Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

МОДЕЛИРОВАНИЕ ПОВЕРХНОСТИ ВОДЫ С ИСПОЛЬЗОВАНИЕМ БЫСТРОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Лелякина Михаила Александровича

Научный руководитель
к. ф.-м. н. С. В. Миронов

Заведующий кафедрой
к. ф.-м. н. С. В. Миронов

Саратов 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настоящая работа посвящена важному направлению в современной индустрии компьютерных игр и кинематографе — проблеме моделирования поверхности воды. На внешний вид поверхности океана или реки влияет множество условий окружающей среды, таких как скорость ветра, скорость течения жидкости, различные источники освещения и окружающие объекты. Для создания реалистичного изображения высокого качества при моделировании необходимо учитывать все перечисленные факторы.

Целью практической части работы является создание фотореалистичной модели поверхности воды в библиотеке DirectX 11 на языке Visual C++. Для быстрого рендеринга поверхности должно быть использован алгоритм быстрого преобразования Фурье. Модель должна включать дополнительные элементы, присутствующие в реальной среде: модель освещения солнечным светом, фотореалистичный горизонт и отражение цветов неба на поверхности воды. Для решения поставленной задачи необходимо:

— Рассмотреть основные алгоритмы моделирования поверхности воды в компьютерной графике
— Реализовать модель поверхности с помощью графической библиотеки DirectX11
— Реализовать наиболее естественную модель освещения водной поверхности
— Реализовать возможность интерактивного взаимодействия с пользователем

1 Методы моделирования водной поверхности

Существует две группы алгоритмов, используемых в современных компьютерных графических системах. Алгоритмы первой группы используют разложение волн, возникающих на поверхности океана в сумму синусоид, с характеристиками, близкими к реальной среде. Алгоритмы второй группы также используют разложение колебаний волны на сумму волн, но при этом характеристика каждой такой волны является случайной величиной.

1.1 Необходимые математические обозначения

Оба рассматриваемых алгоритма из каждой группы имеют похожий математический аппарат, ввиду этого введём общие обозначения.

Будем рассматривать дискретную сетку размером N на M точек. Здесь и далее: L_x, L_z — длина и ширина реального участка водной поверхности, моделируемой дискретной сеткой. λ — длина волны, g — ускорение свободного падения, взятое в данной работе равным $9.8m/s^2$, t — постоянно увеличивающееся время, A — амплитуда волны.

1.2 Существующие алгоритмы моделирования водной поверхности

1.2.1 Аппроксимация суммой синусов

В самом простом случае приближения физической модели можно попытаться задать аналитическую функцию $h(\vec{x})$, которая будет моделировать высоту конкретной точки $\vec{x}$ дискретной сетки. Простой алгоритм аппроксимации суммой синусов подробно описан в статье «Моделирование поверхности воды в режиме реального времени на платформах с многоядерной архитектурой» [?].

Для одной периодической волны с направлением $\vec{k}$, высота в точке $\vec{x}$ равна:

$$h = \sin(\vec{k}\vec{x}) \tag{1}$$

Ввести зависимость от времени можно, добавив постоянное смещение по фазе для дискретного момента времени t:

$$h = \sin(\vec{k}\vec{x} + \phi t) \tag{2}$$

Введём понятие амплитуды волны, как максимального расстояния «подъёма»

точки дискретной сетки. Тогда уравнение примет вид:

$$h = A\sin(\vec{k}\vec{x} + \phi t) \tag{3}$$

В случае n волн достаточно просуммировать высоту, вычисляемую от каждой такой волны:

$$h = \sum_{i=1}^{N} A_i \sin(\vec{k_i}\vec{x} + \phi t) \tag{4}$$

Алгоритм аппроксимации суммой синусов является самым простым из описанных в данной работе. Данный способ даёт очень низкое качество рендеринга. Возможен случай, когда на точки дискретной сетки выпадут средние значения высоты волны. Тогда вся поверхность будет вырождена в плоскость. Эту проблему можно решить переходом к более качественным алгоритмам, либо увеличению размера дискретной сетки.

1.2.2 Волны Герстнера

Метод волн Герстнера описывает наиболее приближенную по физическим характеристикам модель волн, возникающих на водной поверхности.

