



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 «Программное обеспечение ЭВМ и информационные технологии»

## РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

*К КУРСОВОЙ РАБОТЕ*

*НА ТЕМУ:*

*«Разработка программного обеспечения для  
визуализации волн при движении твердого тела»*

Студент ИУ7-53Б  
(Группа)

\_\_\_\_\_  
(Подпись, дата)

**Р. Р. Хамзина**  
(И.О.Фамилия)

Руководитель

\_\_\_\_\_  
(Подпись, дата)

**А. А. Оленев**  
(И.О.Фамилия)

2021 г.

**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)**

---

УТВЕРЖДАЮ  
Заведующий кафедрой ИУ7  
(Индекс)  
И.В.Рудаков  
(И.О.Фамилия)  
« \_\_\_\_\_ » 20 \_\_\_\_ г.

**З А Д А Н И Е  
на выполнение курсовой работы**

по дисциплине Компьютерная графика  
Студент группы ИУ7-53Б

Хамзина Регина Ренатовна  
(Фамилия, имя, отчество)

Тема курсовой работы Разработка программного обеспечения для визуализации волн при движении твердого тела

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

учебная

Источник тематики (кафедра, предприятие, НИР) кафедра  
График выполнения работы: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

**Задание:** разработать программное обеспечение, которое должно предоставлять возможность визуализации волн, образованных при взаимодействии поверхности воды с движущимся твёрдым телом. Реализовать интерфейс, который должен позволять пользователю загружать модель предмета и указывать скорость его движения, изменять указанную скорость в интерактивном режиме, а также управлять положением камеры (вращение, перемещение и масштабирование). Моделью должно быть тело, образованное геометрическими объектами из следующего набора: сфера, параллелепипед, призма трехгранная.

***Оформление курсовой работы:***

Расчетно-пояснительная записка на 25-30 листах формата А4.

Расчетно-пояснительная записка должна содержать постановку задачи, введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.

Перечень графического (илюстративного) материала (чертежи, плакаты, слайды и т.п.)  
На защиту проекта должна быть предоставлена презентация, состоящая из 10-15 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, интерфейс, результаты проведенных исследований.

Дата выдачи задания « \_\_\_\_\_ » 20 \_\_\_\_ г.

**Руководитель курсовой работы**

(Подпись, дата)

А. А. Оленев

(И.О.Фамилия)

**Студент**

(Подпись, дата)

Р. Р. Хамзина

(И.О.Фамилия)

# Содержание

<b>Введение</b>	.	.	.	.	.	.	.	<b>4</b>
<b>1 Аналитическая часть</b>	.	.	.	.	.	.	.	<b>6</b>
1.1	Волновой процесс	.	.	.	.	.	.	6
1.2	Модели предмета и волны	.	.	.	.	.	.	7
1.2.1	Модель предмета	.	.	.	.	.	.	7
1.2.2	Модель волны	.	.	.	.	.	.	7
1.3	Дисперсионное соотношение	.	.	.	.	.	.	10
1.4	Методы визуализации волн	.	.	.	.	.	.	11
1.4.1	Процедурные методы	.	.	.	.	.	.	11
1.4.2	Методы на основе частиц	.	.	.	.	.	.	12
1.4.3	Метод поля высоты	.	.	.	.	.	.	13
1.5	Методы рендеринга отображения	.	.	.	.	.	.	16
1.5.1	DirectX	.	.	.	.	.	.	16
1.5.2	Vulkan	.	.	.	.	.	.	17
1.5.3	OpenGL	.	.	.	.	.	.	17
1.6	Существующие программные обеспечения	.	.	.	.	.	.	19
<b>2 Конструкторская часть</b>	.	.	.	.	.	.	.	<b>21</b>
2.1	Требования к программному обеспечению	.	.	.	.	.	.	21
2.2	Разработка алгоритмов	.	.	.	.	.	.	21
2.3	Описание используемых типов и структур данных	.	.	.	.	.	.	25
2.4	Структура программного обеспечения	.	.	.	.	.	.	26
<b>3 Технологическая часть</b>	.	.	.	.	.	.	.	<b>28</b>

3.1	Средства реализации . . . . .	28
3.2	Реализация алгоритмов . . . . .	29
<b>4</b>	<b>Исследовательская часть . . . . .</b>	<b>31</b>
4.1	Примеры работы программного обеспечения . . . . .	31
4.2	Постановка эксперимента . . . . .	33
4.2.1	Цель эксперимента . . . . .	33
4.2.2	Технические характеристики . . . . .	34
4.2.3	Результаты эксперимента . . . . .	34
	<b>Заключение . . . . .</b>	<b>37</b>
	<b>Список литературы . . . . .</b>	<b>38</b>

# Введение

Компьютерная графика применяется в киноиндустрии и разработке компьютерных игр [1]. Методами компьютерной графики решаются задачи представления объектов и процессов реальной жизни в виртуальной реальности. Способ визуализации предметов и действий оценивают при помощи характеристик — реалистичности результата и скорости выполнения. Для повышения указанных параметров создаются новые алгоритмы и методы моделирования.

Представление жидкости — модель, которую реализуют в дизайне компьютерных игр и кинематографических спецэффектах: моделирование водоёмов, процессов смешивания и движения водных потоков [2]. Важным физическим явлением для создания реалистичного водоема является образование волн на поверхности воды [3]. Для получения более точного изображения визуализируют круговые волны, наложение волн, их прозрачность.

Поверхность воды рассматривают в системе с окружающим миром: при взаимодействии с предметами и препятствиями. Особую сложность для моделирования представляют волны, образованные при движении объектов по воде.

Цель работы — разработать программное обеспечение, которое предоставляет возможность визуализации волн, образованных при взаимодействии поверхности воды с движущимся твердым телом.

Для достижения поставленной цели требуется решить следующие задачи:

- изучить волновой процесс;
- формально описать структуру системы, состоящей из поверхности воды и источника волн;
- проанализировать методы и алгоритмы, моделирующие волновой процесс;
- выбрать алгоритм и структуры данных для визуализации описанной выше системы;
- реализовать выбранный алгоритм моделирования;

- провести анализ производительности программного обеспечения.

# 1 Аналитическая часть

В данном разделе рассматривается система, в которой происходит волновой процесс, состоящая из поверхности воды и предмета. Изучаются и сравниваются методы и алгоритмы, визуализирующие волны. В результате анализа определяется алгоритм, эффективно решающий задачу моделирования волн, образованных движением объекта.

## 1.1 Волновой процесс

Процесс образования волны [4] при условии, что частицы воды находятся в состоянии равновесия, состоит из следующих этапов:

- на поверхность воды оказывается внешнее воздействие (ветер, движение корабля, падение камня), частицы жидкости опускаются вниз, водная поверхность становится вогнутой;
- сила тяжести или сила поверхностного натяжения стремится вернуть частицы в состояние равновесия;
- частицы воды переходят положения равновесия, и поверхность воды становится выпуклой.

Движение передается от одних частиц к другим. Этапы волнового процесса показаны на рисунке 1.1.

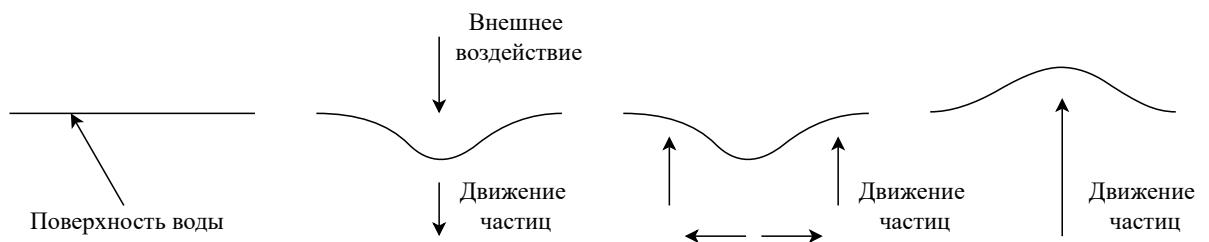


Рисунок 1.1 – Процесс образования волны

Если восстановить равновесие стремится сила тяжести, то волны называются гравитационными, если сила поверхностного натяжения — капилляр-

ными. Волны называют капиллярно-гравитационными, когда силы сопоставимы.

## 1.2 Модели предмета и волны

Внешним воздействием в моделируемой системе является движение предмета.

### 1.2.1 Модель предмета

Для правильной обработки появления волн от твердого тела важны параметры только той части предмета, которая касается воды. В связи с этим нет необходимости рассматривать объект детально.

### 1.2.2 Модель волны

Геометрически волна состоит из следующих элементов:

- гребень — множество точек волны с максимальным положительным отклонением от состояния равновесия;
- подошва — множество точек волны с наибольшим отрицательным отклонением от состояния равновесия.

Кроме того волна обладает следующими параметрами:

- высота — вертикальное расстояние от подошвы до гребня;
- длина — горизонтальное расстояние от гребня до гребня;
- период — временной интервал между прибытием последовательных гребней;
- частота — величина, обратная периоду;

- амплитуда — максимальное отклонение точек волны от положения равновесия;
- скорость распространения.

