**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Заместитель руководителя  Департамента  Программной Инженерии  Факультета Компьютерных Наук  профессор  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е. М. Гринкруг  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |  | УТВЕРЖДАЮ  Академический руководитель  образовательной программы  «Программная инженерия»  профессор, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | **RU.17701729.04.01-01 81** | | **JavaFX ПРИЛОЖЕНИЕ «КВАТЕРНИОНЫ В 3D ГРАФИКЕ»**  **Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.01-01 81 01-1-ЛУ** | | |
|  |  | |
| Исполнитель  студент группы 175 ПИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Т. О. Мартиросян /  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | |
|  | | |
|  | |  |

**Москва 2019**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU. 17701729.04.16-01 81 01-1-ЛУ | |  |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | **RU.17701729.04.01-01 81** | | **JAVAFX ПРИЛОЖЕНИЕ «КВАТЕРНИОНЫ В 3D ГРАФИКЕ»**  **Пояснительная записка**  **RU.17701729.04.01-01 81 01-1**  **Листов 17** | | | | | |
|  |  | | | | |
|  | | | | |
|  | | | | | |
|  | | | |  | |

**Москва 2019**

СОДЕРЖАНИЕ

[1. ВВЕДЕНИЕ 2](#_Toc8823346)

[**1.1.** Наименование разработки 2](#_Toc8823347)

[**1.2.** Основание для разработки 3](#_Toc8823348)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc8823349)

[**2.1.** **Функциональное назначение** 4](#_Toc8823350)

[**2.2.** **Эксплуатационное назначение** 4](#_Toc8823351)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 5](#_Toc8823352)

[**3.1.** **Постановка задачи на разработку программы** 5](#_Toc8823353)

[**3.2.** **Описание алгоритма и функционирования программы** 5](#_Toc8823354)

[**3.2.1 Вращение при помощи кватерниона** 5](#_Toc8823355)

[**3.2.2 Интерполяция кватернионов** 6](#_Toc8823356)

[**3.2.3 Нахождение кватерниона вращения между двумя векторами** 6](#_Toc8823357)

[**3.3.** **Организация входных данных** 7](#_Toc8823358)

[**3.4.** **Организация выходных данных** 7](#_Toc8823359)

[**3.5.** **Описание и обоснование выбора и состава технических и программных средств** 7](#_Toc8823360)

[**3.5.1. Описание технических и программных средств** 7](#_Toc8823361)

[**3.5.2. Обоснование выбора технических и программных средств** 7](#_Toc8823362)

[4. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 7](#_Toc8823363)

[**4.1.** Предполагаемая потребность 7](#_Toc8823364)

[**4.2.** Ориентировочная экономическая эффективность 7](#_Toc8823365)

[**4.3.** **Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами** 8](#_Toc8823366)

[5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 8](#_Toc8823367)

[ПРИЛОЖЕНИЕ 1. JAVA: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ МЕТОДОВ, ПОЛЕЙ И СВОЙСТВ 10](#_Toc8823368)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 16](#_Toc8823369)

1. **ВВЕДЕНИЕ**

## Наименование разработки

**Наименование разработки на русском языке:** JavaFX приложение «Кватернионы в 3D графике»

**Наименование разработки на английском языке:** JavaFX Application «Quaternions 3D Graphics»

## Основание для разработки

**Документ, на основании которого ведется разработка:**

Программа выполнена в рамках задания на курсовую работу в соответствии с учебным планом подготовки бакалавров (НИУ ВШЭ, факультет компьютерных наук) по направлению «Программная инженерия». Приказ декана факультета компьютерных наук Национального исследовательского университета «Высшая школа экономики№ 2.3-02/1504-02 от 15.04.2019 "Об утверждении тем, руководителей курсовых работ студентов образовательной программы Программная инженерия факультета компьютерных наук".

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**
   1. **Функциональное назначение**

К функциональным возможностям программы можно отнести: создание 3D сцены с объектами на ней(сферой или кубом), возможность рассмотреть объект с разных сторон, вращение объекта в пространстве относительно заданной оси при помощи кватерниона, демонстрация информации о примененном для вращения кватернионе, для сферы возможность указать точки на ней, по котором будет сформирован кватернион вращающий сферу таким образом, что одна точка перейдет в другую, возможность приближать и отдалять камеру.

* 1. **Эксплуатационное назначение**

Приложение позволит студентам, школьникам, а также любым желающим познакомиться с практическим применением такой алгебраической структуры как кватернион. Позволит наглядно увидеть, что умеют кватернионы и как они работают.

1. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**
   1. **Постановка задачи на разработку программы**

В рамках курсовой работы требуется реализовать приложение с использованием JavaFX используя Java8, которые покажет некоторые возможности использования кватернионов в прикладных целях.

1. Требуется создать 3D сцену, на которой будет куб. Реализовать возможность вращения куба вокруг заданной пользователем оси, а также на указанный пользователем угол вращения. Пользователю должна быть дана информация о том какой кватернион был сформирован, наглядно должна быть показана ось вращения.
2. Требуется создать 3D сцену, на которой будет сфера. Реализовать возможность указания точек на сфере, что сфера будет повернута таким образом, что одна точка окажется на месте другой. Показать применяемый кватернион для вращения. Реализовать вращение как анимацию, используя интерполяцию кватернионов.
   1. **Описание алгоритма и функционирования программы**

### **3.2.1 Вращение при помощи кватерниона**

Известно, что любая матрица представляет собой преобразование пространства. Если быть более корректным, то id-матрица умноженная на некоторую матрицу есть преобразование, но id-матрица является нейтральным элементом по умножению, поэтому можно опускать и допустить такую неточность. Аффинные преобразования позволяют сохранять углы между векторами и длины в исходном состоянии. В JavaFX повороты осуществляются при помощи матрицы некоторого аффинного преобразования, а именно с помощью матрицы 3х4. По сути, это есть матрица поворота с присоединённым к ней вектором-столбцом, определяющим перемещение в пространстве, трансляцию. Кватернион сам по себе, уже, по сути, какое-то преобразование пространства, как и в случае с любой матрицей.

Кватернион можно определить как некоторую сумму:

,где - мнимые единицы со свойством

На деле получается, что w – это скаляр, а x,y,z -координаты вектора трехмерного пространства.

Получить из угла вращения и заданной оси необходимый кватернион достаточно просто

Где α - угол вращения, а vx,vy,vz – координаты векторов оси вращения.

Преобразовываем этот кватернион в матрицу поворота и средствами JavaFX осуществляем поворот.

Матрица 3х3 выглядела бы так:

Сделать из этой матрицы 4х4 не проблема – приписываем нулевой столбец чтобы никуда не транслировать(перемещать) наш объект.

### **3.2.2 Интерполяция кватернионов**

В данной работе была сделана сферическая интерполяция кватернионов(SLERP)

Формула для интерполяции кватернионов выглядит следующим образом:

В коде на java выглядит так:

double result = start.dot(end);  
  
if (result < 0.0f) {  
 end.x = -end.x;  
 end.y = -end.y;  
 end.z = -end.z;  
 end.w = -end.w;  
 result = -result;  
}  
  
double scale0 = 1 - frac;  
double scale1 = frac;  
  
if ( (1 - result) > 0.1) {  
 double theta = Math.*acos*(result);  
 double invSinTheta = 1 / Math.*sin*(theta);  
  
 scale0 = Math.*sin*((1 - frac) \* theta) \* invSinTheta;  
 scale1 = Math.*sin*((frac \* theta)) \* invSinTheta;  
}  
  
double x = (scale0 \* start.x) + (scale1 \* end.x);  
double y = (scale0 \* start.y) + (scale1 \* end.y);  
double z = (scale0 \* start.z) + (scale1 \* end.z);  
double w = (scale0 \* start.w) + (scale1 \* end.w);  
Quaternion q = new Quaternion(x,y,z,w);

Интерполяция в работе была нужная для того, чтобы показать плавную анимацию вращения сферы.

### **3.2.3 Нахождение кватерниона вращения между двумя векторами**

Пусть есть вектор v1(x,y,z) и v2(x,y,z)

Кватернион вращения должен будет иметь следующий вид:

Где x,y,z – координаты вектора оси вращения.

Осью вращения будет векторное произведение векторов v1 и v2

Выше видно, что вектор, представляющий ось вращения, был нормализован.

так как

Однако, стоит заметить, что в случае если скалярное произведение векторов равно единице, либо -1, необходимо проделать дополнительные операции.

Если скалярное произведение равно 1, значит эти векторы однонаправлены, нужно вернуть id-кватернион.

Если -1, значит направлены в противоположенные стороны, нужно вернуть кватернион вращающий на 180 градусов.

* 1. **Организация входных данных**

Входными данными являются действия пользователя и введенные им значения в соответствующие поля. Введенные данные проверяются на корректность.