Волна Герстнера является точным решением уравнения Эйлера течения несжимаемой жидкости, описывающим перемещение периодичной гравитационной волны на водной поверхности. В источнике [**?**] упоминается, что Волны Герстнера были независимо описаны Франтишеком Йозефом Герстнером в 1802 году и Уильямом Джоном Макуорном Ранкином в 1863 году. Однако применение в компьютерной графике этот способ получил сравнительно недавно.

В методе волн Герстнера, используемом в компьютерной графике используется карта высот для дискретной сетки и на каждой итерации рендеринга для одной волны пересчитываются все три координаты для каждой точки поверхности согласно формулам:

$$\vec{x} = \vec{x_0} - \frac{\vec{k}}{k} A\sin(\vec{k}\cdot\vec{x_0} - \omega t), \tag{5}$$

$$y = A\cos(\vec{k}\cdot\vec{x_0} - \omega t) \tag{6}$$

Начальное положение точки на моделируемом участке — $\vec{x_0}$, а величина k

обратно пропорциональна длине волны λ согласно формуле:

$$k = \frac{2\pi}{\lambda} \tag{7}$$

Векторная величина $\vec{k}$, называемая также «волновым вектором» имеет размерность R^2 и указывает направление движения воды на плоскости, параллельной плоскости дискретной сетки. Частота ω зависит от характеристики глубины моделируемой поверхности. При больших размерах глубины она выражается формулой параметра k, описанной в формуле выше:

$$\omega^2(k) = gk \tag{8}$$

Однако для глубины D, сравнимой с длиной волны, частота имеет вид:

$$\omega^2(k) = gk\,\mathrm{th}(kD) \tag{9}$$

Из уравнения видно, что частота движения волны затухает при глубине D близкой к нулю, а при глубине D намного большей λ величина $\mathrm{th}(kD)$ в предельном переходе обращается в единицу, и уравнение (9) принимает форму уравнения (8).

С первого взгляда кажется, что волны Герстнера — небольшая модификация простой аппроксимации суммой синусов, однако ввиду того, что для волны Герстнера пересчитываются сразу все три координаты, такая волна «сгущает» точки к наиболее острым пикам поверхности, давая высокую точность триангуляции возвышенностей. Сравнение методов простой аппроксимацией суммы синусов и волн Герстнера приводится в статье «Around GPU Gems. Эффективное моделирование воды на основе физических моделей» [**?**].

Уравнения (5), (6) не дают реалистичной картины, так как моделируют лишь одну периодичную волну. Однако набором волновых векторов $\vec{k_i}$, амплитуд A_i, частот ω_i и фаз ϕ_i для n волн $i = 1, 2, ..., N-1, N$ можно задать поверхность, просуммировав перемещения точки, оказанной каждой из N волн по следующим формулам:

$$\vec{x} = x_0 - \sum_{i=1}^{N} \frac{\vec{k_i}}{k_i} A_i \sin(\vec{k_i} \cdot \vec{x_0} - \omega_i t + \phi_i), \tag{10}$$

$$y = \sum_{i=1}^{N} A_i \cos(\vec{k_i} \cdot \vec{x_0} - \omega_i t + \phi_i) \tag{11}$$

С вычислительной точки зрения затратно в бесконечном цикле проводить вычисления величины ω, поэтому используется следующий подход: пусть форма поверхности будет повторяться через интервал времени T. Введём величину ω_0, согласно формуле:

$$\omega_0 = \frac{2\pi}{T} \tag{12}$$

Вместо величины $\omega(k)$ будем пользоваться предварительно вычисленной величиной $\bar{\omega}(k)$, задаваемой формулой:

$$\bar{\omega}(k) = \left\lfloor \frac{\omega(k)}{\omega_0} \right\rfloor \omega_0 \tag{13}$$

1.2.3 Статистический алгоритм

Идея использования случайных чисел для генерирования поверхности воды была впервые описана Джерри Тессендорфом в статье [**?**]. Статистический метод основан на чисто эмпирических наблюдениях за поверхностью океана. Суть метода состоит в суммировании $N \times M$ случайных волн по следующей формуле:

$$h(\vec{x}, t) = \sum_{\vec{k}} \tilde{h}(\vec{k}, t) \exp(i\vec{k} \cdot \vec{x}) \tag{14}$$

Здесь $\vec{k}$ — двумерный вектор, с координатами: $\vec{k} = (k_x, k_z)$, $k_x = 2\pi n / L_x$, $k_z = 2\pi m / L_z$, где числа n и m принадлежат интервалам $-N/2 \leq n < N/2$ и $-M/2 \leq m < M/2$ соответственно.