Модель волны показана на рисунке 1.2.

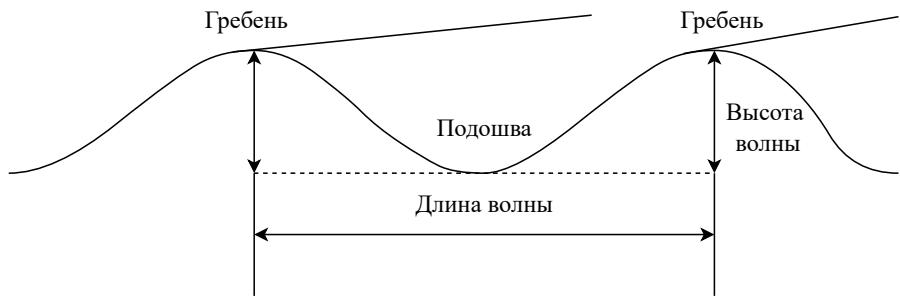


Рисунок 1.2 – Геометрическое строение волны

Параметры волны, показанные на рисунке 1.3, обозначаются следующим образом:

- $h(x, t)$  — высота волны в положении  $x$  в момент времени  $t$ ;
- $\lambda$  — длина;
- $T$  — период;
- $\nu$  — частота;
- $A$  — амплитуда;
- $v$  — скорость распространения.

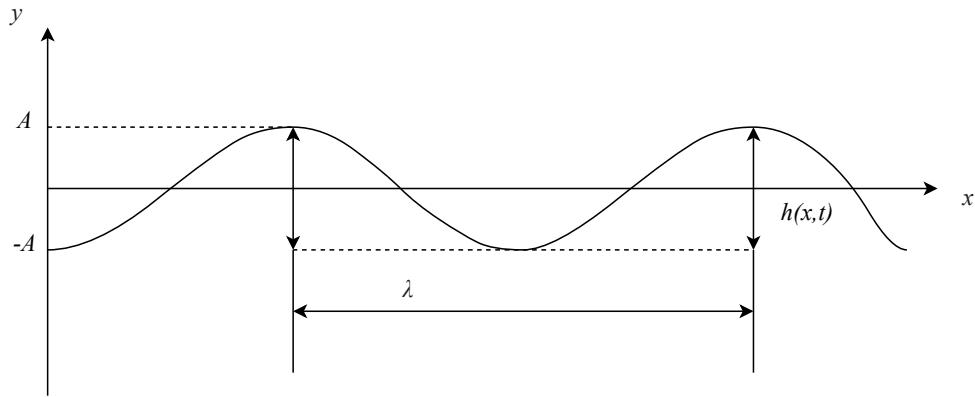


Рисунок 1.3 – График распространения волны

Распространение волн в среде [4] в общем случае описывается волновым уравнением:

$$\Delta h(x, t) = \frac{1}{v^2} \cdot \frac{\partial^2 h(x, t)}{\partial t^2} \quad (1.1)$$

где  $\Delta$  — оператор Лапласа,  $x \in \mathbb{R}^2$ .

Решением волнового уравнения 1.1 является уравнение бегущей волны:

$$h(x, t) = A \cos \omega(t - \frac{x}{v}) \quad (1.2)$$

где  $\omega$  — циклическая частота, которая равняется

$$\omega = 2\pi\nu \quad (1.3)$$

Можно ввести волновое число:

$$k = \frac{2\pi}{\lambda} \quad (1.4)$$

Вектор, абсолютное значение которого равно волновому числу, называют волновым вектором.

Длину волны  $\lambda$  можно выразить следующей формулой:

$$\lambda = vT \quad (1.5)$$

С учетом формул 1.3 и 1.5 формулу 1.4 можно записать так:

$$k = \frac{2\pi}{vT} = \frac{2\pi\nu}{v} = \frac{\omega}{v} \quad (1.6)$$

Отсюда скорость распространения волны:

$$v = \frac{\omega}{k} \quad (1.7)$$

Тогда уравнение бегущей волны 1.2:

$$h(x, t) = A \cos \omega \left( t - \frac{x}{v} \right) = A \cos \left( \omega t - \omega \frac{kx}{\omega} \right) = A \cos(\omega t - kx) \quad (1.8)$$

При поглощении средой энергии волны наблюдается затухание волн. Поэтому уравнение бегущей волны принимает вид:

$$h(x, t) = A \exp(-\beta t) \cos(\omega t - kx) \quad (1.9)$$

где  $\beta$  — коэффициент затухания.

### 1.3 Дисперсионное соотношение

Механизм образования волн подчиняется закону дисперсии.

Дисперсия волн — это зависимость циклической частоты волны от волнового вектора:

$$\omega = \omega(\vec{k}) \quad (1.10)$$

Соотношение означает, что волны разных длин перемещаются с разными фазовыми скоростями. Именно дисперсия создает сложную картину волн, образованных телом в воде.

В зависимости от требований к результату моделирования выбирается определенный метод визуализации. Так как в центре работы дисперсионные волны, для выполнения поставленных задач необходим такой метод моделирования, при котором выполняется дисперсионное соотношение.

## 1.4 Методы визуализации волн

Существует три группы методов моделирования волн:

- процедурные;
- методы на основе частиц;
- метод поля высоты.

### 1.4.1 Процедурные методы

Идея процедурных методов заключается в представлении волновой поверхности наложением периодических функций: циклоиды в ранних работах [5] и синусоиды в дальнейших [6].

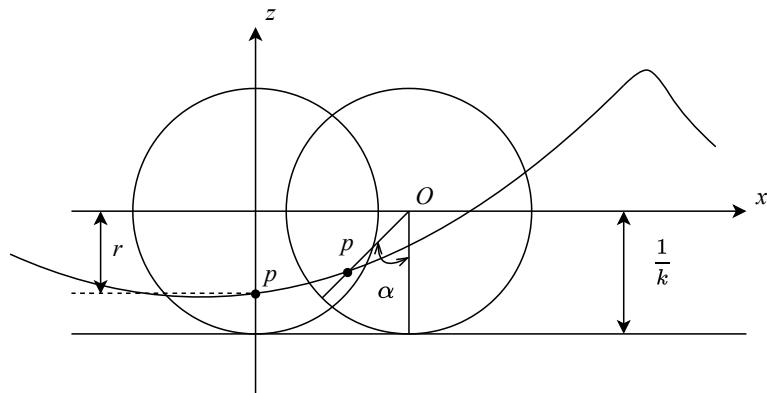


Рисунок 1.4 – Движение точки в процедурном методе

На рисунке 1.4 плоскость покоящейся волновой поверхности параллельна плоскости  $XY$ , ось  $Z$  направлена вверх. Точка  $p$  описывает окружность радиусом  $\frac{1}{k}$  вокруг своего положения покоя — точки  $O$  с координатами  $(x_O, y_O, z_O)$ . Точка  $p$  находится на расстоянии  $r$  от точки  $O$ . Тогда уравнение движения точки:

$$x = x_O + r \sin(kx_O - \omega t) \quad (1.11)$$

$$z = z_0 - r \cos(kx_O - \omega t) \quad (1.12)$$

Волна — траектория движения точки  $p$ . Для создания различных волновых эффектов изменяют параметры уравнений орбиты, например, радиус или фазовый угол  $\alpha$ .

Преимущество процедурных методов — возможность точно контролировать движение волнового фронта — поверхности воды, все точки которой движутся в одинаковой фазе.

Недостаток процедурных методов — сложность получения правильного взаимодействия волн с погруженными телами и границами.

### 1.4.2 Методы на основе частиц

В методах моделирования волновой поверхности на основе частиц [7] вода представляется как система частиц. Каждая частица  $i$  имеет массу  $m_i$  и обладает плотностью  $\rho_i$ , давлением  $p_i$  и объемом  $V_i$ . С течением времени  $t$  положение частицы  $x_i$  и ее параметры меняются в соответствии с учетом скорости  $v_i$ :

$$\frac{dx_i}{dt} = v_i \quad (1.13)$$

Частицы движутся в потоке жидкости, поэтому скорость  $v_i$  определяется формой Лагранжа уравнения Навье-Стокса:

$$\frac{dv_i}{dt} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 v_i + \frac{F_i}{m_i}. \quad (1.14)$$

где  $\nu$  — вязкость,  $-\frac{1}{\rho_i} \nabla p_i$  описывает ускорение частиц из-за давления в жидкости,  $\nu \nabla^2 v_i$  — ускорение, обусловленное силами трения между частицами с разными скоростями,  $\frac{F_i}{m_i}$  — другие ускорения, например ускорение свободного падения.