* 1. **Организация выходных данных**

Выходными данными являются визуализация работы кватернионов и отображаемые данные и сообщения пользователю в ответ на его действия.

* 1. **Описание и обоснование выбора и состава технических и программных средств**

### **3.5.1. Описание технических и программных средств**

Для того чтобы воспользоваться приложением необходим компьютер с установленной JRE 8, в которую входит JavaFx. Также необходимы мышь либо тачпад и клавиатура.

### **3.5.2. Обоснование выбора технических и программных средств**

Приложение было разработано на языке Java 8 версии с использованием JavaFx.

JavaFx позволяет сделать приятный интерфейс для десктопных приложений, а также обладает вполне приемлемым, но не полным набором возможностей для работы с 3D графикой. В связи с этим было решено реализовать программу именно с использованием этих технологий для более глубокого изучения 3D графики.

1. **ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**

## Предполагаемая потребность

Программа создана с целью демонстрации некоторых прикладных назначений кватернионов.

Использование программы может быть интересно студентам, школьникам и даже преподавателям линейной алгебры, желающим показать своим ученикам важность и полезность изучаемого материала.

## Ориентировочная экономическая эффективность

Данная работа не является коммерческим проектом, поэтому расчет экономической эффективности не предусмотрен.

## **Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами**

Программ, имеющих именно такое назначение, найдено не было. Существует большое количество редакторов, позволяющих вращать объекты, которые для вращения используют кватернионы, но в них отсутствует возможность посмотреть информацию о приименном преобразовании. Кроме того, данные программы обычно излишне функциональны и занимают большой объем дискового пространства.

1. **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**
2. ГОСТ 19.102-77 Стадии разработки. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.101-77 Виды программ и программных документов. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.103-77 Обозначения программ и программных документов. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.104-78 Основные надписи. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.105-78 Общие требования к программным документам. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

# ПРИЛОЖЕНИЕ 1. JAVA: ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ МЕТОДОВ, ПОЛЕЙ И СВОЙСТВ

Класс MainWindowController

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Доступ | Тип | Назначение |
| axisGridPane | default | AxisGridPane | Предоставление информации об оси вращение |
| quaternionGridPane | default | QuaternionGridPane | Предоставление информации о применяемом кватернионе |
| axisScene | private | Subscene | Подсцена в которой располагаются координатные оси пространства |
| anchorAngleX | private | double | Для организации вращения камеры |
| anchorAngleY | private | double | Для организации вращения камеры |
| anchorX | private | double | Для организации вращения камеры |
| anchory | private | double | Для организации вращения камеры |
| angleX | private | DoubleProperty | Для организации вращения камеры |
| angleY | private | DoubleProperty | Для организации вращения камеры |
| rotateX | private | Rotate | Представляет собой объект вращения камеры |
| rotateY | private | Rotate | Представляет собой объект вращения камеры |
| zoom | private | Translate | Для организации удаления и приближения камеры |
| camera | private | PerspectiveCamera | Камера |
| contentPane | private | StackPane | Область на которой будет 3D сцена |
| ZOOM\_MAX | private | int | Константа максимальное увеличение |
| ZOOM\_MIN | private | int | Константа минимальное уменьшение |
| ZOOM\_SENSITIVITY | private | double | Константа чувствительность масштабировнания |
|  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Методы | | | | | |
| Имя | Доступ | Тип | Аргументы | Назначение |
| addCameraRotations | private | void |  | Добавляет к сцене возможность вращать камеру |
| buildAxes | private | void |  | Добавляет на сцену координатные оси пространства |
| cos | private | double | double | Функция косинуса |
| sin | private | double | double | Функция синуса |
| prepareAxisScene | private | void |  | Подготавливает сцену с координатными осями |
| prepareCamera | private | void |  | Подготавливает камеру |
| prepateQuaternionSupport | private | void |  | Подготавливает сцену для возможности работы с кватернионами |
| prepareSubScene | private | void |  | Подготавливает сцену |
| zoomCameraBy | private | void | double | Приближает или отдаляет камеру на заданное значение |
| onCubeMenuButtonClicked | private | void | ActionEvent | Обрабатывает нажатие на кнопку «Куб» из списка проектов |
| onSphereMenuButtonClicked | private | void | ActionEvent | Обрабатывает нажатие на кнопку «Сфера» из списка проектов |

Класс Main

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Доступ | Тип | Назначение |
| mainWindowController | private | MainWindowController | Объект контроллера главного окна приложения |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Доступ | Тип | Аргументы | Назначение |
| main | public | void | String[] | Точка входа в программу |
| start | public | void | Stage | Загружает главное окно программы |

