ПРОГРАММИРОВАНИЕ, 2017, № 4, с. 27–38

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.94

МОДЕЛИРОВАНИЕ ОКРУЖАЮЩЕГО ПРОСТРАНСТВА В ЛЕТНОМ СИМУЛЯТОРЕ

© 2017 г. Р.В. Сапронов

Казанский национальный исследовательский технический университет им. А.Н. Туполева
420111 Казань, ул. К. Маркса, д. 10
E-mail: lurky@rambler.ru

Поступила в редакцию 02.02.2017

Предложено решение задачи нормализации карт высот и смещений, генерируемых на основе модели, описанной Д. Тессендорфом. Предложены способы устранения визуальных артефактов в методе проективных сеток при нахождении камеры на большой высоте, при визуализации на границе водно-воздушного интерфейса. Выведена комплексная модель освещения водной поверхности. Предложены способы динамического формирования изображения различных атмосферных эффектов, таких как облака, дождь, снег на основе объемных моделей с использованием процедуры трассировки лучей в экранном пространстве. Предложены эффективные способы реализации указанных алгоритмов, пригодные для использования не только в системе визуализации летного симулятора, но также и других приложениях реального времени.

1. ВВЕДЕНИЕ

При визуализации открытых пространств неизбежно приходится сталкиваться с проблемой реалистичного отображения водной поверхности и атмосферы, изображение которых зачастую составляет большую часть информации, видимой на экране. С учетом специфических особенностей летного симулятора указанная проблема представляется в несколько упрощенном варианте: можно пренебречь взаимодействием объектов с водной поверхностью, не учитывать ряд визуальных эффектов, проявляющихся при нахождении наблюдателя в толще воды, использовать упрощенные модели атмосферы и атмосферных эффектов. Несмотря на это, реалистичное моделирование окружающего пространства является одним из основных критериев, предъявляемых к подобным системам в настоящее время, а также основным предметом этой статьи.

Следует отметить большое количество существующих методов визуализации водной поверхности [1–3], многие из которых основаны тем или иным образом на результатах, полученных Д. Тессендорфом [4], который одним из первых предложил использовать спектральную функцию для синтеза анимированной водной поверхности. В части геометрического моделирования водной поверхности выбор существующих алгоритмов несколько шире — от равномерно-тесселированных мешей, до схем, основанных на динамической детализации [5], радиальных [6] и проективных сеток [7, 8]. Каждый из этих алгоритмов обладает своими преимуществами и недостатками, которых мы коснемся позднее. Ввиду ряда очевидных причин для визуализации водой поверхности выбран метод проективных сеток. Несмотря на его перспективность и универсальность, он обладает рядом недостатков, которые в совокупности с использованием спектральной модели привели к возникновению ряда визуальных артефактов, на устранение которых и направлена данная статья. Немалое внимание научной общественности направлено также на имитацию освещения водной поверхности [3–6, 9–12], которая ввиду своих свойств является весьма трудоемкой задачей, в чистом виде являющей собой компромисс между качеством и производительностью.

Визуализация атмосферных эффектов также является чрезвычайно важной задачей при создании летного симулятора. Однако в интерактивных приложениях временной бюджет, выделяемый на их отображение, существенно ограничен (как правило, не более 10 мс). В этой связи широко распространен подход, при котором небесная сфера представляется в виде кубической или сферической карты, спроецированной на внутреннюю поверхность куба или сферы достаточно большого размера. При этом затрудняется имитация смены времени суток, анимация облаков (в подобных случаях кубическая карта получается путем обработки панорамной фотографии), поскольку в этом случае необходима плавная интерполяция между кубическими картами, полученными с максимальным временным разрешением.

При имитации движения сквозь облака применялся (и до сих пор применяется) подход, основанный на системах частиц [13–15]. Однако в последнее время возросшие требования к реализму изображения и возможности оборудования привели к появлению новых методов моделирования и визуализации атмосферы и облаков, использующих адаптации классических подходов визуализации объемных данных [16, 17], так в некоторых играх появляются облака, основанные на трассировке лучей (Horizon: Zero Dawn, Reset и т.д). В указанных играх используется комбинация предварительно просчитанных октав трехмерного шума различного типа, что накладывает определенные ограничения на анимацию каждой октавы по отдельности, а также на доступную видеопамять и размер самих текстур. Использование же синтеза трехмерного шума в реальном времени сопряжено с большим количеством чтений из текстуры, которые в совокупности с необходимостью дополнительных выборок для освещения существенно ограничивают сферу использования указанного подхода. На устранение этих ограничений, а также поиск алгоритма для имитации освещения облаков в реальном времени, направлена вторая часть данной статьи.

2. МОДЕЛИРОВАНИЕ ВОДНОЙ ПОВЕРХНОСТИ

В результате анализа ряда фотографических изображений водной поверхности, полученных при различных значениях освещения, времени суток, силе ветра, и других параметрах сформулировано предположение, что эффект визуальной правдоподобности при нахождении наблюдателя в широком диапазоне высот (1 м–10 км) достигается при соблюдении следующих условий:

1. Реализуется физически правдоподобная модель распространения волн.

2. Производится реалистичная визуализация водной поверхности с соблюдением соответствующих оптических характеристик (преломление, отражение, оптическая плотность).

3. Моделируются эффекты, характерные при перемещении поверхностных водных масс под действием внешних сил (гребни волн, пена, и т.п.).