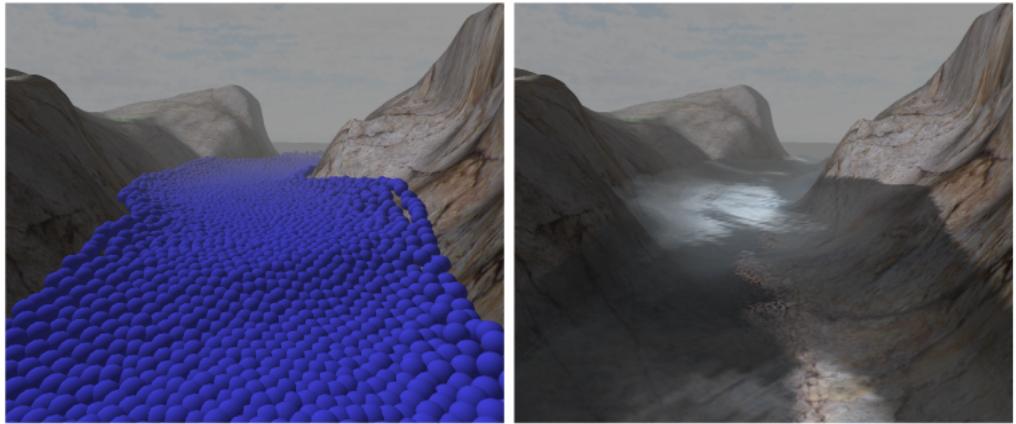


Рисунок 1.5 – Моделирование при помощи метода на основе частиц

Число частиц интерполируется при помощи сглаживающей функции, которая является кубическим сплайном [8].

Преимущество методов на основе частиц — возможность добавлять и удалять расчетные точки во время моделирования.

Недостатки методов на основе частиц:

- необходимость большого числа частиц для создания реалистичного изображения;
- создание неточных границ при контакте жидкости с твердым предметом или другой жидкостью.

### 1.4.3 Метод поля высоты

В методе поля высоты [9] рассматривают волновое уравнение 1.1. Волновая поверхность представляется в виде двумерной функции — поля высоты. Такое упрощение снижает вычислительные затраты. Поверхность дискретизируется в сетку, показанную на рисунке 1.6, и определяется приближенное решение в каждой точке сетки.

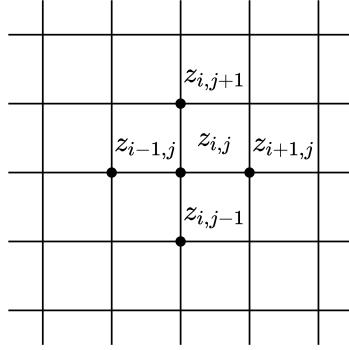


Рисунок 1.6 – Представление волновой поверхности в виде сетки

Значение функции  $h(x_{i,j})$  в точке сетки с координатам  $(i, j)$  обозначается  $z_{i,j}$ . Оператор Лапласа в волновом уравнении:

$$\Delta h(x_{i,j}) \approx \frac{z_{i+1,j} - 2z_{i,j} + z_{i-1,j}}{H^2} + \frac{z_{i,j+1} - 2z_{i,j} + z_{i,j-1}}{H^2} \quad (1.15)$$

$$\Delta h(x_{i,j}) = \frac{z_{i+1,j} + z_{i,j+1} - 4z_{i,j} + z_{i-1,j} + z_{i,j-1}}{H^2} \quad (1.16)$$

где  $H$  – расстояние между двумя точками сетки.

Дискретизация времени обозначается  $\Delta t$ , значение точки с координатами  $(i, j)$  в момент времени  $t = z_{i,j}^t$ . Тогда производная по времени в волновом уравнении:

$$\frac{\partial^2 h(x, t)}{\partial t^2} \approx \frac{z_{i,j}^{t+1} - 2z_{i,j}^t + z_{i,j}^{t-1}}{\Delta t^2} \quad (1.17)$$

С учетом 1.16 и 1.17 решение волнового уравнения:

$$\Delta h(x, t) - \frac{1}{v^2} \cdot \frac{\partial^2 h(x, t)}{\partial t^2} = 0 \quad (1.18)$$

$$\frac{z_{i+1,j} + z_{i,j+1} - 4z_{i,j} + z_{i-1,j} + z_{i,j-1}}{H^2} - \frac{1}{v^2} \frac{z_{i,j}^{t+1} - 2z_{i,j}^t + z_{i,j}^{t-1}}{\Delta t^2} \quad (1.19)$$

$$z_{i,j}^{t+1} = \frac{v^2 \Delta t^2}{H^2} \cdot (z_{i+1,j}^t + z_{i,j+1}^t + z_{i-1,j}^t + z_{i,j-1}^t) + (2 - 4 \frac{v^2 \Delta t^2}{H^2}) \cdot z_{i,j}^t - z_{i,j}^{t-1} \quad (1.20)$$

Преимущества метода поля высоты:

- повышение скорости моделирования;
- точность обработки взаимодействия с препятствиями.

Недостатком метода поля высоты является невозможность создания обрушающихся волн из-за одинаковой скорости распространения всех волн.

Комбинация методов поля высоты и методов, основанных на частицах, позволяет обходить недостатки отдельных [10] и создавать различные эффекты.

## Вывод

В таблице 1.1 представлено сравнение методов визуализации волн. Методы сравнивались по следующим критериям:

- контроль волнового фронта (K1);
- реалистичность (K2);
- затраты памяти (K3);
- точность обработки взаимодействия с предметами (K4).

Таблица 1.1 – Сравнение методов моделирования волн

Метод	K1	K2	K3	K4
Процедурный	высокий	низкая	низкие	низкая
На основе частиц	высокий	высокая	высокие	низкая
Поля высоты	высокий	высокая	низкие	высокая

Процедурные методы и методы на основе частиц имеют сложности при работе с твердыми телами. Метод поля высот корректно обрабатывает взаимодействие с объектом, причем с более высокой скоростью моделирования. Недостатки метода поля высот не имеют значения при решении выбранной задачи. Моделирование волн будет реализовано при помощи метода поля высот.

## 1.5 Методы рендеринга отображения

Рендеринг или отрисовка — процесс получения изображения по математической модели с помощью компьютерной программы.

Для моделирования волн был выбран метод поля высоты, в котором волновая поверхность представляется при помощи сетки. Высокая реалистичность изображения обеспечивается большим числом точек сетки. Для рендеринга поверхности воды не подходит стандартная графическая библиотека, так как в случае ее использования отрисовка займет длительное время. Для ускорения отрисовки необходимо выбрать API, которое позволяет использовать графический ускоритель для рендеринга изображения. Основными API являются DirectX, Vulkan и OpenGL.

### 1.5.1 DirectX

DirectX — это API компьютерной графики от Microsoft, используемый для платформы Windows [11]. Инструмент более близок к архитектуре современных графических процессоров. DirectX фокусируется на рендеринге в реальном времени, поэтому он предназначен для разработчиков игр и систем автоматизированного проектирования.

Его преимущества:

- быстрый рендеринг изображения;
- высокое качество изображения.

Недостатки API:

- отсутствие поддержки старыми видеокартами;
- возможность использования только на операционных системах Windows.

### 1.5.2 Vulkan

Vulkan — кроссплатформенный API для 2D- и 3D-графики, представленный Khronos Group [12].

Инструмент обеспечивает высокую производительность при отображении в реальном времени различных приложений с 3D-графикой, таких как игры или интерактивные книги, на всех платформах.

Его преимущества:

- открытый исходный код;
- высокая производительность работы;
- кроссплатформенность.

Недостатки API:

- прямая работа с GPU;
- отсутствие библиотек для некоторых языков программирования.

### 1.5.3 OpenGL

OpenGL — API, предоставляющий большой набор функций, которые можно использовать для управления графикой и изображениями [11]. OpenGL является спецификацией, разработанной и поддерживаемой Khronos Group.

Спецификация OpenGL определяет, каким должен быть результат работы каждой функции, и каким образом функция должна выполняться. Реализация спецификации зависит от разработчика.

Графическая обработка осуществляется при помощи графического конвейера. Работу конвейера можно разделить на несколько этапов, где каждый следующий этап в качестве входных данных использует выходные данные

из предыдущего этапа. Все эти этапы имеют только одну конкретную функцию и могут легко выполняться параллельно. На каждом этапе запускается программа на графическом процессоре, называемая шейдером. Некоторые шейдеры могут быть изменены разработчиком, что дает возможность более точного контроля над работой конвейера.

Этапы работы графического конвейера:

- вершинный шейдер обрабатывает 3D-координаты вершины;
- все вершины из вершинного шейдера собираются в заданной для примитива форме;
- геометрический шейдер принимает в качестве входных данных набор вершин, образующих примитив, и добавляет дополнительные вершины для формирования других примитивов.
- выходные данные геометрического шейдера передаются на этап растеризации, на котором результирующие примитивы сопоставляются с соответствующими пикселями на экране;
- фрагментный шейдер вычисляет конечный цвет пикселя.

Преимущества OpenGL:

- высокая производительность работы;
- возможность изменения параметров выводимых объектов;
- кроссплатформенность;
- наличие библиотек для работы с различными языками программирования.

Его недостатки:

- сложность работы с новыми возможностями GPU;
- необходимость отдельных инструментов для работы с устройствами ввода.

## Вывод

В таблице 1.2 представлено сравнение методов рендеринга изображения. Методы сравнивались по следующим критериям:

- наличие библиотеки для работы с языком программирования Python (K1);
- открытый код (K2);
- кроссплатформенность (K3).

Таблица 1.2 – Сравнение методов рендеринга изображения

Метод	K1	K2	K3
DirectX	+	-	-
Vulkan	-	+	+
OpenGL	+	+	+

Из результатов сравнения методов рендеринга изображений можно сделать вывод о том, что для рендеринга модели волнового процесса будет использоваться OpenGL.

## 1.6 Существующие программные обеспечения

Одним из популярных программных обеспечений для моделирования жидкости, в том числе и волн, является расширение Blender [13] – Mantaflow [14]. В этом программном продукте существует возможность реализовать моделирование линейных волн, гибкие системы частиц, различные эффекты для жидкости, например, поверхностную турбулентность и вейвлет (небольшую рябь). На рисунке 1.7 показано моделирование волн в Mantaflow.

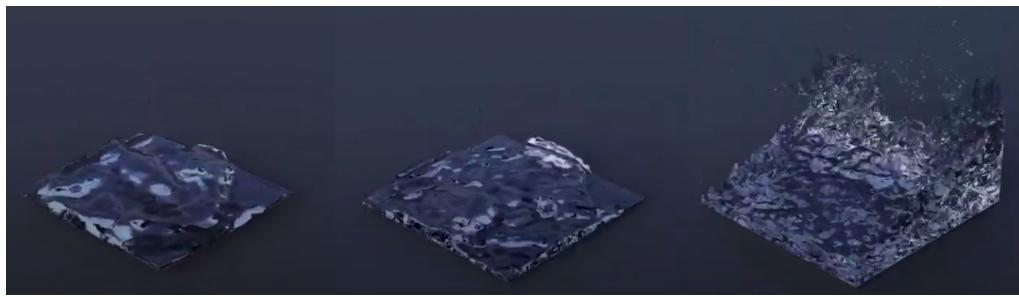


Рисунок 1.7 – Моделирование волн в среде Mantaflow в Blender

В коммерческих и научных проектах используется FLOW-3D [15], который представляет набор инструментов для моделирования жидкостей, с целью исследования динамики их движения. Данный пакет позволяет моделировать линейные и нелинейные распространяющиеся поверхностные волны. На рисунке 1.8 показаны линейный волны, визуализированные в FLOW-3D.

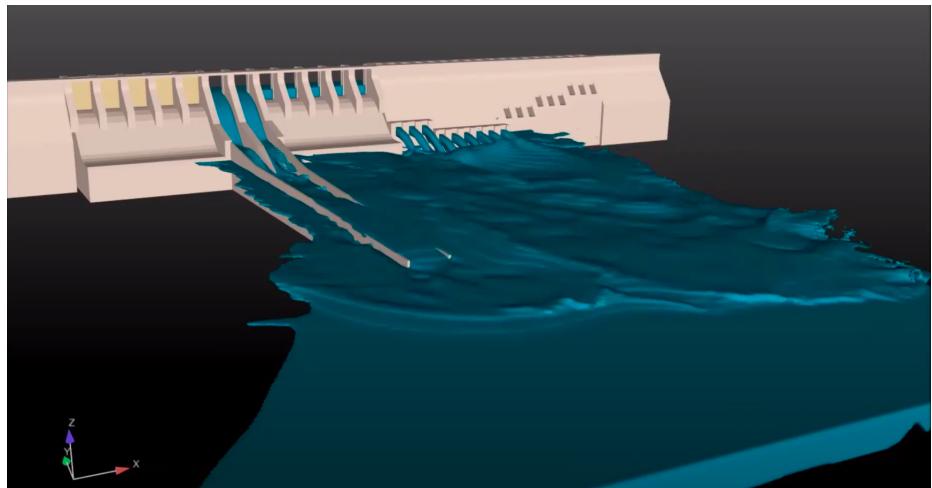


Рисунок 1.8 – Моделирование волн при помощи пакета FLOW-3D

## Вывод

В ходе изучения волнового процесса были рассмотрены модели волны и предмета. Было проведено сравнение методов и алгоритмов, моделирующих волны. В качестве метода визуализации был выбран метод поля высоты, в качестве метода рендеринга изображения — OpenGL.

## **2 Конструкторская часть**

В данном разделе выделяются требования к программному обеспечению, приводятся схемы алгоритмов, описываются используемые типы и структуры данных, и показывается структура программного обеспечения.

### **2.1 Требования к программному обеспечению**

Программное обеспечение должно обеспечить работу следующих функций:

- загрузка модели предмета;
- изменение скорости предмета в интерактивном режиме;
- вращение, перемещение и масштабирование модели.

Требования, которые предъявляются к программе:

- время отклика программы должно быть менее 1 секунды для корректной работы в интерактивном режиме;
- программа должна корректно реагировать на любые действия пользователя.

### **2.2 Разработка алгоритмов**

На рисунке 2.3 представлена схема алгоритма образования волновой поверхности, на рисунке 2.2 — её отрисовка. На рисунке 2.3 показана схема алгоритма перемещения предмета, на рисунке 2.4 — его отрисовка.

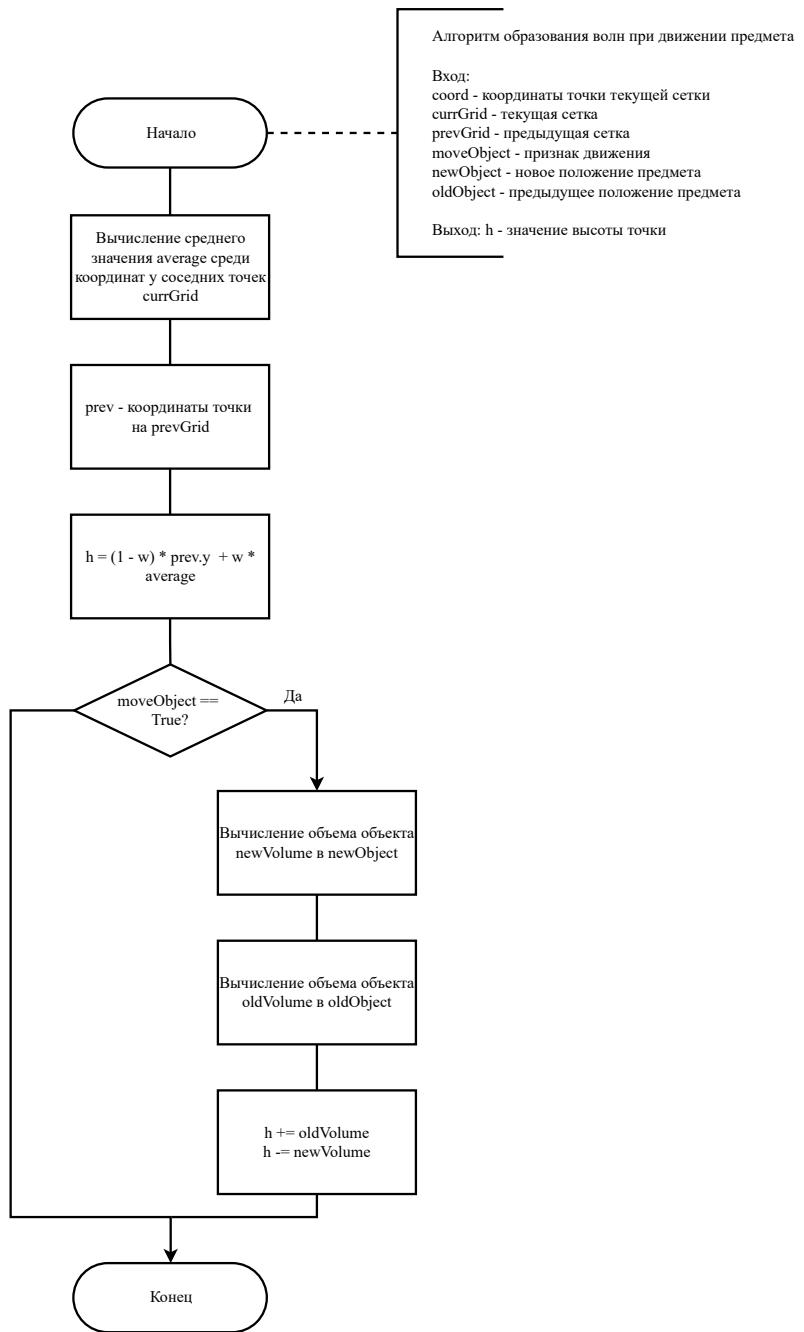


Рисунок 2.1 – Схема алгоритма образования волн при движении предмета

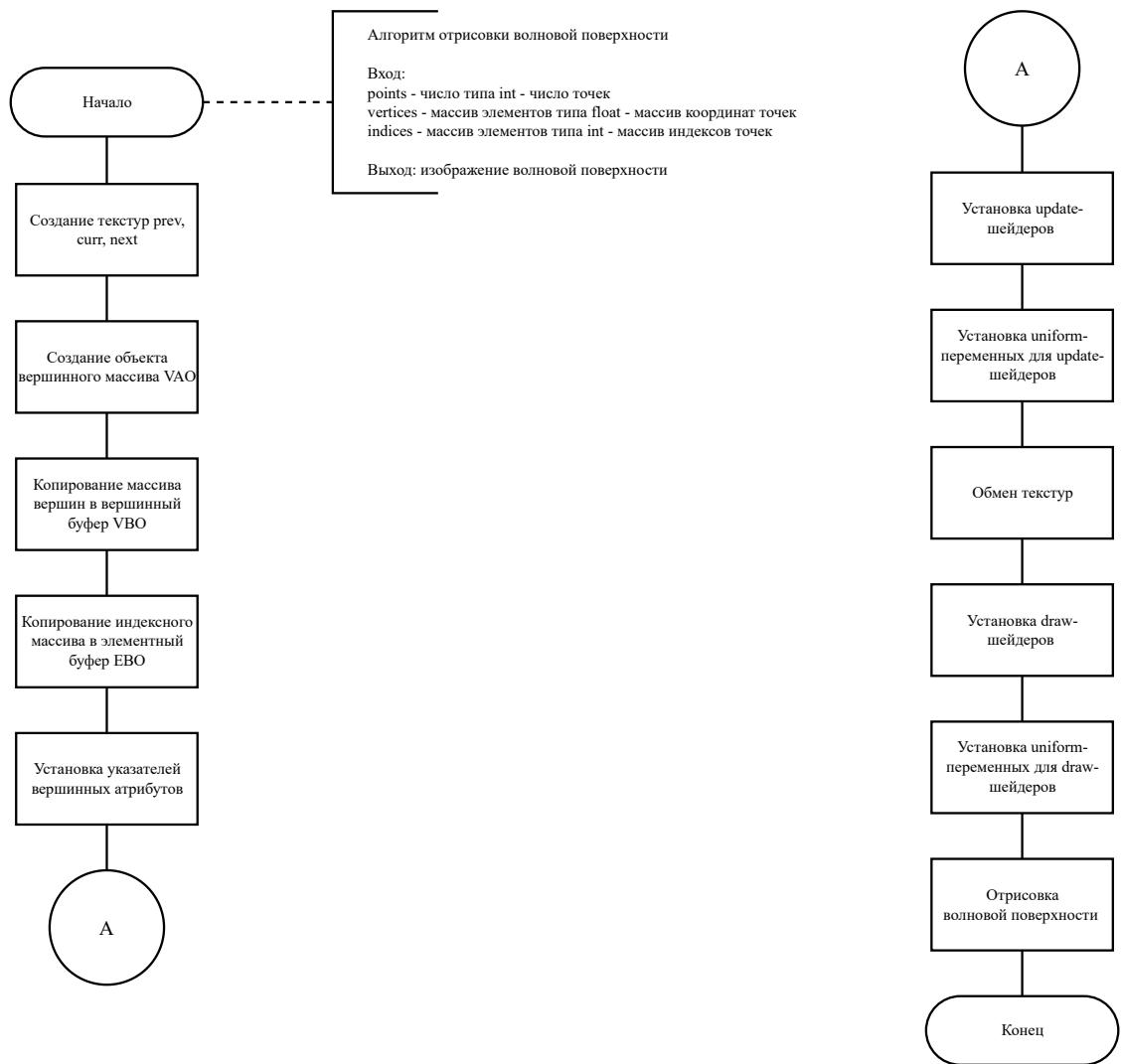


Рисунок 2.2 – Схема алгоритма отрисовки волнистой поверхности

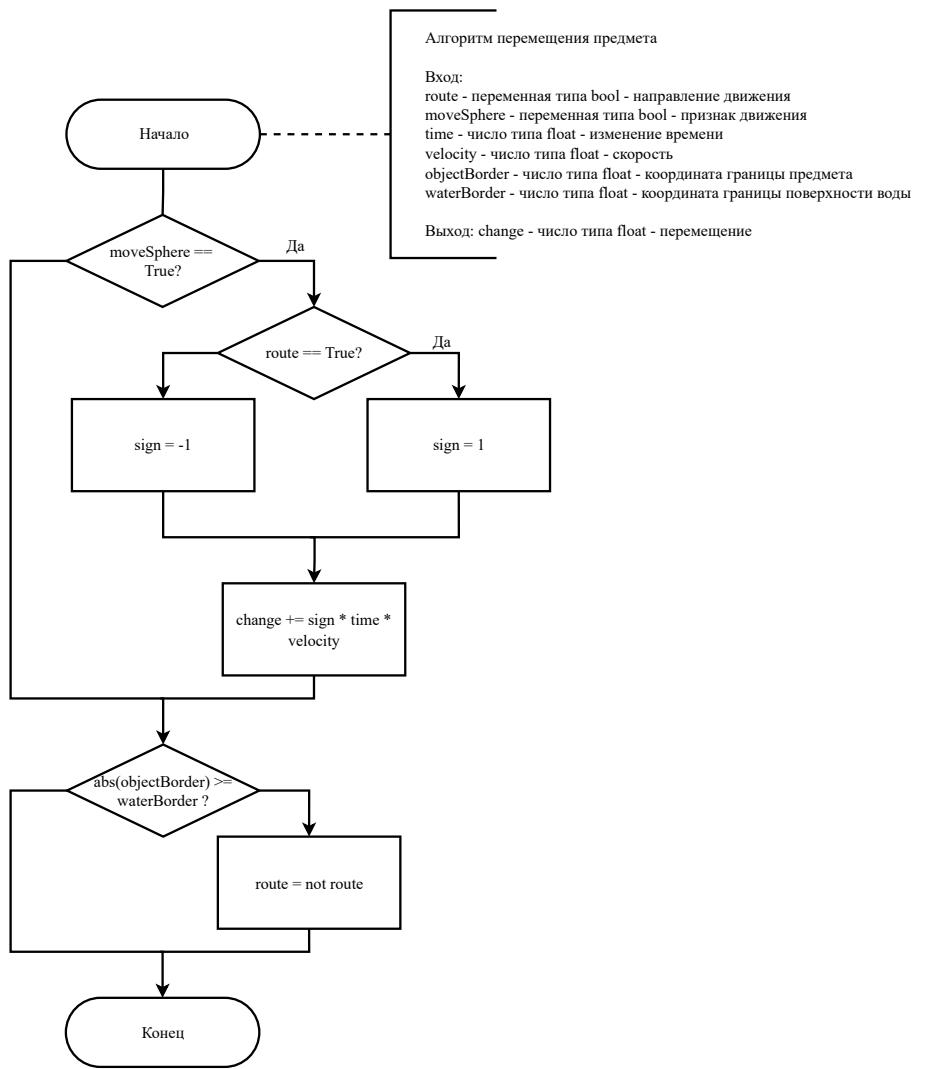


Рисунок 2.3 – Схема алгоритма перемещения предмета

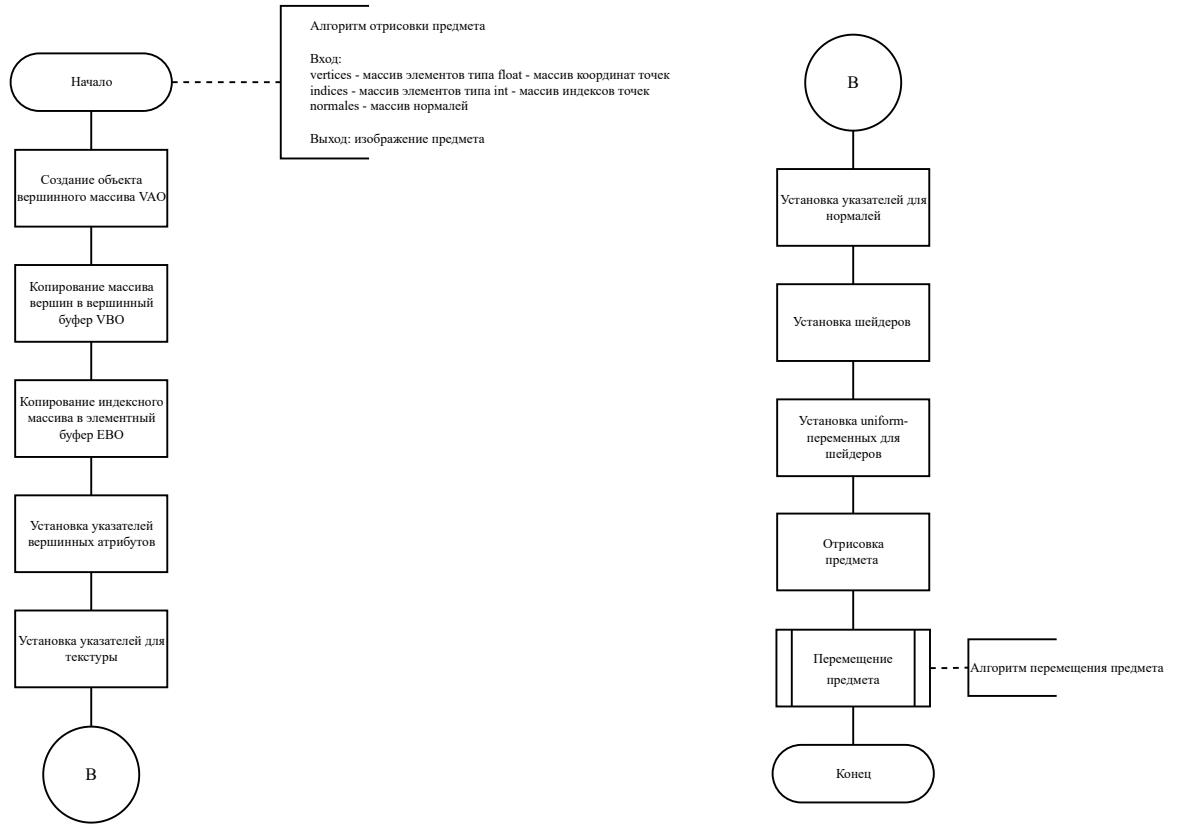


Рисунок 2.4 – Схема алгоритма отрисовки предмета

## 2.3 Описание используемых типов и структур данных

Для реализации поверхности воды используются следующие типы и структуры данных:

- для координат точек сетки — трехмерный массив элементов типа float;
- для индексов точек сетки — трехмерный массив элементов типа int;
- для числа точек сетки — тип данных int.

Для создания предмета используются следующие типы и структуры данных:

- для координат точек предмета — трехмерный массив элементов типа float;
- для индексов точек предмета — трехмерный массив элементов типа int.

Скорость движения задается числом типа int.

## 2.4 Структура программного обеспечения

На рисунке 2.5 представлена структура классов, реализованных в программном обеспечении.

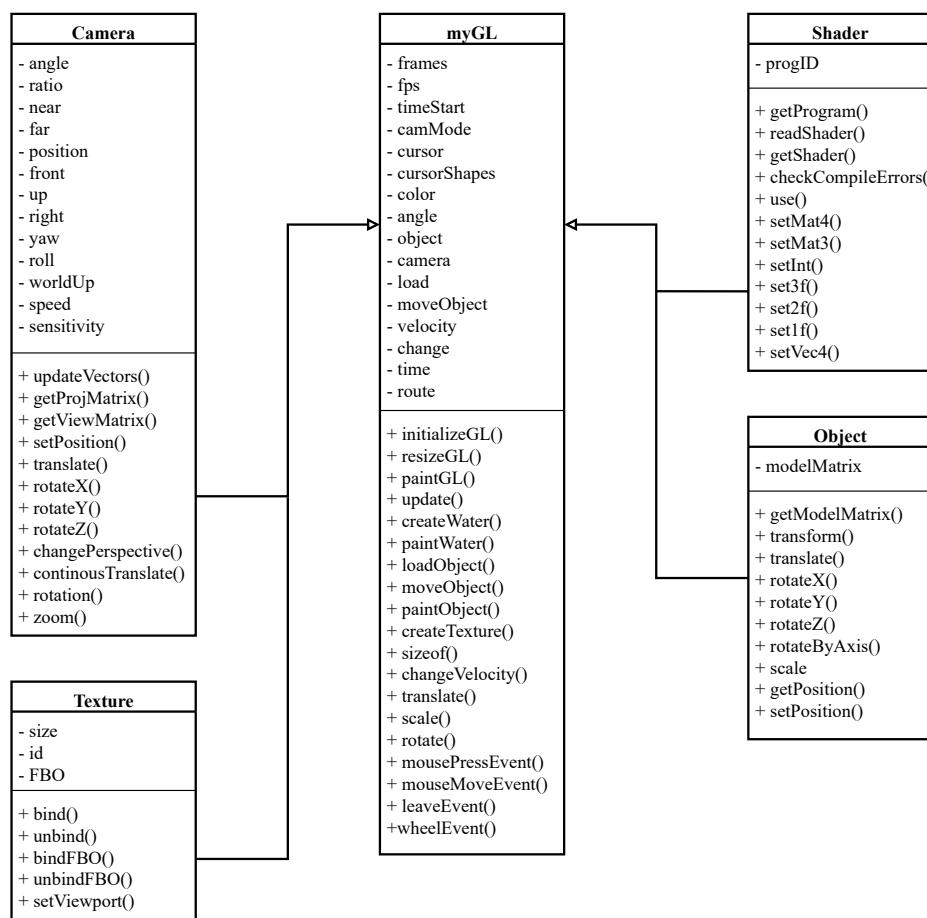


Рисунок 2.5 – Структура реализуемых классов

Описание реализуемых классов:

- myGL — класс, в котором происходит отрисовка всех объектов сцены;
- Camera — класс, предоставляющий функции работы с камерой;
- Texture — класс для работы с текстурой и фрагментным буфером;
- Shader — класс для загрузки шейдеров и установки uniform-переменных;
- Object — класс, предоставляющий функции преобразования объектов сцены.

## Вывод

Были выделены требования к программному продукту, рассмотрены схемы алгоритмов моделирования и типы и структуры данных, при помощи которых реализуются методы визуализации, показана структура программного обеспечения.

### 3 Технологическая часть

В данном разделе описываются средства разработки программного обеспечения и детали реализации.

#### 3.1 Средства реализации

Разработка программного обеспечения выполнялась при помощи языка программирования Python [16]. Выбор языка программирования обусловлен его преимуществами:

- наличие библиотек необходимых для реализации всех требуемых функций;
- возможность использования объектно-ориентированного подхода.

Для написания вершинных и фрагментных шейдеров графического контейнера OpenGL использовался язык GLSL [17]. Данный язык шейдеров был выбран из-за следующих характеристик:

- контроль графического потока без использования машинно-зависимых языков;
- высокая производительность;
- наличие дополнительных функций и типов данных для работы с матрицами и векторами.

Пользовательский интерфейс программного обеспечения создан при помощи Qt Designer [18], так как инструмент позволяет быстро разработать интерфейс.

Для ускорения процесса создания программного обеспечения в качестве среды разработки выбран текстовый редактор Visual Studio Code [19].

## 3.2 Реализация алгоритмов

В листинге 3.1 представлен алгоритм образования волн при движении твердого тела, в листинге 3.2 — алгоритм перемещения предмета.

Листинг 3.1 – Алгоритм образования волн при движении предмета

```
1 in vec2 coord;  
2  
3 const float w = 1.985;  
4  
5 uniform sampler2D currTexture;  
6 uniform sampler2D prevTexture;  
7 uniform bool moveSphere;  
8 uniform float sphereRadius;  
9 uniform vec3 nowCenter;  
10 uniform vec3 oldCenter;  
11 uniform float step;  
12  
13 float volumelnSphere(vec3 sphereCenter)  
14 {  
15     vec3 toCenter = vec3(coord.x * 2.0 - 1.0, 0.0, coord.y * 2.0 -  
1.0) - sphereCenter;  
16     float t = length(toCenter) / sphereRadius;  
17     float dy = exp(-pow(t * 1.5, 6.0));  
18     float ymin = min(0.0, sphereCenter.y - dy);  
19     float ymax = min(max(0.0, sphereCenter.y + dy), ymin + 2.0 *  
dy);  
20  
21     return (ymax - ymin) * 0.1;  
22 }  
23  
24 void main()  
25 {  
26     vec2 dx = vec2(step, 0.0);  
27     vec2 dy = vec2(0.0, step);  
28  
29     float average = (texture(currTexture, coord + dy).y +  
30                         texture(currTexture, coord - dy).y +  
31                         texture(currTexture, coord + dx).y +  
32                         texture(currTexture, coord - dx).y) * 0.25;
```

```

33
34     float prev = texture(prevTexture, coord).y;
35     float h = (1.0 - w) * prev + w * average;
36
37     if (moveSphere)
38     {
39         h += volumeInSphere(oldCenter);
40         h -= volumeInSphere(nowCenter);
41     }
42
43     gl_FragColor = vec4(0.0, h, 0.0, 1.0);
44 }
```

Листинг 3.2 – Алгоритм перемещения предмета

```

1 def moveObject(self):
2     if self.moveSphere:
3         if self.route == POSITIVE:
4             sign = 1
5         else:
6             sign = -1
7
8         self.change += sign * self.time * self.velocity
9
10    if START_SPHERE_CENTER.x + self.change + self.sphereRadius >=
11        POSITIVE_BORDER:
12        self.route = NEGATIVE
13    elif START_SPHERE_CENTER.x + self.change - self.sphereRadius <=
14        NEGATIVE_BORDER:
15        self.route = POSITIVE
```

## Вывод

Были описаны инструменты разработки программного продукта, и представлены алгоритмы моделирования волнового процесса в результате движения объекта.

# 4 Исследовательская часть

В данном разделе приводятся примеры работы программного обеспечения. Кроме того, проводится эксперимент, в котором анализируется производительность программного обеспечения.

## 4.1 Примеры работы программного обеспечения

На рисунке 4.1 показана волновая поверхность в положении равновесия, на рисунках 4.2 и 4.3 представлено образование волн в результате движения сферы, а на рисунке 4.4 — окно панели управления.

