**«Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)»**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.03.04 – Программная инженерия | |
| **Профиль** | Программная инженерия | |
| **Факультет** | ФКТИ | |
| **Кафедра** | МО ЭВМ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | А.А. Лисс |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

БАКАЛАВРА

Тема: создание визуально сложных эффектов с помощью шейдеров в среде разработки Unity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | П.А. Королева |
|  |  | *подпись* |  |  |
| Руководитель |  |  |  | Т.В. Герасимова |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  |

Санкт-Петербург

2025

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ А.А. Лисс |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | Королева П.А. | | | |  | Группа | 1303 | |
| Тема работы: Создание визуально сложных эффектов с помощью шейдеров в среде разработки Unity. | | | | | | | | |
| Место выполнения ВКР: место выполнения ВКР | | | | | | | | |
| Исходные данные (технические требования):  кратко указываются основные требования к ВКР | | | | | | | | |
| Содержание ВКР:  Кратко перечисляются основные разделы ВКР | | | | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал, иные отчетные материалы | | | | | | | | |
| Дополнительные разделы: указывается наименование дополнительного раздела | | | | | | | | |
|  | | | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | | | |
| «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. | | | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. | | | | | |
|  | | |  | | | | | |
| Студент | |  | | П.А. Королева | | | |  |
| Руководитель | |  | | И.И. Иванов | | | |  |
| *(Уч. степень, уч. звание)* | |  | |  | | | |  |
| Консультант | |  | | И.И. Иванов | | | |  |
| *(Уч. степень, уч. звание)* | |  | |  | | | |  |

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ А.А. Лисс |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Королева П.А. |  | Группа | 1303 |
| Тема работы: Наименование темы | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор литературы по теме работы | 00.00 – 00.00 |
| 2 | Наименование раздела | 00.00 – 00.00 |
| 3 | Наименование раздела | 00.00 – 00.00 |
| 4 | Наименование раздела | 00.00 – 00.00 |
| 5 | Оформление пояснительной записки | 00.00 – 00.00 |
| 6 | Оформление иллюстративного материала | 00.00 – 00.00 |

|  |  |  |
| --- | --- | --- |
| Студент |  | П.А. Королева |
| Руководитель |  | Т.В. Герасимова |
| *(Уч. степень, уч. звание)* |  |  |

**РЕФЕРАТ**

Пояснительная записка 65 стр., 19 рис., 16 табл., 24 ист.

КЛЮЧЕВЫЕ СЛОВА, ВАШЕГО, ДИПЛОМА.

**Объектом исследования** являются шейдеры.

**Предметом исследования** является

**Цель работы:** разработка шейдеров для визуализации медицинских данных для обучения врачей.

Аннотация работы порядка 300 символов.

**ABSTRACT**

Перевод аннотации на английский язык.

**СОДЕРЖАНИЕ**

(разделы Задание, Календарный план, Реферат, Abstract в Содержании не указываются).

[ВВЕДЕНИЕ 9](#_Toc193903021)

[1 Обзор литературы 10](#_Toc193903022)

[1.1 Обзор программного обеспечения 10](#_Toc193903023)

[1.2 Обзор существующих решений 15](#_Toc193903024)

[2 Реализация шейдера 19](#_Toc193903025)

[2.1 Теоретические положения 19](#_Toc193903026)

[2.2 Разработка функции движения вершин 19](#_Toc193903027)

[2.3 Написание кода шейдера 26](#_Toc193903028)

[3 Разработка приложения 27](#_Toc193903029)

[4 Тестирование 27](#_Toc193903030)

[ЗАКЛЮЧЕНИЕ 28](#_Toc193903031)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc193903032)

[ПРИЛОЖЕНИЕ А 32](#_Toc193903033)

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

СPU – центральный процессор;

GPU – графический процессор;

Графический API – набор программных инструментов для работы с графическим оборудованием компьютера;

Движок – программное обеспечение, предоставляющее пользователям набор функций и инструментов для ускорения разработки;

Шейдер – программа, запускаемая на графическом процессоре;

ЭКГ – электрокардиография;

## ВВЕДЕНИЕ

Основной текст разделов, параметры — отступ от начала каждого абзаца 1,25 см, 14 шрифт TimesNewRoman, полуторный интервал, выравнивание по ширине, интервал перед и после абзаца нулевой. Эти настройки уже включены в стиль «Основной текст» данного документа.

Во Введении необходимо последовательно (и в разных абзацах) описать Актуальность, Цель работы, Задачи работы, Объект исследования, Предмет исследования, Практическую ценность работы, перечислить публикации по теме работы (при наличии).

## 1 Обзор литературы

### 1.1 Обзор программного обеспечения

#### 1.1.1 Принципы отбора аналогов

В роли сравнительных аналогов были рассмотрены графические движки, поддерживающие 3D графику.

Основным критерием служила возможность работы с шейдерами. Подбор аналогов осуществлялся с использованием ресурса Cyberleninka. Для поиска применялись следующие запросы: «движки шейдеры медицина», «движки шейдеры», «шейдеры медицина».

