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

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА ТЕОРЕТИЧЕСКАЯ ИНФОРМАТИКА И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Графическая визуализация поверхности океана и расчёты заданных параметров с применением программного интерфейса OpenGL

Студент ИУ9-52б _____ М. Д. Соколовский
(Группа) (Подпись, дата) (И.О.Фамилия)

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

Консультант _____ В. В. Соборова
(Подпись, дата) (И.О.Фамилия)

Москва
2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Визуализация поверхности океана может использоваться для формирования фотореалистичных изображений в большом кругу сфер деятельности, таких как: средства массовой информации, различные дизайнерские решения, сфера развлечений (в частности – видеоигры и кинематограф), и т.д. В зависимости от того, в какой сфере деятельности необходимо визуализировать водную поверхность, от системы требуется соблюдение определённых критериев, в частности – степень фотореалистичности и физической корректности, а также производительность этой системы и минимальные и рекомендуемые требования к вычислительным мощностям оборудования.

Данная работа посвящена исследованию и разработке системы визуализации поверхности океана, учитывающей воздействие физических явлений при заданных параметрах. Система реализуется на языке программирования C++ версии стандарта C++17 и предназначена для работы в операционной системе на базе Linux. Для работы с графической подсистемой компьютера используется программный интерфейс OpenGL версии не младше 4.3 в режиме core-profile, с использованием специального языка программирования шейдеров GLSL.

Система визуализации представляет собой оконное приложение, отображающее трёхмерное пространство – визуализируемую сцену, также называемую миром. Приложение позволяет изменять текущее положение и направление взгляда в мире путём считывания ввода с клавиатуры и мыши. Кроме этого, приложение позволяет динамически изменять состояния и параметры системы, а также производить некоторые дополнительные действия.

При визуализации учитываются различные параметры и явления, такие как физические явления, положение, направление, а также прочие свойства условной камеры и прочие настройки отображения, в том числе – связанные с корректным отображением визуализированной модели на экране монитора.

В данной работе будут рассмотрены и сравнены различные подходы к моделированию геометрии поверхности. После расчёта геометрии будут рассмотрены технологии реализации оптических эффектов, связанные с освещением и текстурированием. Для их расчётов в системе учитываются такие показатели, как:

- Глубина воды
- Амплитуда, и прочие свойства волн
- Скорость и направление ветра
- Показатель преломления воды
- Положение, интенсивность и прочие свойства источника света

Также система позволяет задавать пространственные характеристики визуализируемого участка водной поверхности, а также условное разрешение этого участка (количество вычисляемых точек геометрии на единицу площади).

Помимо этого, в процессе работы приложения на экране выводится некоторая отладочная информация, путём визуализации изменяемых строк текста и изображений.

1 Модели расчёта геометрии поверхности

Согласно работе [1], существующие модели расчёта геометрии можно условно разделить на три категории:

1. Приближённые методы, позволяющие получить лишь похожие на реальную поверхности. Они основаны на использовании функций шумов, и прочих подходов.
2. Методы, основанные на использовании тригонометрических функций. Они реализуют приближённую, однако более фотореалистичную визуализацию.
3. Физически корректные, или близкие к этому методы.

Обычно наблюдается обратная зависимость между физической достоверностью и требованиями к мощности вычислительной системы. Однако, часто, для получения подходящего изображения, приближённые методы подходят лучше, нежели физически корректные. Также, большинство моделей могут быть совмещены друг с другом, для объединения преимуществ каждой отдельной модели.

Выбор метода выполняется исходя из поставленной задачи и доступных ресурсов. В большинстве случаев, в которых основным критерием качества является достаточная визуальная достоверность изображения, рационально использовать методы из первой и второй категорий.

Отдельно рассмотрим физически корректные методы. Они используют подход, при котором жидкость рассматривается как множество элементарных объектов – частиц, разделённых по трёхмерной сетке. При этом рассчитывается их поведение на основе физических законов, таких как уравнение Навье-Стокса, и другие. Однако для получения реалистичного изображение число таких частиц должно быть слишком большим, что приведёт к слишком сильному росту требований к вычислительным ресурсам. Поскольку целью данной работы является лишь визуализация поверхности воды, но не симуляция всего объёма жидкости, подобные методы в ней рассматриваться не будут.

Другой приближённый метод визуализации, также основан на физических явлениях, однако он реализует лишь похожую на реальную поверхность воды, так как не учитывает законов поведения жидкости. Он основан на представлении поверхности воды в виде двумерного дискретного поля материальных точек, каждая из которых связана с соседними абстрактными пружинами. Движение такой системы описывается законами механики, такими как закон Гука и другими. Задав свойства этой системы и добавив различные ограничения на перемещение точек, необходимо вывести систему из положения равновесия, после чего она будет совершать колебательные движения, напоминающие волны на воде. Этот метод описан в работе [2].

Далее будут рассматриваться только приближённые методы расчёта. Для всех таких методов, описанных далее в этой работе, поверхность воды разбивается регулярной сеткой на треугольники, как показано на рисунке 1. Методы моделирования изменяют значение высоты (координата Y) узлов сетки, а также могут вводить горизонтальное отклонение узлов от их исходного положения.

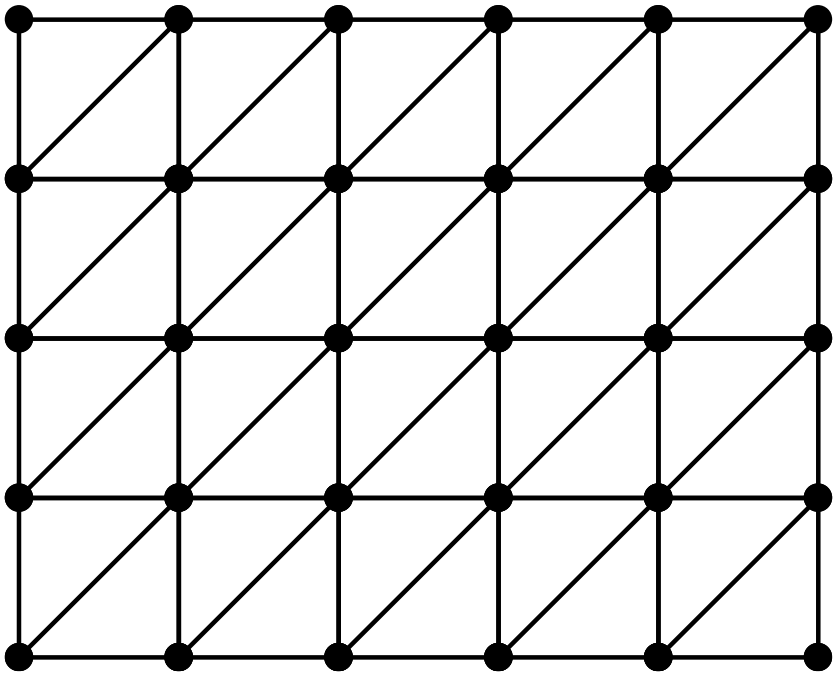

Рисунок 1: Регулярная сетка

1.1 Шум Перлина

Шум – это функция, позволяющая получать случайные значения для заданной точки в пространстве заданной размерности. Для большей универсальности далее будем предполагать, что данная функция будет принимать вещественные значения в интервале от 0 до 1. Определённую таким образом функцию, умножением на константу, можно легко привести к требуемому интервалу значений.

В качестве примера шума можно рассмотреть функцию, которая каждой точке будет сопоставлять значение, независящее от остальных точек пространства, с равной вероятностью принимающее все значения из заданного диапазона. Шум, определённый таким образом называется белый шум. Однако такой шум не является плавным. Поэтому область его применения ограничена.

Для генерации непрерывного и более плавного шума могут использоваться шум Перлина и симплексный шум. Они являются плавными, а также позволяют, путём задания входных параметров, генерировать в пространстве, размытые пятна заданного размера и чёткости. Далее будут рассмотрены только функции шума для пространства размерности два (плоскости). Результатом генерации такого шума является изображение (текстура).

Для вычисления шума Перлина в точке необходимо найти 4 точки с целочисленными координатами, наиболее близких к заданной точке. Так как точки с целочисленными координатами образуют прямоугольную сетку на плоскости, нужными точками являются вершины ячейки этой сетки, внутри которой находится заданная точка. Это показано на рисунке 2. Для каждой вершины любым доступным образом генерируется случайный вектор единичной длины, называемый градиентом в данной точке (на рисунке – вектора a_i). Далее, для каждой из этих 4 точек вектор градиента скалярно умножается на вектор, направленный к заданной точке (на рисунке – вектора b_i).