Наиболее распространенной моделью водной поверхности в настоящее время является статистическая модель. В ее основе лежит применение быстрого преобразования Фурье (БПФ) к спектру волн в частотном диапазоне. Наиболее распространенной моделью для представления волн в открытом море является спектральная функция, предложенная О. Филлипсом [4, 18] (существуют и другие виды спектров, большинство из которых основано на спектральной модели Пирсона-Московица [2]):

$$P_h(\mathbf{k}) = \frac{A}{k^4}\left|\hat{\mathbf{k}}\cdot\hat{\mathbf{w}}\right|^2 e^{-\frac{1}{k^2L^2}}, \qquad (1)$$

где $L = V^2/g$ характеризует наибольшую высоту волны, возникающей при непрерывном ветре скоростью V, g — гравитационная постоянная, $\hat{\mathbf{w}}$ — вектор направления ветра (здесь $\hat{\mathbf{k}}$ и $\hat{\mathbf{w}}$ — вектора единичной длины), $\mathbf{k}$ — волновой вектор, k — его величина, A — некоторая константа [4].

При анализе алгоритма, использующего для синтеза водной поверхности одну из возможных спектральных функций, становится очевидно, что наиболее затратный с т.з. времени этап алгоритма — процедура БПФ. Несмотря на то,

что предложены быстрые алгоритмы реализации с использованием распараллеливания вычислений, практика показывает, что для моделирования участка поверхности размерности большей 1024^2, особенно при использовании оборудования нижнего ценового диапазона, для достижения приемлемой производительности в приложениях реального времени целесообразно предпринять один из следующих шагов:

1. Распределить вычисления по нескольким кадрам, что поможет снизить нагрузку, но может негативно сказаться на плавности анимации.

2. Просчитать анимации заранее, записав их в виде набора изображений либо трехмерной текстуры, что существенно ускорит визуализацию, но накладывает определенные ограничения на длительность анимации.

3. ОТОБРАЖЕНИЕ ВОДНОЙ ПОВЕРХНОСТИ

С целью обеспечения плавной аппроксимации уровней детализации (метод, основанный на LOD, во многих случаях допускает наличие значительного количества полигонов, меньших экранного пикселя, и требует дополнительного рассмотрения способов расширения к горизонту) при моделировании водной поверхности используется метод проективных сеток [8, 19]. Теоретически, указанный выше метод представляет бесконечное количество уровней детализации (при сколь угодно большом приближении к проецируемой плоскости размер полигонов в проективном пространстве остается неизменным — иными словами, размер элемента полигональной сетки, выраженный в пространстве мировых координат, уменьшается пропорционально сокращению расстояния между ближней плоскостью отсечения и рассматриваемой поверхностью). Внеэкранные области, служащие для устранения краевых разрывов при большой высоте волн могут быть уменьшены путем изменения высоты волны при приближении к краям экрана. Эффект “лесенки” (алиасинг) эффективно устраняется выбором проецирующей камеры и стратегии выборки значений карты смещения на расстоянии.

В ходе визуализации водной поверхности для обеспечения корректной работы метода проективных сеток максимальная (и минимальная) высота волн должна быть задана явным образом. Кроме того, в задачах сопоставления различных спектральных моделей и ряда существующих шкал (Бофорта, Дугласа) возникает необходимость в средстве нормализации полученных в ходе БПФ карт высот и смещений. В общем случае для нормализации указанных карт могут использоваться эмпирические соотношения, определяющие зависимость максимальной $H_{\max}$ и значительной $H^{1/3}$ высот волн [20] от спектрального момента нулевого порядка функции $S(\omega)$:

$$H_{\max} \sim 2H^{1/3}; H^{1/3} = 4\sqrt{M_0};\ M_0 = \int_0^\infty S(\omega)d\omega. \tag{2}$$

Здесь, $S(\omega)$ — изотропный спектр частот, ω — соответственно, частота волны, в то же время, $P_h(\mathbf{k})$ в (1) — двумерный спектр плотности волнового действия, выраженный, как функция волнового вектора $\mathbf{k}$, $D(\omega,\theta) = |\hat{\mathbf{k}} \cdot \hat{\mathbf{u}}|^2$ — функция распространения волн (spreading function), где $\theta = \arccos(\hat{\mathbf{k}} \cdot \hat{\mathbf{u}})$ — угол распространения волны. Полный спектр с учетом функции распространения имеет вид:

$$S(\omega,\theta) = S(\omega)D(\omega,\theta). \tag{3}$$

Поскольку суммарная энергия спектра должна совпадать с суммарной энергией волнового спектра (спектра от волнового вектора) $S(\mathbf{k})$ [21], можно записать:

$$S(\omega,\theta)d\theta d\omega = S(k_x,k_y)dk_x dk_y. \tag{4}$$

Воспользовавшись теоремой замены пределов интегрирования, получим:

$$\int_{\omega(k_x,k_y)} \int_{\theta(k_x,k_y)} S(\omega(k_x,k_y),\theta(k_x,k_y))d\omega d\theta = \\ = \int_{k_x}\int_{k_y} S(k_x,k_y)\left|\det(\mathbf{J}_{\omega,\theta})\right| dk_x dk_y. \tag{5}$$

Здесь $|\det(\mathbf{J}_{\omega,\theta})|$ — абсолютное значение определителя матрицы Якоби, составленной из частных производных ω и θ по компонентам вектора $\mathbf{k}$:

$$|\det(\mathbf{J}_{\omega,\theta})| = \left|\frac{\partial\omega}{\partial k_x}\frac{\partial\theta}{\partial k_y} - \frac{\partial\omega}{\delta k_y}\frac{\partial\theta}{\partial k_x}\right|. \tag{6}$$

Рис. 1. Первый слева — артефакты в методе проективных сеток при нахождении наблюдателя под водой, второй слева — корректная визуализация; второй справа — артефакты в методе проективных сеток при нахождении наблюдателя над водной поверхностью, первый справа — корректная визуализация.

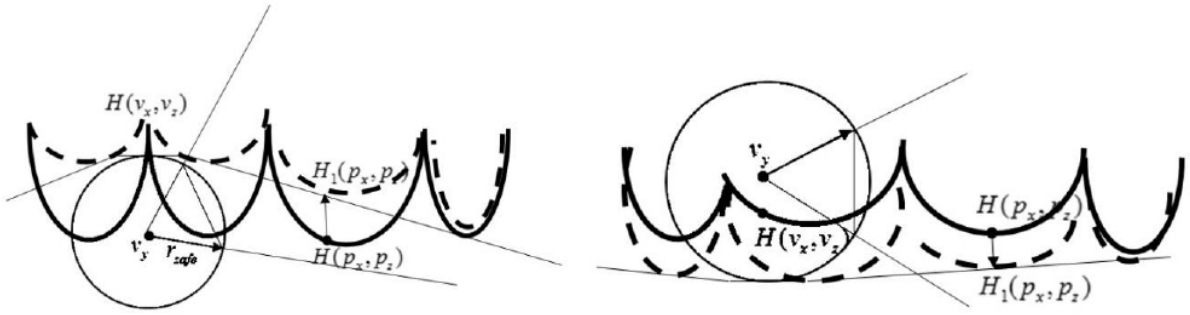

Рис. 2. Слева — схема коррекции при нахождении наблюдателя в толще воды, справа — схема коррекции при положении наблюдателя над водной поверхностью.

В частности, можно записать:

$$d\omega d\theta = |\det(\mathbf{J}_{\omega,\theta})|\, dk_x, dk_y. \tag{7}$$

С учетом того, что $k = \sqrt{k_x^2 + k_y^2}$ и $\theta = \arctan(k_y/k_x)$ найдем частные производные $\frac{\partial\omega}{\partial k_x} = \frac{\partial\omega}{\partial k}\frac{\partial k}{\partial k_x} = \frac{\partial\omega}{\partial k}\frac{k_x}{k}$; $\frac{\partial\omega}{\partial k_y} = \frac{\partial\omega}{\partial k}\frac{\partial k}{\partial k_y} = \frac{\partial\omega}{\partial k}\frac{k_y}{k}$; $\frac{\partial\theta}{\partial k_x} = -\frac{k_y}{k^2}$; $\frac{\partial\theta}{\partial k_y} = \frac{k_x}{k^2}$, подставив которые в (6) получим $|\det(\mathbf{J}_{\omega,\theta})| = \frac{\partial\omega}{\partial k}\frac{1}{k}$. Используя найденное выражение для $|\det(\mathbf{J}_{\omega,\theta})|$ и выражая $S(\mathbf{k})$ из (4), (7) получим:

$$S(\mathbf{k}) = S(\omega,\theta)\frac{\partial\omega}{\partial k}\frac{1}{k}. \tag{8}$$

Учитывая (3) и (8), $\frac{\partial\omega}{\partial k} = \frac{g}{2\sqrt{gk}}$ (для глубокой воды $\omega = \sqrt{gk}$), получим:

$$P_h(\omega) = P_h(\mathbf{k})/\left[\frac{\partial\omega}{\partial k}\frac{1}{k}D\right] = \frac{2g^2A}{\omega^5}\exp\left(-\left(\frac{g}{U}\right)^4\frac{1}{\omega^4}\right). \tag{9}$$

Интегрируя (9) по ω и подставляя результаты в (2), получим искомую зависимость $H_{\max}$ от $P_h(\mathbf{k})$, позволяющую нормализовать карты высот и смещений:

$$M_0 = \frac{AU^4}{2g^2};\ H_{\max} \sim 5,66\sqrt{A}\frac{U^2}{g}. \tag{10}$$

Необходимо отметить, что некоторые из существующих спектров (JONSWAP, Texel MARSEN ARSLOE(TMA) и др.) [1] выведены на основе модели Пирсона-Московица, параметрическая форма которой имеет вид:

$$S(\omega) = 5\pi^4\frac{(H^{1/3})}{T_p^4}\frac{1}{\omega^5}\exp\left[-\frac{20\pi^4}{T_p^4}\frac{1}{\omega^4}\right], \tag{11}$$

где T_p – пиковый период, $H^{1/3}$ — значительная высота. В этом случае удается генерировать спектральную функцию с изначально заданными характеристиками.

При нахождении камеры на большом расстоянии от начала системы координат (что возможно, учитывая рассматриваемый сценарий полета на большие расстояния), а также при больших значениях дальней плоскости отсечения видовой камеры (что также вполне допустимо) точности при вычислении пересечения камеры проектора и плоскости оказывается недостаточно, ввиду чего при повороте и перемещении камеры проявляется ряд видимых артефактов, выражающихся в мерцании и временном пропадании части водной поверхности. Для их устранения горизон-

тальные значения положения исходной камеры были выровнены с началом системы координат, что позволило устранить указанные ошибки вычислений на всем диапазоне работы алгоритма.

При пересечении ближней плоскостью отсечения границы водораздела были отмечены артефакты, которые проявлялись как при нахождении наблюдателя над водой (рис. 1, справа) так и в ее толще (рис. 1, слева).

Для коррекции отображения водной поверхности при нахождении наблюдателя на границе водно-воздушного интерфейса (основные шаги для двух случаев изображены на рис. 2) разработан следующий подход:

1. Вычисляется высота водной поверхности $H(v_x, v_z)$ в горизонтальной проекции положения наблюдателя $\mathbf{v}$;

2. Вычисленная высота сравнивается с положением наблюдателя и, если $v_y < H(v_x, v_z)$, то выполняется привязка высоты водной поверхности к высоте верхней плоскости усеченной пирамиды видимости, в противном случае — к нижней. В разработанном подходе используется концепция расстояния между краями (верхним или нижним) ближней плоскости отсечения и положением наблюдателя $\mathbf{v}$: $r_{safe} = \|\mathbf{p}_{left-top}\mathbf{M}^{-1}_{viewproj} - \mathbf{v}\|$, $\mathbf{p}_{left-top}\mathbf{M}^{-1}_{viewproj}$ — проекция координат левой верхней точки ближней плоскости отсечения, привязка в этом случае осуществляется к $H_1 = lerp(\max(H, r + v_y), H, f)$ при $v_y < \mathrm{H}(v_x, v_z)$, либо к $H_1 = lerp(\min(H, v_y - r_{safe}), H, f)$; $H = \mathrm{H}(p_x, p_z)$ — высота в точке p; $H_1 = \mathrm{H}_1(p_x, p_z)$ — новое значение высоты, соответственно; $f = \max\left(0, \min\left(1, \frac{r_c - r_{safe}}{r_{att}}\right)\right)$ — коэффициент смешивания; $r_e = \|\mathbf{p} - \mathbf{v}\|$ — эффективное расстояние; $r_{att} > 0$ — расстояние, на котором влияние эффекта полностью пропадает.