#### 1.1.2 Аналоги

**Unity** [5, 6] - это популярный мультиплатформенный игровой движок для разработки 2D и 3D приложений и игр.

Он поддерживает работу с шейдерами, предоставляя удобные инструменты для этого - шейдер-граф с встроенными функциями для работы с шейдерами, позволяющими экономить время на написании кода. Также, движок позволяет писать код на языке GLSL, расширяя функциональность шейдер-графа.

**Unreal Engine** [6, 7, 8] - это популярный многофункциональный игровой движок, поддерживающий высокое качество графики и обладающий продвинутыми возможностями. Он использует систему шейдеров, основанную на Material Editor, где шейдеры создаются посредством GUI, также Unreal Engine предоставляет возможность для написания шейдеров посредством кода на языке HLSL для более детальной настройки шейдера.

**CryEngine** [6, 9, 10] - это игровой 3D движок. Шейдеры в нем могут объединяться, накладываться или создавать анимированные текстуры. CryEngine 3 использует унифицированную шейдерную архитектуру, при которой блоки GPU могут выполнять шейдеры любого типа, при этом вычислительная нагрузка равномерно распределяется между блоками GPU, что позволяет достичь высокой производительности. Однако, CryEngine имеет высокий порог входа, из-за чего новым пользователям требуется время на освоение движка.

**Godot Engine** [11, 12] - это открытый кроссплатформенный 2D и 3D игровой движок. Для написания шейдеров используются языки GDScript или Visual Shader Language. Также движок предлагает графовое представление для разработки шейдеров.

**Ogre3D** [13, 14] - это свободный графический движок для рендеринга трехмерной графики с открытым исходным кодом. Ogre3D поддерживает языки шейдеров GLSL для OpenGL и HLSL для Direct3D. Шейдеры можно настраивать в системе материалов, позволяя разработчику задавать параметры текстур, освещения и эффектов, не углубляясь в код.

#### 1.1.3 Критерии сравнения аналогов

Для сравнения были выбраны следующие критерии:

• Удобство настройки шейдеров

• Графический API

• Системные требования

Ниже представлено более подробное описание каждого критерия.

#### Удобство настройки шейдера

Данный критерий оценивает удобство настройки шейдеров и скорость разработки, в зависимости от инструментов, которые предлагает движок. В разработке шейдеров для визуализации медицинских данных могут потребоваться следующие инструменты: тесселяция, текстурирование, нормали, маски и текстурная генерация.

• Если движок не имеет встроенных инструментов и разработчику необходимо целиком реализовывать шейдер, движок получает низкую оценку по этому критерию.

• Если движок имеет часть инструментов, нужных для разработки данных шейдеров, он получает среднюю оценку.

• Если движок предлагает все нужные инструменты для разработки данных шейдеров, он получает высокую оценку.

**Графический API**

Этот критерий описывает, какие графические API поддерживает движок. Выбор графического API в соответствии с операционной системой и видеокартой позволит эффективно использовать ресурсы GPU и достичь наибольшей производительности. Необходимо выбрать такие графические API, которые поддерживаются большинством платформ, чтобы разработанные шейдеры могли эффективно работать на различных устройствах.

**Системные требования**

Данный критерий оценивает минимальные системные требования для разработки.

#### 1.1.4 Сравнение аналогов

Проведенное сравнение представлено в таблице 1.1.

Таблица 1.1 – Сравнение движков

| Аналог | Критерий сравнения | | |
| --- | --- | --- | --- |
| Удобство настройки шейдеров | Графический API | Системные требования |
| Unity | Высокое | OpenGL,  DirectX,  Vulkan | Низкие  (4 GB RAM,  Storage 4 GB) |
| Unreal  Engine | Высокое | OpenGL,  DirectX,  Vulkan | Высокие  (8 GB RAM, Storage 40 GB) |
| CryEngine | Высокое | OpenGL,  DirectX,  Vulkan | Средние  (4 GB RAM, Storage 8 GB) |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| Аналог | Критерий сравнения | | |
| Удобство настройки шейдеров | Графический API | Системные требования |
| Godot Engine | Среднее | DirectX,  Vulkan,  Metal,  WebGPU | Низкие  (4 GB RAM, Storage 1 GB) |
| Ogre3D | Низкое | OpenGL,  DirectX | Низкие  (2 GB RAM, Storage 1 GB) |

Среди графических API, поддерживаемых рассмотренными движками, представлены:

• OpenGL [15] - кроссплатформенное графическое API. Считается устаревшим, так как перестал обновляться, последняя версия вышла в 2022 году.

• DirectX [16] - графическое API, предназначенное для разработки под Microsoft Windows.

• Vulkan [17] - кроссплатформенное развивающееся современное графическое API, последняя версия вышла 3 декабря 2024 года.

• Metal [18] - графическое API для операционных систем IOS.

• WebGPU [19] – экспериментальное графическое API, позволяющее разработчикам использовать GPU для отрисовки изображений в браузере. На данных момент поддерживается только в Internet Explorer.

В результате сравнения графических движков можно сказать, что нет универсальных движков, каждый имеет преимущества и недостатки. Ниже представлены краткие выводы по каждому движку:

• Unity имеет низкие системные требования и предоставляет нужные инструменты для реализации шейдеров в рамках объекта исследования. Поддерживает современное API Vulkan, поэтому подойдет для разработки шейдеров под любую платформу, и специлизированное API DirectX для разработки под Microsoft Windows.