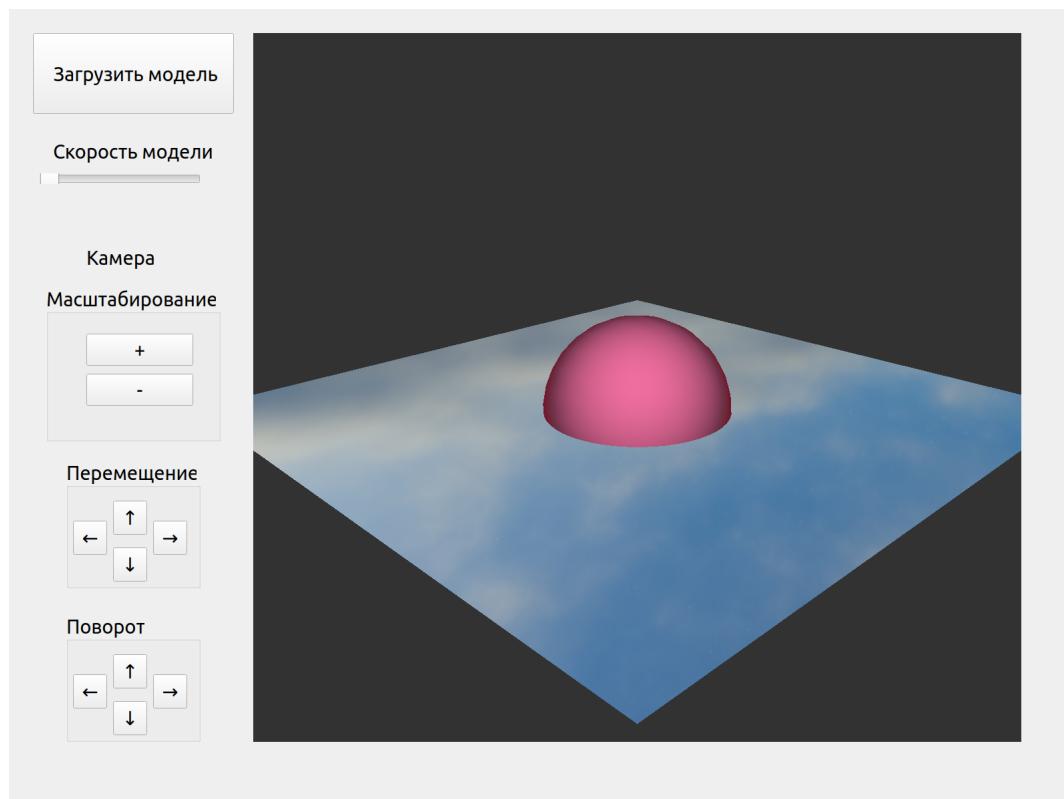


Рисунок 4.1 – Пример волновой поверхности в положении равновесия

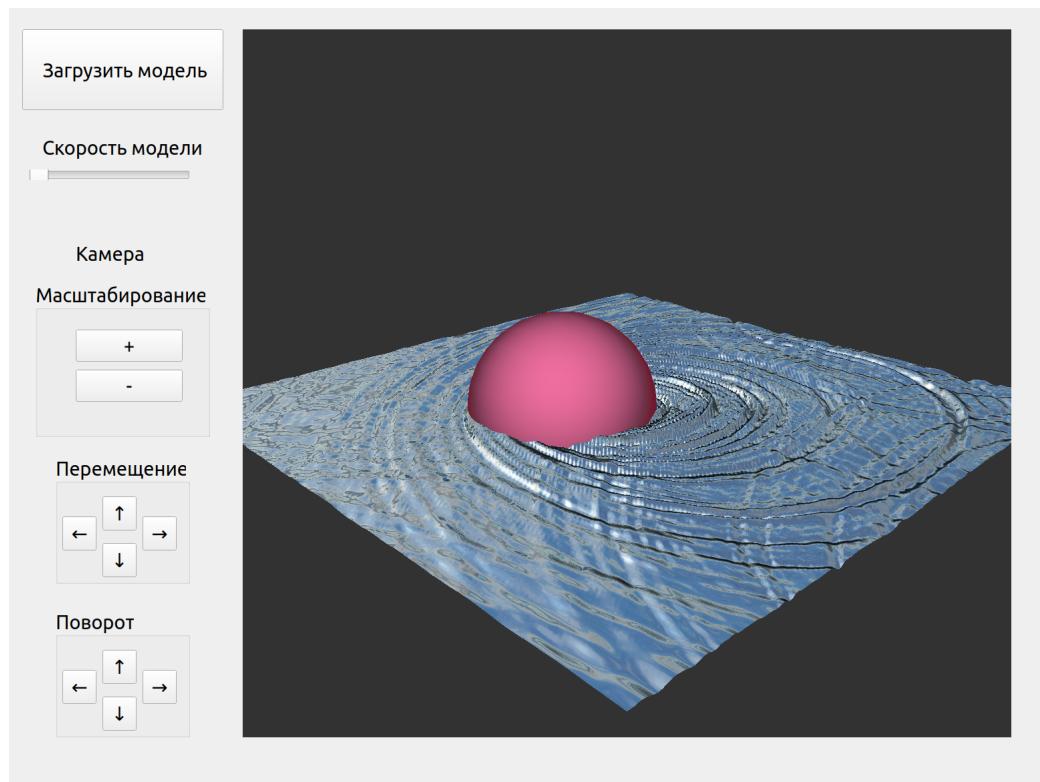


Рисунок 4.2 – Пример образования волн при движении сферы (вид 1)

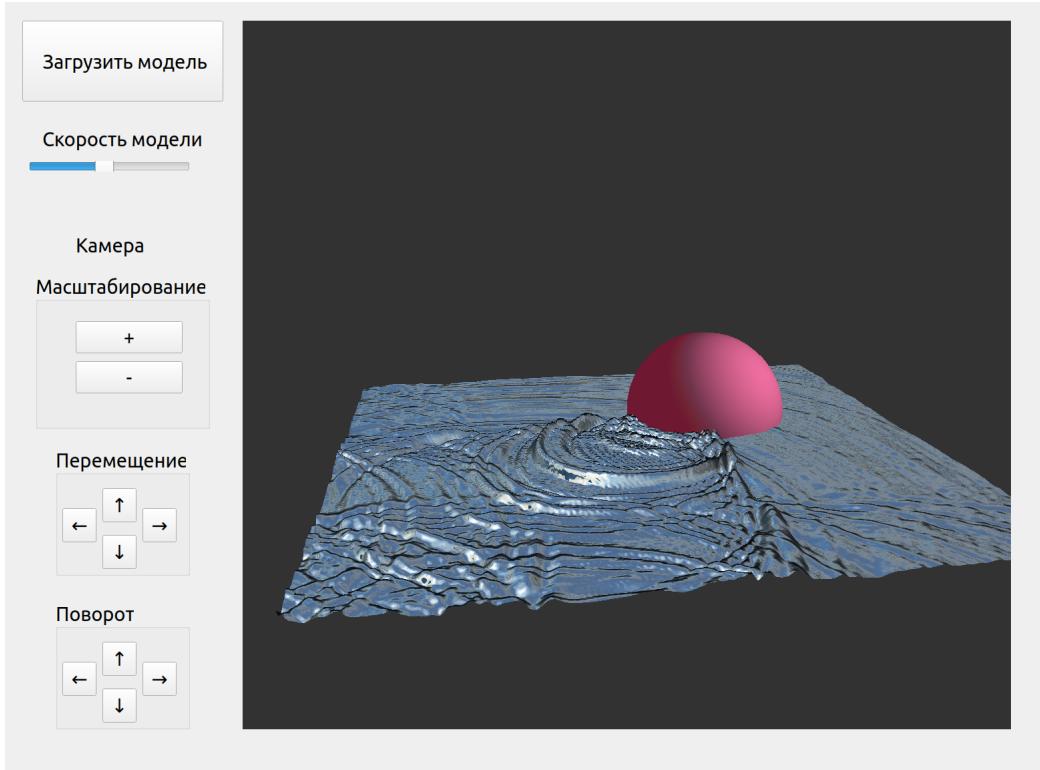


Рисунок 4.3 – Пример образования волн при движении сферы (вид 2)

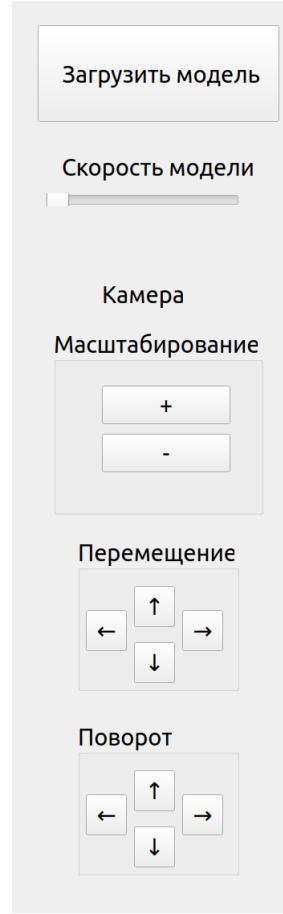


Рисунок 4.4 – Окно панели управления

## 4.2 Постановка эксперимента

Метод моделирования оценивают по его скорости выполнения. Количественную характеристику скорости выполнения функций программного обеспечения называют его производительностью.