Визуализация поверхности воды выполняется с учетом преломления и отражения лучей на границе водной поверхности, при этом для определения доли отраженного (преломленного) излучения используется аппроксимация формулы Френеля [22].

Текстура отражений содержит изображение сцены, полученное на предыдущем кадре [23], при этом зеркальные отражения (преломления) вычисляются путем нахождения пересечения отраженного (преломленного) луча с поверхностью сцены с использованием нескольких шагов линейного и бинарного поиска (количество промежуточных шагов может быть значительно сокращено при использовании вспомогательных структур, аналогичных MM-пирамиде [24], построенных на основе линейной карты глубины).

Отражения объектов сцены, имеющих значительную протяженность у краев видимой области (небо, облака), требуют большого количества шагов процедуры трассировки и подвержены краевым артефактам, связанным с отсутствием информации об элементах сцены за пределами геометрического буфера, строятся процедурно. Так для построения облаков для карты отражений используется оригинальный алгоритм с меньшим количеством итераций, в случае небесной сферы возможно использование как оригинального алгоритма, так и интерполяции между значениями яркости в нескольких точках небесной сферы.

Необходимо добавить, что для обеспечения быстрого доступа к текстуре отражений небосвода, а также устранения краевых артефактов при построении протяженных объектов (в т.ч., облаков и небо), целесообразно использовать сферическую проекцию, где направление трассировки определяется из двумерных текстурных координат.

Для получения отраженного солнечного света используется локальная модель освещения, предложенная В. Россом и Э. Брунетоном [9, 25]:

$$I_{sun} \approx L_{sun}\Omega_{sun}p(\zeta_h)\frac{R + (1-R)(1-\mathbf{v}\cdot\mathbf{h})^5}{4h_z^4\cos\theta_v(1 + \wedge(a_v) + \wedge(a_l))}. \quad (12)$$

Здесь L_{sun} — яркость солнечного излучения; Ω_{sun} — величина телесного угла, на котором значения модели освещения являются постоянными; выражение в числителе является аппроксимацией коэффициента Френеля; $h_z = \mathbf{n}\cdot|\mathbf{l} + \mathbf{v}|$, $\mathbf{n}, \mathbf{l}$ и $\mathbf{v}$ — вектор нормали, вектор на источник освещения и наблюдателя соответственно;

Рис. 4. Слева — демонстрация результатов локальной модели освещения; справа — имитация объемного освещения.

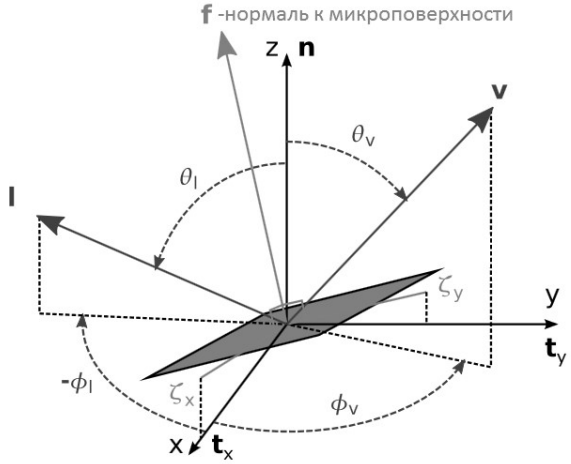

Рис. 3. Координатная система локальной модели освещения водной поверхности.

остальные параметры выражаются следующим образом:

$$\wedge(a_i) = \frac{\exp(-a_i^2) - a_i\sqrt{\pi}\mathrm{erfc}(a_i)}{2a_i\sqrt{\pi}},\ i \in \{v, l\} \quad (13)$$

$$a_i = (2(\sigma_x^2 \cos^2\phi_i + \sigma_y^2 \sin^2\phi_i)\tan\theta_i)^{-1/2}. \quad (14)$$

$$p(\zeta) = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left(-\frac{1}{2}\left(\frac{\zeta_x^2}{\sigma_x^2} + \frac{\zeta_y^2}{\sigma_y^2}\right)\right). \quad (15)$$

Соотношения между величинами ϕ_l, ϕ_v, θ_v, θ_l, компонентами вектора нормали к микрограни ζ_x и ζ_y и нормалью к поверхности $\mathbf{n}$ приведены на рис. 3, а также в [25]:

Несмотря на множество интенсивных вычислений, отраженных в (12–15) и анализе производительности исходного алгоритма, выявлено, что наибольшие затраты по времени вызывает расчет и выборка значений параметра σ^2, для расчета которого применяется либо численное интегрирование в вершинном и пиксельном шейдерах (что, очевидно, существенно уменьшает производительность) либо выборка из предварительно построенной трехмерной текстуры, что возлагает дополнительные требования к оборудованию. В ходе анализа оригинального алгоритма эмпирически выявлена зависимость указанного параметра от величины вертикальной проекции видового вектора $\mathbf{v}_y$, при помощи которой определяются пределы суммирования σ^2:

$$\left[\sigma_x^2\sigma_y^2\right]^T = \sum_{N_{\min}}^{N_{\max}} [k_{i,x}^2 k_{i,y}^2]^T / \|\mathbf{k}_i\|^2 \left(1 - \sqrt{1 - \|\mathbf{k}_i\|^2 h_i^2}\right). \quad (16)$$

Ввиду указанной зависимости и необходимости повышения эффективности вычислений принято решение о замене указанной функцией с использованием следующего соотношения:

$$[\sigma_x^2\sigma_y^2]^T \sim \mathbf{a} - \log_2(C/\mathbf{v}_y)(\mathbf{a} - \mathbf{b}). \quad (17)$$

Здесь $\mathbf{a}$ и $\mathbf{b}$ — значения σ^2 в двух заранее выбранных точках. На практике было выявлено, какие компоненты указанных векторов принимают значения, пропорциональные значениям σ_{total}^2, полученным суммированием по всем возможным длинам волн, кроме того, возможно управление дисперсией по одному из двух горизонтальных направлений выбором соответствующих значений. Замена выборки из трехмерной текстуры простыми арифметическими действиями обеспечила улучшение производительности алгоритма на 5%, позволяя успешно использовать его на системах нижнего ценового диапазона (указанные в статье

изображения получены в реальном времени при разрешении 1920×1080 на машине с мобильным процессором и видеокартой).

В соответствии с описанным ранее, выведен алгоритм анимации и визуализации водной поверхности, реализованный в программном коде, вершинном и пиксельном шейдерах, представляющий собой следующую последовательность действий (результаты, полученные в ходе моделирования водной поверхности приведены в конце статьи на рис. 4):

1. Построить карты смещения, нормалей и карты складок, используя один из доступных спектров и параллельную реализацию БПФ.

2. Найти пересечение потенциально видимой области с базовой плоскостью согласно алгоритму проективных сеток [8].

3. Передать четыре найденные координаты ($\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ и $\mathbf{d}$) проекции потенциально видимой области на базовую плоскость S_{base} в вершинный шейдер, выполнить линейную интерполяцию между ними, используя в качестве параметра экранные координаты соответствующей вершины $x, y \in (0, 1)$: $\mathbf{p} = \mathbf{a}x + (1-x)\mathbf{b} + (1-y)(\mathbf{c}x + (1-x)\mathbf{d})$, при необходимости расширить вершины за границы видимой области для компенсации горизонтальной составляющей смещения.

4. На основании полученных мировых координат вычислить координаты для выборки из текстур.

5. Применить карты смещения к координате вершины путем сложения координаты $\mathbf{p}$ с масштабированным значением вектора смещения, выбранного по текстурным координатам, полученным на предыдущем шаге.

6. Скорректировать положения вершин согласно описанному ранее подходу (скорректировать вершины при нахождении наблюдателя на границе водно-воздушного интерфейса, при необходимости убрать горизонтальные смещения для краевых вершин).

7. Передать необходимые данные в пиксельный шейдер.

8. Выбрать значения яркости преломленного L_{sky} и отраженного освещения L_{ref} из соответствующих текстур.

9. На основании значения глубины найти значение цвета воды: $L_{sea} = attenuation \cdot L_{ref} + inscatter \cdot L_{fog}$ $attenuation = e^{-c(s-p_y+l)}$, c — плотность, s — базовый уровень водной поверхности, $\mathbf{p}$ — точка пересечения видового луча и поверхности воды, $\mathbf{o}$ — точка пересечения видового луча и морского дна, $l = \|\mathbf{p} - \mathbf{o}\|$, $inscatter = e^{-c(s-o_y)} \left[\frac{e^{(d_y-1)cl}-1}{(d_y-1)c}\right]$, $\mathbf{d} = \frac{\mathbf{p}-\mathbf{o}}{l}$.

10. $I = FL_{sky} + (1-F)L_{sea} + I_{sun} + I_{foam}$.

Здесь F — коэффициент Френеля, I_{sun} — отраженный свет Солнца, полученный в соответствии с локальной моделью освещения, описанной ранее, I_{foam} — цвет нескольких слоев пены, смешанный в соответствии с величиной, полученной из карты складок.

Анализ и измерение времени выполнения алгоритма позволяют предположить, что его производительность зависит в существенной мере от разрешения экрана, поскольку большая часть работы происходит в пиксельных шейдерах. Тем не менее, время выполнения алгоритма для Full-HD разрешения (1920×1080) составляет 16 мс (что соответствует частоте обновления 63 кадров/с) на аппаратном обеспечении 8-летней давности (мобильная видеокарта AMD поколения HD5000), что позволяет использовать его в приложениях реального времени.

4. МОДЕЛИРОВАНИЕ АТМОСФЕРНЫХ ЭФФЕКТОВ

4.1. Визуализация неба

В соответствии с общепринятой методикой небо моделируется в виде полусферической полигональной сетки с нормалями, направленными к центру указанной полусферы. Количество света, поступившее в одном и рассеянное в другом направлении, определяется при помощи фазовой функции, предложенной

Хеньи-Гринштейном [26, 27]. Для каждой видимой точки полусферы цвет рассчитывается по формуле, приведенной в работах Т. Нишиты и И. Добаши [28, 29].

На практике удалось обеспечить построение двумерных таблиц рассеяния Ми и Релея на графическом процессоре предыдущего поколения с достижением частоты реального времени (60 кадров/с), однако, в связи с высокой интенсивностью вычислений в целом и специфики разрабатываемого приложения, были внесены следующие дополнительные изменения в оригинальный алгоритм:

1. Небо моделируется полусферой, центр которой совпадает с текущим положением видовой камеры, в связи с этим цвета по-прежнему рассчитываются при пересечении лучей от наблюдателя до верхнего края атмосферы, но проецируются на локальный сферический меш сферы.

2. Земная поверхность представлена высотной картой, проецируемой на плоскость.

3. Выход за пределы земной атмосферы не рассматривается ввиду особенности моделируемых летательных аппаратов, что уменьшает количество возможных вариантов и, соответственно, шейдерных программ.

Кроме того,

1. Проведено кэширование значений яркостей для суточной траектории движения Солнца в ряд таблиц с возможностью имитации смены его положения путем линейной интерполяции значений соседних таблиц, что упрощает вычисления до двух чтений двумерной текстуры, превосходя по производительности и используемым ресурсам видеопамяти ряд методов, использующих трехмерные текстуры [30], при этом отсутствие множественного рассеяния может быть скомпенсировано рядом подходов, среди которых снижение насыщенности при помощи эмпирической функции, а также использование процедуры трассировки лучей.

2. Устранены ресурсоемкие вычисления, связанные с вычислением двойных интегралов в уравнении освещения [29, 31], при этом возможно вычисление значений следующей таблицы в параллельном потоке средствами центрального процессора.

4.2. Визуализация облаков

При визуализации атмосферы неизбежно возникает задача реалистичного моделирования облаков. В этой связи необходимо отметить несколько наиболее распространенных подходов:

1. Представление облаков в виде анимированных во времени двумерных текстур, проецируемых на поверхность небесной сферы (либо кубических карт, созданных из фотографий, содержащих в себе изображения облаков), к этой категории можно отнести множество компьютерных игр.

2. Моделирование при помощи полигональной сетки [32].

3. Представление облаков в виде одиночного или нескольких слоев — отдельного полигона, либо группы полигонов параллельных земной поверхности [33].

4. Моделирование облаков в виде системы полигональных частиц, обращенных к наблюдателю (импосторов) [13, 15].

5. Представление облаков в виде элементов объемного изображения, вокселов (облака задаются в виде объемной функции плотности) [16, 17, 34].

Так, в большинстве симуляторов полета (включая известные серии Microsoft Flight Simulator [35], X-Plane [36], Flight Gear [37]) и многих компьютерных играх используется система частиц (импосторов), размещаемых и анимируемых вручную. При таком подходе возможен полный контроль формы и поведения моделируемых объектов, увеличивая вместе с тем трудозатраты, связанные с ручной настройкой всех необходимых параметров. Большинство алгоритмов, позволяющих получить наиболее реалистичное изображение, зачастую слишком медленны (хотя и выполняются отдельно от других компонентов системы в интерактивное время), либо требуют расширенной аппаратной

поддержки для достижения максимального качества.

В соответствии с разработанной моделью, функция плотности облака математически представляется в виде суммы нескольких октав трехмерного шума:

$$q(\mathbf{p}, t_i) = \sum_{k=0}^{N_{oct}} w_k N_k(\mathbf{p}, \mathbf{v_k}, a_k, b_k, c_k, t_i), \qquad (18)$$

где $w_k < w_{k-1}$; $a_k > a_{k-1}$; параметры a_k, b_k, c_k, w_k, количество октав N выбираются эмпирически, $\mathbf{v_k}$ — вектор скорости смещения k-ой октавы, N_{oct} — количество октав шума во времени. В ходе анализа фотографических изображений различного типа облаков выведено следующее соотношение для $N_k(\mathbf{p}, \mathbf{v_k}, a_k, b_k, c_k, t_i)$:

$$N_k(\mathbf{p}, \mathbf{v_k}, a_k, b_k, c_k, t_i) = 1 - |1 - \\ -(N(\mathbf{p}a_k + \mathbf{v_k}t_i)b_k + c_k)|. \qquad (19)$$

При моделировании перистых и перисто-слоистых облаков координаты вектора $\mathbf{p}$ смещаются на значения, определяющиеся той же шумовой функцией $\mathbf{p}_k = \mathbf{p}_{k-1} + N_{k-1}\mathbf{s}_k$, где $\mathbf{s}_k$ — вектор масштабирования, компоненты которого характеризуют величину искажений по каждой из осей.

В качества шумовой функции используется виртуальная трехмерная текстура, получающаяся из двумерной шумовой текстуры смещением каждого слоя на константу, текстурные координаты при этом определяются так:

$$\begin{aligned} \mathbf{a}_{xy} &= \mathbf{p}_{xz} + \lfloor \mathbf{p} \rfloor_y k_o \\ \mathbf{b}_{xy} &= \mathbf{p}_{xz} + (\lfloor \mathbf{p} \rfloor_y + 1) k_o \end{aligned}, \qquad (20)$$

где k_o — смещение между слоями, выбираемое некоторым нечетным числом, результат выборок из текстуры по указанным координатам смешивается с коэффициентом $\{\mathbf{p}\}_y$. Преимущества указанного подхода над выборкой из трехмерного объема (текстуры) очевидны: при выборке из трехмерного объема 1024^3 потребуется до 4 Гб памяти для размещения одной текстуры, в случае же двумерной текстуры объем памяти не превысит 4 Мб. Указанной особенностью объясняется низкая детализация облаков в большинстве недавних компьютерных игр (Horizon: Zero Dawn [38], Reset и т.д.). Кроме того, в случае использования указанного подхода возможно задавать уникальные коэффициенты смещения $\mathbf{v_k}$ для каждого из слоев, что затруднено в случае использования трехмерной текстуры. Однако количество выборок из двумерной текстуры оказывается существенно выше, чем в случае использования трехмерной текстуры, в связи с чем предложено следующее решение. Семплы по координатам $\mathbf{a}_{xy}$ и $\mathbf{b}_{xy}$ записываются в разные каналы одной текстуры (в оставшиеся 2 канала записываются смещенные значения для компоненты освещения), что существенно повышает производительность (в 3–4 раза по сравнению с использованием первоначального подхода).

Алгоритм трассировки лучей представляет собой классическую процедуру трассировки лучей [16, 17, 34]. При этом, значения цвета и прозрачности на текущем шаге — C_i и a_i, получаются в ходе следующей последовательности действий:

1. На основании координаты точки $\mathbf{p}_i$ получить значение плотности $N_{\sum}(\mathbf{p}_i, t_i)$ путем сложения нескольких октав трехмерной шумовой функции.