• Unreal Engine имеет большой набор встроенных функций, что позволяет быстро достичь желаемого результата при разработке шейдеров. Как и Unity поддерживает API Vulkan и DirectX, что позволяет разрабатывать кроссплатформенные шейдеры. Однако движок имеет высокие системные требования.

• CryEngine позволяет создавать реалистичную графику и имеет средние системные требования, однако имеет высокий порог вхождения. Поддерживает API Vulkan и DirectX, обеспечивает разработку кроссплатформенных шейдеров.

• Godot Engine имеет низкие системные требования, однако среднее удобство настройки шейдеров. Поддерживает большое количество разных графических API, специализированных для Microsoft Windows, IOS и браузера, а также кроссплатформенное API – OpenGl для остальных операционных систем.

• Ogre3D имеет низкие системные требования, но не предоставляет встроенных инструментов для реализации шейдеров, что замедляет процесс разработки. Поддерживает графический API DirectX, предназначенный для Microsoft Windows и кроссплатформенный, но устаревший графический API - OpenGL.

#### 1.1.5 Выбор программного обеспечения

Для текущей задачи создания шейдеров, визуализирующих медицинские данные, был выбран движок Unity. Он обеспечивает необходимые инструменты для разработки - тесселяцию, текстурирование на основе карты высот, нормали, маски и текстурную генерацию. Unity не требователен к ресурсам системы и поддерживает кроссплатформенное и развивающееся графическое API - Vulkan, что позволит использовать шейдеры на любом устройстве.

Решение должно представлять шейдер, разработанный на движке Unity.

### 1.2 Обзор существующих решений

#### 1.2.1 Принципы отбора аналогов

В роли сравнительных аналогов были использованы программные решения для обучения студентов ЭКГ.

Поиск аналогов осуществлялся с помощью ресурса Google Scholar. Для поиска применялись запросы: «ЭКГ визуализация сердца обучение», «ECG education visualization», «ECG education visualization», «ECG education», «ECG education video».

#### 1.2.2 Аналоги

Были найдены следующие аналоги:

**EKGtolkning.com** [20] – веб-приложение, разработанное для обучения студентов Каролинского института. В нем содержатся теоретические материалы, иллюстративные видео и тесты для проверки знаний. Видео содержат схематичные изображения частей сердца.

**Interactive teaching of medical 3D cardiac anatomy** [21] – приложение дополненной реальности, в котором пользователю предоставляется детализированная 3D модель предсердий на основе данных микро-КТ. Приложение визуализирует сигналы синоартериального узла с помощью анимации фронта распространения деполяризационной волны предсердий.

**EduECG** [22] – приложение, в котором пользователю предлагается на выбор несколько вариантов ЭКГ для здорового сердца и сердца с патологиями, визуализируются желудочки сердца и вектора электрических импульсов.

**Educational Material for Taking High Quality ECG** [23] – видео материалы, для обучения снятию ЭКГ, разработанные для студентов университета SAMK. Материалы включают схематичное изображение разреза сердца, на котором

**Визуальная модель сердца для обучения студентов-медиков** [24] – воксельная модель сердца, позволяющая строить сечения сердца, моделирующая и визуализирующая возбуждение миокарда с помощью системы уравнений Алиева–Панфилова.

#### 1.2.3 Критерии сравнения аналогов

В качестве критериев сравнения были выбраны:

• Свободное использование

• Визуализация сердца

• Визуализация сердечного ритма

• Настройка ЭКГ

Ниже представлено более подробное описание каждого критерия.

**Свободное использование**

Данный критерий описывает степень свободы, с которой любой пользователь может использовать решение. Оценивание по этому критерию:

• «Отсутствует», если решения нет в свободном доступе.

• «Частичное», если решение может бесплатно использоваться любым пользователем в качестве готового продукта, но без возможности изучения и изменения исходного кода.

• «Полное», если решение и его исходный код предоставляется в пользование и изменение любому пользователю.

Этот критерий имеет значение для рассматриваемой темы, так как необходимо разработать решение, которое свободно смогут применять и усовершенствовать под свои нужды в бюджетных образовательных учреждениях.

**Визуализация сердца**

Этот критерий описывает степень и формат визуализации сердца в решении. Оценка по критерию может иметь следующие варианты:

• «Отсутствует», если в решении не используется модель сердца.

• «Частичная», если представлены только некоторые отделы сердца, только желудочки или только предсердия.

• «Полная», если сердце визуализировано полностью.

Последние два пункта дополнительно имеют уточнение, двумерная или трехмерная модель сердца используется, соответственно «2D» или «3D».

Этот критерий важен для понимания студентами работы всех отделов сердца.

**Визуализация сердечного ритма**

Данный критерий описывает, как на модели сердца визуализируется сердечный ритм. Этот критерий позволит определить, какого метода визуализации среди существующих решений еще нет.

**Настройка ЭКГ**

В процессе обучения важно рассмотреть разные варианты сердечного ритма, данный критерий оценивает возможность настройки сердечного ритма или возможность отображения разных вариантов.

• «Отсутствует», если в решении визуализируется только один вариант сердечного ритма.

• «Выбор варианта», если пользователь выбирает из списка заготовленных вариантов сердечного ритма.

• «Детальная настройка», если пользователь может настраивать интервалы, сегменты и зубцы ЭКГ.

#### 1.2.4 Сравнение аналогов

Проведенное сравнение представлено в таблице 1.2.

Таблица 1.2 – Сравнение решений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Аналог | Критерий сравнения | | | |
| Свободное использование | Визуализация сердца | Визуализация сердечного ритма | Возможность настройки |
| EKGtolkning.com | Отсутствует | Частичная, 2D | Не  уточняется | Отсутствует |
| Interactive teaching of medical  3D cardiac anatomy | Отсутствует | Частичная, 3D | Движение фронта  поляризации миокарда | Выбор  варианта |
| EduECG | Частичное | Частичная, 2D | Вектор  поляризации миокарда | Выбор  варианта |
| Educational Material for Taking High Quality ECG | Отсутствует | Полная, 2D | Отсутствует | Отсутствует |
| Визуальная модель сердца для обучения студентов-медиков | Отсутствует | Полная, 3D | Изменение потенциала миокарда с помощью изменения цвета | Отсутствует |

Среди рассмотренных аналогов нет решения, удовлетворяющего всем критериям, т.е. предоставляющего полностью свободное использование, полную 3D модель сердца, возможность детальной настройки ЭКГ. Среди вариантов визуализации сердечного ритма нет визуализации с помощью анимации движения миокарда.

#### 1.2.5 Выбор метода решения

Исходя из поставленной цели и сравнительного анализа решений, были сформулированы следующие требования к данной работе:

1. Решением является шейдер.
2. Требуется использовать инструменты с небольшой вычислительной нагрузкой, так как решение направлено на использование студентами-медиками, чьи устройства могут иметь небольшую вычислительную мощность.
3. Представление полной 3D модели сердца.
4. Визуализация сердечного ритма в виде анимации движения сердца.
5. Настройка сердечного ритма с помощью ввода пользователем параметров ЭКГ, в формате секунд и милливольт.
6. Использование решения должно быть полностью свободным.

## 2 Реализация шейдера

### 2.1 Теоретические положения

Шейдеры выполняются на графических процессорах, которые, благодаря распараллеливанию выполнения кода, позволяют достичь большей скорости вычислений, чем на CPU. Но распараллеливание выполнения кода эффективно только в случае, когда в коде нет условного ветвления.

Хотя шейдеры имеют инструмент для условного ветвления, его использование требует осторожности, так как приводит к замедлению вычислений. В представленной работе важна производительность шейдера, так как пользователями шейдера являются студенты медицинских вузов, которые могут не иметь большие вычислительные мощности. Поэтому принято решение отказаться от условного ветвления в пользу ускорения вычислений.

### 2.2 Разработка функции движения вершин

Для реализации анимации необходимо задавать смещение вершины в каждый момент времени. Параметр должен увеличиваться и уменьшаться для моделирования сердцебиения.

Так как в шейдерах использование условного ветвления нежелательно, в силу замедления вычислений, разработанная функция должна основываться на элементарной функции, имеющей периоды возрастания и убывания. Такими функциями являются синус и косинус.

За основу взято уравнение косинуса, представленное в уравнении (1). Функция косинуса выбрана из-за симметричности, которая облегчает вывод итоговой формулы.

Для того, чтобы обеспечить периоды, когда отдел сердца находится в покое, берется часть синусоиды выше оси X, уравнение (2).

Таким образом график принимает вид, представленный на рисунке 2.1.

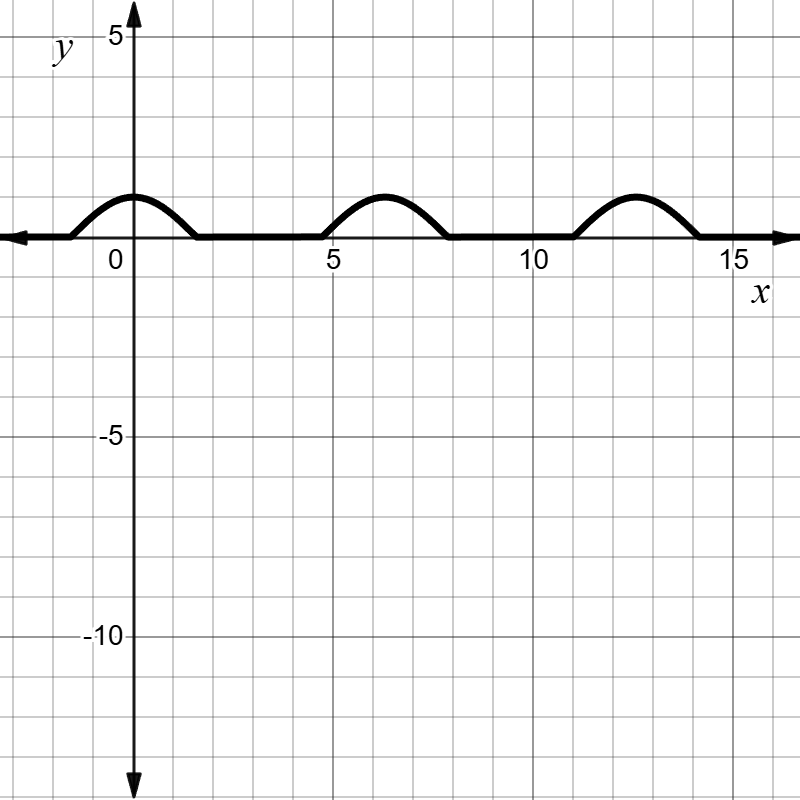


Рисунок 2.1 – График уравнения №2.

Для моделирования ЭКГ потребуется независимо менять следующие параметры:

1. Максимальное значение функции, отвечающее силе импульса, созданного сердцем.
2. Длительность первой половины зубца синусоиды, отвечающая длительности сокращения конкретного отдела сердца.
3. Длительность второй половины зубца синусоиды, отвечающая длительности расслабления конкретного отдела сердца.
4. Период синусоиды, отвечающий интервалу между ударами конкретного отдела сердца.
5. Смещение по оси X, отвечающее интервалу между началом ударов разных отделов сердца.

Для регулирования ширины зубцов независимо от периода, добавляется коэффициент *f*, который сложением с коэффициентом *a* будет растягивать график вдоль оси Y и прибавлением к коэффициенту *d* опускать график по оси Y таким образом, чтобы максимум оставался в той же точке, графически это представлено на рисунке 2.2.

В результате уравнение примет вид, представленный в уравнении (3).

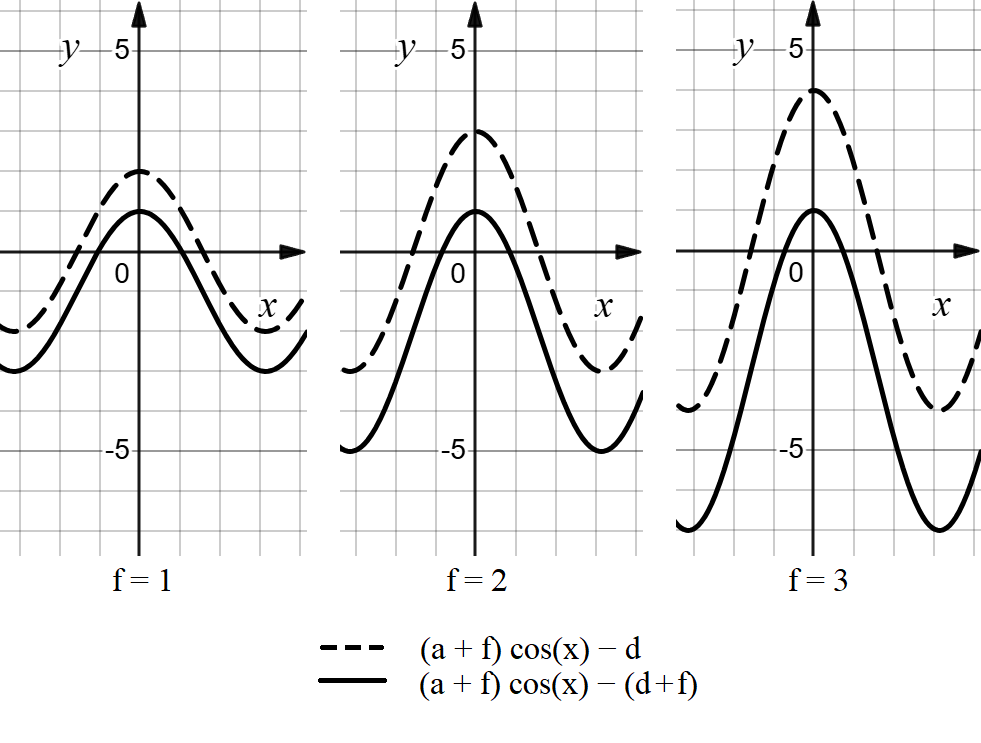


Рисунок 2.2 – Влияние коэффициента f.

Поскольку для изменения максимума функции, достаточно менять только *d*, примем *a* = 1.

Чтобы из данных ЭКГ – интервалов в секундах и силы в вольтах получать параметры синусоиды, необходимо сначала вывести эти данные из уравнения, чтобы определить от каких переменных они зависят. Затем вывести из полученных формул параметры, зависящие от данных в секундах и вольтах.

Так, для времени *t* одной половины синусоиды, необходимо вывести x, результат в уравнении (4).

Период *T*, равный времени между ударами отдела сердца, представлен в формуле (5).

Максимум *A* функции, равный силе электрического импульса, представлен в формуле (6).

Таким образом, можно вывести зависимости параметров синусоиды от периода и максимума, результат представлен в уравнениях (7-8).

С использованием уравнений (7-8) можно вывести зависимость параметра *f* от периода и максимума, результат представлен в уравнении (9).

Сердце имеет разную длительность сокращения и расслабления, следовательно, нужно задавать отдельные параметры *f* для периода возрастания и убывания функции. Для этого используется функция *q*, представленная в уравнении (10).

График функции q представлен на рисунке 2.3.

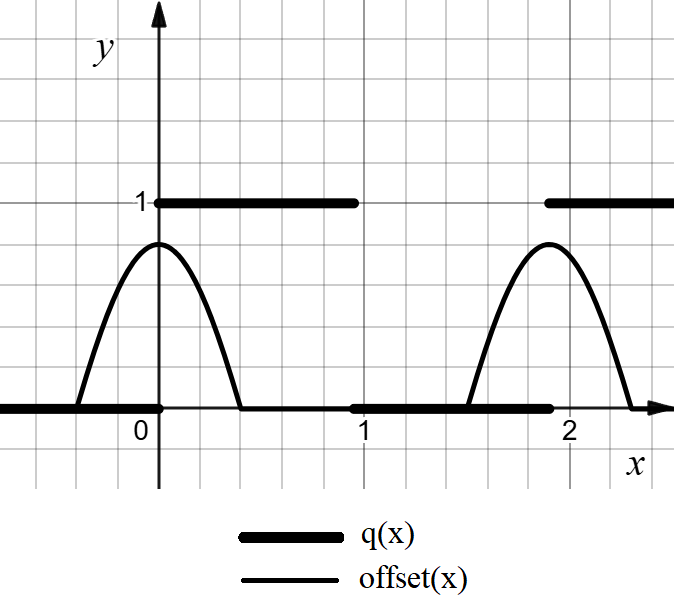


Рисунок 2.3 – График функции q.

Функция *q* используется для переключения коэффициента *f*. В зависимости от *x*, коэффициент *f* будет умножаться на 1 или на 0.

Таким образом, должно использоваться выражение, представленное в уравнении (11). Для удобства, обозначим его *z*.

где fв – коэффициент для возрастающей части функции, fу – коэффициент для убывающей части функции.

Итоговая функция смещения вершины для одного отдела сердца примет вид уравнения (12).

При задании разных значений параметрам *tв* – время возрастания, и *tу* – время убывания, получается график, представленный на рисунке 2.4.

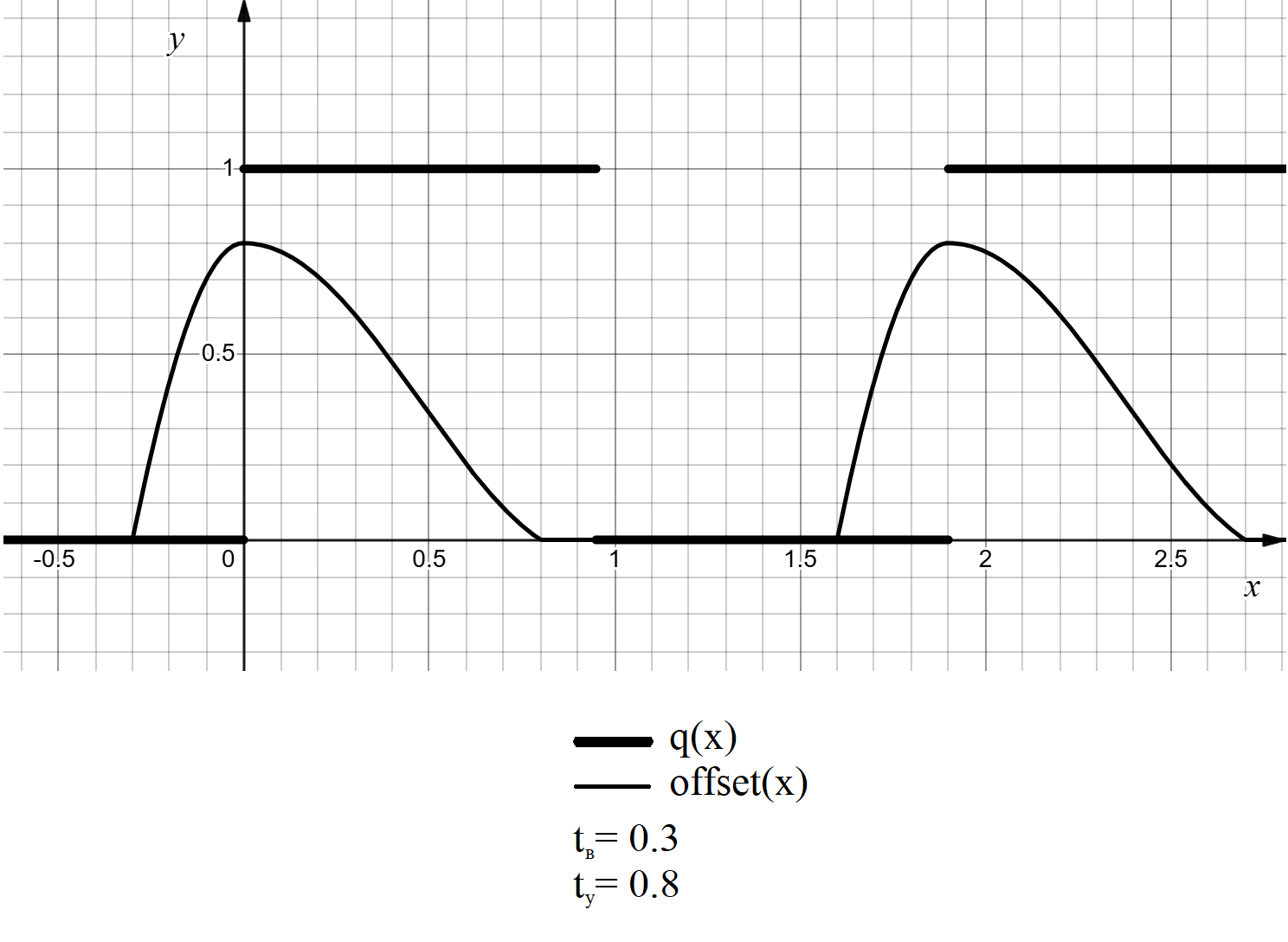


Рисунок 2.4 – График функции offset(x) при разных значениях параметров

tв и tу.

Функции смещения вершин для других отделов сердца также имеют вид уравнения (12), но должны задаваться со смещением по оси *X*, так как предсердия и желудочки бьются по очереди.

С учетом смещения по оси *Х*, функция offset(x) примет вид (13), а функция *q* примет вид (14).

где *c* – смещение начала удара относительно начала координат. Таким образом, *c* задает время, через которое начинается возрастание функции offset(x). В ЭКГ этот промежуток между ударом предсердий и желудочков соответствует сегменту PQ.

Соответствие параметров полученной функции offset(x) и параметров ЭКГ представлено в таблице 2.1.

Таблица 2.1 – Соответствие параметров функции и параметров ЭКГ

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр ЭКГ | Единица измерения | Параметр функции | Пояснение параметра функции |
| Интервал P | Секунды | tпв | Время возрастания функции смещения вершин предсердий. |
| Сегмент P-S | Секунды | tпу | Время убывания функции смещения вершин предсердий. |
| Интервал QRS | Секунды | tжв | Время возрастания функции смещения вершин желудочков. |
| Интервал S-T | Секунды | tжу | Время убывания функции смещения вершин желудочков. |
| Интервал P-P | Секунды | T | Время между максимумами функции. Общий параметр. |
| Зубец P | Милливольт | Aп | Максимум функции смещения вершин предсердий. |
| Зубец R | Милливольт | Aж | Максимум функции смещения вершин желудочков. |
| Сегмент P-Q | Секунды | cж | Смещение по оси X функции смещения вершин желудочков. |

Пример функций смещения вершин для предсердий и желудочков с стандартными параметрами ЭКГ представлен на рисунке 2.5.

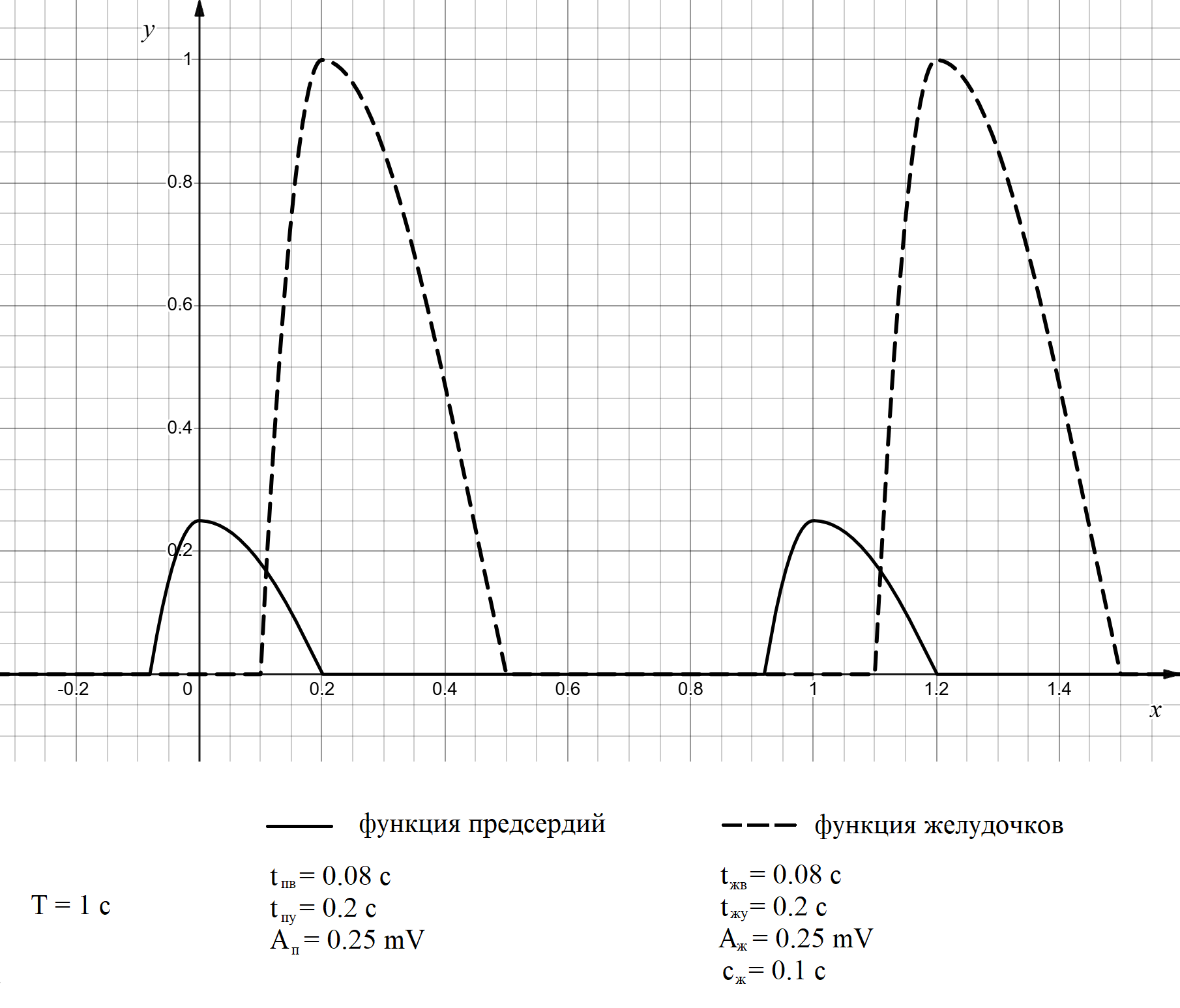


Рисунок 2.5 – График функций смещения вершин предсердий и желудочков.

### 2.3 Написание кода шейдера

В шейдере в качестве функции *q*, из уравнения (10), используется встроенная функция Unity – step(a, b). Математически она описывается уравнением (11).

В качестве переменной *x*, в функцию передается встроенная переменная времени \_Time.y, хранящая время в секундах с момента загрузки шейдера.

## 3 Разработка приложения

## 4 Тестирование

## 5 Недостатки решения

## 6 Направление дальнейшего развития

## ЗАКЛЮЧЕНИЕ

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сыроежко А.А. Технология трехмерной визуализаии // Современное состояние, проблемы и перспективы развития отраслевой науки - Москва, 2016 - с.466-474
2. Аванесова Татьяна Панайотовна Эффективность компьютерной технологии обучения // Вестник Северного (Арктического) федерального университета. Серия: Гуманитарные и социальные науки. 2013. №4.
3. Свободное программное обеспечение в государственном секторе. Сборник материалов. — М. INFO-FOSS.RU, 2007. — 112 с
4. Тихомирова А.А., Котиков П.Е. О перспективах перехода на свободное программное обеспечение в здравоохранении / А.А. Тихомирова, П.Е. Котиков // Детская медицина Северо-Запада. — 2018. Т. 7. № 1. С. 315-316.
5. Unity : офиц.сайт - URL: https://unity3d.com (дата обращения: 02.12.2024 )
6. Бровкина А. С., Сейидов Р. Х. ИГРОВЫЕ ДВИЖКИ // Форум молодых ученых. 2017. №1 (5).
7. Unreal Engine : офиц.сайт - URL: https://www.unrealengine.com/en-US (дата обращения: 02.12.2024 )
8. Пасько Д. Н. Современные игровые движки // Инновационная наука. 2016. №2-3 (14).
9. CryEngine : офиц.сайт - URL: https://www.cryengine.com/ (дата обращения: 02.12.2024 )
10. Усков Максим Александрович ОБЗОР ПРЕИМУЩЕСТВ И НЕДОСТАТКОВ ИГРОВЫХ ДВИЖКОВ. ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТОВ И ТЕХНОЛОГИЙ РАЗРАБОТКИ КЛИЕНТСКОЙ ЧАСТИ ИГРОВЫХ ПРИЛОЖЕНИЙ // Глобус: технические науки. 2020. №5 (36).
11. Godot Engine : офиц.сайт - URL: https://godotengine.org/ (дата обращения: 02.12.2024 )
12. Ломайкин А. С. GODOT ENGINE В СОВРЕМЕННОЙ ИГРОВОЙ ИНДУСТРИИ // E-Scio. 2023. №6 (81).
13. Абалаков И. Н., Зотин А. Г. Обзор 3D -движков с возможностью визуализации ландшафтных изображений и природных эффектов // Решетневские чтения. 2012. №16.
14. Ogre3D : офиц.сайт - URL: https://www.ogre3d.org/ (дата обращения: 02.12.2024 )
15. OpenGL Registry: офиц.сайт - URL: https://registry.khronos.org/OpenGL/index\_gl.php (дата обращения: 10.01.2025)
16. The Microsoft DirectX: офиц.сайт - URL: https://www.microsoft.com/en-us/download/details.aspx?id=35 (дата обращения: 10.01.2025)
17. Vulkan : офиц.сайт - URL: https://www.vulkan.org/#latest-vulkan-api-extensions-and-additions (дата обращения: 10.01.2025)
18. Metal: офиц.сайт - URL: https://developer.apple.com/metal/ (дата обращения: 11.01.2025)
19. WebGPU: URL: https://developer.mozilla.org/en-US/docs/Web/API/WebGPU\_API (дата обращения: 11.01.2025)
20. Nilsson M. et al. Evaluation of a web-based ECG-interpretation programme for undergraduate medical students //BMC medical education. – 2008. – Т. 8. – С. 1-7.
21. Potyagaylo D. et al. Interactive teaching of medical 3D cardiac anatomy: atrial anatomy enhanced by ECG and 3D visualization //Frontiers in Medicine. – 2024. – Т. 11. – С. 1422017.
22. Glądys K. et al. Teaching the Anatomical Three-Dimensional Relation of the ECG: EduECG.
23. Rajala N., Mahlamäki J. Educational Material for Taking High Quality ECG: educational video. – 2021.
24. Бодин О. Н., Кузьмин А. В., Митрошин А. Н. Разработка визуальной модели сердца для обучения студентов-медиков //Известия высших учебных заведений. Поволжский регион. Медицинские науки. – 2007. – №. 2. – С. 3-10.

## ПРИЛОЖЕНИЕ А