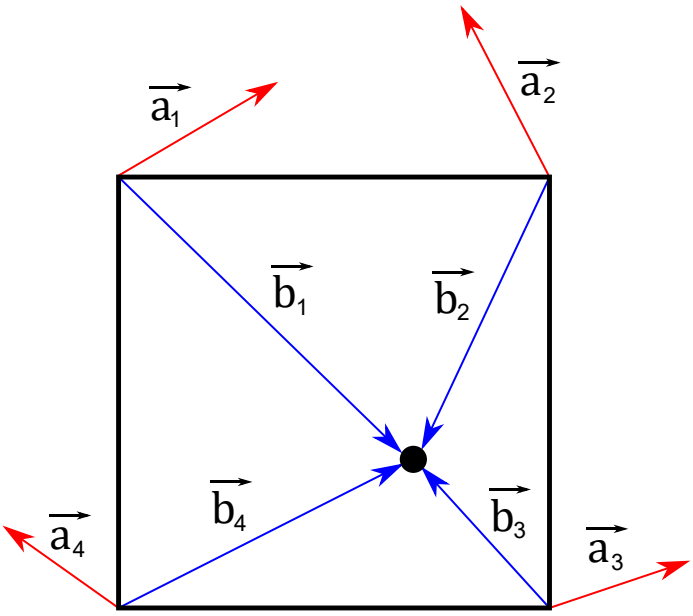

Рисунок 2: Вычисление шума Перлина

Полученные значения необходимо интерполировать, для получения итогового результата. Однако, при использовании линейной интерполяции на границах ячеек сетки могут возникать неровности. Поэтому в качестве интерполирующей функции может использоваться, например, одна из стандартных функций языка GLSL – `smoothstep`, а также любая другая функция, не приводящая к нарушению непрерывности на границах ячейки. Сам процесс интерполяции в двумерном пространстве выполняется за 3 этапа: на первых двух этапах интерполируются значения между соседними вершинами, а на последнем – результат двух предыдущих.

Далее, мы можем уменьшить шаг ячеек сетки в два раза, и для такой сетки заново вычислить шум. Каждая следующая функция шума, полученная таким образом называется октавой. Для получения более размытых результатов нужно просуммировать несколько октав шума.

Таким образом, шум Перлина может быть гибко настроен для различных задач путём выбора таких параметров, как функция выбора случайных градиентов, функция интерполяции и количество суммируемых октав.

Двумерный шум, определённый выше, естественным образом может быть изменён для пространства нужной размерности. В целях упрощения и ускорения генерации допустимо также делить пространство не на квадратные ячейки, а на треугольники (в случае трёхмерного пространства – на тетраэдры вместо кубов, и т.д.). Полученный таким образом шум называется симплексным, и был также введён Перлином как улучшение своего предыдущего шума.

Шум Перлина может быть использован для создания двумерной текстуры, определяющий изменение высоты узлов сетки. Генерация такого шума обладает низкими требованиями к вычислительным ресурсам системы, позволяя с небольшими затратами получить плавную модель водной поверхности.

Для визуализации поверхности, изменяющейся во времени, необходимо сгенерировать трёхмерное поле шума, а для визуализации использовать сечение полученного поля плоскостями, перпендикулярными третьей оси, и двигающимися вдоль неё с течением времени.

Однако, использование такого метода не позволяет получать достоверное изображение волн, так как в нём не учитываются физические параметры. Несмотря на это, данный метод может использоваться в низкопроизводительных системах для получения удовлетворительного изображения.

Тем не менее, шум также может быть использован и в других целях, таких как введение дополнительного случайного начального отклонения узлов сетки, либо для генерации текстур воды, морской пены и т.д.

1.2 Использование тригонометрических функций

Так как основной вклад в геометрию водной поверхности вносят волны, вызванные различными явлениями, то для расчётов логично использовать тригонометрическую функцию синуса. Однако у методов, основанных на данном подходе есть два существенных недостатка.

Первый недостаток заключается в необходимости подбора набора входных параметров системы, в частности – свойств визуализируемых волн. Для решения этой проблемы можно использовать различные эмпирические наблюдения, а также использовать физические законы в области теории волн.

Другой проблемой является высокая однообразность получаемой поверхности: все волны выглядят одинаково и находятся на одинаковом расстоянии друг от друга. Для уменьшения такого эффекта можно ввести начальное отклонение узлов сетки, задаваемое шумом Перлина.

1.2.1 Использование сумм синусов

Данный подход заключается в использовании сумм тригонометрических функций для моделирования волн на поверхности. Рассмотрим одну из его реализаций, описанную в статье [3]. В ней будет визуализироваться несколько волн с заданными постоянными параметрами: амплитудой, частотой волны и скоростью её изменения.

В начале вводится функция, позволяющая определять высоту каждого узла сетки поверхности в заданный момент времени. Рассмотрим случай визуализации одной волны. Для построения функции вводится промежуточное значение S. Для этого вектор координат точки скалярно умножается на вектор направления волны, после чего результат умножается на ряд коэффициентов, как показано в формуле (1).

$$S(\overrightarrow{pos}, t) = \left(\overrightarrow{pos} \cdot \overrightarrow{dir}\right) \cdot \ freq + velocity \cdot \ freq \cdot t \qquad (1)$$

Где $\overrightarrow{pos}$ – двумерный вектор координат узла сетки, $\overrightarrow{dir}$ – двумерный вектор направления распространения волны, $freq$ – частота волны, $velocity$ – скорость изменения волны, t – параметр времени.

Далее вводится функция зависимости изменения высоты узла от времени (далее – функция волны) по формуле (2). На данном этапе возможно добавить отрицательное смещение после нормализации значения синуса для получения отрицательного изменения высоты узлов.

$$f_i(\overrightarrow{pos}, t) = A \cdot \left(\frac{sin\left(S(\overrightarrow{pos}, t)\right) + 1}{2} \right)^{stepness} \quad (2)$$

Где $\overrightarrow{pos}$ – вектор координат узла сетки, t – параметр времени, $stepness$ – показатель крутизны волны, A – амплитуда волны.

Для введения дополнительных волн в модель, необходимо для каждой точки просуммировать значения функций волн по формуле (3).

$$height(\overrightarrow{pos}, t) = \sum f_i(\overrightarrow{pos}, t) \quad (3)$$

Данный метод является простым в реализации и не требует затратных вычислений для моделирования, однако качество модели, полученной данным методом, остаётся низким: волны получаются не реалистичными.

1.2.2 Волны Герстнера

Более обоснованное с физической точки зрение улучшение описанного метода представлено в работе [4]. Кроме физической обоснованности, отличие этого подхода заключается в том, что в данной модели используется также горизонтальное смещение узлов сетки.

Для получения достоверных результатов узлы сетки должны совершать круговые движения в то время, когда она находится на пути прохождения волн. Такое движение задаётся системой уравнений (4). Она позволяет вычислить новые координаты для каждой точки регулярной сетки (подразумевается, что изначально все точки имеют нулевое значение высоты).

$$\begin{cases} \vec{v} = \overrightarrow{v_0} - \dfrac{\vec{k}}{|\vec{k}|} \cdot A \cdot sin\left(\vec{k} \cdot \overrightarrow{v_0} - \omega t\right) \\ y = A \cdot cos\left(\vec{k} \cdot \overrightarrow{v_0} - \omega t\right) \end{cases} \quad (4)$$

Где $\overrightarrow{v_0}$ – двумерный вектор, содержащий начальные координаты точки поверхности в плоскости XZ, $\vec{v}$ - двумерный вектор, содержащий новые координаты точки поверхности в плоскости XZ, A – амплитуда рассматриваемой волны, $\vec{k}$ – волновой вектор длинны $|\vec{k}| = \frac{2\pi}{\lambda}$ (далее – волновое число), λ – длина волны, ω – фазовая скорость.

Значение фазовой скорости вычисляется на основе закона дисперсии по одному из следующих уравнений. Уравнение (5) применяется в случае, если расстояние до дна велико, а волны имеют длину более 1 сантиметра. Для визуализации воды в мелководье применяется уравнение (6). Для работы с волнами длинной менее 1 сантиметра применяется уравнение (7).

$$\omega(k) = \sqrt{gk} \quad (5)$$

$$\omega(k) = \sqrt{gk \cdot tanh(kD)} \quad (6)$$

$$\omega(k) = \sqrt{gk(1 + k^2 L^2)} \quad (7)$$

Где $g = 9.8\text{м}/\text{с}^2$ – ускорение свободного падения, k – волновое число, D – расстояние до дна, L – длинна волн.

При необходимости получить периодическую модель, необходимо дополнительно преобразовать полученную функцию фазовой скорости по формуле (8). Это может быть использовано для повышения качества получаемой модели, либо в целях оптимизации.

$$\omega(k) = \left\lfloor \frac{T \cdot \omega_0(k)}{2\pi} \right\rfloor \cdot \frac{2\pi}{T} \tag{8}$$

Где T – требуемый период модели, $\omega_0(k)$ – одна из определённых выше функций вычисления фазовой частоты.

Также, от значения произведения kA зависит форма волны. Когда это значение равно нулю, волна вырождается в плоскость. При плавном увеличении этого значения до единицы, получаемая волна будет заостряться к вершинам. При дальнейшем увеличении этого значения, волна начнёт образовывать петлю на вершине, что также является вырожденным случаем. Данный эффект показан для сечения волны вертикальной плоскостью на рисунке 3. Цифрами указано значение kA для каждого графика. Таким образом, это значение должно принадлежать промежутку от 0 до 1.

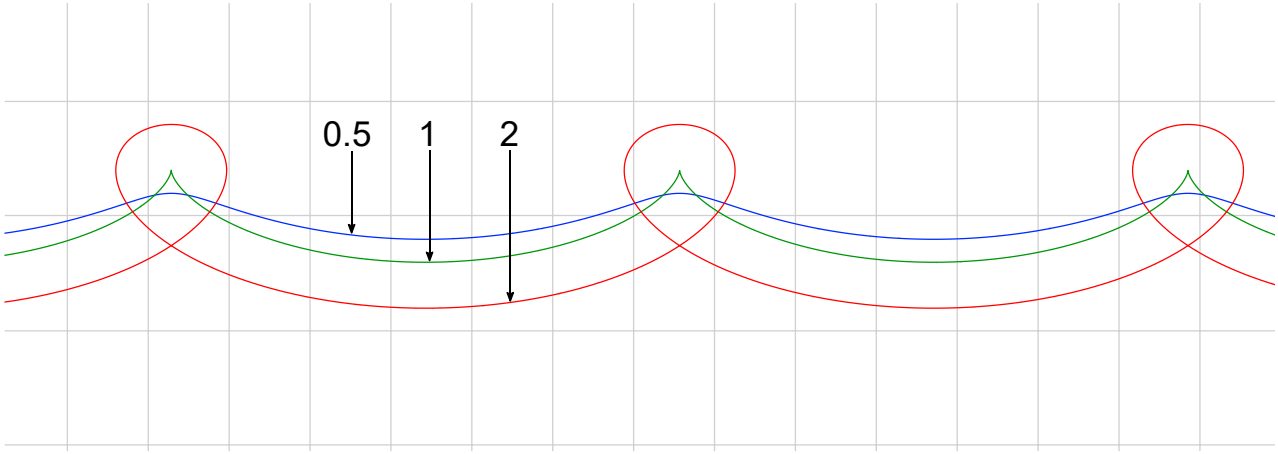

Рисунок 3: Сечения волн Герстнера

Для моделирования нескольких волн, аналогично методу суммы синусов, возможно просуммировать несколько волн. В таком случае, система будет описываться системой, показанной на формуле (9).

$$\begin{cases} \vec{v} = \overrightarrow{v_0} - \sum \frac{\overrightarrow{k_i}}{|\overrightarrow{k_i}|} \cdot A_i \cdot sin\left(\overrightarrow{k_i} \cdot \overrightarrow{v_0} - \omega_i t + \phi_i\right) \\ y = \sum A_i \cdot cos\left(\overrightarrow{k_i} \cdot \overrightarrow{v_0} - \omega_i t + \phi_i\right) \end{cases} \quad (9)$$

Где ϕ_i – фазовый сдвиг волн, остальные параметры определяются аналогично формуле (4).

1.3 Статистический метод

Данный подход также описан в [4] и заключается в разложении функции высоты узлов на сумму синусоид с комплексными, зависящими от времени амплитудами, как показано на формуле (10). После чего к полученной функции применяется обратное дискретное преобразование Фурье.

$$y = \sum_k \tilde{h}\left(\vec{k}, t\right) \cdot exp\left(i \cdot \left(\vec{k} \cdot \overrightarrow{v_0}\right)\right) \quad (10)$$

Где $\tilde{h}$ – функция, определяющая поле высот водной поверхности, $\vec{k}$ и $\overrightarrow{v_0}$ – волновой вектор и двумерные координаты узла сетки, t – параметр времени.

Важно отметить, что параметры в данной модели отличаются от аналогичным им в рассмотренных ранее методах. В частности, будет рассматриваться только одна волна, волновой вектор которой задаётся функцией, зависящей от двумерных координат узла (x, y) и вычисляющийся по формуле (11).

$$\vec{k}(x, y) = (2\pi x, 2\pi y) \quad (11)$$

Определение функции $\tilde{h}$ является ключевым шагом, влияющим на свойства и внешний вид получаемой модели. Её можно определить различными способами, некоторые из которых приведены ниже.

В статье [1] предлагается два варианта определения этой функции задаваемых формулой (12). Для варианта, предложенного Пирсоном и Московицом, функция h_0 задаётся формулой (13). Улучшение метода, предложенное Хассельманом, показано на формуле (14).

$$\tilde{h}(\vec{k},t) = h_0(\vec{k}) \cdot exp(i\omega t) \tag{12}$$

$$h_0(\vec{k}) = h_{pm}(\vec{k}) = \frac{0.0081\, g^2}{16\,\pi^4\, f^5} exp\left(-\frac{5}{4}\left(\frac{f_m}{f}\right)^4\right),\ f_m = \frac{0.13g}{U_{10}} \tag{13}$$

$$h_0(\vec{k}) = h_{pm}(\vec{k}) \cdot exp\left(ln(3.3) \cdot exp\left(\frac{-(f-f_m)^2}{2\sigma^2 f_m^2}\right)\right),\ \sigma = \begin{cases} 0.07, f \le f_m \\ 0.09, f > f_m \end{cases} \tag{14}$$

Где $\vec{k}$ – волновой вектор, ω – фазовая скорость, определённая также, как и для волн Герстнера, t – параметр времени, f – частота волны, f_m – пик частоты, U_{10} – скорость ветра на высоте 10 метров от поверхности воды.

Другой подход, описанный в [4], основан на спектре Филипса P_h, определяемого формулой (15). В данном методе функция $\tilde{h}$ определяется по формуле (16), в которой используется функция h_0 (а также комплексно сопряжённая с ней h_0^*), определённая по формуле (17).

$$P_h(\vec{k}) = A \cdot \frac{exp\left(-\left(|\vec{k}| \cdot E\right)^{-2}\right)}{|\vec{k}|^4} \cdot |cos(\phi)|^2, E = \frac{V^2}{g} \tag{15}$$

$$\tilde{h}(\vec{k},t) = h_0(\vec{k}) \cdot exp(i\omega t) + h_0^*(-\vec{k}) \cdot exp(-i\omega t) \tag{16}$$

$$h_0(\vec{k}) = \frac{1}{\sqrt{2}}(\xi_r + i\xi_i)\sqrt{P_h(\vec{k})} \tag{17}$$

Где V – модуль скорости ветра, A – амплитуда волн, ϕ – угол между волновым вектором и вектором скорости ветра, ξ_r и ξ_i – случайные числа, полученные с математическим ожиданием 0 и стандартным отклонением 1.

Для увеличения качества модели, возможно также добавить горизонтальные смещения узлов сетки, также получаемые применением обратного дискретного преобразования Фурье покомпонентно к двумерному полю, задаваемому функцией, показанной на формуле (18).

$$\vec{D} = \lambda \cdot \sum_{k} \frac{-i \cdot \vec{k}}{|\vec{k}|} \cdot \tilde{h}(\vec{k}, t) \cdot exp\left(i \cdot \left(\vec{k} \cdot \overrightarrow{v_0}\right)\right) \quad (18)$$

Где $\vec{D}$ – двумерный вектор смещения точек, λ – произвольный коэффициент, определяющий дальность смещения точек (далее будем считать его равным -1), а остальные параметры определяются аналогично формуле (10).

Таким образом, данный метод может гибко изменяться путём выбора функций $\tilde{h}$ и h_0, а также за счёт выбора используемых параметров, таких как скорость ветра над поверхностью воды.

Данный алгоритм позволяет достичь результатов лучшего качества, по сравнению с остальными алгоритмами, представленными в данной работе. Однако он наиболее затратен по вычислениям, так как требует вычисления от одного до трёх обратных дискретных преобразований Фурье над двумерным полем точек водной поверхности. В целях оптимизации целесообразно использовать для этого быстрое обратное преобразование Фурье, вычисляемое методом Кули-Тюки.

2 Освещение и текстурирование

Следующим за определением геометрии шагом является определение цвета треугольников, образованных регулярной сеткой. Для этих целей необходимо задать модель освещения.