2. Получить значение прозрачности a_i, используя найденное ранее значение плотности $a_i = N_{\sum}(\mathbf{p}_i, t_i) \cdot g((p_y - h_{cloud})/\delta_{cloud})$, где p_y — высота точки $\mathbf{p}_i$, $h_{cloud} = (h_{\min} + h_{\max})/2$ — высота слоя, δ_{cloud} — толщина указанного слоя, $g(x)$ — линейная функция вида: $g(x) = ax + b$.

3. На основании значения, смещенного по направлению к источнику $N_{\sum}(\mathbf{p}_i + \mathbf{l}m, t_i)$ освещения получить направленную компоненту освещения C_{sc} (в большинстве известных подходов [38] для получения компоненты направленного освещения используется 6 и более выборок текстуры по направлению к источнику освещения, что в совокупности с большим количеством шагов по лучу приводит к существенному падению производительности);

4. Найти цвет в данной точке как сумму количества света, полученного от источника C_d, отраженного земной поверхностью C_{gr} и рассеянного освещения C_a: $C_i = lerp(C_a, C_d, df) + C_{gr}$, где

Рис. 5. Слева — результат визуализации облаков; справа — результат визуализации дождя.

$C_a = lerp(L_{sky}, L_{sun}, \mathrm{gr})$; L_{sky}, L_{sun}, gr — цвет неба, Солнца и высотный градиент, соответственно; $C_d = L_{sun}\gamma_1(C_{cs})f_{HG}(\theta)$, $f_{HG}(\theta)$ — фазовая функция Хеньи-Гринштейна [26]; $C_{gr} = L_{gr}(1 - \mathrm{gr})^m$; L_{gr} характеризует количество освещения, отраженного земной поверхностью, k и n, m — некоторые константы, выбираемые эмпирически; $df = \gamma_2(C_{cs})$, где $\gamma_j(x) = (kx + m)^{n_i}$.

4.3. Визуализация погодных эффектов

При реализации погодных эффектов возникает необходимость реалистичного отображения таких явлений, как дождь и снег. Традиционно существует большое количество техник, использующих системы частиц, которые моделируют отдельные капли дождя или хлопья снега [39–41], а также анимированных текстур, проецируемых, например, на коническую поверхность [42, 43]. Ввиду того, что указанные эффекты обладают объемной природой, а также в связи с необходимостью учета взаимодействия указанных эффектов и окружающих поверхностей (например, намокания участков поверхности) и с целью унификации средств конфигурирования указанных эффектов был избран процедурный подход, аналогичный примененному для моделирования облаков. Здесь одна и та же шумовая текстура задает как представление отдельных капель, так и внешний вид поверхности, подвергшейся воздействию одного из указанных эффектов. При этом текстурные координаты для представления отдельных капель определяются следующим соотношением:

$$uv = \left[\frac{1}{Th} + t\mathbf{v}_v, \frac{1}{L} + t\mathbf{v}_h\right]\mathbf{M}. \qquad (21)$$

Здесь, Th характеризует толщину капель, L — длина капель (при моделировании снежных хлопьев выбираем $L \sim Th$, для имитации дождевых капель $L \gg Th$), t — переменная времени, $\mathbf{v}_v$ и $\mathbf{v}_h$ — вертикальная и горизонтальная компоненты вектора скорости соответственно, $\mathbf{M}$ — двумерная матрица ориентации эффекта относительно мировой системы координат. Для имитации эффекта объема используется несколько анимируемых плоскостей, расположенных перпендикулярно вектору наблюдения (в большинстве случаев достаточно 10 плоскостей).

5. ЗАКЛЮЧЕНИЕ

Результаты визуализации водной поверхности представлены на рис. 4, результаты визуализации атмосферных эффектов — на рис. 5. Все результаты получены в реальном времени (50–65 кадров/с) на мобильной видеокарте и процессоре (Intel Core i7-Q720M, ATI Radeon HD5870). Исходя из анализа полученных результатов и учитывая результаты других исследователей, можно заключить, что использование трассировки лучей в реальном времени (имея в виду постоянный рост вычислительной мощности оборудования) является перспективной техникой, пригодной для применения как задач освещения (например, эффекты отражения, преломления), так и для построения объектов с учетом их объемной природы (облака, дождь, туман, снег). Однако для ряда задач трассировка лучей в экранном пространстве требует особого подхода ввиду недоступности информации о визуализируемой сцене за пределами экрана и необходимости использования расширенного представления

моделируемой сцены, что является предметом для дальнейших исследований.

