УДК 004.92 URL: https://ptsj.bmstu.ru/catalog/icec/auto/1045.html

ПРИМЕНЕНИЕ БИБЛИОТЕКИ VULKAN ДЛЯ ВИЗУАЛИЗАЦИИ ГЕОМЕТРИЧЕСКИХ МОДЕЛЕЙ

Д.Ф. Хакимова khakimovadf@student.bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Работа посвящена анализу и разработке инструмента визуализации (визуализатора) трехмерных геометрических моделей для системы автоматизированного проектирования (САПР) с применением библиотеки Vulkan. Представлен алгоритм триангуляции геометрических моделей, рассмотрены последующие этапы обработки данных визуализатором. Выполнен обзор таких инструментов, как OpenGL, Mantle, DirectX, Metal и Vulkan, по результатам которого обоснован выбор библиотеки Vulkan как средства эффективного управления аппаратными ресурсами и обеспечения максимальной производительности. Представлен алгоритм триангуляции геометрических моделей, рассмотрены последующие этапы обработки данных визуализатором. Рассмотрены основные компоненты визуализатора для САПР, разобран принцип их работы, приведено описание программной реализации визуализатора трехмерных моделей, включая алгоритм визуализации трехмерных моделей и управления камерой.

Ключевые слова: САПР, геометрическая модель, визуализация, триангуляция, Vulkan, графический API, графический конвейер, рендеринг

Введение. Графическая подсистема (подсистема машинной графики) является важной частью современных систем автоматизированного проектирования (САПР) [1]. Визуализация в САПР позволяет лучше представить форму проектируемой детали или механизма и выявить возможные проблемы на ранних этапах разработки [2, 3]. Для получения качественного объемного изображения используют развитые инструменты визуализации (визуализаторы) трехмерных объектов (рис. 1).

При разработке САПР компания может купить готовый специализированный визуализатор, который нужно будет интегрировать с разработанным геометрическим ядром либо создать собственный [4]. Однако большая часть готовых инструментов визуализации изначально создавалась для игровой индустрии, поэтому они не соответствуют особым требованиям инженерных САПР-приложений. Программы для визуализации геометрических моделей в САПР должны обеспечивать возможность создания, редактирования и просмотра 3D-моделей на экране. Для этого необходимо передать рассчитанные геометрическим ядром точки в визуализатор, разработать алгоритм визуали-

 ISSN 2541-8009

зации, модуль управления камерой, пользовательский интерфейс и модуль управления элементами модели. В данной работе представлен обзор основных библиотек для трехмерной визуализации геометрических моделей, рассмотрен этап подготовки исходных данных к визуализации, алгоритм триангуляции и приведено описание разработанного модуля визуализации.

Рис. 1. Визуализация модели механизма Фергюсона в САПР «Компас-3D» [3]

Обзор инструментов для визуализации. Для создания визуализатора можно использовать различные графические библиотеки и API (Application programming interface). Графический API — программный интерфейс, который предоставляет функции, позволяющие графическому приложению использовать вычислительные ресурсы видеокарты: графический процессор, видеопамять. Наиболее популярными графическими API на данный момент являются OpenGL, Vulkan, DirectX и Metal.

OpenGL некоторое время был стандартом в индустрии графических API. Он поддерживается на широком спектре устройств и операционных систем, обладает большим набором графических возможностей: текстурирование, освещение, трансформация и прозрачность [5]. Однако производительность OpenGL ограничена его архитектурой, количеством доступных потоков для параллельной обработки, поэтому данный API считается устаревшим.

API Mantle был разработан компанией AMD с целью предоставления разработчикам прямого доступа к графическому процессору и улучшения эффективности взаимодействия с ним [6]. Mantle является низкоуровневым API, он позволяет разработчикам оптимизировать использование ресурсов

и достичь повышенной производительности в играх и графических приложениях. Он также предоставляет доступ к некоторым особенностям аппаратной части AMD, что может быть полезно при создании специализированных решений для этой платформы. Однако поддержка и развитие Mantle были приостановлены. Разработчики компании AMD увидели потенциал в новом Vulkan API и предоставили создателям Vulkan свое решение для совместной работы.

Vulkan является эволюционным продолжением OpenGL. Он разработан с учетом современных требований к производительности и масштабируемости. Данный API позволяет разработчикам эффективно использовать ресурсы и многопоточность, предоставляет возможность полностью контролировать выделение и использование памяти, что ведет к повышению производительности в сложных графических приложениях. Со спецификацией Vulkan API можно ознакомиться в источнике [7]. Vulkan является кроссплатформенным API, он предоставляет низкий уровень абстракции и обширный доступ к графическим возможностям устройства. Это может быть полезно для опытных разработчиков, работающих на конкретной платформе. Однако из-за своей относительной новизны Vulkan поддерживается не на всех платформах.

Набор API DirectX был разработан для решения задач, связанных с программированием под Microsoft Windows [8]. Данный API поддерживает множество эффектов, текстур, шейдеров и других технологий, но предназначен и оптимизирован исключительно для Windows, что обеспечивает более высокую производительность и качество графики на этих системах по сравнению с другими API. DirectX 12 считается более низкоуровневой системой, полноценно использует многоядерные процессоры и позволяет разработчикам получать полный доступ к ресурсам компьютера, как в API Vulkan. Однако различия между данными API довольно сильные [9]. Традиционно считается, что под DirectX лучше оптимизированы видеокарты от NVIDIA, а под Vulkan — продукты AMD.

Metal является низкоуровневым графическим API. Он предназначен для разработки приложений, использующих графику и вычисления на устройствах компании Apple [10]. Metal обеспечивает прямой доступ к графическим ядрам устройства, обладает высокой производительностью, поддерживает многопоточность и реалистичное отображение графики. До 2018 года компания Apple использовала OpenGL наряду с Metal, однако сейчас устройства этой компании поддерживают только Metal API.

На основании приведенного анализа библиотек для создания визуализации выбран интерфейс Vulkan API. Данный API современный, обеспечивает высокую производительность и обладает всеми возможностями, которые

 ISSN 2541-8009

предоставляет DirectX 12, и при этом он является кроссплатформенным, что позволяет использовать созданную программу на различных устройствах. На данный момент компании, занимающиеся созданием визуализаторов для САПР, уходят от использования OpenGL и работают над внедрением VulkanAPI в свои продукты.

Подготовка исходных данных для визуализации. Исходными данными для визуализатора служат трехмерные геометрические модели, генерируемые геометрическим ядром САПР. Ядро выполняет все расчеты, необходимые для построения 2D-эскизов и 3D-моделей [1]. Существует множество способов создания объемных тел:

– выдавливание контура в определенном направлении;

– тела, полученные операцией вращения контура вокруг оси;

– заметание контура вдоль направляющей кривой;

– добавление геометрических примитивов (куб, цилиндр, сфера, пирамида, тор)

В результате работы алгоритмов для создания объемных тел получается геометрическая модель, представленная в общем случае набором поверхностей и линий их пересечения, которые нужно отобразить на экране (визуализировать).

Процесс визуализации трехмерных объектов в современных графических приложениях проходит через ряд этапов [11]:

– получение триангуляции исходной поверхности;

– обработка команд пользователя по управлению камерой;

– визуализация.

Триангуляцией называется планарное разбиение плоскости на M фигур, из которых одна является внешней бесконечной, а остальные — треугольниками. Задачей построения триангуляции по заданному набору двумерных точек называется задача соединения заданных точек непересекающимися отрезками так, чтобы образовалась триангуляция [12].

Оптимальная триангуляция позволяет получить такой граф, сумма длин всех ребер которого минимальна среди всех возможных триангуляций, построенных на тех же исходных точках. Однако для большинства реальных задач существующие алгоритмы построения оптимальной триангуляции неприемлемы ввиду слишком высокой трудоемкости. На практике применяют приближенные алгоритмы, одним из которых является следующий жадный алгоритм построения триангуляции.

1. Генерируется список всех возможных отрезков, соединяющих пары исходных точек, после чего этот список сортируется по длинам отрезков.

2. Начиная с самого короткого последовательно выполняется вставка отрезков в триангуляцию. Если отрезок не пересекается с другими ранее вставленными отрезками, то он вставляется, иначе он отбрасывается.

Вычислительная сложность жадного алгоритма при некоторых его улучшениях составляет $O(N^2 \lg N)$, где N — число элементов. В связи со столь большой вычислительной сложностью на практике он почти не применяется. Кроме оптимальной и жадной триангуляции также широко известна триангуляция Делоне.

Триангуляция Делоне — выпуклая триангуляция, в которой внутрь окружности, описанной вокруг любого построенного треугольника, не попадает ни одна из заданных точек триангуляции. Ее основные особенности заключаются в том, что она обладает максимальной суммой минимальных углов всех своих треугольников среди всех возможных триангуляций, а также она обладает минимальной суммой радиусов окружностей, описанных около треугольников, среди всех возможных триангуляций [12].

Модуль управления камерой. Модуль управления камерой в визуализаторе для САПР обеспечивает пользователю возможность просмотра с разных сторон трехмерной модели, детализированный просмотр небольших элементов в увеличенном отображении. Камера представляет собой объект, имеющий собственные матрицы преобразования вида и проекции.

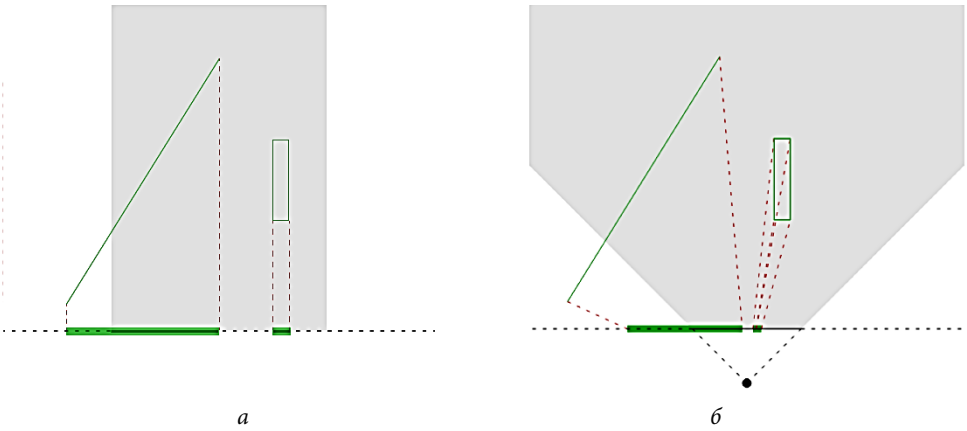
Матрица преобразования вида определяет положение и ориентацию камеры в пространстве. Она преобразует мировые координаты объектов в координаты видового пространства. Это позволяет определить, как сцена будет видна со стороны камеры.

Проекция в терминах рендеринга — это способ преобразования мира из одной размерности в другую [13]. При ортографической проекции ось или плоскость проекций перпендикулярна (ортогональна) направлению проектирования (рис. 2, *а*). Однако человеческий глаз не видит мир через ортографическую проекцию, поэтому в визуализаторе для САПР была использована перспективная проекция — это проекция трехмерных объектов на двухмерную плоскость с учетом их расстояния до точки наблюдения (рис. 2, *б*). В 2D форма перспективной проекции представляет собой правильную трапецию, в 3D такая форма называется усеченной пирамидой. Матрица преобразования проекции позволяет менять угол обзора и дальность отсечения видимости для перспективной проекции.

Управление камерой в приложениях САПР должно быть интуитивно понятным и точным. Для обработки событий в разработанном модуле визуализации используется библиотека GLFW. Изменение матриц преобразования происходит при нажатии клавиш на клавиатуре. Клавиши W, A, S, D позво-

 ISSN 2541-8009

ляют перемещать камеру вперед, влево, назад и вправо, клавиши Q, E — вверх и вниз. Поворот камеры относительно осей X и Y осуществляется нажатием стрелок на клавиатуре.



а *б*

Рис. 2. Виды проекций:

а — ортогональная; *б* — перспективная; сцена проецируется на черную пунктирную линию, серый фон представляет часть мира, видимую для проекции; части сцены за пределами этой области не видны; черная точка на рисунке *б* обозначает зрителя [14]

Алгоритм визуализации. На основе документации Vulkan [13] и руководств [15, 16] разработан алгоритм визуализации 3D-модели с помощью Vulkan API. Рассмотрим разработанный алгоритм.

1. Создание графического окна приложения.

2. Создание экземпляра Vulkan. Экземпляр Vulkan собирает все поддерживающие Vulkan устройства вместе, логически отделяет состояние создаваемого приложения от других приложений.

3. Выбор физического устройства. Физическое устройство — графические карты, ускорители или другие компоненты.

4. Создание логического устройства. Логическое устройство — это программная конструкция, оборачивающая физическое устройство и представляющая зарезервированный набор ресурсов, связанных с конкретным физическим устройством.

5. Создание графического конвейера. Графический конвейер — это последовательность операций, в результате которой вершины и текстуры геометрического объекта преобразуются в пиксели на экране.

6. Создание цепочки обмена изображениями. Цепочка обмена изображениями меняет положения кадров с определенной частотой.

7. Создание изображений и буферов. После обработки графическим конвейером подготовленное к выводу изображение отправляется в буфер кадров. Для графического приложения нужно несколько таких буферов, так как пока из одного кадр выводится на экран с помощью цепочки обмена, во второй загружается новый кадр. Третий буфер позволяет уменьшить время задержки кадра.

8. Создание командных буферов. Командные буферы служат для хранения команд, которые будут выполнены на графическом процессоре.

9. Запуск цикла, который ожидает от пользователя закрытие окна.

10. Нахождение матриц перемещения и поворота зрителя, изменение положения заднего фона и зрителя.

11. Расчет матрицы вида объекта камеры.

12. Установление перспективной проекции.

13. Обработка трехмерного объекта графическим конвейером, которая состоит из семи этапов:

а) получение геометрических данных для рендеринга из вершинного буфера. В коде программы вершинный буфер заполняется координатами, полученными из геометрического модуля методической САПР;

б) применение вершинного шейдера. Вершинный шейдер — это программа, которая производит математические операции с вершинами, изменяет их параметры и освещение;

в) этап тесселяции. На этапе тесселяции примитивы разбиваются на более мелкие части, что позволяет гибко управлять детализацией объектов в зависимости от расстояния до наблюдателя;

г) применение геометрического шейдера. Геометрически шейдер обладает возможностью изменять тип примитива для отрисовки. Примитивы могут быть точками, отрезками, треугольниками или их специальными вариантами;

д) этап растеризации. Растеризация — фундаментальная часть всей графики в Vulkan. Во время растеризации примитивы раскладываются на множество пиксельных элементов, каждый из которых имеет свои атрибуты на основе значений, вычисленных ранее;

е) применение фрагментного шейдера. Фрагментный шейдер определяет цвета пикселей и другие их атрибуты, также он может выполнять текстурирование, накладывание теней, освещения и отображать другие визуальные эффекты;

ж) этап смешивания цветов. На этапе смешивания цветов происходит смешивание различных фрагментов, относящихся к одному и тому же пикселю в буфере кадров. Фрагменты могут перекрывать друг друга или смешиваться в зависимости от прозрачности.

 ISSN 2541-8009

14. Возврат к началу цикла.

15. Если окно закрыто, то очистить ресурсы.

Основной метод разработанного инструмента визуализации изображен в виде блок-схемы на рис. 3. При добавлении модуля управления камерой к циклу ожидания закрытия окна в алгоритме визуализации добавляется получение информации о текущем времени кадра, произошедших событиях, нажатых клавишах. На основе этой информации меняется положение зрителя.

Рис. 3. Блок-схема основного метода

Примеры визуализации. На рис. 4 продемонстрирован результат работы визуализатора для методической САПР. Из геометрического модуля методической САПР были получены вершины и индексы объекта, на основе которых была создана модель, представленная на рис. 4.

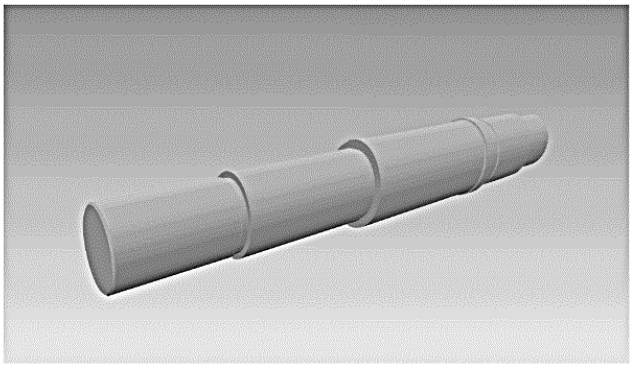


Рис. 4. Отображение трехмерной модели вала
с помощью созданного визуализатора

Заключение. Современные графические библиотеки визуализации, такие как OpenGL, Vulkan, Metal, обладают богатыми возможностями по графическому представлению трехмерных объектов. Однако для создания инструмента визуализации геометрических моделей для САПР-приложений необходимо решить задачи подготовки исходных данных, включая триангуляцию поверхностей, и управления камерой. Представленные в статье алгоритмы подготовки исходных данных, управления камерой и трехмерного представления модели с использованием библиотеки Vulkan реализованы в инструменте визуализации. Реализованный инструмент позволяет отобразить геометрию модели с разных ракурсов — благодаря управлению камерой пользователь имеет возможность изменять масштаб, вращать и перемещать модель на экране. Таким образом, представленный в статье инструмент является прототипом графической подсистемы САПР-приложения, который может быть использован в методических целях для знакомства с работой подсистем машинной графики САПР.

Литература

[1] Норенков И.П. *Основы автоматизированного проектирования*. Москва, Изд-во МГТУ им. Н.Э. Баумана, 2009, 430 с.

[2] Тарабарин В.Б., Козов А.В. Исследование структуры, кинематики и 3D-моделирование механизмов Фергюсона. *Наука и образование: научное издание МГТУ им. Н.Э. Баумана*, 2015, № 6, с. 517–532.

[3] Tarabarin V., Kozov A. Analysis of structure, kinematic and 3D-modeling of Ferguson's mechanisms. *History of Mechanism and Machine Science*, 2016, vol. 32, pp. 183–194. https://doi.org/10.1007/978-3-319-31184-5_17

[4] Максименко Э. *Визуализация в САПР: зачем мы написали еще один 3D-движок и как он работает.* URL: https://habr.com/ru/companies/ascon/articles/354636/ (дата обращения 06.09.2024).

[5] Shreiner D., Sellers G., Kessenich J. *OpenGL programming guide: the official guide to learning OpenGL, version 4.3.* Bill Licea-Kane, the Khronos OpenGL ARB Working Group, Addison-Wesley, 2013, 935 p.

[6] Smith R. *Understanding AMD’s Mantle: A Low-Level Graphics API For GCN*. URL: https://www.anandtech.com/show/7371/understanding-amds-mantle-a-lowlevel-graphics-api-for-gcn (дата обращения 02.11.2024).

[7] *Khronos Vulkan Working Group. Vulkan 1.3.295 — A Specification (with all registered extensions)*. URL: https://registry.khronos.org/vulkan/specs/1.3/html/ (дата обращения 09.09.2024).

 ISSN 2541-8009

[8] *Руководство по программированию для DirectX 12*. URL: https://learn.microsoft.com/ru-ru/windows/win32/direct3d12/directx-12-programming-guide (дата обращения 09.09.2024).

[9] Shiraef J., Liang Y., Kizza J., Tanis C. *An exploratory study of high performance graphics application programming interfaces*. The University of Tennessee at Chattanooga, 2016, 87 p.

[10] *Apple Developer, Metal Overview*. URL: https://developer.apple.com/metal/ (дата обращения 11.09.2024).

[11] Дижевский А.Ю. Общий подход к реализации методов построения триангуляций неявно заданных поверхностей, использующих разбиение пространства на ячейки. *Вычислительные методы и программирование*, 2007, т. 8, с. 286–296. URL: https://num-meth.ru/index.php/journal/article/view/270 (дата обращения 14.11.2024).

[12] Скворцов А.В. Обзор алгоритмов построения триангуляции Делоне. *Вычислительные методы и программирование*, 2002, № 3, с. 14–39. URL: https://num-meth.ru/index.php/journal/article/view/45 (дата обращения 14.11.2024).

[13] *Vulkan Documentation*. URL: https://docs.vulkan.org/spec/latest/index.html (дата обращения 09.09.2024).

[14] Jason L. *McKesson Learning Modern 3D Graphics Programming*. 2012, 360 p. URL: https://paroj.github.io/gltut/ (дата обращения 02.11.2024).

[15] Overvoorde A. *Vulkan Tutorial*. 2023, 286 p. URL: https://vulkan-tutorial.com/resources/vulkan_tutorial_en.pdf (дата обращения 14.11.2024)

[16] Селлерс Г. *Vulkan*. Руководство разработчика. Москва, ДМК Пресс, 2017, 394 с.

Поступила в редакцию 25.01.2025

Хакимова Диляра Фаридовна — студентка кафедры «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Научный руководитель — Козов Алексей Владимирович, кандидат технических наук, старший преподаватель кафедры «Системы автоматизированного проектирования», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Ссылку на эту статью просим оформлять следующим образом:
Хакимова Д.Ф. Применение библиотеки Vulkan для визуализации геометрических моделей. *Политехнический молодежный журнал*, 2025, № 3 (98). URL: https://ptsj.bmstu.ru/catalog/icec/auto/1045.html

VULKAN LIBRARY APPLICATION IN VISUALIZING THE GEOMETRIC MODELS

D.F. Khakimova khakimovadf@student.bmstu.ru

Bauman Moscow State Technical University, Moscow, Russian Federation

The paper is devoted to analyzing and developing a visualization tool (visualizer) for the three-dimensional geometric models in the computer-aided design (CAD) system using the Vulkan library. It presents an algorithm for triangulating the geometric models, and considers subsequent stages in data processing by the visualizer. A review of such tools as OpenGL, Mantle, DirectX, Metal, and Vulkan is provided substantiating selection of the Vulkan library as a means of efficient management of the hardware resources ensuring the maximum performance. The paper presents an algorithm for triangulating the geometric models, and considers subsequent stages in data processing by the visualizer. Main components of the CAD visualizer are studied, and their operating principle is analyzed. The paper provides a description of the three-dimensional model visualizer software implementation, including the three-dimensional model visualization algorithm and the camera control algorithm.

Keywords: CAD, geometric model, visualization, triangulation, Vulkan, graphics API, graphics pipeline, rendering

Received 25.01.2025

Khakimova D.F. — Student, Department of Computer-Aided Design Systems, Bauman Moscow State Technical University, Moscow, Russian Federation.

Scientific advisor — Kozov A.V., Ph. D. (Eng.), Senior Lecturer, Department of Computer-Aided Design Systems, Bauman Moscow State Technical University, Moscow, Russian Federation.

Please cite this article in English as:
Khakimova D.F. Vulkan library application in visualizing the geometric models. *Politekhnicheskiy molodezhnyy zhurnal*, 2025, no. 3 (98). URL: https://ptsj.bmstu.ru/catalog/icec/auto/1045.html

 ISSN 2541-8009