DOI: 10.17117/na.2016.12.02.127

<http://ucom.ru/doc/na.2016.12.02.127.pdf>

Поступила (Received): 21.12.2016

Сесин И.Ю., Холкин И.И.**К выбору программно-аппаратных средств,
применяемых для разработки рендереров****Sesin I.Yu., Kholkin I.I.****Software and hardware used for renderer development**

В статье рассматриваются средства, применяемые для написания и тестирования рендереров. Делается упор на охват большинства популярных подходов к написанию рендереров на текущий момент, но также рассматриваются и нетипичные требования к оборудованию и программному обеспечению вследствие особенностей применяемого алгоритма рендеринга

Ключевые слова: рендеринг, рендерер, программное обеспечение, аппаратное обеспечение

Article covers hardware and software requirements for purposes of renderer development. Article focuses on requirements befit for general setups widely used for renderer development presently. However, some specific cases stemming from traits of algorithms used are also covered

Key words: rendering, renderer, software, hardware

Сесин Игорь Юрьевич

Бакалавр, магистрант

Московский технологический университет

г. Москва, пр. Вернадского, 78

Sesin Igor Yurevich

Bachelor, master

Moscow technological university

Moscow, Vernadskogo ave., 78

Холкин Игорь Иванович

Кандидат технических наук, доцент

Московский технологический университет

г. Москва, пр. Вернадского, 78

Kholkin Igor Ivanovich

Candidate of Engineering Sciences, Associate

Professor

Moscow technological university

Moscow, Vernadskogo ave., 78

Рендеринг (англ. rendering – «визуализация») в терминах компьютерной графики – процесс получения растрового изображения, исходя из некоторого модельного описания визуализируемых объектов с использованием ЭВМ для выполнения ряда необходимых расчетов [1]. Рендерером же называют программное средство или программный комплекс, осуществляющий рендеринг.

Соответственно, для осуществления разработки рендерера необходимо наличие ЭВМ с программным обеспечением, ориентированным на разработку ПО, а конкретно, включающим в себя среду разработки и пакет компиляции или интерпретации языка в машинный код.

Классически для разработки приложений, в которых критично быстродействие, использовались языки С и С++, иногда с использованием ассемблерных вставок.

Однако, с развитием технологий ситуация изменилась – в большинстве комплектаций персональных компьютеров теперь присутствуют видеокарты, снабженные графическими процессорами. Эти графические процессоры специально разрабатывались для ускорения ряда узконаправленных графических расчетов, и они ориентированы на проведение большого количества параллельных вычислений с числами с плавающей точкой.

Подобная доступность высоких вычислительных мощностей не прошла незамеченной, и было создано много инструментов, упрощающих рядовому программисту доступ к графическому процессору.

Таким образом, появилось множество самых разнообразных решений для реализации расчетов рендерера на видеокарте – от OpenGL API или DirectX до тяжеловесных решений вроде CUDA.

Иначе говоря, исчезло жесткое ограничение на язык, на котором следует разрабатывать рендерер – оно по-прежнему существует, но гораздо в более мягкой форме, и зависит от конкретного применяемого метода рендеринга.

Для большинства классических методов рендеринга хорошо подходит расчет напрямую в шейдере – это означает, что основную математику расчета придется написать на GLSL (OpenGL Shading Language [2])/HLSL (High-Level Shader Language), однако остальная часть рендерера может быть реализована на любом языке, для которого имеется используемая графическая библиотека.

В частности, на текущий момент, одним из популярных подходов является разработка рендереров в JavaScript с использованием интерфейса, предоставленного WebGL.

С другой стороны, если выбранный метод рендеринга не может быть эффективно рассчитан при помощи шейдеров, для того, чтобы задействовать все вычислительные мощности видеокарты, необходимо использовать методы GPGPU (General-purpose computing for graphical processing units – неспециализированные вычисления на графических процессорах). Говоря простым языком, методы GPGPU позволяют запускать непосредственно код программы на видеокарте.

Однако использование этого подхода немедленно накладывает ограничение на используемый язык – например, применение CUDA SDK возможно только с определенным набором языков, в частности, с C, C++ [3] и Fortran [4].

Применение OpenCL API требует владения C-подобным языком, на котором пишутся программы, выполняемые на вычислительных устройствах (не всегда GPU).

