Государственное образовательное учреждение высшего профессионального образования

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

Моделирование раскачивания объекта на водной поверхности

Студент Медведев А.В.

(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта Деон А.Ф.

(Подпись, дата) (И.О.Фамилия)

Оглавление

[1 Введение 5](#_Toc501062460)

[2 Аналитический раздел 6](#_Toc501062461)

[2.1 Алгоритмы удаления невидимых линий и поверхностей 6](#_Toc501062462)

[Алгоритм Робертса 6](#_Toc501062463)

[Алгоритм трассировки лучей 6](#_Toc501062464)

[Алгоритм Варнока 7](#_Toc501062465)

[Алгоритм, использующий z-буфер 8](#_Toc501062466)

[2.2 Распространённые алгоритмы закрасок 9](#_Toc501062467)

[Модель освещения Ламберта 9](#_Toc501062468)

[Затенение по Гуро 10](#_Toc501062469)

[Закраска по Фонгу 11](#_Toc501062470)

[2.3 Алгоритмы генерации морской волны 12](#_Toc501062471)

[Основные подходы к генерации морской волны 12](#_Toc501062472)

[Система уравнений Навье-Стокса 14](#_Toc501062473)

[Метод Эйлера 16](#_Toc501062474)

[2.4 Обоснование выбора алгоритмов 17](#_Toc501062475)

[3 Конструкторский раздел 17](#_Toc501062476)

[3.1 Общий алгоритм программы 17](#_Toc501062477)

[3.2 Алгоритмы генерации водной поверхности 18](#_Toc501062478)

[Метод суммирования гармонических колебаний 19](#_Toc501062479)

[Моделирование статических волн 19](#_Toc501062480)

[Моделирование динамических волн 20](#_Toc501062481)

[3.3 Построение волн 22](#_Toc501062482)

[4 Технологический раздел 22](#_Toc501062483)

[4.1 Выбор языка программирования и среды разработки 22](#_Toc501062484)

[4.2 Схемы классов 23](#_Toc501062485)

[Общая схема классов 23](#_Toc501062486)

[4.3 Интерфейс программы 24](#_Toc501062487)

[5 Исследовательский раздел 25](#_Toc501062488)

[6 Заключение 26](#_Toc501062489)

[7 Список использованной литературы 27](#_Toc501062490)

# Введение

Одним из популярны направлений использование компьютерной техники стала обработка изображений, и прежде всего - синтез изображений. С каждым годом растут требования к росту реалистичности моделируемого трехмерного изображения. Оно требуется в графике, компьютерных играх, создании сценических эффектов, виртуальной и дополненной реальности,

Целью курсового проекта была разработка программы, моделирующей в реальном времени реалистичное поведение трехмерного объекта (бочки) на поверхности воды.

В рамках реализации проекта должны быть решены следующие задачи:

* Изучение и анализ алгоритмов компьютерной графики, использующихся для создания реалистичной модели взаимно перекрывающихся объектов, и выбор наиболее подходящего для решения поставленной задачи.
* Подробно изучен выбранный алгоритм для поставленной задачи.
* Проектирование архитектуры программы и ее интерфейс.
* Реализация выбранных алгоритмов и структур данных.
* Проведение исследования с помощью программы.

Итогом работы стала программа, демонстрирующая поведение объекта на волнах, позволяющая менять характеристики волн. В программе предусмотрена возможность поворота камеры, приближения и отдаления объектов сцены.

# Аналитический раздел

## Алгоритмы удаления невидимых линий и поверхностей

Для того чтобы выбрать подходящий алгоритм построения изображения, необходимо провести обзор известных алгоритмов и осуществить выбор наиболее подходящего для решения поставленной задачи.

### Алгоритм Робертса

Алгоритм Робертса представляет собой первое известное решение задачи об удалении невидимых линий. Это метод, работающий в объектном пространстве. В соответствии с алгоритмом, прежде всего удаляются из каждого тела те ребра или грани, которые перекрываются самим телом. Затем каждое из видимых ребер каждого тела сравнивается с каждым из оставшихся тел для определения того, какая его часть или части, если таковые есть, перекрываются этими телами.

Преимущества данного алгоритма в том, что математические методы, используемые в нем просты, мощны и точны. Более поздние реализации алгоритма, например, использующие предварительную сортировку вдоль оси z, демонстрируют почти линейную зависимость от числа объектов.

Минус этого алгоритма в том, что теоретически вычислительная трудоемкость алгоритма Робертса растет как квадрат числа объектов. Реализация оптимизированных алгоритмов весьма сложна.