### 4.2.1 Цель эксперимента

Целью эксперимента является проведение замеров производительности при создании сцен различной нагруженности. Так как при моделировании волн методом поля высоты поверхность воды представляется сеткой, то параметр, изменяющий нагрузку — число точек сетки.

Мерой измерения производительности является число кадров в секунду,

соответствующее параметру нагрузки.

При проведении эксперимента скорость движения предмета должна быть одинаковой для каждого числа точек сетки.

### 4.2.2 Технические характеристики

Тестирование проводилось на устройстве со следующими техническими характеристиками:

- операционная система: Ubuntu 20.04.1 Linux x86\_64 [20];
- память : 8 GiB;
- процессор: AMD® Ryzen™ 3 3200u @ 2.6 GHz [21];
- видеокарта: Radeon™ Vega 3 Graphics [21].

Тестирование проводилось на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, а также непосредственно системой тестирования.

### 4.2.3 Результаты эксперимента

Результаты эксперимента приведены в таблице 4.1. На рисунке 4.5 представлен график зависимости производительности программного обеспечения от числа точек сетки.

Таблица 4.1 – Зависимость производительности программного обеспечения от числа точек сетки

Число точек сетки, шт.	Производительность, к/с
800	78
900	74
1000	61
1100	48
1200	41
1300	35
1400	32
1500	27
1600	23
1700	21
1800	19
1900	17
2000	15

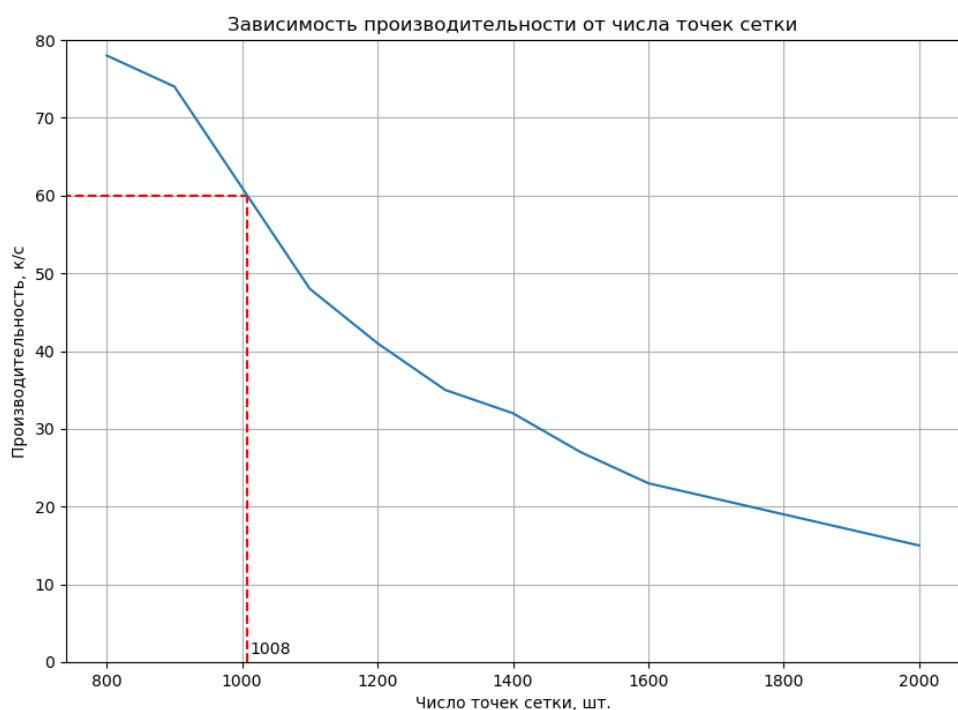


Рисунок 4.5 – График зависимости производительности программного обеспечения от числа точек сетки

В результате эксперимента получено, что при линейном увеличении числа точек сетки водной поверхности число кадров в секунду экспоненциально уменьшается.

Для комфортной работы кадровая частота должна быть 60 кадров в секунду [22]. На графике зависимости производительности от числа точек сетки видно, что оптимальное число точек сетки равняется 1008 штукам.

## Вывод

Были приведены примеры работы программного обеспечения. Был проведен анализ производительности программного продукта, в результате которого было выяснено, что при линейном увеличении нагрузки на программное обеспечение, его производительность экспоненциально уменьшается.

# Заключение

В ходе выполнения курсовой работы было разработано программное обеспечение, которое предоставляет возможность визуализации волн, образованных при взаимодействии поверхности воды с движущимся твердым телом. В процессе выполнения данной работы были выполнены следующие задачи:

- изучен волновой процесс;
- формально описана структура системы, состоящая из поверхности воды и источника волн;
- проанализированы методы и алгоритмы, моделирующие волновой процесс;
- выбраны алгоритм и структуры данных для визуализации описанной выше системы;
- реализован выбранный алгоритм моделирования;
- проведен анализ производительности программного обеспечения.

В ходе исследования было выявлено, что производительность программного обеспечения снижается экспоненциально при линейном росте числа точек сетки, представляющей волновую поверхность.

# Список литературы

- [1] Irving Geoffrey, Guendelman Eran. Efficient Simulation of Large Bodies of Water by Coupling Two and Three Dimensional Techniques. Boston, Massachusetts, 2006. c. 805–811.
- [2] Jeschke Stefan, Wojtan Chris. Water Wave Packets // ACM Trans. Graph. New York, NY, USA, 2017. 07. C. 4–16.
- [3] Canabal, Miraut David, Thuerey Nils. Dispersion Kernels for Water Wave Simulation // ACM Trans. Graph. New York, NY, USA, 2016. 11. C. 10–15.
- [4] G. Airy. Tides and Waves. 1849.
- [5] Fournier Alain, Reeves William T. A Simple Model of Ocean Waves // SIGGRAPH Comput. Graph. New York, NY, USA, 1986. 08. c. 75–84.
- [6] Emmanuelle Darles, Crespin Benoît. A Survey of Ocean Simulation and Rendering Techniques in Computer Graphics // Computer Graphics Forum - CGF. New York, NY, USA, 2011. 09. C. 1–30.
- [7] Ihmsen Markus, Orthmann Jens. SPH Fluids in Computer Graphics - Eurographics State-of-the-art report // VRIPHYS 2011 - 8th Workshop on Virtual Reality Interactions and Physical Simulations. Darmstadt, 2014. 01. C. 1–15.
- [8] Premoze Simon, Tasdizen Tolga. Particle-Based Simulation of Fluids // Computer Graphics Forum. New York, NY, USA, 2003. 06. C. 1–22.
- [9] Kass Michael, Miller Gavin. Rapid, Stable Fluid Dynamics for Computer Graphics // SIGGRAPH Comput. Graph. New York, NY, USA, 1990. 09. T. 24, № 4. c. 49–57.
- [10] Solenthaler Barbara, Bucher Peter. SPH Based Shallow Water Simulation. // VRIPHYS 2011 - 8th Workshop on Virtual Reality Interactions and Physical Simulations. Darmstadt, 2011. 01. C. 39–46.
- [11] Hillesland Karl. OpenGL and DirectX. Hong Kong, Hong Kong, 2013. C. 79–85.

- [12] Bailey Mike. Introduction to the Vulkan® Computer Graphics API. Brisbane, Queensland, Australia, 2019. C. 155–160.
- [13] Blender. Режим доступа: <https://www.blender.org/> [Электронный ресурс].
- [14] Mantaflow. Режим доступа: <http://mantaflow.com/> [Электронный ресурс].
- [15] FLOW-3D. Режим доступа: <https://www.flow3d.com/modeling-capabilities/waves/> [Электронный ресурс].
- [16] Welcome to Python [Электронный ресурс]. Режим доступа: <https://www.python.org>.
- [17] OpenGL Shading Language [Электронный ресурс]. Режим доступа: [https://www.khronos.org/opengl/wiki/OpenGL\\_Shading\\_Language](https://www.khronos.org/opengl/wiki/OpenGL_Shading_Language).
- [18] Qt Designer Manual [Электронный ресурс]. Режим доступа: <https://doc.qt.io/qt-5/qtdesigner-manual.html>.
- [19] Visual Studio Code [Электронный ресурс]. Режим доступа: <https://code.visualstudio.com/docs>.
- [20] Ubuntu 20.04 LTS (Focal Fossa) Beta [Электронный ресурс]. Режим доступа: <http://old-releases.ubuntu.com/releases/20.04.1/>.
- [21] Мобильный процессор AMD Ryzen™ 3 3200U с графикой Radeon™ Vega 3 [Электронный ресурс]. Режим доступа: <https://www.amd.com/ru/products/apu/amd-ryzen-3-3200u>.
- [22] Debattista Kurt, Bugeja Keith. Frame Rate vs Resolution: A Subjective Evaluation of Spatiotemporal Perceived Quality Under Varying Computational Budgets // Computer Graphics Forum. Chichester, 2017. 09. C. 1–37.