Величина $\tilde{h}(\vec{k}, t)$ вычисляется через карту высот в момент времени $t = 0$. Карта высот в начальный момент времени имеет вид h_0:

$$h_0\vec{(k)} = \frac{1}{\sqrt{2}}(\xi_r + i\xi_i)\sqrt{P_h(\vec{k})} \tag{15}$$

Здесь величины ξ_r и ξ_i распределены по нормальному закону с математическим ожиданием, равным нулю, и единичным среднеквадратичным отклонением.

В указанной формуле величина $P_h(\vec{k})$ — спектр Филлипса, который име-

ет вид:

$$P_h(\vec{k}) = A\frac{\exp(-1/(kL)^2)}{k^4}|\vec{k}\vec{\omega}|^2 \tag{16}$$

Спектр Филлипса задаёт начальную структуру водной поверхности. При различных задачах дизайна можно изменять вид этой формулы на практике, домножая одну или несколько составляющих формулы на эмпирически подобранные коэффеценты.

Один раз вычислив начальную карту высот, можно в момент времени t вычислить величину $\tilde{h}(\vec{k}, t)$ по формуле:

$$\tilde{h}(\vec{k}, t) = \tilde{h_0}(\vec{k})\exp(i\omega(k)t) + \tilde{h_0^*}(-\vec{k})\exp(-i\omega(k)t) \tag{17}$$

Пусть $N = M = 512$. Тогда при вычислении высоты точки методом «в лоб», потребуется произвести NM вычислений для нахождения величины $\tilde{h}(\vec{k}, t)$ для NM волн. Также на поверхности у нас находится NM вершин, для которых необходимо вычислять величину $\tilde{h}(\vec{k}, t)$ NM раз. Общая сложность вычислений для сетки равна $512^4 = 2^{36}$.

2 Реализация приложения

Было разработано оконное Windows приложение с двумя кнопками — для перехода в режим полного экрана и для остановки процесса непрерывного течения воды. Пользователь может перемещаться по сцене нажатием стрелочных клавиш, либо набором кнопок «W», «S», «A», «D» на клавиатуре. Также возможно управлять вращением камеры с помощью мыши. Примеры запусков приложения представлены на рисунках 1 и 2.