СПИСОК ЛИТЕРАТУРЫ

1. *Frechot J.* Realistic Simulation of Ocean Surface using Wave Spectra, Proc. First Int. Conf. Comput. Graph. Theory Appl. (GRAPP 2006), 2006, v. 4, p. 76–83.
2. *Hinsinger D., Neyret F., Cani M.-P.M.* Interactive animation of ocean waves, Symp. Comput. Animat., 2002, v. 46, p. 5471–5481.
3. *Jensen L.S., Golias R.* Deep-water animation and rendering, Game Developer's Conference (Gamasutra), 2001. http://www.gamasutra.com/gdce/jensen/ /jensen_03.htm
4. *Tessendorf J.* Simulating Ocean Water, Environment, 2001, v. 2, p. 1–19.
5. *Yang X. et al.* GPU-Based Real-time Simulation and Rendering of Unbounded Ocean Surface, Ninth Int. Conf. Comput. Aided Des. Comput. Graph., 2005, v. 1, p. 428–433.
6. *Kryachko Y.* Using vertex texture displacement for realistic water rendering, GPU Gems, 2005, v. 2, p. 283–294.
7. *Chiu Y.F., Chang C.F.* GPU-based ocean rendering, 2006 IEEE International Conference on Multimedia and Expo, ICME 2006, p. 2125–2128.
8. *Johanson C., Lejdfors C.* Real-time water rendering, Lund Univ., 2004. p. 42.
9. *Ross V., Dion D., Potvin G.* Detailed analytical approach to the Gaussian surface bidirectional reflectance distribution function specular component applied to the sea surface // Journal of the Optical Society of America A, Optics, image science, and vision, 2005, v. 22, no. 11, pp. 2442–2453.
10. *Premoze S., Ashikhmin M.* Rendering natural waters, Computer Graphics Forum, 2001, v. 20, no. 4, pp. 189–199.
11. *Iwasaki K., Dobashi Y., Nishita T.* Efficient rendering of optical effects within water using graphics hardware, Proc. Ninth Pacific Conf. Comput. Graph. Appl. Pacific Graph, 2001, p. 374–383.
12. *Hu Y. et al.* Realistic, real-time rendering of ocean waves, Comput. Animat. Virtual Worlds, 2006, v. 17, no. 1, p. 59–67.
13. *Wang N.* Realistic and Fast Cloud Rendering, J. Graph. Tools, 2004, v. 9, p. 21–40.
14. *Wenzel C.* Real-Time Atmospheric Effects in Games, 2006. http://amd-dev.wpengine.netdna-cdn.com/ /wordpress/media/2012/10/Wenzel-Real-time_ _Atmospheric_Effects_in_Games.pdf
15. *Harris M.J., Lastra A.* Real-Time Cloud Rendering, Computer Graphics Forum, 2001, v. 20, p. 76–85.
16. *Kajiya J.T., Von Herzen B.P.* Ray tracing volume densities, ACM SIGGRAPH Computer Graphics, 1984, v. 18, pp. 165–174.
17. *Engel K. et al.* Real-time volume graphics, Proceedings of the conference on SIGGRAPH 2004 course notes — GRAPH'04, 2004, 29-es p.
18. *Phillips O.M.* On the generation of waves by turbulent wind, J. Fluid Mech, 1957, v. 2, no. 3, p. 417–445.
19. *Mahsman J.D.* Projective grid mapping for planetary terrain. University of Nevada, Reno, 2010, 152 p.
20. *Bouws E., Draper L., Laing A.K. et al.* Guide to Wave Analysis and Forecasting, WMO-No. 702, Geneva: WMO, 1998, 159 p.
21. *Rodriguez O.C. et al.* Modeling arrival scattering due to surface roughness, Proc. 10th European Conference on Underwater Acoustics, 2010, p. 1–8.
22. *Schlick C.* An Inexpensive BRDF Model for Phsyically-based Rendering, Computer Graphics Forum, 1994, v. 13, no. 3. p. 233–246.
23. *Shishkovtsov O.* Deferred shading in stalker, GPU Gems 2: Programming Techniques for High-Perfomance Graphics and General-Purpose Comput., 2005, v. 2, p. 143–166.
24. *Tevs A., Ihrke I., Seidel H.-P.* Maximum mipmaps for fast, accurate, and scalable dynamic height field rendering, Proc. 2008 Symp. Interact. 3D Graph. games — SI3D'08, 2008, p. 183–190.
25. *Bruneton E., Neyret F., Holzschuch N.* Real-time realistic ocean lighting using seamless transitions from geometry to BRDF, Computer Graphics Forum, 2010. v. 29, no. 2. p. 487–496.
26. *O'Neil S.* Accurate atmospheric scattering, GPU Gems, 2005, v. 2, pp. 253–268.
27. *Nielsen R.S.* Real Time Rendering of| Atmospheric Scattering Effects for Flight Simulators, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2003.

28. *Miyazaki R., Dobashi Y., Nishita T.* Simulation of cumuliform clouds based on computational fluid dynamics, Proc. Eurographics 2002 Short Present., 2002, p. 405–410.

29. *Dobashi Y., Nishita T., Yamamoto T.* Interactive rendering of atmospheric scattering effects using graphics hardware, Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, 2002, v. 2. p. 99–108.

30. *Bruneton E., Neyret F.* Precomputed atmospheric scattering, Computer Graphics Forum, 2008. v. 27, no. 4, p. 1079–1086.

31. *Nishita T., Sirai T.* Display of The Earth Taking into Account Atmospheric Scattering, Special Interest Group on Graphics and Interactive Techniques, 1993. p. 175–182.

32. *Bouthors A. et al.* Interactive multiple anisotropic scattering in clouds, ACM Symposium on Interactive 3D Graphics and Games, 2008, p. 173–182.

33. *Bouthors A., Neyret F., Lefebvre S.* Real-time realistic illumination and shading of stratiform clouds, Proc. Eurographics Workshop on Natural Phenomena, Vienna, Austria, 2006, p. 1–10.

34. *Wrenninge M., Zafar N.Bin.* Production Volume Rendering Fundamentals, Special Interest Group on Graphics and Interactive Techniques 2011 Course Notes, 2011, p. 71.

35. Microsoft Flight Simulator — Product Information, https://www.microsoft.com/Products/Games/ /FSInsider/product/Pages/

36. X-Plane 11 is Here | More Powerful. Made Usable, http://www.x-plane.com/

37. FlightGear Flight Simulator — sophisticated, professional, open-source, http://www.flightgear.org/

38. Schneider A. Real-Time Volumetric Cloudscapes // GPU Pro 7 Adv. Render. Tech. CRC Press, 2016. p. 97.

39. *Tatarchuk N., Isidoro J.* Artist-directable real-time rain rendering in city environments // Natural Phenomena, 2006, p. 61–73.

40. *Tariq S.* Rain // NVIDIA Whitepaper. 2007, http://developer.download.nvidia.com/ /whitepapers/2007/SDK10/RainSDKWhitePaper.pdf

41. *Rousseau P., Jolivet V., Ghazanfarpour D.* Realistic real-time rain rendering, Computer Graphics, 2006, v. 30, no. 4, p. 507–518.

42. Game environments — Part B: rain | fxguide, https://www.fxguide.com/featured/game-environments-partb/

43. *Wang N., Wade B.* Rendering falling rain and snow // ACM Special Interest Group on Graphics and Interactive Techniques 2004 Sketches. 2004. p. 14.