. (дата обращения: 08.01.2022).

8. What are 3D texture. – Режим доступа: https://gamedev.stackexchange.com/questions/9668/what-are-3d-textures. (дата обращения: 08.01.2022).

9. What is Ray tracing? – Режим доступа: https://www.techradar.com/news/ray-traing#:~:text=Ray%20tracing%20is%20a%20rendering,in%20the%20computer%2Dgenerated%20world. (дата обращения: 08.01.2022).

10. Perlin noise. – Режим доступа: https://www.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-noise/a/perlin-noise. (дата обращения: 08.01.2022).

11. Worley noise. – Режим доступа: https://thebookofshaders.com/12/. (дата обращения: 08.01.2022).

12. Curl noise. – Режим доступа: https://www.sidefx.com/docs/houdini/nodes/vop/curlnoise.html. (дата обращения: 08.01.2022).

13. *Schneider, A.* Real-Time Volumetric Cloudscapes, GPU Pro 7, CRC Press. – 2016. – P. 97–127.

14. *Phong, B. T.* Illumination for Computer Generated Pictures, 1975. – Режим доступа: http://www.cs.northwestern.edu/~ago820/cs395/Papers/Phong_1975.pdf. (дата обращения: 08.01.2022).

15. *Beer, A.* Bestimmung der Absorption des rothen Lichts in farbigen Flüssigkeiten

(Determination of the Absorption of Red Light in Colored Liquids) / A. Beer. – 1852. – С. 78–88.

16. *Henyey, L. G.* Diffuse radiation in the Galaxy / L. G. Henyey, Greenstein J. L. // Astrophysical Journal. – 1941. – P. 70–83.

17. Schneider, A. Real-Time Volumetric Cloudscapes, GPU Pro 7, CRC Press, 2016. – P. 97–127.

18. Хьюбел, Д. Глаз, мозг, зрение / Д. Хьюбел. – 1990. – 240 с.

19. Technique enables real-time rendering of scenes in 3D. – Режим доступа: https://news.mit.edu/2021/3-d-image-rendering-1207 (дата обращения: 08.01.2022).

Сырых Александр Сергеевич — магистрант 1-го года обучения кафедры МО ЭВМ Воронежского государственного университета.
E-mail: zsyryhz@mail.ru
ORCID iD: https://orcid.org/0000-0001-8326-3743

Медведев Сергей Николаевич — канд. физ.-мат. наук, доц., доцент кафедры вычислительной математики и прикладных информационных технологий Воронежского государственного университета.
E-mail: s_n_medvedev@mail.ru
ORCID iD: https://orcid.org/0000-0002-6971-5069

DOI: https://doi.org/10.17308/sait/1995-5499/2022/2/93-109 ISSN 1995-5499
Received 30.11.2021
Accepted 29.06.2022

PROCEDURAL GENERATION OF VOLUMETRIC CLOUDS IN REAL TIME

© 2022 A. S. Syryh✉, S.N. Medvedev

Voronezh State University
1, Universitetskaya Square, 394018 Voronezh, Russian Federation

Annotation. The article presents the developed algorithm for generating realistic three-dimensional clouds in real time. A detailed analysis of existing cloud generation solutions is given. Based on the analysis, the Nubis base technology was chosen for further work. The article describes the theoretical material necessary for the implementation of the algorithm. The developed algorithm is described in detail, and for each of its stages, the results of work based on the implemented software are demonstrated. An important part of the article is a detailed description of the computational experiment on setting the parameters of the algorithm. Based on the analysis of the results obtained, conclusions are drawn about the operation of the algorithm in real time and the necessary settings for the parameters for the high-quality operation of the algorithm are proposed.
Keywords: procedural generation, ray marching, Perlin noise, Worley noise, Curl noise.

CONFLICT OF INTEREST

The authors declare the absence of obvious and potential conflicts of interest related to the publication of this article.