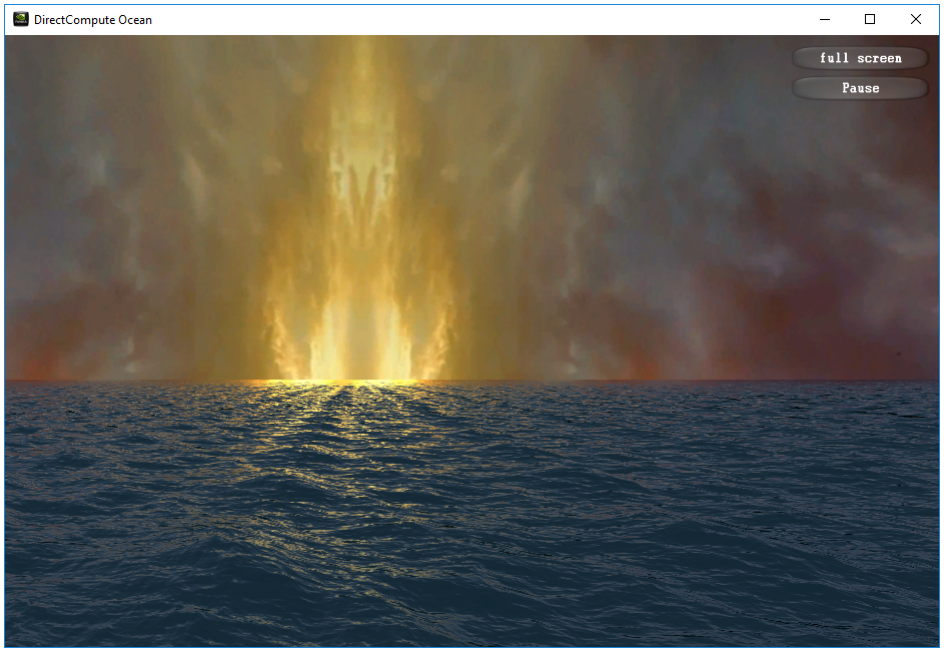

Рисунок 1 – Результат запуска приложения

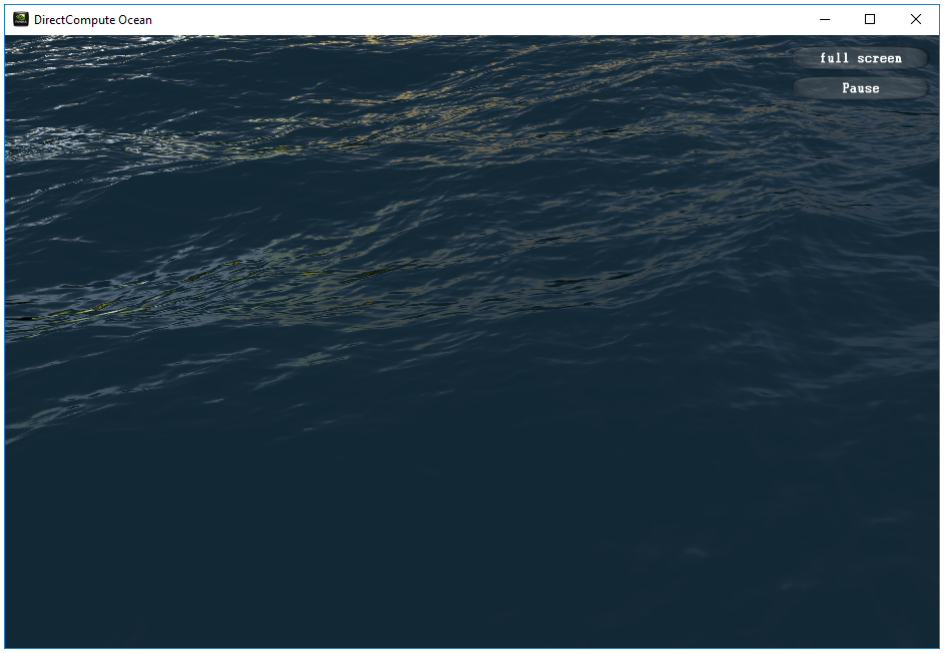

Рисунок 2 – Результат запуска приложения. Крупный масштаб.

В процессе работы приложение в режиме реального времени моделирует водную поверхность океана. На поверхности учитывается источник света — Солнце и отражение цветов неба (скайбокса).

2.1 Используемые инструменты и технологии

Приложение реализовано в среде Visual Studio 2008 под графическую платформу DirectX, которая согласно статье [**?**] является стандартом разработки графических приложений под семейство операционных систем Windows. Были использованы комплект разработчика DirectX SDK и графический фреймворк компании Microsoft — DXUT. Разработка велась под операционной системой Windows 10. Для компиляции приложения необходимо установить набор библиотек DirectX SDK, согласно инструкции, приведённой в [**?**]. При разработке использовалась версия DirectX SDK 11 (2010 June). Для версионирования изменений файлов была использована система контроля версий Git 2.12.0.

Для увеличения скорости разработки был использован графический фреймворк с открытым исходным кодом — DXUT. С помощью данного инструмента возможно избежать работы с Win32 API для создания окна, управляющих элементов и отлавливания событий пользователя и переложить данную задачу на фреймворк. Процесс подключения фреймворка описан в официальной документации Microsoft в [**?**].

2.2 Реализация модели поверхности

В приложении реализован статистический алгоритм с использованием быстрого преобразования Фурье. В качестве моделей освещения использованы модели равномерного освещения, бликовая модель освещения Фонга и диффузная модель. Для большей реалистичности написан алгоритм рендеринга уровня яркости в зависимости от угла нормали согласно уравнению Френеля в текстурную карту.

Для реализации статистического алгоритма необходимо:

1. Реализовать рендеринг карты высот в момент времени $t = 0$ с помощью вычисления величины h_0.
2. Описать алгоритм вычисления величины, описанной в уравнении (17).
3. Описать процедуру быстрого преобразования Фурье для обновления карты высот в режиме реального времени.

Для моделирования поверхности океана в приложении создан специальный класс `Ocean Simulator` со структурой, изображённой на рисунке 3:

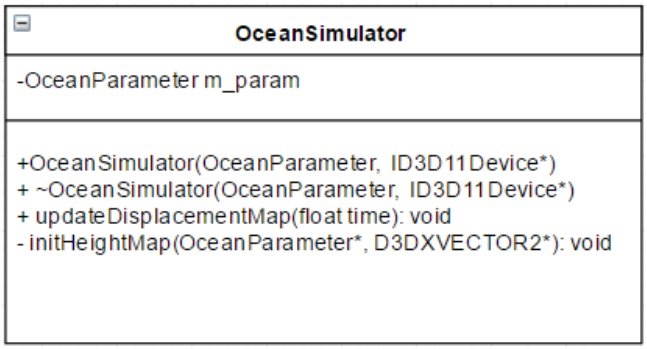

Рисунок 3 – UML представление класса «Ocean Simulator»

Все характеристики поверхности описаны в отдельной структуре `Ocean Parameter`, имеющей следующие составляющие:

```
struct OceanParameter
{
 // Размер дискретной сетки
 int dmap_dim;
 // Длина участка поверхности
 float patch_length;
 // Множитель для ускорениязамедления/ течения
 float time_scale;
 // Множитель для увеличение амплитуды волн
 float wave_amplitude;
 // Направление ветра
 D3DXVECTOR2 wind_dir;
 // Скорость ветра
 float wind_speed;
 float wind_dependency;
};
```

В конструкторе класса `OceanSimulator` производится начальное вычисление величины $h_0(\vec{k})$ из формулы (15) с помощью метода `initHeightMap` и вызова метода `updateDisplacementMap` для задания карты высот в момент времени $t = 0$.

Для простоты реализации алгоритма была взята дискретная сетка с равными размерами по длине и ширине. То есть: $N = M, L_x = L_y$. В приведённом выше коде размер дискретной сетки содержится в переменной `height_map_dim`, а реальный размер поверхности в переменной `patch_length`. Так как h_0 — комплексная величина, результат вычисления приходится хранить в массиве двумерных векторов `_h0` для запоминания как действитель-

ной, так и мнимой части каждой величины h_0. Действительная и мнимая часть величины h_0 вычисляются домножением значения Филлипса на случайные числа `Gauss()` в строчках кода 17-20.

Функция `Phillips` также определена в файле «ocean_simulator.cpp» и работает согласно определению формулы (16). Её код полностью реализует вычисления формулы (16).

При вычислении величины $h_0(\vec{k})$ используется функция `Gauss`. Так как стандартная библиотека языка C++ не содержит генератора случайных чисел, распределённых по нормальному закону, используется метод полярных координат для получения нормально рспределённых случайных чисел.

Вычисление величины из формулы (17) производится в вызове метода `updateDisplacementMap` с помощью функции `UpdateSpectrum`, описанной в шейдере «ocean_simulator.hlsl».

В функции для каждой волны $\vec{k} = (k_x, k_y)$ берётся значение `g_inputH0`, вычисленное в конструкторе класса `OceanSimulator`. Далее для волны находится сопряжённая ей волна $\vec{k^*} = (N - k_x, M - k_y)$ с помощью индекса, описанного в строчке [3]. Находится второе соответствующее ей значение из массива `g_inputH0`. Далее по требованию формулы (17) оба значения требуется умножить на величины $\exp(i\omega(k)t)$ и $\exp(-i\omega(k)t)$ соответственно. Так как умножение комплексного числа h_0 на экспоненту, равнозначно повороту вектора данного комплексного числа на аргумент экспоненты, в строке 11 и 24 вычисляются синус и косинус аргумента, с помощью функции языка HLSL — sincos, а в строках 19 — 20 и 25 — 26 осуществляется умножение комплексных чисел с помощью матрицы поворота двумерного вектора. Далее результаты складываются и отправляются в глобальный массив `g_OutputHt`. Полностью файл «ocean_simulator.cpp» представлен в приложении B.

2.3 Быстрое преобразование Фурье

В файле «fft_shader.hlsl» описано преобразование «Бабочка» восьмого порядка `fft8`:

```
1 void fft8(inout float2 D[8]) {
2   batterfly(D[0], D[4], 0);
3   batterfly(D[1], D[5], 0);
4   batterfly(D[2], D[6], 0);
5   batterfly(D[3], D[7], 0);
6
```

```
7   batterfly(D[0], D[2], 0);
8   batterfly(D[1], D[3], 0);
9
10  batterfly(D[4], D[6], -PI / 2.0);
11  batterfly(D[5], D[7], -PI / 2.0);
12
13  batterfly(D[0], D[1], 0);
14  batterfly(D[4], D[5], -PI / 4.0);
15  batterfly(D[6], D[7], -PI / 4.0 * 3);
16  batterfly(D[2], D[3], -PI / 2.0);
17 }
```

В приведённом выше коде вызову `batterfly` соответствует операция «Бабочка» первого порядка, где третьим аргументом задаётся угол поворота второго элемента преобразуемой пары. Так как двухточечное преобразование Фурье записывается через операцию «Бабочка» с единичным коэффициентом, то двухточечным преобразованиям точек (`x[0]`, `x[4]`), (`x[2]`, `x[6]`), (`x[1]`, `x[5]`), (`x[3]`, `x[7]`) соответствуют вызовы функции `batterfly` с нулевым углом поворота. Также согласно схеме необходимо выполнить двухточечные преобразования точек (`D[0]`, `D[2]`) и (`D[1]`, `D[3]`) с нулевым коэффициентом. Далее в строчках 10, 11, 16 выполняется двухточечное преобразование с коэффициентом W^1. В строчке 15 — преобразование с множителем W^3.

Сам код функции `batterfly` выглядит как представлено ниже:

```
1  void batterfly(inout float2 a, inout float2 b,
2      float phase) {
3    float2 c = a;
4    float2 d = b;
5    //Повернём копию второго элемента на угол phase
6    rotate(d, phase);
7
8   // a = a + Wb;
9    a.x = c.x + d.x;
10   a.y = c.y + d.y;
11
12   // b = a - Wb;
13   b.x = c.x - d.x;
14   b.y = c.y - d.y;
15 }
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были изучены различные способы рендеринга водной поверхности в современной компьютерной графике. Реализован статистический алгоритм моделирования водной поверхности в компьютерной графике и основные приёмы, использующиеся при решении данной задачи: метод скайбокса, различные модели освещения. Был получен опыт разработки под графическую платформу DirectX11, реализовано интерактивное приложение на языке Visual C++, моделирующее водную поверхность океана.

Разработанное приложение можно использовать как в учебных целях, так и в целях рендеринга простых изображений водной среды. Реализованную систему файлов можно интегрировать в существующие DirectX приложения, в которых требуется рендеринг водной поверхности хорошего качества.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Моделирование поверхности воды в режиме реального времени на платформах с многоядерной архитектурой [Электронный ресурс]. — URL: https://software.intel.com/ru-ru/articles/real-time-deep-ocean-simulation-on-multi-threaded-architectures (Дата обращения 10.03.2017). Загл. с экр. Яз. англ.

2 *Франк, Филлип.* Дифференциальные и интегральные уравнения математической физики / Филлип Франк, Рихард Мизес. — Ленинград, 1987. — P. 421.

3 Around GPU Gems. Эффективное моделирование воды на основе физических моделей [Электронный ресурс]. — URL: https://cg.siomax.ru/index.php/around-gpu-gems/8-gpugems-water-modeling-1 (Дата обращения 15.05.2017). Загл. с экр. Яз. рус.

4 *Tessendorf, Jerry.* Simulating ocean wather / Jerry Tessendorf. — ACM Press, 2004.

5 *Luna, Frank D.* Introduction to 3D game programming with DirectX11 / Frank D. Luna. — Mercury learning and information, 2012.

6 Where is the DirectX SDK? [Электронный ресурс]. — URL: https://msdn.microsoft.com/en-us/library/windows/desktop/ee663275(v=vs.85).aspx (Дата обращения 12.05.2017). Загл. с экр. Яз. англ.

7 DXUT Tutorial Win32 Sample [Электронный ресурс]. — URL: https://code.msdn.microsoft.com/windowsdesktop/DXUT-Tutorial-Win32-Sample-fe15e440 (Дата обращения 12.05.2017). Загл. с экр. Яз. англ.

10.06.2017