Далее будет рассмотрена модель освещения и текстурирования океана, подразумевающая, что вокруг не будет объектов, требующих расчёт своего отражения. Однако будет учитываться отражение неба, а также дополнительные блики от солнца (являющегося основным источником освещения в сцене).

2.1 Модель освещения Блинна-Фонга

Модель Блинна-Фонга проста в реализации, в связи с чем она используется на начальном этапе визуализации. Несмотря на то, что в ней напрямую не учитываются отражения и преломления света, она позволяет получить приемлемое изображение модели как для отладки моделей геометрии, так и для использования в задачах, не требовательных к качеству изображения.

Для работы с этой моделью необходимо определить характеристики источника (или источников) света и свойства материала полигонов. Они являются константными показателями. Также необходимо определить нормали к полигонам сетки. В целях оптимизации, они вычисляются только для узлов сетки, и интерполируются на полигоны. Нормали в узлах можно вычислить аналитически (если для этого достаточно данных), либо усреднив нормали ко всем прилегающим в данному узлу полигонам.

Интерполяция нормалей на полигон соответствует тонировке Фонга, однако, в случае критической нехватки вычислительных ресурсов, допустимо использовать тонировку Гуро – при которой цвет будет вычисляться лишь для узлов, и интерполироваться на полигоны. В этом случае, очевидно, дальнейшие действия по тонировке не могут выполняться, поэтому процесс визуализации будет закончен на этом шаге.

Освещение в модели Блинна-Фонга складывается из 3 компонент – фонового, диффузного и зеркального освещения. Эта модель является улучшением модели Фонга, и отличается от неё лишь методом вычисления третьей компоненты. Каждая компонента рассчитывается с учётом соответствующих коэффициентов как со стороны освещаемой поверхности, так и со стороны источника света. Далее упоминание данных коэффициентов будет опущено, для краткости.

Для расчёта фонового освещения необходимо определить лишь факт доступности данного объекта светом. Данная компонента учитывает только ненаправленный свет. Однако, обычно, её влияние не велико, в связи с чем, её вычисление не заслуживает отдельного внимания.

Интенсивность диффузного освещения рассчитывается как скалярное произведение между вектором нормали к поверхности и вектором – направлением на источник света. Данная компонента даёт основной вклад в итоговый цвет полигона.

В модели освещения Фонга интенсивность зеркального освещения рассчитывается как скалярное произведение между вектором, направленным на точку наблюдателя (камеру), и вектором отражённого света. Однако данный подход работает некорректно, когда результат векторного произведения получается меньше нуля. Эта проблема решается в модели Блинна-Фонга. В ней интенсивность равна векторному произведению между нормалью и медианным вектором – вектором единичной длинны, находящимся посередине между направлением на камеру и на источник света.

Также, для получения менее интенсивных бликов на матовых поверхностях, значение интенсивности зеркального отражения возводится в степень заданного для материала коэффициента отражения (блеска).

Описанная модель освещения для работы требует задания параметров материала, таких цвета компонент модели: фоновый, диффузный и отражаемый, а также коэффициент блеска. Возможно также задание излучаемого цвета, однако в данной работе это не используется.

Применение модели освещения Блинна-Фонга позволяет получить приемлемый, но не достоверный результат. Несмотря на реалистичную визуализацию бликов солнца, а также затенение волн, позволяющее видеть геометрию волн, результат визуализации получается однотонным, отдалённо напоминающим пластик. Поэтому, такая модель без дополнительных улучшений является удобной на этапе отладки, однако, не подходит для итогового решения.

2.2 Текстурирование

Одним из путей исправления указанного выше недостатка является использование текстур, изменяющих начальный цвет воды. Наиболее естественно применять такие текстуры для изменения фонового и диффузного цветов поверхности воды.

Возможно два способа применения таких текстур:

- Текстура полностью замещает собой некоторые компоненты цвета в модели освещения. Таким образом текстура является единственным источником цвета воды
- Текстура влияет на установленные на предыдущем этапе компоненты цвета путём умножения, либо других математических операций

Оба этих метода не обладают существенными преимуществами друг перед другом, а также легко могут быть преобразованы. Однако во втором случае становится возможным применение текстуры обычного шума (например – рассмотренного ранее шума Перлина). Также, алгоритмы применения текстур во втором случае крайне разнообразны, что позволяет получать широкий спектр возможных результатов.

В ходе работы использовался второй метод. При запуске программы создаётся квадратная текстура шириной 256 пикселей с шумом Перлина, состоящим из 8 октав. Эта текстура накладывается на визуализируемый участок поверхности, после чего к синей компоненте диффузного освещения прибавляется значение шума, уменьшенное на 0.3, умноженное на 0.3.

2.3 Оптические эффекты

Помимо компонент освещения, используемых в модели освещения Блинна-Фонга, для реалистичной визуализации необходимо учитывать прозрачность воды и эффект отражения ей цвета неба.

При падении на поверхность воды луч света делится на две части. Первая часть отражается от поверхности. Вторая часть луча, преломившись, продолжает движение сквозь воду. Данный эффект показан на рисунке 4. На нём буквой Q обозначен отражённый, а буквой S – преломлённый лучи.

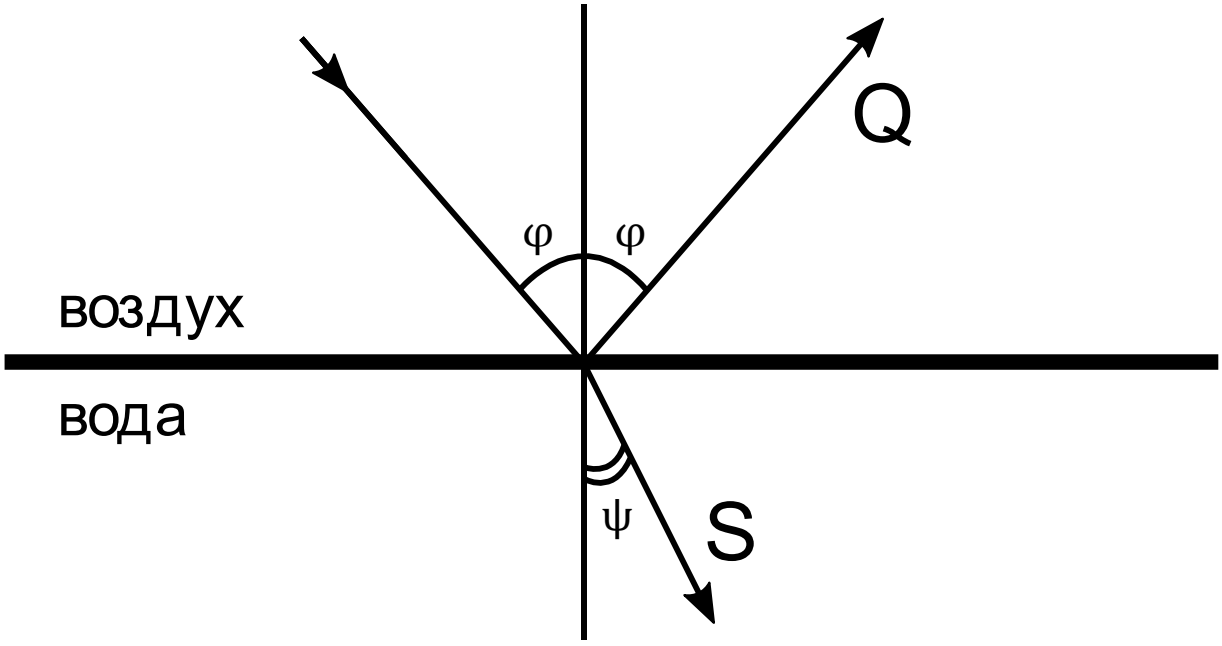

Рисунок 4: Отражение и преломление луча света

На рисунке углы ϕ – равные углы падающего и отражённого лучей. Угол ψ – угол преломлённого луча, который вычисляется по закону Снелиуса, показанному на формуле (19).

$$n_1 Sin(\phi) = n_2 Sin(\psi) \tag{19}$$

Где n_1 – показатель преломления первой среды (для воздуха он равен 1), n_2 – показатель преломления второй среды (для морской воды он равен 1.35).

Пусть R – отношение энергии отражённого света к мощности падающего. По закону сохранения энергии, аналогичное отношение для преломлённого луча при сложении с R будет давать единицу (при этом подразумевается, что поглощённая мощность мала). Согласно работе [5] значение этого коэффициента может быть вычислено по формулам Френеля по формуле (20).

$$R = \left(\frac{n_1 Cos(\phi) - n_2 Cos(\psi)}{n_1 Cos(\phi) + n_2 Cos(\psi)}\right)^2 \quad (20)$$

Полученный коэффициент показывает, какая часть падающего света будет отражаться в данной точке, поэтому он затрагивает диффузную и зеркальную компоненты освещения в модели Фонга. Для того, чтобы это учесть, нужно умножить этот коэффициент на полученные ранее коэффициенты диффузной и зеркальной освещённости.

