|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 | Робототехника и комплексная автоматизация (РК) |
| КАФЕДРА | Системы автоматизированного проектирования (РК6) |

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***«Разработка графического программного обеспечения для визуализации трехмерных объектов»***

|  |  |  |
| --- | --- | --- |
| Студент РК6-73Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Губанов Д.А.** |
|  | (Подпись, дата) | И.О. Фамилия |
| Руководитель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | И.О. Фамилия |

*2025 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 РК6

А.П. Карпенко

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме: Разработка графического программного обеспечения для визуализации трехмерных\_\_\_\_\_\_\_\_ объектов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_РК6-83Б\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Губанов Даниил Александрович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.) \_учебная\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к 5 нед., 50% к 11 нед., 75% к 14 нед., 100% к 16 нед.

Техническое задание:\_Разработка системы перемещения, вращения и масштабирования 3D-объектов с использованием матричных преобразований и кватернионов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на 16 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «1» май 2025 г.

|  |  |  |
| --- | --- | --- |
| **Руководитель НИР** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | И.О. Фамилия |
| **Студент** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Губанов Д.А.** |
|  | (Подпись, дата) | И.О. Фамилия |

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc199980246)

[1. Аффинные преобразования 5](#_Toc199980247)

[2. Проблемы матричного вращения 8](#_Toc199980248)

[3. Кватернионы в 3D-графике 10](#_Toc199980249)

[4. Системы координат и пространства 13](#_Toc199980250)

[ЗАКЛЮЧЕНИЕ 15](#_Toc199980251)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc199980252)

ВВЕДЕНИЕ

Целью данной работы является разработка системы преобразований 3D-объектов, объединяющей матричные и кватернионные подходы. В рамках исследования рассматриваются математические основы аффинных преобразований, анализируются проблемы матричного представления вращений, и демонстрируются преимущества кватернионов для плавной анимации. Практическая часть включает реализацию системы на выбранном языке программирования с сравнением производительности методов.  
 Современные технологии трехмерной графики активно применяются в компьютерных играх, виртуальной реальности, инженерном проектировании и кинематографе. Одной из ключевых задач при работе с 3D-объектами является управление их перемещением, вращением и масштабированием, что требует эффективных математических методов. Традиционно для этого используются матрицы аффинных преобразований, однако в ряде случаев, например при интерполяции вращений, более предпочтительными оказываются кватернионы.

Актуальность работы обусловлена растущими требованиями к визуализации в реальном времени, где важно не только корректное отображение объектов, но и оптимизация вычислений. Результаты исследования могут быть использованы в графических движках, симуляторах и других приложениях, требующих точного управления 3D-преобразованиями.

1. Аффинные преобразования

Аффинные преобразования представляют собой фундаментальный математический аппарат для работы с 3D-объектами, позволяющий осуществлять их перемещение, изменение ориентации и масштабирование. В данном разделе рассматриваются основные виды аффинных преобразований и их матричное представление.

Основные виды аффинных преобразований:

1. Перенос (Трансляция). Преобразование, изменяющее положение объекта в пространстве. Матрица переноса T в однородных координатах имеет вид:
2. Масштабирование. Преобразование, изменяющее размеры объекта. Матрица масштабирования S:
3. Вращение. Преобразование, изменяющее ориентацию объекта. Основные матрицы вращения вокруг осей:

Аффинные преобразования обладают свойством композиции - последовательность преобразований может быть представлена единой матрицей, равной произведению матриц отдельных преобразований. Важно учитывать порядок умножения матриц, так как матричное умножение некоммутативно.

Общая формула преобразования точки:

Реализация преобразования точки представлена в листинге 2.

Листинг 2 – Реализация преобразовании точки.

FMatrix FTransform::GetMatrix() const

{

return Position.GetTranslationMatrix() \* Quaternion.GetMatrix() \*

Scale.GetScaleMatrix();

}

В сложных 3D-сценах объекты часто организуются в иерархические структуры, где каждый дочерний объект существует в локальной системе координат родительского объекта. Для корректного отображения и взаимодействия таких объектов необходимо:

1. Определять глобальные координаты дочерних объектов
2. Обрабатывать каскадные преобразования
3. Обеспечивать эффективное обновление при изменении иерархии

Глобальная трансформация объекта вычисляется как произведение матриц преобразований всех родительских объектов:

Реализация представлена в листинге 3, в виде рекурсивного алгоритма вычисления родительских матриц.

Листинг 3 – Реализация расчета глобальной трансформации.

FMatrix RSceneComponent::GetWorldTransformMatrix() const

{

if (const std::shared\_ptr<RSceneComponent> Parent = GetParentComponent())

{

return Parent->GetWorldTransformMatrix() \* GetRelativeTransformMatrix();

}

return GetRelativeTransformMatrix();

}

1. Проблемы матричного вращения

Несмотря на широкое применение матричных преобразований в компьютерной графике, данный подход обладает рядом существенных ограничений, которые необходимо учитывать при разработке систем управления 3D-объектами. В настоящем разделе рассматриваются ключевые проблемы, возникающие при использовании матричного представления вращений.

Наиболее известной проблемой матричного вращения является Gimbal Lock, возникающий при совпадении двух осей вращения.

Рассмотрим классический пример с последовательностью вращений ZYX:

1. При повороте вокруг оси Y на 90° (тангаж), оси X и Z совпадают в одной плоскости.
2. Последующие вращения вокруг этих осей становятся эквивалентными — система больше не может различить, вокруг какой именно оси происходит вращение.
3. Математически это проявляется как вырождение матрицы вращения, когда две компоненты матрицы становятся линейно зависимыми.

Традиционные методы пытались обойти проблему:

1. Ограничение углов (например, запрет поворотов на ±90° по второй оси)
2. Переключение осей при приближении к опасному углу
3. Реортогонализация матриц для компенсации ошибок

Эти ограничения привели к поиску принципиально иного подхода — кватернионов. В отличие от углов Эйлера, кватернионы:

1. Избегают Gimbal Lock
2. Описывают вращение как единое преобразование вокруг произвольной оси
3. Не используют последовательность элементарных поворотов
4. Обеспечивают плавную интерполяцию
5. Метод SLERP (Spherical Linear Interpolation) дает естественные промежуточные состояния
6. Вычислительно эффективны
7. Требуют меньше операций, чем матричная реортогонализация
8. Легко конвертируются в матрицы для рендеринга

Кватернионы не просто решают проблему Gimbal Lock — они предлагают принципиально более совершенный способ представления вращений в 3D-пространстве.

1. Кватернионы в 3D-графике

Кватернионы представляют собой мощный математический аппарат для описания вращений в трехмерном пространстве, лишенный недостатков матричного подхода. Впервые предложенные Уильямом Гамильтоном в 1843 году, они нашли широкое применение в компьютерной графике, робототехнике и системах виртуальной реальности.

Кватернион q — это гиперкомплексное число вида:

где w - скалярная часть, (x,y,z) - векторная часть,

Единичный кватернион может представлять вращение на угол θ вокруг оси n:

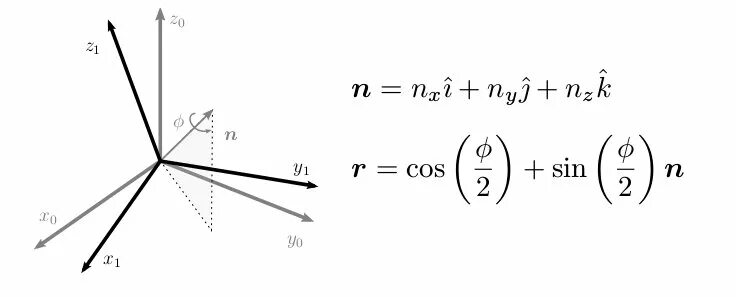


Рис 1. Представление кватерниона в пространстве

Реализация кватернионов наследована от реализации библиотеки GLM и добавлена реализация преобразования эйлеровых углов в кватернион и представлена в листинге 4.

Листинг 4 – Реализация работы эйлеровых углов и кватернионов.

FMatrix RSceneComponent::GetWorldTransformMatrix() const

{

if (const std::shared\_ptr<RSceneComponent> Parent = GetParentComponent())

{

return Parent->GetWorldTransformMatrix() \* GetRelativeTransformMatrix();

}

return GetRelativeTransformMatrix();

}

FQuat::FQuat(float Pitch, float Yaw, float Roll)

{

\*this = glm::normalize(

glm::angleAxis(glm::radians(Yaw), FVector::UpVector)

\* glm::angleAxis(glm::radians(Pitch), FVector::RightVector)

\* glm::angleAxis(glm::radians(Roll), FVector::ForwardVector));

UpdateEditorAngles();

}

void FQuat::AddPitch(float Pitch)

{

\*this = \*this \* glm::angleAxis(glm::radians(Pitch), FVector::RightVector);

UpdateEditorAngles();

}

void FQuat::AddWorldPitch(float Pitch)

{

\*this = glm::angleAxis(glm::radians(Pitch), FVector::RightVector) \* \*this;

UpdateEditorAngles();

}

void FQuat::AddYaw(float Yaw)

{

\*this = glm::normalize(\*this \* glm::angleAxis(glm::radians(Yaw), FVector::UpVector));

UpdateEditorAngles();

}

void FQuat::AddWorldYaw(float Yaw)

{

\*this = glm::normalize(glm::angleAxis(glm::radians(Yaw), FVector::UpVector) \* \*this);

UpdateEditorAngles();

}

void FQuat::AddRoll(float Roll)

{

\*this = glm::normalize(\*this \* glm::angleAxis(glm::radians(Roll), FVector::ForwardVector));

UpdateEditorAngles();

}

void FQuat::AddWorldRoll(float Roll)

{

\*this = glm::normalize(glm::angleAxis(glm::radians(Roll), FVector::ForwardVector) \* \*this);

UpdateEditorAngles();

}

1. Системы координат и пространства

В компьютерной графике и 3D-моделировании корректное описание положения и ориентации объектов требует четкого понимания систем координат и их взаимных преобразований. В данном разделе рассматриваются ключевые концепции, связанные с системами координат, используемыми при работе с 3D-объектами.

При работе с 3D-графикой используются несколько взаимосвязанных систем координат:

1. Локальная система координат (Object Space) - система, связанная с конкретным 3D-объектом. В этой системе задается исходная геометрия объекта, причем начало координат обычно совпадает с его центром или точкой привязки (pivot point).
2. Мировая система координат (World Space) - глобальная система отсчета, относительно которой располагаются все объекты сцены. В этой системе определяется положение и ориентация каждого объекта.
3. Система координат камеры (View Space) - система, связанная с виртуальной камерой. Все объекты сцены преобразуются в эту систему для последующего проецирования на экран.

Переход между системами координат осуществляется с помощью матриц преобразований:

1. Матрица модели (Model Matrix) - преобразует координаты из локальной системы объекта в мировую систему. Эта матрица включает в себя информацию о положении, ориентации и масштабе объекта.
2. Матрица вида (View Matrix) - преобразует мировые координаты в систему координат камеры. Эта матрица учитывает положение и ориентацию камеры в мировом пространстве.
3. Матрица проекции (Projection Matrix) - выполняет перспективное или ортографическое проецирование трехмерных координат на двумерную плоскость экрана.

Координата вершины преобразуется в координаты пространства отсечения следующим образом и представлена в листинге 5:

Листинг 5 – Реализация расчета матриц проекции и вида.

FMatrix RCameraComponent::GetProjectionMatrix() const

{

auto Engine = REngine::GetEngine();

RCheckReturn(Engine, {});

auto Editor = Engine->GetEditor();

RCheckReturn(Editor, {});

auto Frame = Editor->GetFrame();

RCheckReturn(Frame, {});

const FIntVector FrameSize = Frame->GetFrameSize();

const GLfloat FrameRatio = static\_cast<GLfloat>(FrameSize.x) / static\_cast<GLfloat>(FrameSize.y);

return glm::perspective(glm::radians(FOV), FrameRatio, 0.1f, 100.0f);

}

FMatrix RCameraComponent::GetViewMatrix() const

{

const FVector WorldPosition = GetWorldPosition();

return glm::lookAt(WorldPosition, WorldPosition + GetForwardVector(), GetUpVector());

}

Для удобства выполнения аффинных преобразований в 3D-графике используются однородные координаты - расширение трехмерных координат до четырехмерных. В однородных координатах:

1. Точка представляется как (x, y, z, 1)
2. Вектор представляется как (x, y, z, 0)

Использование однородных координат позволяет объединять линейные преобразования (вращение, масштабирование) и трансляцию (перемещение) в одной матрице размером 4×4.

ЗАКЛЮЧЕНИЕ

Проведенное исследование позволило разработать комплексную систему преобразований 3D-объектов, эффективно сочетающую матричные и кватернионные подходы. В ходе работы были успешно решены ключевые задачи компьютерной графики, связанные с представлением и интерполяцией пространственных преобразований. Особое внимание уделялось устранению принципиальных ограничений матричного подхода, таких как блокировка карданова подвеса и накопление вычислительных ошибок при последовательных преобразованиях. Реализованная система демонстрирует устойчивую работу в различных сценариях - от простых вращений отдельных объектов до сложных иерархических анимаций.

Основным теоретическим результатом работы стало строгое обоснование преимуществ кватернионного представления вращений для задач трехмерной графики. Практическая значимость подтверждается разработанными оптимизациями, включая кэширование преобразований, векторные вычисления и эффективные алгоритмы композиции вращений. Особого внимания заслуживает предложенный метод гибридного хранения трансформаций, когда для внутренних вычислений используются кватернионы, а для рендеринга - матрицы, что позволило сохранить совместимость с существующими графическими конвейерами.

Перспективы дальнейшего развития исследования видятся в нескольких направлениях. Во-первых, представляет интерес адаптация системы для работы с деформируемыми объектами и мягкими телами. Во-вторых, актуальной задачей является оптимизация вычислений для массовых параллельных преобразований на GPU. Особенно важным представляется исследование возможностей применения результатов работы в областях виртуальной и дополненной реальности, где требования к плавности и точности преобразований особенно высоки. Полученные результаты уже сейчас могут быть полезны разработчикам графических приложений и инженерам, работающим с трехмерными визуализациями.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Уроки по OpenGL с сайта OGLDev / [Электронный ресурс] // Уроки по OpenGL с сайта OGLDev : [сайт]. — URL: <https://triplepointfive.github.io/ogltutor/> (дата обращения: 04.06.2025).
2. Joey de Vries OpenGL / Joey de Vries [Электронный ресурс] // Learn OpenGL : [сайт]. — URL: <https://learnopengl.com/> (дата обращения: 04.06.2025).
3. Song Ho Ahn (안성호) OpenGL / Song Ho Ahn (안성호) [Электронный ресурс] // OpenGL : [сайт]. — URL: <https://www.songho.ca/opengl/> (дата обращения: 30.11.2024).
4. Sam Buss 3D Computer Graphics: A Mathematical Introduction with (Modern) OpenGL / Sam Buss [Электронный ресурс] // 3D Computer Graphics: A Mathematical Introduction with (Modern) OpenGL : [сайт]. — URL: <https://mathweb.ucsd.edu/~sbuss/MathCG2/>(дата обращения: 04.06.2025).
5. Куров, В. Л. Компьютерная графика: модели и алгоритмы [Текст] / Куров, В. Л. — 1-е издание. — М.: Наука, 2023 — 752 c.
6. Khronos Group OpenGL Programming Guide / Khronos Group [Электронный ресурс] // OpenGL Programming Guide : [сайт]. — URL: https://www.khronos.org/opengl/wiki/ (дата обращения: 04.06.2025).
7. Epic Games Unreal Engine 5.2 Documentation / Epic Games [Электронный ресурс] // Unreal Engine 5.2 Documentation : [сайт]. — URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-2-documentation?application\_version=5.2 (дата обращения: 04.06.2025).