REFERENCES

1. *Bespalov D. P., Devyatkin A. M., Dovgalyuk Yu. A. [et al.]* (2011) Atlas oblakov [Cloud Atlas] ed. by L. K. Surygina. *Sankt-Peterburg : RIF «D'ART»*. 252 p. (in Russian)

✉ Syryh Alexander S.
e-mail: zsyryhz@mail.ru

2. Opacity shadow maps. – Available at: http://crystalspace.ru/download/opacity-shadow-maps.pdf. (accessed: 08.01.2022).

3. Development of Red Dead Redemption. – Available at: https://gamerant.com/long-red-dead-redemption-2-development. (accessed: 08.01.2022).

4. The Art of the Witness – Clouds. – Available at: http://www.artofluis.com/3dwork/the-art-of-the-witness/clouds/. (accessed: 08.01.2022).

5. The technical art of sea of thieves. – Available at: https://dl.acm.org/doi/10.1145/3214745.3214820. (accessed: 08.01.2022).

6. Horizon Dawn Zero – Режим доступа: https://www.guerrillagames.com/play/horizon. (accessed: 08.01.2022).

7. NUBIS: Authoring Real-Time Volumetric Cloudscapes With The Decima Engine. – Available at: https://www.guerrilla-games.com/read/nubis-authoring-real-time-volumetriccloud-scapes-with-the-decima-engine. (accessed: 08.01.2022).

8. What are 3D texture. – Available at: https://gamedev.stackexchange.com/questions/9668/what-are-3d-textures. (accessed: 08.01.2022).

9. What is Ray tracing? – Available at: https://www.techradar.com/news/ray-traing#:~:-text=Ray%20tracing%20is%20a%20rendering,in%20the%20computer%2Dgenerated%20world. (accessed: 08.01.2022).

10. Perlin noise. – Available at: https://www.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-noise/a/perlin-noise. (accessed: 08.01.2022).

11. Worley noise. – Available at: https://thebookofshaders.com/12/. (accessed: 08.01.2022).

12. Curl noise. – Available at: https://www.sidefx.com/docs/houdini/nodes/vop/curlnoise.html. (accessed: 08.01.2022).

13. *Schneider A.* (2016) Real-Time Volumetric Cloudscapes, GPU Pro 7, CRC Press. P. 97–127.

14. *Phong B. T.* (1975) Illumination for Computer Generated Pictures. – Available at: http://www.cs.northwestern.edu/~ago820/cs395/Papers/Phong_1975.pdf. (accessed: 08. 01.2022).

15. *Beer A.* (1852) Bestimmung der Absorption des rothen Lichts in farbigen Flüssigkeiten (Determination of the Absorption of Red Light in Colored Liquids). P. 78–88.

16. *Henyey L. G. and Greenstein J. L.* (1941) Diffuse radiation in the Galaxy, Astrophysical Journal. P. 70–83.

17. *Schneider A.* (2016) Real-Time Volumetric Cloudscapes, GPU Pro 7, CRC Press. P. 97–127.

18. Hubel David H. Eye, Brain, and Vision (1990) (Scientific American Library). 240 p.

19. Technique enables real-time rendering of scenes in 3D. – Available at: https://news.mit.edu/2021/3-d-image-rendering-1207 (accessed: 08.01.2022).

Syryh Alexander S. — master student of the 1st year of the Department of Mathematical Support of Computing Electronic Computers, Applied Mathematics, Informatics and Mechanics Faculty, Voronezh State University.
E-mail: zsyryhz@mail.ru
ORCID iD: https://orcid.org/0000-0001-8326-3743

Medvedev Sergei N. — PhD in Physics and Mathematics, Associate Professor of the Department of Computing Mathematics and Applied Information Technology, Applied Mathematics, Informatics and Mechanics Faculty, Voronezh State University.
E-mail: s_n_medvedev@mail.ru
ORCID iD: https://orcid.org/0000-0002-6971-5069