УДК 004.921
DOI: 10.15827/0236-235X.119.353-358
Дата подачи статьи: 21.03.17
2017. Т. 30. № 3. С. 353–358

ВОСПРОИЗВЕДЕНИЕ ВИДЕОДАННЫХ ВЫСОКОЙ ЧЕТКОСТИ В ВИРТУАЛЬНОЙ ТРЕХМЕРНОЙ СРЕДЕ ИМИТАЦИОННО-ТРЕНАЖЕРНЫХ СИСТЕМ

А.М. Гиацинтов, научный сотрудник, algts@inbox.ru;
К.А. Мамросенко, к.т.н., руководитель Центра, kirillam@ya.ru
(Центр визуализации и спутниковых информационных технологий ФНЦ НИИСИ РАН, Нахимовский просп., 36, корп. 1, г. Москва, 117218, Россия)

В статье приводятся разработанные методы воспроизведения видеоматериалов высокой четкости в подсистеме визуализации тренажерно-обучающих систем.

Тренажерно-обучающие системы, как правило, содержат значительное количество разнородных информационных ресурсов. Особый интерес представляют такие виды аудиовизуальной учебной информации, как динамические графики развития процессов, иллюстративные материалы изучаемых объектов, трехмерные модели объектов и их частей, результаты работы моделирующих комплексов в форме видеообразов с сохранением управляемости приложения, видеоматериалы. Подсистема визуализации обеспечивает отображение результатов моделирования внешней среды и объекта управления с помощью устройств отображения информации. Отображение видеоматериалов в виртуальной трехмерной сцене является одним из требований к тренажерно-обучающим системам.

Воспроизведение видеоматериалов внутри виртуальной трехмерной сцены является сложной задачей, так как необходимо учитывать такие факторы, как производительность подсистемы визуализации и производительность видеокарты. Подсистема визуализации должна обеспечивать отображение трехмерной сцены с приемлемой частотой кадров (не менее 25 кадров в секунду) и при этом быть способной реагировать на внешние воздействия, в том числе на изменения параметров трехмерной сцены или загрузку дополнительных объектов.

Для отображения нескольких видеоматериалов высокой четкости в виртуальной трехмерной сцене авторами была разработана и реализована архитектура декодера кадров видео. Архитектура включает декодер видео, в котором происходит декодирование видео- и аудиопакетов; подсистему воспроизведения декодированного звука; управляющую структуру, необходимую для запуска видео, паузы воспроизведения, выставления громкости воспроизводимого видео и т.д.; интерфейс взаимодействия с движком, необходимый для обновления видеокадров.

Ключевые слова: *тренажерно-обучающие системы, система визуализации, видеоматериалы, тренажер, рендеринг, видео, декодер.*

Увеличение числа технических систем и повышение их сложности приводят к необходимости создания новых видов технических средств обучения. Одним из таких средств являются *тренажерно-обучающие системы* (ТОС). Подобные системы незаменимы в тех отраслях, где ошибки при обучении на реальных объектах могут привести к негативным последствиям, а их устранение – к значительным финансовым затратам: в военном деле, медицине, атомной энергетике, авиации и космосе. Из этого следует актуальность задачи построения современных ТОС.

Под ТОС оператора *сложной технической системы* (СТС) будем понимать техническое средство для подготовки операторов в едином информационном окружении, отвечающее требованиям методик подготовки, создающее условия для получения знаний, навыков и умений, реализующее модель таких систем и обеспечивающее контроль над действиями обучаемого, а также для проведения исследований [1, 2].

Как правило, ТОС содержат значительное количество информационных ресурсов. Особый интерес представляют такие виды аудиовизуальной учебной информации, как динамические графики развития процессов, иллюстративные материалы изучаемых объектов, трехмерные модели объектов и их частей, результаты работы моделирующих комплексов в форме видеообразов с сохранением управляемости приложения, видеоматериалы [3].

Подсистема визуализации обеспечивает отображение результатов моделирования внешней среды и объекта управления с помощью устройств отображения информации. Отображение разнородных видеоматериалов в виртуальной трехмерной сцене – одно из требований к ТОС.

Воспроизведение видеоматериалов внутри виртуальной трехмерной сцены является сложной задачей, так как необходимо учитывать такие факторы, как производительность подсистемы визуализации и производительность видеокарты. Подсистема визуализации должна обеспечивать отображение трехмерной сцены с приемлемой частотой кадров (не менее 25 кадров в секунду) и при этом быть способной реагировать на внешние воздействия, в том числе на изменения параметров трехмерной сцены или загрузку дополнительных объектов. Для отображения нескольких видеоматериалов высокой четкости в виртуальной трехмерной сцене авторами была разработана и реализована архитектура декодера кадров видео, а для обеспечения требуемой производительности подсистемы визуализации при отображении видеоматериалов высокой четкости – методы передачи данных в видеопамять.

Декодер видеоданных

Структура видеофайла – контейнер с общей информацией о его содержимом (количество видео- и аудиопотоков, наличие субтитров, информация о разделах, метаданные – название, год создания, автор и т.д.) и различных потоках информации (видео, аудио, субтитры и т.д.). Некоторые контейнеры допускают использование только определенных кодеров для представления информации. Наиболее популярными являются AVI, OGV, MP4, MKV, MOV, WMV.

В большинстве контейнеров данные представляются в виде пакетов (в некоторых форматах – куски (chunks) или сегменты (segments)). Пакеты хранят закодированную информацию определенного типа – видео, аудио или субтитры и т.д. В большинстве видеофайлов количество аудиопакетов значительно меньше количества видеопакетов – приблизительно 75 % видеопакетов и 25 % аудиопакетов. В зависимости от типа контейнера и используемого кодера расположение пакетов внутри контейнера может быть разным. Зачастую пакеты расположены в виде последовательности очередей пакетов разного типа, например, около 20 видеопакетов, расположенных подряд, затем несколько аудиопакетов, после которых снова подряд расположены около 20 видеопакетов.

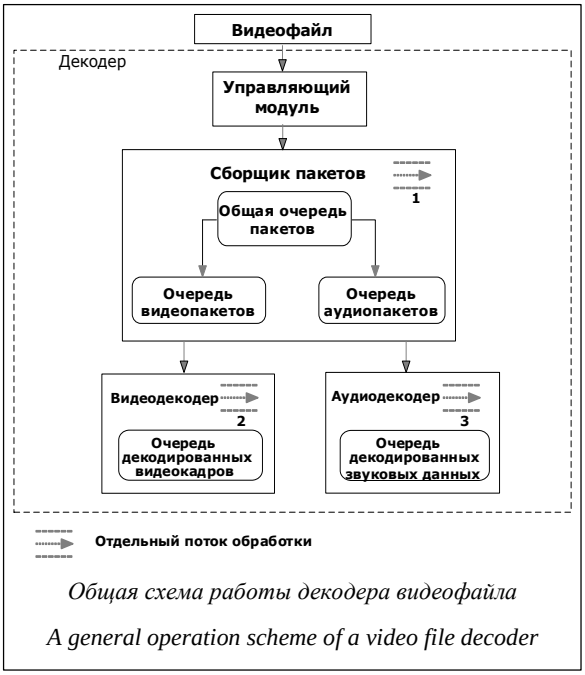
Пакеты хранят в себе определенный сегмент аудио- или видеоинформации. В большинстве случаев в видеопакете хранится целый видеокадр, но в некоторых форматах один видеокадр может состоять из нескольких пакетов. Это обусловлено тем, как кодер сохраняет информацию о видеокадре. Например, в формате MPEG 2 присутствуют три типа кадров: независимо сжатые кадры (I-кадры), кадры, сжатые с использованием предсказания движения в одном направлении (P-кадры), и кадры, сжатые с использованием предсказания движения в двух направлениях (B-кадры). Соответствующие группы кадров от одного I-кадра до другого образуют GOP (group of pictures) группу кадров [4].

В пакетах, кроме закодированных данных и информации о размере и типе данных данного пакета, присутствует информация о времени, когда необходимо декодировать данный пакет (последовательности декодирования пакетов), и времени, когда декодированная информация должна быть отображена или воспроизведена.

Аудиоинформация видеофайла может быть закодирована в различных форматах сжатия, таких как MP3, AAC, AC3, WMA и т.д. Для корректного воспроизведения аудиоинформации необходимо знать количество каналов (моно-, стерео-, 5.1, 7.1 и т.д.) и частоту дискретизации. После декодирования звукового пакета декодированные данные представляются в формате PCM (pulse code modulation, импульсно-кодовая модуляция) и передаются звуковой карте для воспроизведения звука.

Разработанная архитектура графической подсистемы позволяет декодировать и отображать одновременно несколько видео высокой четкости в трехмерной сцене. Состав архитектуры: декодер видео, в котором происходит декодирование видео- и аудиопакетов; подсистема воспроизведения декодированного звука; управляющая структура, необходимая для запуска видео, паузы воспроизведения, выставления громкости воспроизводимого видео и т.д.; интерфейс взаимодействия с движком, необходимый для обновления видеокадров; интерфейс управления графической подсистемой.

Общая схема работы декодера представлена на рисунке.



Общая схема работы декодера видеофайла

A general operation scheme of a video file decoder

Декодер состоит из четырех независимых частей:

– блок управления, осуществляющий запуск, остановку, а также обработку ошибок в случае их возникновения;

– блок получения пакетов из файла или внешнего источника, их демультиплексирование (при необходимости) и сортировка видео- и аудиопакетов по соответствующим очередям;

– блок декодирования видеопакетов;

– блок декодирования аудиопакетов.

Для каждого отображаемого видеоматериала создается свой экземпляр декодера. Во время инициализации декодера считываются все необходимые параметры о видеофайле, такие как кодеки, используемые для сжатия видео- и аудиоинформации, количество звуковых и видеопотоков в файле, ширина и высота (разрешение) видео, количество каналов звука и т.д.

Размещение пакетов в файле может различаться в зависимости от формата видеофайла, следовательно, сложно предсказать, в каком порядке будут

следовать видео- и аудиопакеты. Это может привести к ситуации, когда очередь видеопакетов заполнена, а очередь аудиопакетов пуста. Узнать тип следующего пакета невозможно до его чтения. Если следующий видеопакет окажется закодированным видеокадром, необходимо либо поместить этот пакет в очередь видеопакетов, то есть превысить заданные рамки очереди, либо удалить его, что приведет к артефактам на конечном изображении или пропущенным кадрам и рывкам воспроизводимого видео. Для решения этой проблемы была создана третья очередь пакетов, в которую помещаются пакеты различного типа из видеофайла, а затем сортируются по типу пакета (аудио- или видеопакет) в одну из очередей. Сортировка осуществляется по методу FIFO («первый вошел, первый вышел»).

Видеодекодер проверяет наличие пакетов в очереди видеопакетов и при обнаружении забирает пакет из начала очереди. Процесс декодирования видеопакета состоит из двух этапов – декодирование пакета и преобразование декодированного кадра в необходимую цветовую модель. После декодирования и преобразования в другую цветовую модель кадр помещается в очередь декодированных кадров. Аналогично аудиодекодер проверяет наличие пакетов в очереди и при обнаружении декодирует первый пакет из очереди. После декодирования звуковые данные в формате PCM помещаются в очередь декодированных звуковых данных.

Так как подсистема визуализации принимает изображения только в цветовой модели BGRA, после декодирования видеокадр будет преобразован в эту цветовую модель. Цветовая модель BGRA выбрана неслучайно: большинство форматов изображений и большая часть оборудования хранят изображения в формате BGRA. При использовании других цветовых моделей на центральном процессоре осуществляется преобразование используемой цветовой модели в модель BGRA. Так как любое лишнее преобразование снижает производительность графической подсистемы, использование цветовой модели BGRA является оптимальным.

Процесс декодирования ресурсоемкий. Например, декодирование одного видеопакета с разрешением 3 860×2 160 (4K) на процессоре Intel Core i7 3 ГГц занимает от 10 до 20 мс без использования специализированных аппаратных средств, таких как Intel QuickSync [5]. Декодирование пакета звуковых данных занимает значительно меньше времени (0,5–2 мс), но, если одновременно декодируется звук для нескольких воспроизводимых видео, это может серьезно снизить производительность. Скорость получения пакетов из видеофайла ограничивается производительностью *системы хранения данных* (СХД), что, в свою очередь, влияет на производительность подсистемы визуализации. Задержки происходят, если в момент считывания пакетов из видеофайла СХД обрабатывала другие команды либо файл являлся фрагментированным, что заставляет СХД искать фрагменты данного файла. Для преодоления проблем с производительностью была использована многопоточность.

Все потоки выполняются в адресном пространстве процесса. Выполняющийся процесс имеет как минимум один (главный) поток. Применение многопоточности наиболее эффективно при использовании многоядерных процессоров. На одноядерных процессорах многопоточное приложение также будет работать, но повышение производительности будет незначительным; возможно даже снижение производительности, если разные потоки одного процесса выполняют ресурсоемкие операции [6].

Хотя многопоточность способна существенно повысить производительность приложения, применять ее следует с осторожностью. Внедрение многопоточности существенно повышает сложность самого приложения и его отладки, а при некорректном использовании может привести к таким проблемам, как «состояние гонки» (когда работа системы зависит от того, в каком порядке выполняются части кода) и взаимная блокировка потоков (несколько потоков находятся в состоянии бесконечного ожидания ресурсов, занятых самими этими потоками), а также к трудностям при обработке ошибок (исключений) внутри приложения.

На данный момент каждая независимая часть декодера, кроме блока управления, работает в собственном потоке. Многопоточность позволяет значительно повысить производительность подсистемы декодирования при использовании многоядерного процессора. Однако данный подход имеет и свои недостатки. При воспроизведении значительного количества видеоматериалов одновременно будет создано большое количество потоков, превышающее количество доступных процессорных ядер (например, для пяти одновременно воспроизводимых видео будет создано 15 потоков). В то же время, чем больше потоков работают, тем чаще операционная система должна выполнять контекстное переключение, в течение которого текущий обрабатываемый поток приостанавливает свою работу, сохраняется вся информация о выполняемых операциях и данных, а затем управление передается другому потоку или процессу. Каждое переключение выполняется в течение определенного времени, которое можно было бы использовать для декодирования. Соответственно, в определенный момент добавление новых потоков не повысит, а снизит общую производительность приложения [7]. Так как каждый из потоков выполняет ресурсоемкие операции, может возникнуть ситуация, когда для одного из воспроизводимых видеоматериалов требуется отобразить следующий видеокадр, который еще не был декодирован. Выходом из данной ситуации может быть использование пула потоков.

Пул потоков – это структура, содержащая в себе определенное количество потоков обработки, обеспечивающая прием задач обработки от управляющей программы и их распределение по потокам. Зачастую количество потоков в пуле задается на основе информации о количестве доступных ядер процессора для использования всех доступных вычислительных мощностей процессора и снижения накладных расходов на контекстное переключение. Наиболее совершенные реализации пулов потоков позволяют перераспределять поставленные в обработку задачи между потоками для равномерной загрузки процессорных ядер [8].

Применительно к декодированию данных использование пула потоков позволит более равномерно распределять нагрузку при одновременной обработке нескольких видеоматериалов и таким образом готовить декодированные данные для большего числа одновременно воспроизводимых видеоматериалов, чем при использовании отдельных потоков для каждой части декодера.

Обновление видеоматериалов в подсистеме визуализации

Загрузка новых кадров видеоматериала (видеотекстуры) в подсистеме визуализации осуществляется в основном потоке программы (потоке рендеринга). Каждый кадр подсистемы визуализации проверяется на необходимость обновления видеотекстуры в трехмерной сцене. Если обновление требуется, то видеокадр берется из очереди декодированных кадров и помещается в память видеокарты в качестве текстуры, которая может быть наложена на поверхность любого трехмерного объекта [9].

При запуске видео создается пустая текстура с размерами видеокадра, которая затем применяется к материалу трехмерного объекта (материал объекта определяет, какие текстуры и шейдеры использует данный трехмерный объект). Перед применением новой текстуры к материалу происходит резервное копирование старого материала для возможности возвращения к исходным текстурам объекта после окончания воспроизведения видео. Также для повышения производительности используются два так называемых пиксельных буфера. Они применяются для хранения пиксельных данных (текстуры) в видеопамяти и позволяют значительно снизить временные затраты на обновление видеотекстуры в подсистеме визуализации. Количество используемых пиксельных буферов было подобрано таким образом, чтобы получить лучшее соотношение объема используемой оперативной памяти и величины задержки при отображении декодированных видеокадров.

При использовании стандартной функциональности подсистемы визуализации для обновления текстур в видеопамяти сначала из видеокадра в промежуточный массив копируются данные, а затем из промежуточного массива помещаются в текстуру. При обновлении текстуры в видеопамяти сначала происходит выделение нового места под данные и только потом копирование данных из промежуточного массива в видеопамять. Все описанные выше действия выполняются центральным процессором, что значительно снижает производительность подсистемы визуализации. Например, при воспроизведении FullHD-видео (1 920×1 080) время копирования данных из видеокадра в промежуточный массив составляет приблизительно 3–4 мс, а обновление текстуры из промежуточного массива в видеопамяти – от 5 до 8 мс.

Преимущество пиксельных буферов состоит в том, что затраты процессорного времени необходимы только на помещение информации в буфер, а обновление текстуры из буфера происходит практически мгновенно (около 0.1 мс) за счет того, что обработка перемещения данных из пиксельного буфера в текстуру возлагается на видеокарту [10]. Так как операция обновления текстуры из пиксельного буфера является асинхронной, выполняется она очень быстро, в то время как операция обновления информации в пиксельном буфере может вызвать синхронизацию процессора и видеокарты, что приведет к задержкам. Также видеокарте после обновления пиксельного буфера может понадобиться некоторое время для обработки полученных данных. Поэтому при обновлении текстуры сразу после обновления пиксельного буфера может возникнуть задержка. Если же производить обновление видеотекстуры перед загрузкой новых данных, это даст видеокарте больше времени на обработку данных пиксельного буфера.

При воспроизведении одного видео с разрешением вплоть до 1 920×1 080 (FullHD) в подсистеме визуализации либо нескольких видео с разрешением не выше 720×576 (576p, SD) время генерации одного кадра не превышает 40 мс (в зависимости от аппаратной платформы). Однако при одновременном воспроизведении нескольких видео высокой четкости наблюдается значительное увеличение времени генерации кадра подсистемой визуализации.

Во время работы над решением проблемы производительности при одновременном воспроизведении видеоматериалов высокой четкости было разработано несколько методов. В первом разработанном методе новые декодированные кадры копировались в пиксельные буферы в отдельном потоке, в то время как их передача в память видеокарты осуществлялась в основном потоке визуализации. Однако данный метод себя не оправдал, так как узким местом было не время копирования информации в оперативной памяти, а передача новых данных в память видеокарты.

Второй метод основывался на использовании единого пиксельного буфера, который содержал

бы кадры всех воспроизводимых видео. Преимущество данного метода в однократном обновлении пиксельного буфера для всех воспроизводимых видеоматериалов во время *итерации цикла* (ИЦ) обработки подсистемы визуализации. Однако у него обнаружились и существенные недостатки: необходимость сохранять предыдущие видеокадры, чтобы избежать рассинхронизации видео и аудио; сложность реализации динамического расширения и уменьшения пиксельного буфера при добавлении или удалении видео в ходе работы приложения. Данный метод также не смог предоставить необходимый уровень производительности при обновлении нескольких видеоматериалов.

Метод, позволивший получить требуемую производительность при одновременном воспроизведении видеоматериалов высокой четкости, базируется на построении очереди воспроизводимых видео на основе весовых коэффициентов.

Каждому видеоматериалу при воспроизведении присваиваются два вида весовых коэффициентов: коэффициент разрешающей способности видео и коэффициент выполняемой операции. Коэффициент разрешающей способности определяется один раз при запуске видеоматериала и основан на разрешении видео: при более низком разрешении значение коэффициента увеличивается, а при более высоком уменьшается. Это связано с тем, что видеоматериал с более высоким разрешением обрабатывается дольше, чем с более низким. На данный момент коэффициент может принимать три различных значения: 8 (высокий приоритет обновления), 4 (средний приоритет обновления), 0 (низкий приоритет). Например, видеоматериалу с разрешающей способностью до 720×576 будет присвоен высокий приоритет обновления, с разрешением до 1 280×720 – средний, с разрешением выше – низкий.

Коэффициент выполняемой операции определяется заново при каждом обновлении видеоматериалов в системе. Коэффициенту присваивается значение, определяющее операцию, которая будет выполнена на следующей итерации цикла обработки подсистемы визуализации. Могут быть присвоены следующие значения: 3 (обновление видеотекстуры), 2 (обновление видеотекстуры и пиксельного буфера), 1 (обновление пиксельного буфера), 0 (исключение видео из обработки). Значение 0 коэффициенту присваивается в случае, когда на следующей ИЦ подсистемы визуализации видеотекстура не требует обновления, а все пиксельные буферы заполнены.

Значения коэффициентов подобраны таким образом, чтобы преобладающим был коэффициент разрешающей способности видео, так как разрешение видеоматериала в большей степени влияет на производительность, чем порядок выполнения операций. Список обновляемых видеоматериалов определяется на каждой ИЦ подсистемы визуализации. На основе коэффициента разрешающей способности видео и коэффициента выполняемой операции определяется позиция видео в списке обновления. При первом обновлении видеоматериалов в системе порядок обновления определяется только по коэффициенту разрешающей способности видео, а коэффициенту выполняемой операции присвоено высшее значение 3 (обновление видеотекстуры).

Разработанный метод отображения видеоматериалов высокой четкости позволяет одновременно воспроизводить несколько видео в высоком разрешении. На тестовом оборудовании среднего класса одновременно воспроизводилось до трех видеоматериалов в разрешении 4K или до 6 в разрешении FullHD с сохранением управляемости ТОС и частоты кадров подсистемы визуализации на уровне 30 кадров в секунду. При улучшении аппаратного обеспечения увеличивается количество одновременно воспроизводимых видеоматериалов или появляется возможность использования видео с большим разрешением (5K, 8K).

Заключение

Задача разработки ТОС является комплексной и требует решения ряда сопутствующих задач, в частности, задач обработки и визуализации видеоданных. Преимуществом использования в ТОС таких информационных ресурсов, как видеоматериалы, является возможность визуализации как реальных, так и моделируемых процессов. За счет одновременного воспроизведения могут быть отображены параллельно выполняемые процессы или же один процесс с различных ракурсов. Разработанный метод воспроизведения видеоматериалов в виртуальной трехмерной сцене на основе весовых коэффициентов для формирования порядка обновления позволяет одновременно отображать несколько видео в разрешении 4K с сохранением требуемой производительности подсистемы визуализации. Планируется дальнейшее развитие разработанного метода, в частности, применение пула потоков и отказ от цветового преобразования на процессоре в пользу преобразования с использованием мощностей видеокарт.

Работа выполняется при поддержке РФФИ, грант № 17-07-00169.

Литература

1. Мамросенко К.А., Решетников В.Н. Моделирование подстилающей поверхности в имитационных системах // Программные продукты и системы. 2015. № 4. С. 70–74.

2. Решетников В.Н. Тригонометрические воспоминания. Сер.: Космические телекоммуникации. СПб: Изд-во РИП СПБ, 2015. 138 с.

3. Гиацинтов А.М., Мамросенко К.А. Метод рир-проекции в подсистеме визуализации тренажерно-обучающей системы // Программные продукты и системы. 2014. № 4. С. 31–37.

4. MPEG-2 video compression. URL: http://www.bbc.co.uk/

rd/publications/rdreport_1996_02 (дата обращения: 11.03.2017).

5. Intel® Quick Sync Video. URL: http://www.intel.com/content/www/ru/ru/architecture-and-technology/quick-sync-video/quick-sync-video-general.html (дата обращения: 11.03.2017).

6. Miloš Ljumović. C++ Multithreading Cookbook. Packt Publ. Ltd, 2014, 422 p.

7. Уильямс Э. Параллельное программирование на C++ в действии. М.: ДМК Пресс, 2012. 672 с.

8. Thread Pools. URL: https://msdn.microsoft.com/ru-ru/library/windows/desktop/ms686760(v=vs.85).aspx (дата обращения: 11.03.2017).

9. Гиацинтов А.М. Отображение разнородных видеоматериалов на гранях трехмерных объектов в подсистеме визуализации тренажерных обучающих систем // Программные продукты и системы. 2012. № 3. С. 80–86.

10. Pixel Buffer Object – OpenGL Wiki. URL: https://www.khronos.org/opengl/wiki/Pixel_Buffer_Object (дата обращения: 11.03.2017).

Software & Systems
DOI: 10.15827/0236-235X.119.353-358

Received 21.03.17
2017, vol. 30, no. 3, pp. 353–358

HIGH RESOLUTION VIDEO PLAYBACK IN VIRTUAL 3D ENVIRONMENT IN TRAINING SIMULATION SYSTEMS

A.M. Giatsintov [1], *Research Associate, giatsintov@niisi.ras.ru*
K.A. Mamrosenko [1], *Ph.D. (Engineering), Head of Department, mamrosenko_k@niisi.ras.ru*

[1] *Center of Visualization and Satellite Information Technologies SRISA, Nakhimovsky Ave. 36/1, Moscow, 117218, Russian Federation*

Abstract. The article describes the developed methods of high-resolution video playback in a visualization subsystem of training simulation systems (TSS). Usually TSS have a number of heterogeneous data resources. The following resource types are of interest for personnel training: dynamic process development charts; graphic materials of studied objects; 3D models of objects and their parts; simulation results in a video form; video records. Visualization subsystem provides rendering simulation results of environment and dynamic objects, and displaying the rendered image using display devices. Displaying videos in virtual 3D scenes is one of the requirements for TSS.

Displaying videos inside a virtual 3D scene is a complex task, as many factors should be taken in account, such as performance of a video card and a visualization subsystem (renderer). A renderer should visualize a 3D scene with acceptable frame rate (not less than 25 frames per second) and respond to commands, such as a 3D scene transformation or new objects loading.

The authors have developed and implemented a new decoder architecture in order to display several high definition videos in a virtual 3D scene. The architecture includes the following components: a decoder that decompresses audio and video packets; an audio playback system; a control module that allows starting, stopping playback, setting volume, etc.; an interface that interacts with a visualization subsystem, which is required for updating video images.

Keywords: training simulation systems, visualization system, video, video material, trainer, rendering, decoder.

Acknowledgements. *The work has been done with the support of the RFBR, grant no. 17-07-00169.*

References

1. Mamrosenko K.A., Reshetnikov V.N. Terrain modelling in simulation training systems. *Programmnye produkty i sistemy* [Software & Systems]. 2015, no. 4, pp. 70–74 (in Russ.).

2. Reshetnikov V.N. *Trigonometricheskie vospominaniya. Ser. Kosmicheskie telekommunikatsii* [Trigonometric Memories. Series “Space Telecommunications”]. St. Petersburg, RIP SPB Publ., 2015, 138 p.

3. Giatsintov A.M., Mamrosenko K.A. Rear-projection method in visualization subsystem of training simulation system. *Programmnye produkty i sistemy* [Software & Systems]. 2014, no. 4, pp. 31–37 (in Russ.).

4. *MPEG-2 Video Compression*. Available at: http://www.bbc.co.uk/rd/publications/rdreport_1996_02 (accessed March 11, 2017).

5. *Intel® Quick Sync Video*. Available at: http://www.intel.com/content/www/ru/ru/architecture-and-technology/quick-sync-video/quick-sync-video-general.html (accessed March 11, 2017).

6. Miloš Ljumović. *C++ Multithreading Cookbook*. Packt Publ. Ltd, 2014, 422 p.

7. Williams A. *C++ Concurrency in Action: Practical Multithreading*. Manning Publ., 2012, 528 p. (Russ.ed.: Moscow, DMK Press, 2012, 672 p.).

8. *Thread Pools*. Available at: https://msdn.microsoft.com/ru-ru/library/windows/desktop/ms686760(v=vs.85).aspx (accessed March 11, 2017).

9. Giatsintov A.M. Playback of heterogeneous videos on 3D object’s edges in visualization subsystem of training simulation system. *Programmnye produkty i sistemy* [Software & Systems]. 2012, no. 3, pp. 80–86 (in Russ.).

10. *Pixel Buffer Object – OpenGL Wiki*. Available at: https://www.khronos.org/opengl/wiki/Pixel_Buffer_Object (accessed March 11, 2017).