Далее, к полученному цвету необходимо прибавить цвет окружения в направлении отражённого луча (в случае, если окружение задано, например, через skybox, иначе можно использовать цвет неба независимо от направления), также умноженный на коэффициент R. При этом цвета, заданные в модели Фонга должны быть существенно меньше, чем использованные ранее, так как именно цвет неба должен вносить основной вклад в цвет воды.

В ходе этих преобразований не была затронута компонента фонового освещения, так как оно не связано с отражением конкретных лучей. Тем не менее, оно продолжает использоваться в данной модели, и может задавать цвет неосвещённых участков (например, непосредственно под камерой).

Таким образом, за счёт применения законов Френеля, данная модель освещения является намного более приближенной к реальному освещению, нежели модель Блинна-Фонга.

3 Реализация

Для реализации системы визуализации используется язык C++ стандарта C++17 с использованием следующих библиотек:

- OpenGL – программный интерфейс для работы с графической системой компьютера
- GLFW3 – работа с окнами операционной системы
- GLEW – работа с расширениями OpenGL
- GLM – работа с векторами и матрицами
- FreeType – отображение текста

Помимо этого, в ходе работы использовались также исходный код библиотек для записи и чтения изображений из файлов `stb_image` и `stb_image_write` (они также относятся к списку библиотек).

Исходный код программы состоит из заголовочных файлов, файлов исходного кода C++, файлов шейдеров, а также дополнительных ресурсных файлов, таких как шрифты и текстуры. Все файлы проекта организованы в следующую структуру:

- include – папка с заголовочными файлами
- src – папка с файлами исходного кода
- shaders – папка с файлами шейдеров
- resources – папка с файлами ресурсов

В файле src/main.cpp находится точка входа в программу (функция `main`), а также функции инициализации графического интерфейса, задание параметров системы, главный цикл отрисовки и обработка пользовательского ввода.

Класс debugInformer отвечает за вывод текстовой отладочной информации на экран. Он определяется в двух файлах debugInformer.hpp и debugInformer.cpp. У этого класса есть метод `show`, выводящий текущее состояние, а также методы установки значений состояния (сеттеры). Выводятся следующие показатели: позиция в мире, направление взгляда, число кадров в секунду (FPS) и среднее время, затрачиваемое на отрисовку одного кадра.

Класс WaterMeshChunk находится в файлах waterMeshChunk.hpp и waterMeshChunk.cpp Он представляет собой визуализируемую модель воды и отвечает за расчёт геометрии (путём использования шейдеров) и вывод модели на экран. Этот класс содержит методы задания и чтения параметров модели (сеттеры и геттеры), метод обновления состояния модели с течением времени – `computePhysics`, а также метод вывода модели – `show`.

В программе используются шейдеры разного типа. Условно их можно разделить на шейдеры конвейера отрисовки – вершинные, геометрические и фрагментные, и вычислительные шейдеры. Первые используются для вывода графики на экран. Также они отвечают за расчёты освещения. Вычислительные шейдеры используются для расчётов геометрии.

Помимо перечисленного выше, в программе также присутствует набор вспомогательных функций и классов. Они находятся в папках include/util и src/util. Они включают в себя:

- Класс абстрактной камеры
- Класс для загрузки шрифтов и отрисовки текста
- Класс для загрузки и примитивной работы с изображениями
- Класс для работы с шейдерами
- Некоторые математические функции

3.1 Вычисление геометрии поверхности

Первым этапом создания очередного кадра является расчёт новых позиций узлов регулярной сетки. Помимо этого, на этом этапе необходимо также рассчитывать нормали к поверхности в этих узлах.

Поскольку рассматриваемые методы расчёта подразумевают выполнение однотипных, независимых друг от друга действий над большим числом точек сетки, целесообразно проводить их вычисление на видеокарте (далее – GPU). Поскольку результатом вычисления должны быть данные о координатах точек, для решения этой задачи используются вычислительные шейдеры OpenGL.

Вычислительные шейдеры позволяют проводить действия на группе данных, называемых рабочей группой (Workgroup). При этом внутри рабочей группы возможно создание общей памяти (shared memory). Несмотря на то, что методы, используемые в данной работе, не требуют использование общей памяти, в вычислительных шейдерах используются рабочие группы размера 8x8, так как при такой конфигурации достигается наилучшая эффективность работы.

На данном этапе расчёты выполняются по следующей схеме:

1. Устанавливаются параметры шейдеров (время, свойства волн и т.д.), путём задания uniform переменных шейдеров и заполнением входных буферов
2. Вызывается шейдер или группа шейдеров расчёта геометрии, в результате чего в вершинном буфере должны оказаться координаты точек
3. Вызывается шейдер расчёта нормалей, который на основе данных вершинного буфера формирует карту нормалей

В ходе работы были рассмотрены два метода расчётов – основанный на использовании волн Герстнера и статистический.

Входными данными для первого метода является набор волн произвольного размера и их свойства. Для того, чтобы передать их в шейдер используется объект буфера типа shader storage. Он создаётся и заполняется данными при инициализации программы. Так как значения смещений узлов для этого метода явным образом выражаются одной формулой, геометрия рассчитываются за один вызов вычислительного шейдера. В нём в цикле вычисляются значения волновых функций, после чего их значения складываются, а сумма записывается в вершинный буфер.

Для повышения реалистичности этого метода, к нему было добавлено дополнительное смещение высоты узлов, получаемое из шума Перлина. Для этого при запуске программы генерируется текстура с шумом, после чего она передаётся в шейдер, где полученные значения шума прибавляются к результату, будучи умноженными на коэффициент.

В статистическом методе используется быстрое обратное дискретное преобразование Фурье методом Кули-Тюки (далее – преобразование Фурье), в связи с чем вычисления для него выполняются группой шейдеров. Метод оптимизации такого вычисления описаны в работе [6].

Для проведения такого преобразования в начале необходимо определить следующую операцию, далее именуемую операцией бабочки. Она принимает на вход два комплексных числа и комплексные коэффициенты, и возвращает два новых комплексных числа. Эта операция названа бабочкой в связи с видом описывающий её диаграммы, которая показана на рисунке 5.

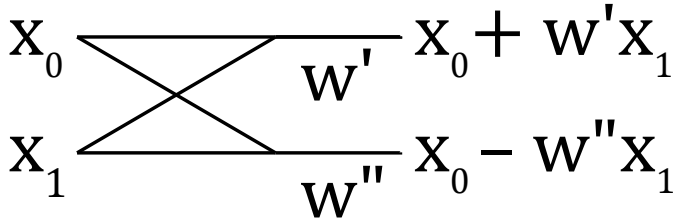

Рисунок 5: Операция бабочки

Пусть N – ширина и высота сетки. Для получения наилучших результатов, предполагается, что сетка квадратная, а N является степенью двойки (иначе нужно дополнить её нулями). Для того, чтобы применить преобразование, операцию бабочки необходимо провести $log_2 N$ сперва для каждой строки, а потом для каждого столбца. На каждом из этапов преобразования коэффициент w определяется по формуле (21).

$$w_N^k = \left(exp\left(-\frac{2\pi i}{N}\right)\right)^2, k = n\frac{N}{2^s} \tag{21}$$

Где N – число элементов преобразования (длина строки или столбца), n – индекс элемента в строке или столбце, s – текущий этап преобразования.

Исходя из свойств функции exp видно, что к коэффициенту k может быть применена операция остатка от деления на N.

Также на каждом этапе для каждого элемента необходимо определить индексы элементов, над которыми должна проводиться операция. На первом этапе их индексами является результат применения к ряду чисел от 0 до N операции обращения битов. На каждом следующем этапы индексы получаются по следующим правилам:

- Элемент является верхней частью крыла, если $(n \ mod \ 2^{s+1}) < 2^s$
- В верхней части крыла индексами являются числа n и $n + 2^s$
- В нижней части крыла индексами являются числа $n - 2^s$ и n

На рисунке 6 приведён пример преобразования для строки размера 4. На рисунке подразумевается, что каждый коэффициент w имеет нижний индекс 4.

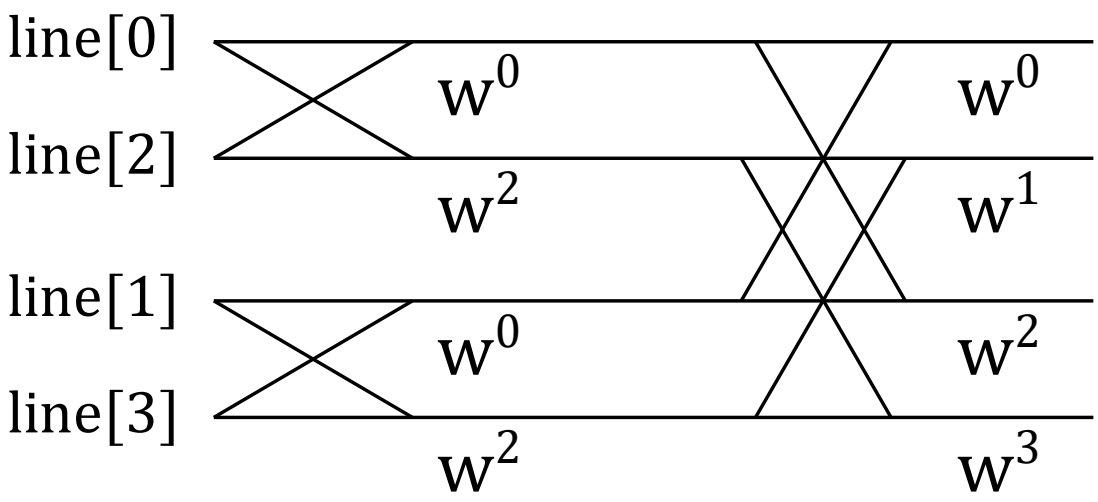

Рисунок 6: Два этапа применения операции бабочки

На последнем этапе необходимо умножить значения вещественной части результата на $\pm N^{-2}$ (отрицательное число для узлов с нечётной суммой координат на сетке). После этого преобразование Фурье будет завершено.

Для ускорения работы преобразования, при старте программы генерируется текстура, содержащая информацию для проведения операций бабочки на каждом этапе. В красной и зелёной компоненте содержатся коэффициенты для операции бабочки, а в синей и альфа компонентах – индексы элементов. Текстура имеет размеры N на $log_2 N$ пикселей.

Помимо этого, на начальном вычисляется и сохраняется текстура, со значениями функции $h_0(\vec{k})$ и $h_0^*(-\vec{k})$. Результатами этой функции являются комплексные числа, поэтому их вещественные части записываются в красную и синюю компоненты пикселя, а мнимые – в зелёную и альфа компоненты соответственно. Вычисления проводятся без использования GPU.

Для каждого кадра вычисляется текстура со значениями функции $\tilde{h}(\vec{k}, t)$. В результате этого формируется три текстуры, содержащие соответствующие значения для смещения вдоль каждой оси. Это выполняется в вычислительном шейдере ht.comp.

Далее выполняется преобразование Фурье для каждой из трёх полученных текстур. Сперва применяется шейдер бабочки $log_2 N$ раз в горизонтальном и вертикальном направлениях. Для проведения одного этапа операции бабочки используется шейдер butt.comp. Он также использует одну дополнительную текстуру для сохранения результата. После чего применяется финальный шейдер преобразования fourier.comp, который принимает три обработанные текстуры и формирует значения для вершинного буфера.

Последним этапом обоих методов является расчёт нормалей. Он выполняется за один проход соответствующего вычислительного шейдера norm.comp. Для каждого пикселя вычисляются направления на соседние пиксели. Выполняя векторное произведение соседних векторов, находятся нормали соседних с рассматриваемой вершиной граней. Полученные нормали нормируются, после чего результатом является их среднее значение.

3.2 Тонировка

В работе было рассмотрено два подхода к тонировке. В первом подходе используется модель освещения Блинна-Фонга с применением дополнительной текстуры с шумом Перлина. Второй основан на эффекте Френеля и использует физические свойства света для формирования цвета, в основном, на основе отражаемого неба.

Для вывода полигонов сетки на экран используется связка вершинного и фрагментного шейдеров (на этапе отладки иногда также использовался геометрический шейдер).

Вершинный шейдер находится в файле water.vert, и выполняет следующие действия: применяет матрицу преобразования к координатам для задания координат узла сетки с учётом положения и направления камеры, пробрасывает глобальные координаты узла во фрагментный шейдер и вычисляет текстурные координаты узла для последующего использования текстур.

Фрагментный шейдер находится в файле water.frag и рассчитывает цвет каждого пикселя полигона исходя из модели тонировки.

Для реализации модели Блинна-Фонга во фрагментном шейдере вычисляются: вектор нормали в данной точке, вектор направления на точку обзора (камеру) и медианный вектор. Также вычисляется интенсивность освещённости данной точки (скалярное произведение вектора направления на солнце и нормали) и значение шума из текстуры шума Перлина. После чего вычисляются интенсивности компонент освещения, которые складываются в итоговый цвет в данной точке.

Вторая модель, основанная на эффекте Френеля, также реализуется во фрагментном шейдере. В начале, так же, как и в предыдущей модели, вычисляются значения интенсивности фоновой, диффузной и зеркальной компонент освещения. Дальше находится направления и углы отражённого и преломлённого лучей, после чего вычисляется коэффициент мощности отражённого луча. Далее интенсивность диффузной и зеркальной компонент умножаются на полученный коэффициент, после чего цвета компонент суммируются с учётом полученных коэффициентов. Последним этапом к результату прибавляется умноженный на коэффициент цвет неба в направлении отражённого луча.

4. Результаты

В ходе работы последовательно реализовывались различные методы моделирования геометрии поверхности и её тонировки. Далее будут рассмотрены и сравнены промежуточные результаты работы.

Все модели геометрии, кроме последней представлены с использованием тонировки Блинна-Фонга. Для некоторых моделей также использовалась текстура с шумом Перлина, прибавляемая к цвету некоторых компонент.

Также большинство моделей требуют задания большого числа параметров (например, набора волн и их параметров), которые не будут указываться далее. При этом будет показан наилучший из достигнутых результатов для каждой модели, полученный путём наблюдения за моделью при задании различных входных параметрах.

Первым реализованным методом был метод суммы синусов. Несмотря на высокую производительность, эта модель является не реалистичной за счёт часто повторяющейся и однообразной структуры волн. Эта модель показана на рисунке 7.

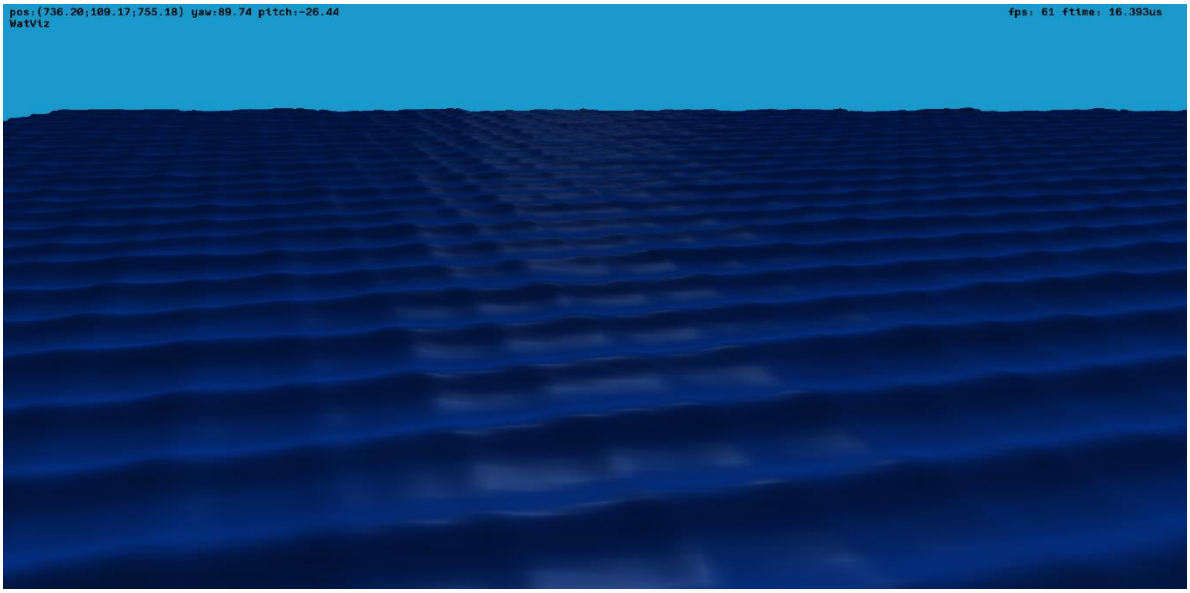

Рисунок 7: Модель суммы тригонометрических функций

Модель, показанная на рисунке 8, использует волны Герстнера. Она является усовершенствованием предыдущей модели, и показывает лучшие результаты, однако всё ещё состоит из повторяющихся волн.

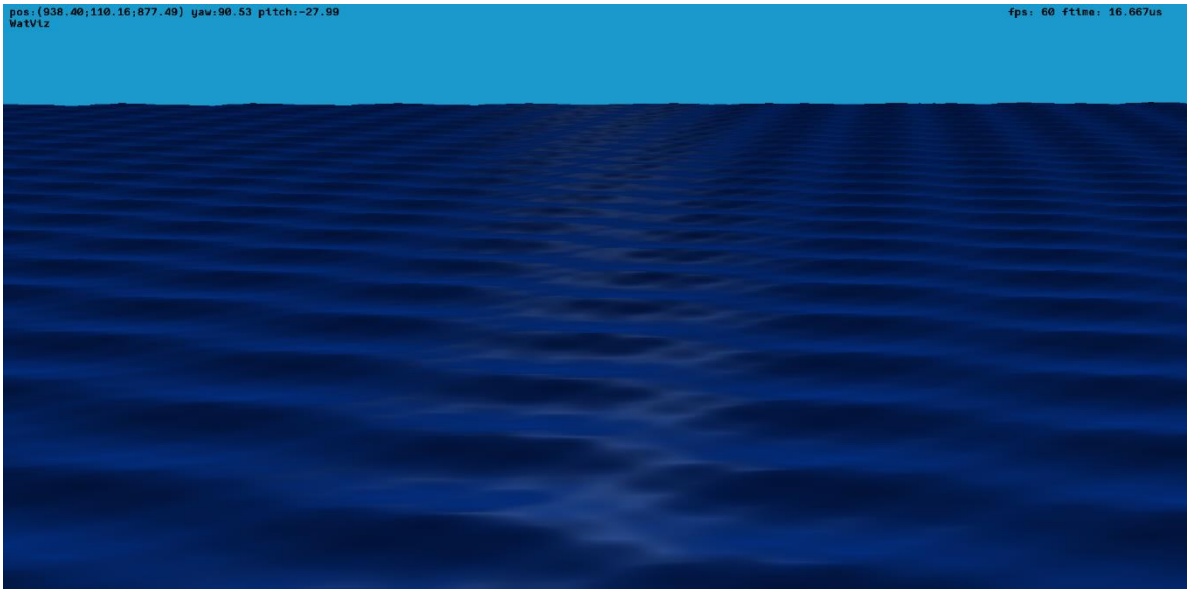

Рисунок 8: Волны Герстнера

Последним этапом усовершенствования этих моделей является добавление начального смещения высоты узлов, получаемого из текстуры с шумом Перлина. Результат показан на рисунке 9. На нём также к цвету воды добавлен шум.

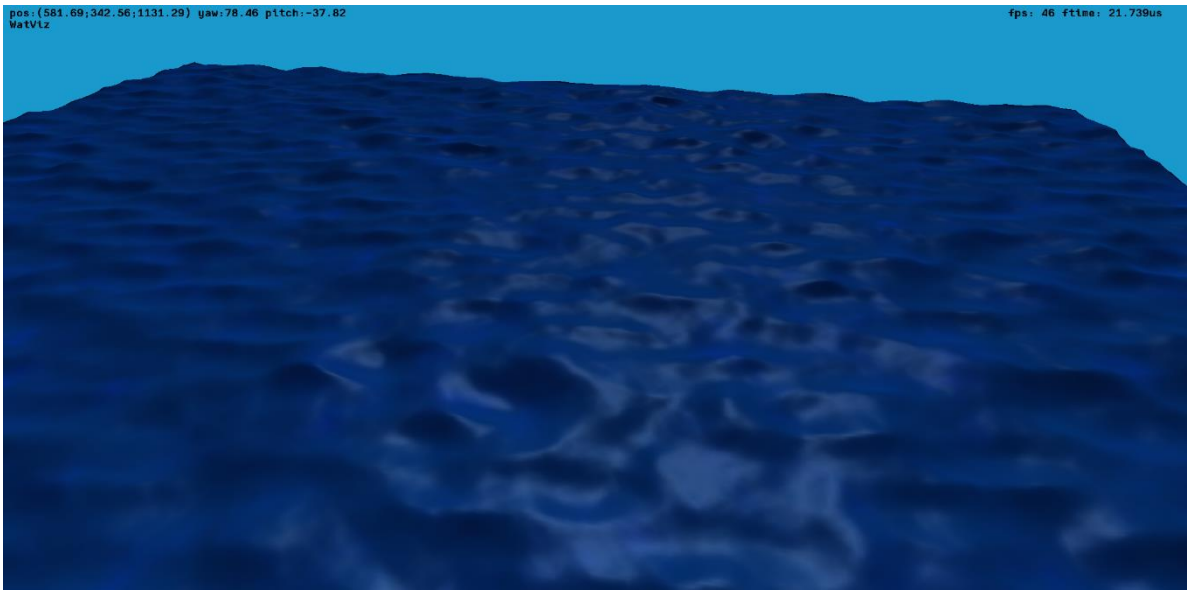

Рисунок 9: Сложение волн Герстнера с шумом Перлина

Далее был реализован статистический метод – основной метод, рассматриваемый в данной работе. Для него было реализовано две модели геометрии. Первая показана на рисунке 10.

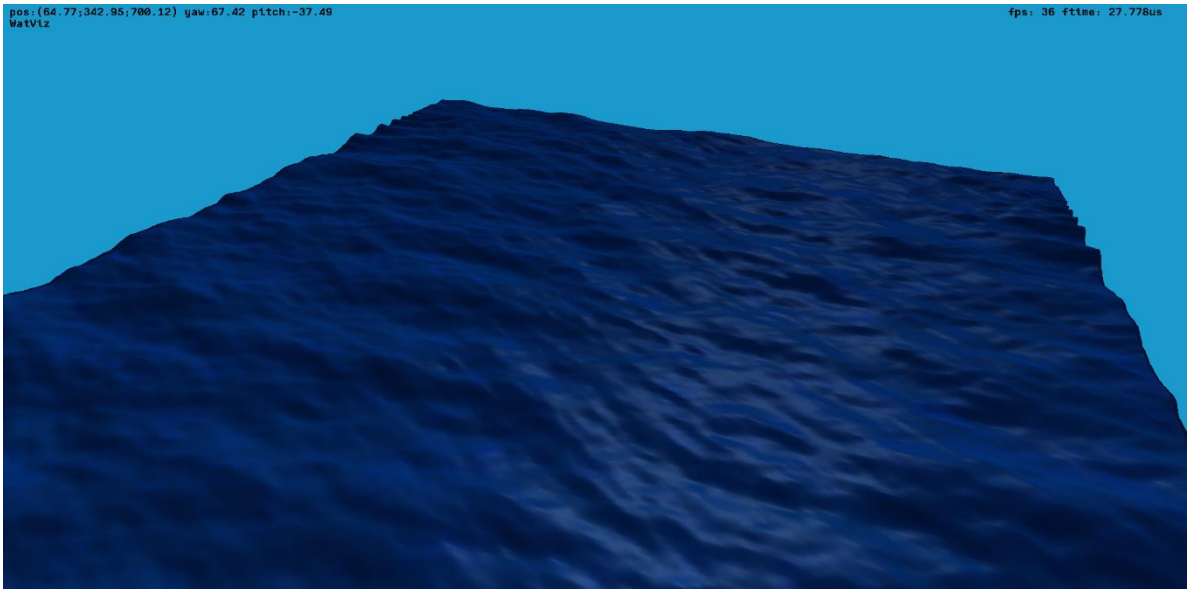

Рисунок 10: Статистическая модель без горизонтальных смещений

Затем, для повышения реалистичности, к модели были добавлены горизонтальные смещения узлов. Полученная модель показана на рисунке 11 и является наилучшей моделью, среди рассмотренных в этой работе.

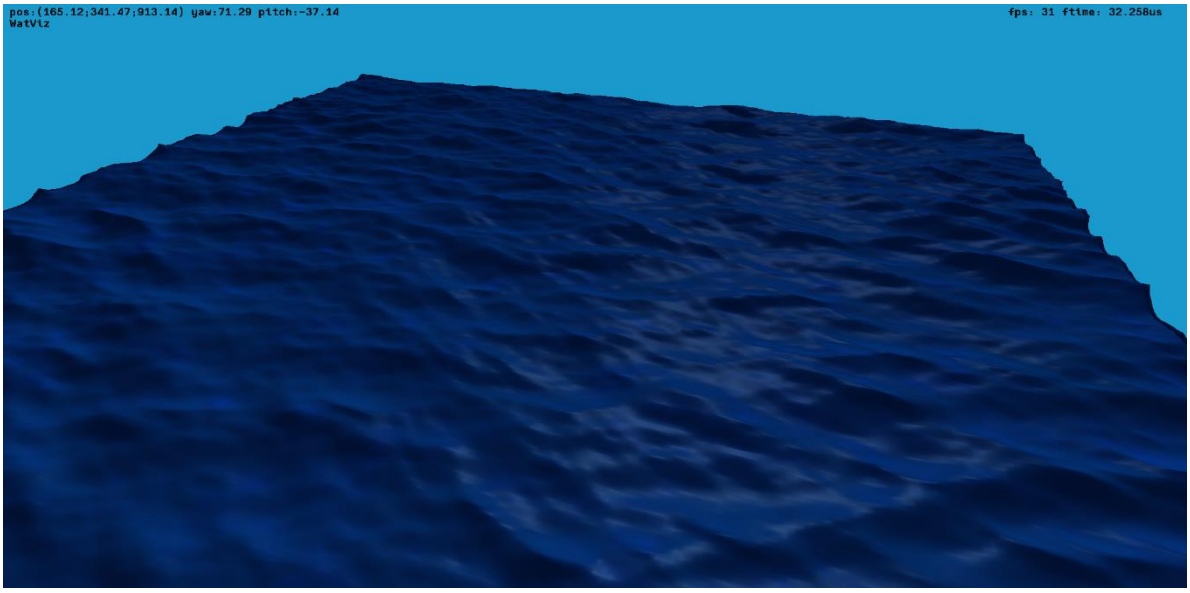

Рисунок 11: Статистическая модель с горизонтальными смещениями

Следующим этапом работы является улучшение модели освещения. При этом были изменены цвета неба и материалов, для соответствия новой системе расчётов. На рисунке 12 показан результат применения новой модели.

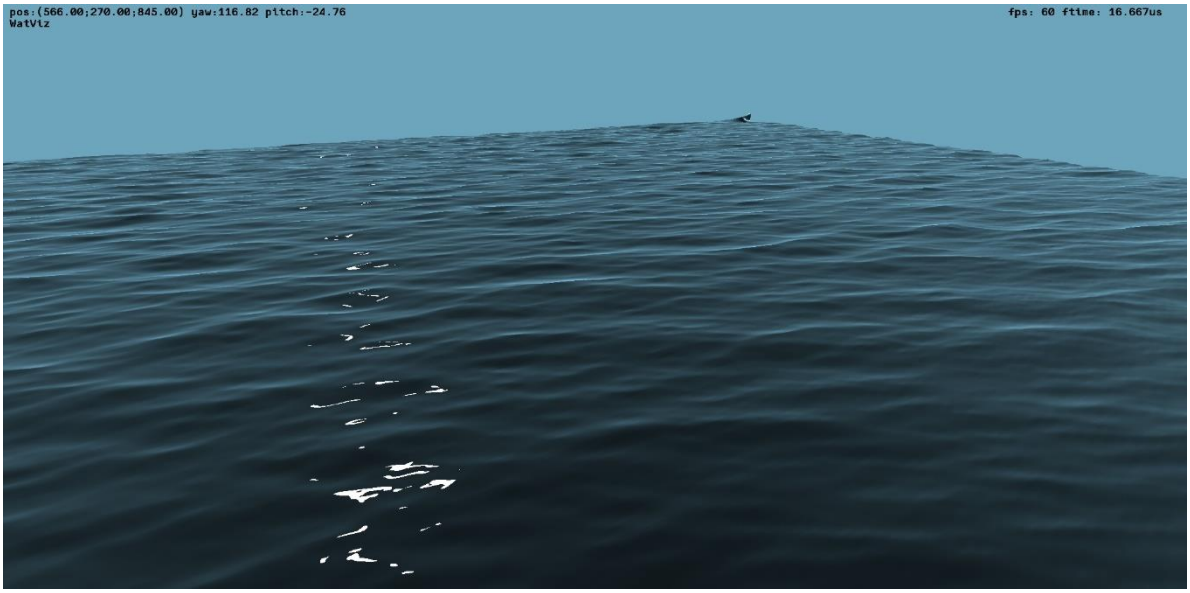

Рисунок 12: Визуализация с эффектом Френзеля

Для достижения лучшего результата, к этой модели дополнительно были добавлены некоторые коэффициенты, в результате чего итоговая модель стала выглядеть так, как показано на рисунке 13.

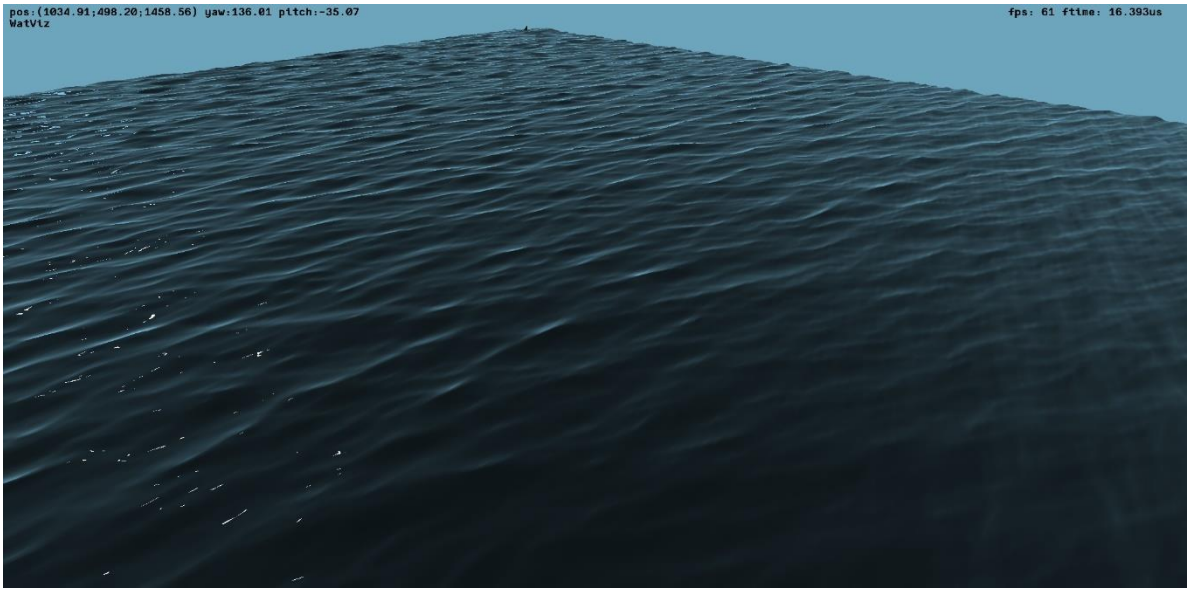

Рисунок 13: Итоговая модель визуализации

ЗАКЛЮЧЕНИЕ

В ходе работы была получена реалистичная модель визуализации поверхности океана, основанная на использовании статистических данных и обратного дискретного преобразования Фурье. Была проведена оптимизация работы данной модели с выполнением вычислений на видеокарте.

Для реализации программы использовались такие технологии параллельных вычислений на видеокарте, как обычные и вычислительные шейдеры OpenGL. Была изучена реализация быстрого обратного дискретного преобразования Фурье методом Кули-Тюки на вычислительных шейдерах.

Также, в процессе создания итоговой модели были созданы промежуточные модели визуализации, которые также могут быть использованы для решения некоторых задач, связанных с визуализацией океана.

Помимо простой визуализации созданная модель может быть адаптирована, например, для расчётов физики плавающих тел, вычисления преломления лучей для визуализации подводных объектов и визуализации каустики на них.

Дальнейшими этапами развития созданной программы является повышение качества визуализации, путём реализации таких эффектов, как визуализация пены и пузырьков воздуха, подповерхностное рассеянное освещение. Также может быть добавлена поддержка множественных и объёмных источников освещения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Куров А.В., Завалин А.А. Визуализация водной поверхности на основе спектрального метода // Символ науки. 2017. Том №01-2. С. 71-77.
2. Мальковский А. Визуализация природных явлений // Введение в компьютерную графику. URL: https://www.graphicon.ru/oldgr/courses/cg02b/assigns/hw-5/hw5_wat.htm (Дата обращения 04.10.2020).
3. Adam T Lake Real-Time Deep Ocean Simulation on Multi-Threaded Architectures. URL: https://software.intel.com/content/www/us/en/develop/articles/real-time-deep-ocean-simulation-on-multi-threaded-architectures.html (Дата обращения: 04.10.2020).
4. Tessendorf J. Simulating ocean waters. ACM SIGGRAPH Computer Graphics. 2001.
5. Thorvald C. Grindstad, Runar J. F. Rasmussen Deep water ocean surface modelling with ship simulation // Norges teknisk-naturvitenskapelige universitet, Fakultet for ingeniørvitenskap og teknologi, Institutt for marin teknikk. 2011.
6. Fynn-Jorin Flügge Realtime GPGPU FFT ocean water simulation // Institute of Embedded Systems. 2017.