### Алгоритм трассировки лучей

В этом методе для каждого пикселя картинной плоскости определяется ближайшая к нему грань, для чего через этот пиксель выпускается луч, находятся все его пересечения с гранями и среди них выбирается ближайшая.

К достоинствам данного алгоритма можно отнести возможность получения изображения гладких объектов без аппроксимации их примитивами (например, треугольниками). Вычислительная сложность метода линейно зависит от сложности сцены. Нетрудно реализовать наложение света и тени на объекты. Качество полученного изображения получается очень реалистичным, этот метод отлично подходит для создания фотореалистичных картин.

Серьёзным недостатком алгоритма трассирования является производительность. Для получения изображения необходимо создавать огромное число лучей, проходящих через сцену и отражаемых от объекта. Это приводит к существенному снижению скорости работы программы.

### Алгоритм Варнока

Алгоритм Варнока работает в пространстве изображений. В основу алгоритма положен принцип "разделяй и властвуй", состоящий в разбиении области рисунка на более мелкие подобласти (окна). Для каждой подобласти (окна) определяются связанные с ней многоугольники и те из них, видимость которых определить "легко", изображаются на экране. В противном же случае разбиение повторяется, и для каждой из вновь полученных подобластей рекурсивно применяется процедура принятия решения. Предполагается, что с уменьшением размеров области ее перекрывает все меньшее и меньшее количество многоугольников. Считается, что в пределе будут получены области, содержащие не более одного многоугольника, и решение будет принято достаточно просто. Если же в процессе разбиения будут оставаться области, содержащие не один многоугольник, то следует продолжать процесс разбиения до тех пор, пока размер области не станет совпадать с одним пикселом. В этом случае для полученного пикселя необходимо вычислить глубину (значение координаты Z) каждого многоугольника и визуализировать тот из них, у которого максимальное значение этой координаты.

### Алгоритм, использующий z-буфер

Данный алгоритм удаления невидимых поверхностей является одним из простейших. Этот алгоритм работает в пространстве изображения. Здесь обобщается идея о буфере кадра. Буфер кадра используется для заполнения атрибутов (интенсивности) каждого пикселя в пространстве изображения. Наряду с буфером кадра вводится Z-буфер, представляющий собой специальный буфер глубины, в котором запоминаются координаты Z (глубина) каждого видимого пикселя в пространстве изображения. В процессе работы глубина (значение координаты Z) каждого нового пикселя, который надо занести в буфер кадра, сравнивается с глубиной того пикселя, который уже занесен в Z-буфер. Если это сравнение показывает, что новый пиксель расположен ближе к наблюдателю, чем пиксел, уже находящийся в буфере кадра, то новый пиксель заносится в буфер кадра. Помимо этого, производится корректировка Z-буфера: в него заносится глубина нового пикселя. Если же глубина (значение координаты Z) нового пикселя меньше, чем хранящегося в буфере, то никаких действий производить не надо. В сущности, алгоритм для каждой точки (x,y) находит наибольшее значение функции Z(x,y).

Несмотря на свою простоту, этот алгоритм позволяет удалять сложные поверхности и визуализировать пересечения таких поверхностей. Сцены могут быть произвольной сложности, а поскольку размеры изображения ограничены размером экрана дисплея, то трудоемкость алгоритма имеет линейную зависимость от числа рассматриваемых поверхностей. Элементы сцены заносятся в буфер кадра в произвольном порядке, поэтому в данном алгоритме не тратится время на выполнение сортировок, необходимых в других алгоритмах.

## Распространённые алгоритмы закрасок

### Модель освещения Ламберта

Одной из самых простых моделей освещения является модель Ламберта. Она учитывает только идеальное диффузное отражение света от тела. Считается, что свет падающий в точку, одинакового рассеивается по всем направлениям полупространства (см Рисунок 1). Таким образом, освещенность в точке определяется только плотностью света в точке поверхности, а она линейно зависит от косинуса угла падения. При этом положение наблюдателя не имеет значение, т.к. диффузно отраженный свет рассеивается равномерно по всем направлениям.

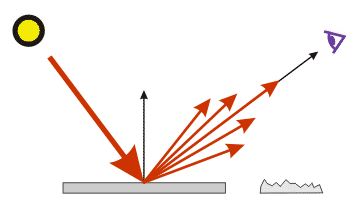


Рисунок 1.

Пусть

α – угол между вектором направления света и нормалью к поверхности

I – результирующая интенсивность света в точке

I0 – интенсивность источника

k – способность материала к диффузному отражению

Тогда формула расчёта интенсивности будет иметь следующий вид:

I = I0\*k\*cos(α) (1)

Большим недостатком данной модели является то, что согласно приведённой выше формуле, все точки грани будут иметь одинаковую интенсивность.