Исходя из исторического аспекта можно видеть, что множество языков, используемых для написания высокопроизводительных рендереров, несомненно, расширилось, но применение нетривиальных методов рендеринга по-прежнему сопряжено с использованием узкого круга высокопроизводительных и, зачастую, низкоуровневых языков.

Что касается средств разработки – как правило, работают те же принципы, что и обычно, то есть стоит выбирать такую среду, в которой наиболее комфортно работать конкретной команде программистов. Разработка рендереров, как таковая, обычно не накладывает дополнительных ограничений, однако

иногда имеются оптимальные комбинации среды разработки и используемого API, когда среда поддерживает интеграцию API. Так же желательно, но не обязательно, иметь редактор растровых изображений с функцией сравнения изображений – это помогает сравнивать результаты тестовых прогонов с целью поиска неочевидных артефактов из-за ошибок округления, проверки на то, что рендерер действительно реализует рендеринг без допущений (англ. unbiased rendering). Помимо этого, при отладке программ шейдеров удобно иметь программу, просто отображающую результат работы шейдера для устранения ошибок в «оболочке» рендерера.

Разберем требования к аппаратному обеспечению. Так как в процессе разработки программного средства-рендерера его, так или иначе, потребуется тестировать, это накладывает определенные ограничения на используемую для тестирования ЭВМ.

Для осуществления рендеринга изображений большого размера за приемлемое время требуются аппаратные средства с большими вычислительными мощностями, так как в подавляющем большинстве техник рендеринга требуется проведение большого количества расчетов.

Это означает, что ЭВМ, на которой проходит тестирование рендерера (может и не совпадать с ЭВМ, используемой для разработки), должна обладать адекватными вычислительными мощностями для многократного тестирования рендерера.

Необходимо отметить, что не все методы рендеринга могут быть хорошо реализованы в условиях параллельных расчетов на GPU – существуют методы, которые не могут быть исполнены параллельно. Помимо этого, существуют методы, опирающиеся на особенности работы CPU с числами с плавающей точкой [5] или на работу с памятью вместо тяжеловесных расчетов. В определенных случаях при наличии большого количества ветвления в коде рендерера и использовании методов оптимизации, таких как таблицы поиска, расчет таким методом на CPU может быть быстрее, нежели чем только на GPU! Следовательно, выбирая алгоритм, скорость CPU тоже надо брать в расчет.

Итак, для хорошо распараллеливаемых алгоритмов следует использовать мощные GPU для обсчета шейдеров, CPU и оперативная память не принципиальны, пока они не становятся узким местом системы, и в качестве языка может использоваться практически любой язык, для которого существуют библиотеки OpenGL или подобные ей.

Для нетривиальных алгоритмов по-прежнему рекомендуются низкоуровневые языки вроде C/C++ и набор инструментов CUDA или OpenCL, которые так же подразумевают использование мощного GPU. Важно, что CPU начинает играть большую роль, так как вычислительная мощь CPU влияет на скорость компиляции и сборки программы, что играет немаловажную роль в разработке больших проектов.

Так же стоит отметить важность наличия достаточного количества оперативной памяти, так как зачастую специфика работы рендереров подразумевает

создание изображений большого размера, или множества изображений. При создании масштабируемых рендереров стоит заранее предусматривать механизм потоковой обработки и выгрузки изображения.

Список используемых источников:

1. Сесин И.Ю., Холкин И.И. Рендеринг – состояние и перспективы // Научный альманах. 2015. N 10-3(12). С. 233-236.
2. J. Kessenich, D. Baldwin, R. Rost. GLSL Language Specification, Version 1.10.59.
URL: <https://www.opengl.org/registry/doc/GLSLangSpec.Full.1.10.59.pdf>
3. NVIDIA CUDA Toolkit. URL: <https://developer.nvidia.com/cuda-toolkit>
4. NVIDIA FORTRAN. URL: <https://developer.nvidia.com/cuda-fortran>
5. Сесин И.Ю., Холкин И.И. Проблематика использования чисел с плавающей точкой в задачах рендеринга // Научный альманах. 2016. N 4-3(18). С. 182-185.

© 2016, Сесин И.Ю., Холкин И.И.

К выбору программно-аппаратных средств,
применяемых для разработки рендереров

© 2016, Sesin I.Yu., Kholkin I.I.

Software and hardware used for renderer
development