Класс Matrix4x3

Класс-наследник класса Affine, позволяющий преобразовать кватернион в матрицу поворота

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Доступ | Тип | Аргументы | Назначение |
| rotate | public | Matrix4x3 | Quaternion | Получает из кватерниона матрицу поворота |
| rotateGeneric | public | Matrix4x3 | Quaternion | Применяет к текущей матрице поворота, кватернион преобразованный в матрицу поворота. В итоге получается необходимая матрица поврота |

Класс Quaternion

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Доступ | Тип | Назначение |
| x | private | double | x-координата кватерниона |
| y | private | double | y-координата кватерниона |
| z | private | double | z-координата кватерниона |
| w | private | double | скаляр |
| EPSILON | private | double |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Доступ | Тип | Аргументы | Назначение |
| sin | private | double | double | Функция синуса |
| cos | private | double | double | Функция косинуса |
| Quaternion | public |  |  | Конструктор. Создает id-кватернион |
| Quaternion | public |  | double x,y,z,w | Конструктор. Создает кватернион по заданным параметрам |
| mul | public | Quaternion | Quaternion | Перемножение кватернионов |
| dot | public | double | Quaternion | Скалярное произведение |
| fromRotationBetween | public | Quaternion | Point3D a, Point3D b | Возвращает кратчайший кватернион вращения между двумя векторами |
| normalize | public | void |  | Нормализует кватернион |

Класс QuaternionSphere

Класс-наследник класса Sphere, с поддержкой вращения при помощи кватернионов

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Доступ | Тип | Назначение |
| qrt | private | QuaternionTransition | объект поворота в режиме анимации |
| state | private | Quaternion |  |
| points | private | ArrayList<Point3D> | хранит точки |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Доступ | Тип | Аргументы | Назначение |
| QuaternionSphere | public | Void | double radius | Конструктор. Создает сферу с заданным радиусом |
| setDuration | public | void | Duration | Устанавливает продолжительность анимации |
| play | public | void |  | Запускает анимацию |
| rotateByQuaternion | public | void | Quaternion q | Вращает сферу используя кватернион |

Класс QuaternionTransition

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Доступ | Тип | Назначение |
| sphere | private | QuaternionSphere | Сфера которую будет вращать текущий объект |
| duration | private | ObjectProperty<Duration> | длительность вращения |
| start | private | Quaternion | Для интерполяции. Начальный кватернион |
| end | private | Quaternion | Для интерполяции. Финишный кватернион |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Доступ | Тип | Аргументы | Назначение |
| QaternionTransition | public | QuaternionSphere |  | Конструктор. Создает объект вращения переданной сферы |
| interpolate | public | void | double frac | Интерполирует start и end кватернионы с параметром frac [0,1] |
| getDuration | public | Duration |  | возращает продолжительность вращения |
| durationProperty | public | ObjectProperty<Duration> |  | Возращает свойство продолжительности |
| setStart | public | void | Quaternion startQuaternion | Устанавливает начальный кватернион |
| setEnd | public | void | Quaternion startQuaternion | Устанавливает конечный кватернион |

Класс P2PCylinder

Класс-наследник класса Cylinder. Служит для рисования цилиндра в пространстве от одной точки до другой

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Доступ | Тип | Аргументы | Назначение |
| P2PCylinder | public |  | Point3D origin, Point3D target | Конструктор. |
| createConnection | public | void | Point3D origin, Point3D target | Изменяет текущий объект таким образом, чтобы цилиндр был от одной точки до другой |

Класс XYZAxis

|  |  |  |  |
| --- | --- | --- | --- |
| Поля | | | |
| Имя | Доступ | Тип | Назначение |
| CX | private | Cylinder | Цилиндр для оси X |
| CY | private | Cylinder | Цилиндр для оси Y |
| CZ | private | Cylinder | Цилиндр для оси Z |
| S | private | Sphere | Сфера на пересечении цилиндров |
| yCone | private | MeshView | Конус для оси Y |
| xCone | private | MeshView | Конус для оси X |
| zCone | private | MeshView | Конус для оси Z |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Методы | | | | |
| Имя | Доступ | Тип | Аргументы | Назначение |
| setMaterials | private | void |  | Устанавливает цвета для конусов и цилиндров |
| positioning | private | void |  | Располагает оси в пространстве правильным образом |
| createCone | private | TriangleMesh | float radius, float height | Создает меш для конуса по заданному радиусу и высоте |

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированх |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |