УДК 004.921
DOI: 10.15827/2311-6749.17.3.5

ВИЗУАЛИЗАЦИЯ ВИРТУАЛЬНЫХ ТРЕХМЕРНЫХ СЦЕН НА ОДНОКРИСТАЛЬНЫХ СИСТЕМАХ

(Работа выполняется в рамках проекта № 2.9 программы фундаментальных исследований ОНИТ)

Гиацинтов А.М., зав. отделом, algts@inbox.ru;
Баженов П.С., ведущий программист, pav.bazhenov@gmail.com
(Центр визуализации и спутниковых информационных технологий ФНЦ НИИ системных исследований РАН, Нахимовский просп., 36, к. 1., г. Москва, 117218, Россия)

В статье рассмотрена технология визуализации виртуальных трехмерных сцен на однокристальных системах с использованием программного интерфейса OpenGL ES, а также общая структура графического модуля, обеспечивающего отображение виртуального окружения и способы адаптации графического модуля для работы с OpenGL ES. Выявлены ограничения визуализации трехмерных сцен на однокристальных системах и предложены способы их устранения.

Ключевые слова: *визуализация, тренажерно-обучающая система, однокристальная система, графический модуль, OpenGL ES, шейдеры.*

Для подготовки операторов сложных технических систем разработана *тренажерно-обучающая система* (ТОС) – техническое средство, отвечающее требованиям методик подготовки, обеспечивающее получение знаний, навыков и умений, реализующее модель таких систем и осуществляющее контроль над действиями обучаемого. ТОС могут применяться для формирования индивидуальных профессиональных навыков и умений, для отработки групповых операций, а также для исследований [1]. Подсистема визуализации ТОС обеспечивает отображение результатов моделирования внешней среды и объекта управления с помощью устройств отображения информации. Она должна обеспечивать воспроизведение созданной виртуальной сцены с достаточно подробным содержанием, позволяющим операторам ТОС успешно выполнять поставленные задачи [2].

Однокристальная система, или система на кристалле (SoC), представляет собой интегральную схему, способную выполнять функции целого устройства (компьютера), элементы которого размещены на одном кристалле [3]. Особенности использования такой системы в том, что устройство в ней может быть компактным, потреблять мало энергии и, следовательно, выделять меньше тепла. Но вследствие этого однокристальные системы могут быть ограничены в вычислительных ресурсах [4]. Для работы с графикой в этих системах, как правило, используется графический интерфейс OpenGL ES.

Необходимо реализовать поддержку API OpenGL ES в графическом модуле. Существуют отличия графического API OpenGL ES от OpenGL. Например, OpenGL ES поддерживает работу с такими примитивами растеризации, как точки, линии и треугольники, но не поддерживает работу с квадами (quads). Также OpenGL ES не поддерживает фиксированный графический конвейер (fixed-function graphics pipeline). Вместо этого необходимо использовать вершинные и фрагментные шейдерные программы. В OpenGL ES отсутствуют функции для задания усеченной пирамиды видимости, необходимо вычислить свою собственную матрицу трансформации и передать ее в шейдер.

Общая структура графического модуля

Графический модуль представляет собой систему, обеспечивающую визуализацию виртуальной трехмерной сцены. Система графического модуля может иметь ряд компонентов: интерфейс взаимодействия, компоненты управления ресурсами, визуализацией, сценой, обработки шейдеров.

Интерфейс взаимодействия обеспечивает взаимосвязь между всеми компонентами системы. Он позволяет выполнить инициализацию всех компонентов, регистрирует типы ресурсов и узлов, а также функции прорисовки трехмерных объектов.

Компонент управления ресурсами обеспечивает загрузку, хранение, поиск и передачу ресурсов другим компонентам. Ресурс – это объект, необходимый для рендеринга сцены. Данный компонент позволяет работать с различными типами файлов: изображения (текстуры), шейдеры, модели, материалы, геометрические данные. Каждому ресурсу присваивается имя, уникальное для каждого типа ресурсов, при помощи которого этот ресурс смогут использовать различные объекты. Также он следит за тем, чтобы файлы были загружены только один раз и впоследствии могли использоваться повторно. Если на ресурс больше не ссылаются какие-либо объекты, он может быть удален. Таким образом, может производиться «сборка мусора» для освобождения вычислительных ресурсов. Графический модуль использует отло-

женную загрузку ресурсов. Это означает, что создание ресурса и его загрузка являются отдельными этапами. Преимуществом такого разделения является то, что ресурс не должен быть доступен сразу же после создания, а может быть загружен в фоне в отдельном потоке. При создании ресурса он инициализируется с данными по умолчанию, специфичными для конкретного типа данных, а другие объекты получают возможность ссылаться на этот ресурс. После создания ресурс может быть загружен и заполнен необходимыми данными. При создании ресурса задается его базовая структура.

Компонент управления визуализацией отвечает за прорисовку трехмерной графики с помощью графического интерфейса OpenGL ES. Он обеспечивает работу с текстурами, а именно возможность создавать и связывать текстуру с объектом и по необходимости производить замену текстуры, при этом модуль позволяет работать с различными поддерживаемыми текстурными форматами. Также в данном компоненте происходят создание и компиляция шейдерных программ. Значимой частью компонента является работа с буферами рендеринга. Для рендеринга используется подход программируемого конвейера. Ресурсом программируемого конвейера является XML-документ, описывающий этапы процесса рендеринга. Программируемый конвейер может определять цели рендеринга (render target) – выходные буферы, хранящие временные результаты рендеринга. Они могут быть различного размера и формата, а также содержать несколько буферов цвета и глубины. Несколько команд программируемого конвейера определяют, какая геометрия рендерится в какой буфер и каким образом. Выбор прорисовываемой геометрии определяют классы материалов, а техника рендеринга определяется выбранным контекстом шейдера. Команды объединены в этапы, каждый из которых можно включить или выключить. После заполнения буфера его можно использовать снова в качестве входной текстуры на следующем этапе рендеринга. Такое использование входных и выходных буферов позволяет реализовывать различные эффекты пост-обработки и улучшенные техники рендеринга.

Компонент управления сценой отвечает за работу с различными типами узлов виртуальной сцены: источники света, камера, геометрические модели или группа узлов. Данный компонент позволяет добавлять или удалять узел (или группу узлов) по запросу, а также осуществлять поиск узла по имени. Компонент управления сценой поддерживает расширение типов узлов, то есть имеется возможность добавления новых типов. Управление сценой происходит с помощью графа сцены. Граф сцены используется для представления логической или пространственной структуры сцены, которую необходимо визуализировать. Граф сцены реализован в виде дерева. Каждый узел может иметь неограниченное число дочерних узлов, но графический модуль накладывает некоторые ограничения на иерархию и определяет, какие типы узлов к какому типу родительского узла могут быть присоединены. Только такие сущности, как модели, частицы и т.д., имеющие трансформации и, соответственно, местонахождение в виртуальном мире, представляются как узлы. Все другие абстрактные сущности, такие как материалы, представлены как свойства узлов. Данный подход позволяет сократить количество узлов в графе и повысить производительность операций над графом сцены. Для каждого узла применимы два типа трансформаций – локальные и глобальные. Локальные трансформации, иногда также называемые трансформациями объекта, относительны к родительскому объекту и могут быть явно заданы в приложении. Глобальные трансформации рассчитываются автоматически путем прохода по иерархии родительского – дочерних объектов и накопления трансформаций. То есть, когда происходит трансформация узла, все его дочерние объекты также трансформируются.

Управление шейдерными программами осуществляется компонентом обработки шейдеров. Шейдер – это программа, которая выполняется на графическом процессоре. Шейдер обеспечивает отображение визуальных эффектов, таких как отражение, вода, погодные условия (туман, дождь, снег), частицы и т.д. (рис. 1). В разрабатываемом графическом модуле шейдеры разделяются на две секции: секция FX и секция кода шейдерных программ. В секции FX указываются глобальные переменные и семплеры (samplers), которые будут использоваться в шейдерах, а также шейдеры, используемые для конкретного контекста. Глобальные переменные передаются из программы в шейдер как константы. Семплеры предназначены для хранения текстурных данных. В секции кода находятся тексты шейдерных программ. При написании шейдеров используется подход uber shader. Он позволяет избавиться от написания множества шейдеров, которые могут отличаться, к примеру, только одной командой, представив

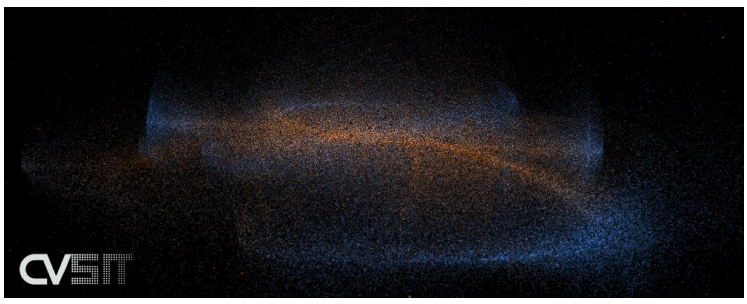

Рис. 1. Применение вычислительных шейдеров в графическом модуле

их в виде единого кода, в котором шейдеры разделяются с помощью флагов. Компонент обработки анализирует входной файл шейдеров, определяя наличие ошибок, и при успешном анализе формирует итоговую шейдерную программу, которая в последующем отправляется на компиляцию.

Ограничения визуализации трехмерных сцен на однокристальных системах

При адаптации трехмерных приложений, использующих графический интерфейс OpenGL, для работы на системах на кристалле, поддерживающих только графический API OpenGL ES, можно столкнуться с рядом ограничений. Одним из них является поддержка расширений (Extensions). Графические процессоры имеют заданный набор поддерживаемых расширений. Это приводит к тому, что, например, одни процессоры поддерживают определенный формат текстур, а другие не поддерживают, или одни поддерживают геометрические шейдеры, а другие нет. К примеру, графический процессор Mali-T628 от ARM не поддерживает формат RGBA16F (расширение EXT_color_buffer_float), что в определенных ситуациях может привести к неправильному отображению из-за недостатка точности для хранения информации. Также графические процессоры могут не поддерживать анизотропную фильтрацию, которая позволяет устранять артефакты текстур на поверхностях трехмерных объектов, при этом сохраняя более высокую детализацию изображения, чем другие методы фильтрации.

Еще одно ограничение – использование цветовых моделей текстур в OpenGL ES. По умолчанию в OpenGL применяется цветовая модель BGRA, в то время как в OpenGL ES используется RGBA, а на многих графических процессорах нет поддержки BGRA. Из-за этого текстуры могут отображаться с некорректными цветами. Для перевода значений из одной цветовой модели в другую применяется перестановка компонентов цвета (swizzling) пикселя текстуры.

Ограничения могут затронуть и шейдерные программы. Например, наиболее современные графические процессоры могут поддерживать геометрические и вычислительные шейдеры. Данные виды шейдеров могут служить хорошим инструментом для реализации сложных графических эффектов. В шейдерах на OpenGL ES необходимо указывать точность для вещественных типов данных: lowp, mediump, highp. Это связано с тем, что некоторые графические процессоры могут не поддерживать высокую точность вычислений. Одной из причин этого может служить ограничение на производительность процессора.

Для работы в режиме OpenGL ES в графический модуль был внедрен новый компонент управления визуализацией, использующий команды графического API OpenGL ES (рис. 2). В графическом модуле была добавлена проверка поддерживаемых используемым графическим процессором расширений OpenGL ES и их инициализация. В числе расширений, предоставляющих дополнительную функциональность для этого модуля, EXT_texture_filter_anisotropic, EXT_color_buffer_float, EXT_texture_compression_s3tc, EXT_disjoint_timer_query, EXT_texture_border_clamp, EXT_geometry_shader.

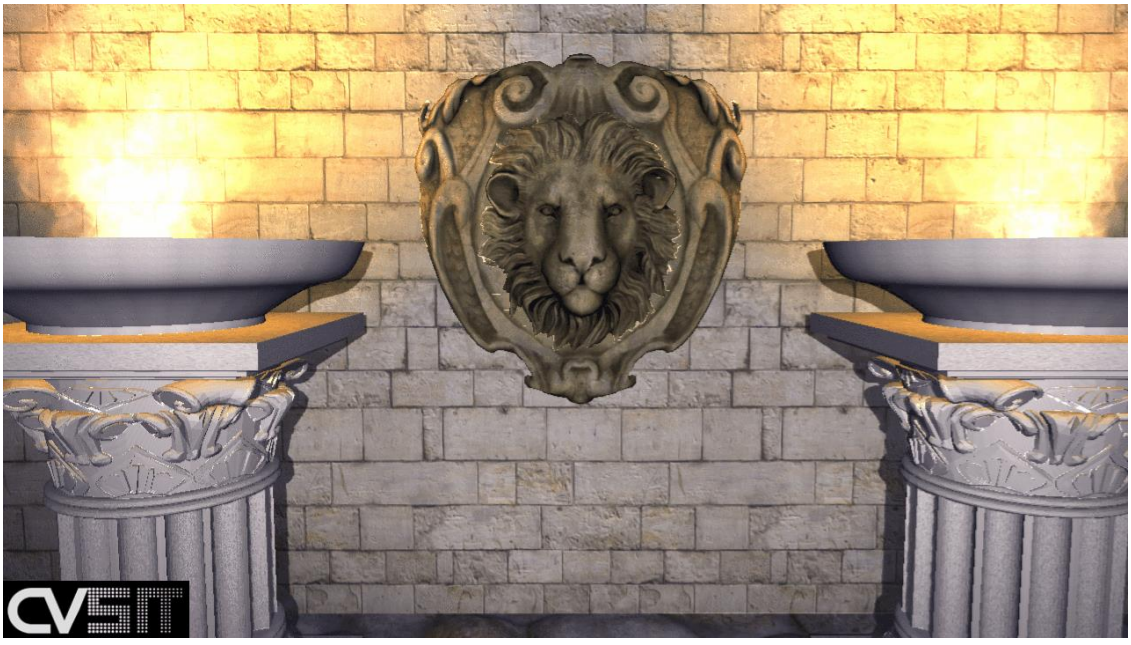

Рис. 2. Визуализация сцены с использованием опытного образца однокристальной системы

Для корректной работы шейдерных программ были определены значения точности типов данных с плавающей точкой. Из-за того, что в OpenGL ES не поддерживается (без использования расширений) цветовая модель BGRA, применяемая по умолчанию в OpenGL, было внедрено преобразование цветовой модели загружаемых текстурных данных из BGRA в RGBA.

Заключение

Рассматриваемая в работе структура графического модуля может быть использована для построения подсистем визуализации ТОС, обеспечивающих отображение высокореалистичного виртуального окружения. Особенностями структуры являются поддержка технологии хранения данных шейдеров uber shader, а также модульность и масштабируемость. Выявлены ограничения визуализации трехмерных сцен на однокристальных системах. Предложены подходы к адаптации структуры графического модуля, использующего API OpenGL, для работы с графическим API OpenGL ES. С помощью описанного графического модуля были созданы тесты для проверки корректного отображения трехмерных виртуальных сцен на однокристальных системах.

Литература

1. Гиацинтов А.М., Мамросенко К.А., Решетников В.Н. Инструментальные средства предтренажерной и тренажерной подготовки операторов сложных технических систем // Программные продукты, системы и алгоритмы. 2014. № 1. URL: http://swsys-ru/simulator-training-operators.html (дата обращения: 12.07.2017).

2. Гиацинтов А.М., Мамросенко К.А. Методы отображения трехмерных объектов при применении отложенной визуализации // Информационные технологии. 2016. № 7. С. 510–514.

3. Technologies P. System on a Chip. URL: http://www.palomartechnologies.com/applications/system-on-a-chip (дата обращения: 14.07.2017).

4. Aryashev S.I., Rogatkin B.Yu., Barskikh M.E. Modern methods of functional verification of rtl units of vlsi microprocessor // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2015. № 2. С. 119–122.