### Затенение по Гуро

Метод Гуро является одним из способов устранения дискретности интенсивностей закрашивания.

Данный алгоритм предполагает следующие шаги:

1. Вычисление векторов нормалей к каждой грани.
2. Вычисление векторов нормали к каждой вершине грани путем усреднения нормалей к граням (см. рисунок 2)
3. Вычисление интенсивности в вершинах грани.
4. Интерполяция интенсивности вдоль ребер грани.
5. Линейная интерполяция интенсивности вдоль сканирующей строки.

Закраска по Гуро хорошо сочетается с диффузным отражением. Данный метод интерполяции обеспечивает лишь непрерывность значений интенсивности вдоль границ многоугольников, но не обеспечивает непрерывность изменения интенсивности, следовательно, возможно появление полос Маха.

На рисунке 2 Nv – нормаль к вершине – усреднённое значение нормалей N1…NN.

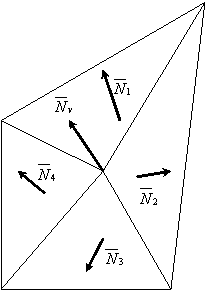


Рисунок 2.

### Закраска по Фонгу

Закраска по Фонгу требует б*о*льших вычислительных затрат, чем Гуро. При такой закраске, в отличие от метода Гуро, вдоль сканирующей строки интерполируется значение вектора нормали, а не интенсивности. Используя закраску по Фонгу, можно достичь лучшей локальной аппроксимации кривизны поверхности.

Шаги алгоритма:

1. Вычисление векторов нормалей в каждой грани.
2. Вычисление векторов нормали к каждой вершине грани.
3. Интерполяция векторов нормалей вдоль ребер грани.
4. Линейная интерполяция векторов нормалей вдоль сканирующей строки.
5. Вычисление интенсивности в очередной точке сканирующей строки.

Изображение получается более реалистичным. Закраска по Фонгу хорошо сочетается с зеркальным отражением.

Основными недостатками данного алгоритма являются ресурсоемкость и большая вычислительная сложность.

## Алгоритмы генерации морской волны

### Основные подходы к генерации морской волны

Для представления динамики океана можно использовать различные модели: параметрическое описание, спектральное описание, а также модели из «Расчетной жидкостной динамики» (CFD) и, более конкретно, уравнение Навье-Стокса (NSE).

Первая категория направлена на вычисление пути частиц воды и описывает свободную поверхность с параметрическими уравнениями, основанными на реальных наблюдениях, полученных из буев или спутниковых измерений. Вторая категория аппроксимирует состояние моря и распределение волн по их амплитудам и частотам. Наконец, NSE может представлять динамику всех типов жидкостей, включая динамическое поведение океана.

Методы моделирования океана в области компьютерной графики могут быть классифицированы по двум основным категориям: параметрическим /спектральным методам, использующим океанографические модели, и физическим методам, основанным на NSE.

Параметрические/спектральные модели лучше всего работают в глубоких водах, где они точно представляют периодическое движение моря. Но поскольку они не учитывают взаимодействия с дном моря на мелководье, только физически основанные методы, получающие приближенные решения от NSE, могут воспроизвести сложность динамики океана вблизи берега.

Другими важными характеристиками океанов для компьютерной графики являются их сложные оптические свойства. Например, цвет океанских вод, варьирующий от зеленого до глубокого синего. Для повышения реалистичности на стадии рендеринга возможно включение моделирования несколько других явлений, таких как пена, спреи, свойства воды (мутность, пузырьки и т.д.)

Моделированием водной поверхности занималось достаточное количество разработчиков, наиболее успешным среди которых был Тессендорф (Tessendorf). Разработанную им модель океана была использована, например, в фильме «Титаник». В последние годы ряд других разработчиков решали схожие задачи с учетом возможностей моделирования в режиме реального времени. Например, Мигель Гомес (Miguel Gomez), в своей книге «Интерактивное моделирование водных поверхностей» (Interactive Simulation of Water Surfaces) описал решение для вычисления полей высот. В некоторых случаях такое решение можно считать оптимальным, однако его главным недостатком является необходимость сохранения в памяти по меньшей мере двух сеток рендеринга – вычисленной ранее и текущей, чтобы рассчитать новую. Еще один минус этого решения – необходимость получения данных о соседних вершинах перед вычислением новой позиции каждой из них. Прямое же решение, не требующее таких данных, было описано Марком Финчем (Mark Finch) в книге «Эффективное моделирование водных поверхностей на основе физических моделей» (Effective Water Simulation from Physical Models). Оно обладает целым рядом преимуществ:

• для определения новых координат вершин не требуются данные о соседних вершинах, поэтому код легче распараллелить;

• поскольку не требуется данных соседних вершин, упрощается реализация вершинных шейдеров,

• полностью параметризованное моделирование позволяет управлять геометрией модели;

• существует возможность обновления нормалей с учетом данных только по локальным вершинам, что упрощает распараллеливание в процессе реализации вершинных шейдеров; в то же время обновление нормалей можно проводить с учетом данных по соседним вершинам;

• упрощается масштабирование и расширяемость кода, что позволяет реализовать новые функциональные возможности. Этому способствует и возможность тонкой настройки параметров;

• алгоритм многофункционален, поскольку один и тот же подход пригоден как для моделирования поверхностей с большими и редкими волнами, так и для поверхностей с мелкой рябью от ветра.

Джон Айсидоро (John Isidoro) в книге «Рендеринг океанской воды» (Rendering Ocean Water) описал метод суммирования синусоидальных волн, схожий с рассмотренным в книге Марка Финча.

### Система уравнений Навье-Стокса

Всю сложность движущихся потоков жидкости способна описать система уравнений Навье-Стокса. В случае несжимаемой жидкости, к которой можно отнести и воду, система состоит из двух уравнений: уравнения движения и уравнения неразрывности.

Использование уравнений Навье-Стокса – это физический подход к моделированию поверхности воды. Такой подход позволяет получать реалистичную модель поверхности жидкости, но требует большого количества вычислительных ресурсов.

Существуют два основных метода решения уравнений Навье-Стокса: метод Лагранжа и метод сеток Эйлера (конечных разностей).

В методе Лагранжа жидкость рассматривается как набор частиц, подчиняющихся физическим законам. Каждая частица имеет набор параметров, таких, как масса, скорость и др. и может оказывать влияние на соседние частицы.

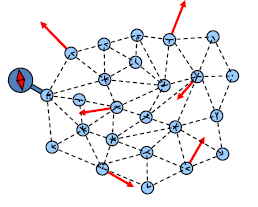
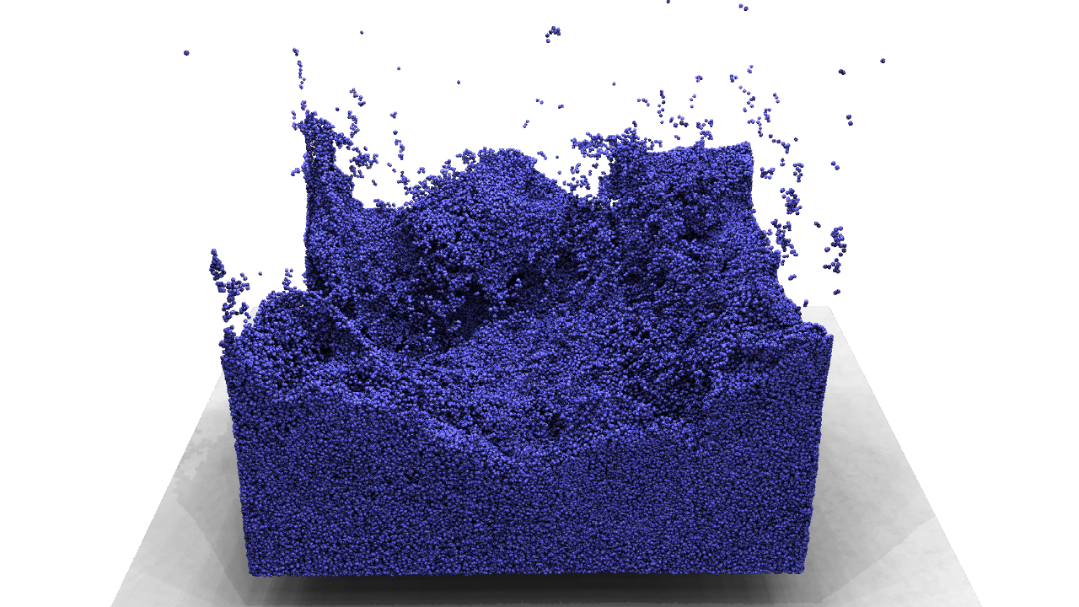


Рисунок 3. Взаимодействие частиц в методе Лагранжа

Методы решения уравнений, основанные на взаимодействии частиц, интуитивно понятны, их легче кодировать, к тому же, вычисление решения уравнений не требует много времени, т.к. не нужно находить решения систем уравнений. Однако, данный метод менее точен, чем метод Эйлера и не способен описать всю сложность внутреннего взаимодействия жидкости.

Высокую реалистичность визуализации жидкости может обеспечить модель гидродинамики сглаженных частиц. Главная идея данного метода – представление каждой отдельно взятой частицы как неявной поверхности, способной воздействовать на соседние частицы.



**Рисунок 4. Визуализация жидкости с использованием метода гидродинамики сглаженных частиц**

Тем не менее, этому методу присущи две проблемы: модель теряет свою реалистичность по мере того, как объем воды, представленной частицами, изменяется с течением времени, кроме того, для детализированной визуализации большого объема воды требуется огромное число частиц, что приводит к высоким требованиям к вычислительным ресурсам.

### Метод Эйлера

Метод Эйлера предполагает дискретизацию потока жидкости с помощью введения двухмерной сетки и моделирование поверхности жидкости как движение её между узлами сетки.

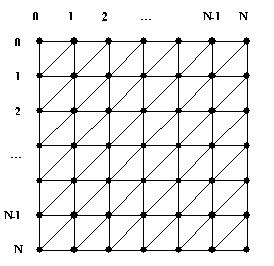


Рисунок 5. Общий вид равномерной сетки высот

Сеточные методы более точны, чем методы взаимодействия дискретных частиц, т.к. вычисления легче проводить с регулярной сеткой, чем с неупорядоченным набором частиц. Более того, использование регулярной сетки позволяет упростить геометрию моделируемой водной поверхности, получить более качественное изображение гладкой поверхности воды, к тому же, к координатам каждой вершины сетки легко получить доступ через двухмерный массив вершин. Однако, этот метод не лишён недостатков. Вычисления с помощью метода Эйлера будут производиться несколько дольше, чем с использованием метода Лагранжа, т.к. понадобится решение систем уравнений, с помощью сеточных методов невозможно моделировать некоторые трёхмерные свойства жидкости, например, разбивающиеся волны океана.

## Обоснование выбора алгоритмов

Для реализации водной поверхности был выбран сеточный метод, так как он позволяет упростить геометрию моделируемой водной поверхности и получить более качественное изображение гладкой поверхности воды. Для отображения сцены был выбран алгоритм трассировки лучей, так как он позволяет получить очень достоверное изображение.

# Конструкторский раздел

## Общий алгоритм программы

После запуска программы загружается модель объекта из файла в формате 3ds, устанавливаются параметры волн, и после нажатия кнопки «Старт» начинается генерация изображения. В время работы программы пользователь может изменять параметры любой из 4-х волн, также менять положением камеры, рассматривая поверхность и модель с разных точек.

Алгоритм работы программы:

1. Загрузка модели из 3ds файла.
2. Установка первоначальных параметров для 4х волн.
3. Генерация поверхности воды.
4. Вычисление координат и положения объекта (наклон в разных плоскостях)
5. Преобразование координат (повороты моделей, масштабирование и перенос) относительно камеры.
6. Отображение объекта и поверхности воды
7. Считывание введенных пользователем параметров волн и расположения камеры
8. Возвращение к п.3

## Алгоритмы генерации водной поверхности

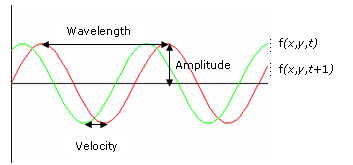
Рассмотрим несколько основных понятий волновой теории.

**Амплитуда волны (amplitude)**: величина пика или впадины относительно положения равновесия. Сумма абсолютных величин пика и впадины волны называется двойной амплитудой.

**Длина волны (wavelength)**: расстояние между двумя ближайшими гребнями волны.

**Скорость волны (velocity)**: определяет перемещение волны в среде с течением времени. φ, фазовая постоянная, характеризует распространение волны в среде и определяется по формуле φ = скорость волны \* *(2 π)/длина волны*

### Метод суммирования гармонических колебаний



**Рисунок 6. Физические свойства волны**

На рисунке 6 представлены графики гармонических колебаний. На графиках отмечены: *длина волны* (расстояние между двумя ближайшими гребнями), *скорость* (смещение волны в единицу времени) и *амплитуда* (максимальное отклонение от исходного положения).

### Моделирование статических волн

Для получения периодической волны с контролируемыми параметрами необходимо взять синусоидальную функцию, описываемую формулой:

https://software.intel.com/sites/default/files/oceansim2.png

В нашем случае мы используем нормализованное пространство, в котором значения всех синусоидальных функций находятся в пределах от 0 до 1. Так будет проще определить действия, необходимые для задания координат моделируемой водной поверхности. Изменим функцию таким образом, чтобы ее значения были всегда положительны:

https://software.intel.com/sites/default/files/oceansim3.png

Из формулы видно, что значения функции будут всегда больше либо равно нулю. В то же время часть из них будут больше 1, поэтому преобразуем функцию еще раз, чтобы все ее значения находились в интервале от 0 до 1:

https://software.intel.com/sites/default/files/oceansim4.png

Очень часто, особенно при моделировании [относительно гладкой поверхности воды](https://software.intel.com/sites/default/files/m/d/4/1/d/8/IntegratedGFX_MTADeepOceanWave_Demo.zip), предпочтительно использовать именно такие масштабные гармонические колебания. Однако, если необходимо смоделировать море перед приближающейся бурей, требуются более крутые волны. Такой эффект позволяет получить новый параметр – крутизна, которая добавлена в формулу в виде показателя степени:

https://software.intel.com/sites/default/files/oceansim5.png

Для реализации следующей цели – управления высотой волн (амплитудой) – добавим в формулу масштабный коэффициент:

https://software.intel.com/sites/default/files/oceansim6.png

### Моделирование динамических волн

Теперь мы имеем формулу синусоидальной функции, крутизну и амплитуду которой мы можем контролировать. Но для полноценного моделирования водной поверхности этого недостаточно: необходимо учесть скорость, направление, а также длину волн.

Поскольку моделируется двухмерное поле высот, нужно предусмотреть движение волн в обоих направлениях. Для этого необходимо вычислить проекцию заданных координат х и у на вектор направления волны путем скалярного произведения. Предположим, что вектор направления волны параллелен плоской поверхности и не имеет координаты z (это упростит вычисления). Результатом скалярного произведения двух векторов является скалярная величина, которую мы обозначим как S:

https://software.intel.com/sites/default/files/oceansim7.png

Теперь предусмотрим возможность изменения частоты волны. Из курса физики известно соотношение, описывающее связь длины волны с ее частотой: частота = 2 pi / длина волны. Поэтому воспользуемся длиной волны в качестве исходных данных для вычисления частоты по формуле, описанной выше. Добавим в формулу S новый множитель – частоту, чтобы иметь возможность изменять частоту нашей синусоидальной функции:

https://software.intel.com/sites/default/files/oceansim8.png

В нашей формуле учтены направление и длина волны, необходимые для определения координат поверхности, но не предусмотрено движение волны по этой поверхности. Преобразуем формулу таким образом, чтобы учесть скорость волны. Фазовая постоянная φ связана с волновой скоростью следующим соотношением:

https://software.intel.com/sites/default/files/oceansim9.png

Преобразуем нашу формулу следующим образом (t – время):

https://software.intel.com/sites/default/files/oceansim10.png

В итоге мы получили функцию, в которой учтены длина, амплитуда, скорость, направление, положение и крутизна волны:

https://software.intel.com/sites/default/files/oceansim11.png  
Где

https://software.intel.com/sites/default/files/oceansim12.png  
и

https://software.intel.com/sites/default/files/oceansim13.png

## Построение волн

Функция построения одиночной волны пригодна лишь для простейшего моделирования, в то время как нашей задачей является моделирование максимально реалистичной водной поверхности. Сидя на берегу моря, вы наверняка видели на его поверхности множество волн, приходящих со всех сторон. Волны сталкиваются, образуя водные вершины и впадины, положение которых постоянно меняется. Для моделирования такого процесса необходимо использовать суммирование нескольких синусоидальных функций в каждой точке поверхности. Опытным путем установлено, что четырех синусоидальных волн вполне достаточно для создания довольно реалистичной, изменчивой морской поверхности. Для моделирования вершины волны с координатами (x,y) воспользуемся формулой:

https://software.intel.com/sites/default/files/oceansim14.png

.

# Технологический раздел

## Выбор языка программирования и среды разработки

В качестве языка программирования был выбран язык C#. Этот язык поддерживает объектно-ориентированную парадигму программирования, что позволяет естественным образом декомпозировать задачу и легко модифицировать программу. Язык C# содержит встроенные библиотеки для работы с многопоточностью, что позволяет использовать больше ресурсов компьютера.

В качестве среды была выбрана Visual Studio 2017 и .NET Framework 4.7. Данная среда обладает удобным редактором кода и отладчиком, широким набором настроек, вместе со средой поставляются графические библиотеки.

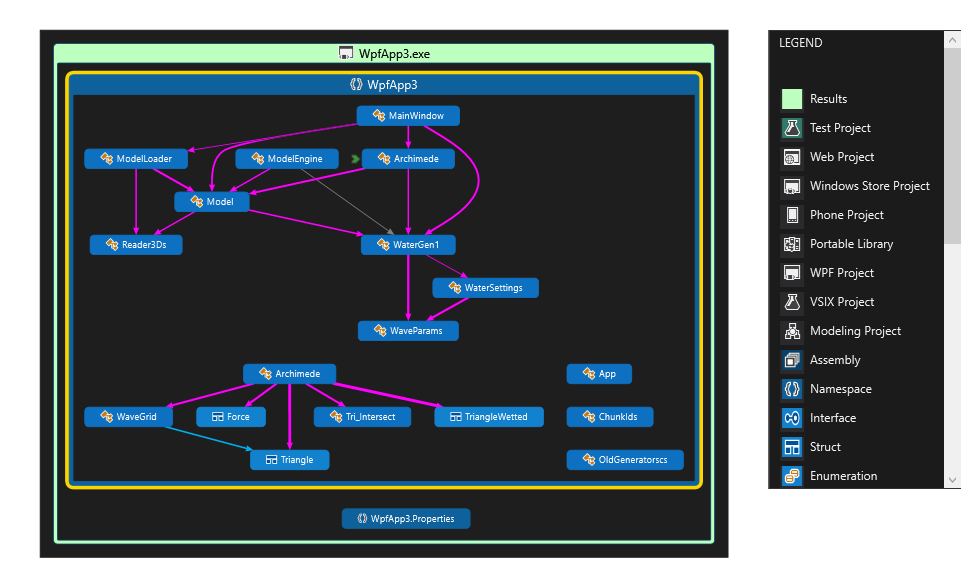
Была создана программа, которая позволяет загрузить трехмерную модель объекта (бочки) из 3ds файла, а также генерирует водную поверхность с волнами. Программа позволяет в процессе работы изменять параметры водной поверхности, а также позволяет управлять камерой, двигая ее по полусфере, приближая и отдалять изображение.

## Схемы классов

В данном подразделе представлены основные классы приложения.

### Общая схема классов

Классовая диаграмма проекта на рисунке 7:



Назначение классов:

* MainWindow – класс пользовательского интерфейса, который отображает сцену и позволяет настраивать параметры камеры
* ModelLoader – класс, обрабатывающий загрузку сцены
* Reader3DS- Класс -парсер, читающий модель из 3DS файла
* Model – Класс, хранящий все параметры модели, такие как список точек, нормалей и примененных к телу трансформаций.
* ModelEngine -Класс, проводящий преобразования модели
* WaterGen1- Класс, имитирующий поведение водной поверхности
* WaterSettings,WaveParams - настройки параметров волны
* Archimede и связанные с ним классы рассчитывают поведение модели в зависимости от состояния водной поверхности

## Интерфейс программы

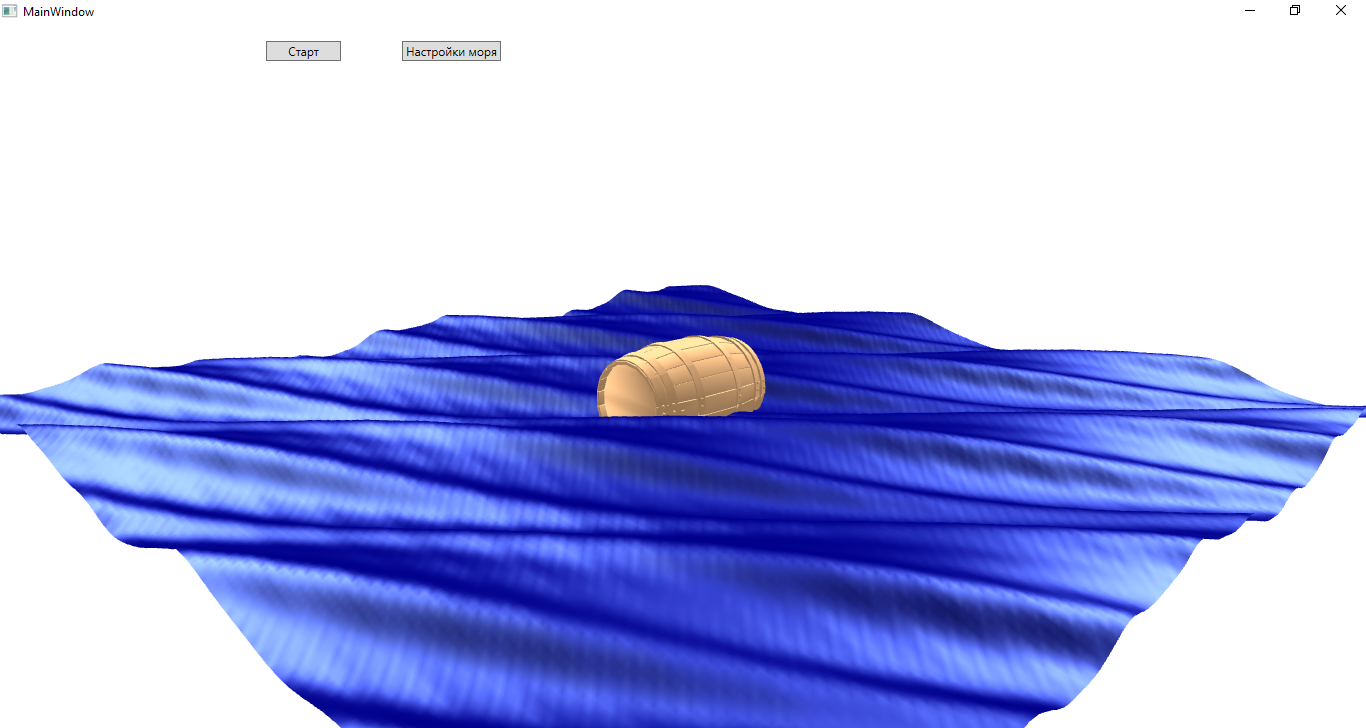


Рисунок 6. Главное окно программы

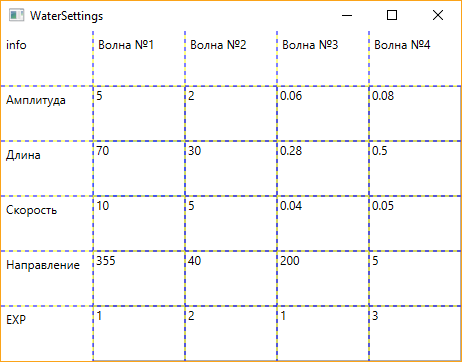


Рисунок 7. Интерфейс настройки параметров волн

# Исследовательский раздел

Была измерена скорость вычислений в представленной программе в зависимости от размера текстуры.

Рисунок 8.

Для проведения замеров времени использовались следующие параметры:

* Размер текстуры 100x100, 150x150, 200x200,250x250,300x300

Исходя из полученных данных, следует вывод, что использование большой площади водной поверхности не выгодно. С увеличение площади водной поверхности, увеличивается время необходимое на генерацию каждого кадра изображение из-за необходимости обработки дополнительных точек и нормалей.

Технические характеристики ЭВМ, на которой были проведены тесты:

* Процессор: Intel® Core™ i7-4500U CPU @ 2.40GHz
* ОЗУ: 8,00 Гб DDR3
* Интегрированный графический процессор: Intel® HDGraphics 4400

# Заключение

При написании проекта были рассмотрены и проанализированы основные алгоритмы генерации морской волны, проанализированы их достоинства, недостатки, а также возможность использования для решения поставленной задачи. Для реализации данной задачи были выбраны советующие алгоритмы. Проведено обобщение теоретических и экспериментальных исследований, по материалам которых подготовлена расчетно-пояснительная записка.

Разработанная программа позволяет получать на экране дисплея реалистичную модель поверхности моря с плавающим объектом (бочкой). Пользователь может менять через интерфейс параметры волн и положение камеры.

Приложение имеет несколько направлений дальнейшего развития. Например, на данном этапе не была реализована возможность создать прозрачную водную гладь и создания пены, для создания более реалистичного изображения.

# Список использованной литературы

1. Д. Роджерс. Алгоритмические основы машинной графики./Д. Роджерс. – М.: Мир, 1989. – 512с
2. Gustav Olsson. Real-Time Water Animation and Rendering using Wavefront Parameter Interpolation – Chalmers University of Technology, 2017 – 74с
3. Maila Sepri. Application of wave generator theory to the development of a Wave Energy Converter, 2008 -208c
4. Sanjay Sen. Methods of Deep Ocean Simulation, 2013
5. Y. Nishidate and G. P. Nikishkov. Fast Water Animation Using the Wave Equation with Damping, 2005.
6. Николай А. Елыков, Игорь В. Белаго. Генерация изображения морской поверхности в реальном времени – Институт Автоматики и Электрометрии СО РАН – 2000
7. Peter Shirley and Steve Marschner. Fundamentals of Computer Graphics - 3rd Edition, 2009. 782с.