УДК 681.3.012

М. А. Городилов, **Б. С. Долговесов**, **И. Д. Храмцов**, **А. Х. Радостев**

Институт автоматики и электрометрии СО РАН
пр. Коптюга, 1, Новосибирск, 630090, Россия

gorodilovm@gmail.com, bsd@iae.nsk.su, khramtsov9203@ya.ru
idargerogue@gmail.com

ОСОБЕННОСТИ ПОСТРОЕНИЯ СИСТЕМ ДЛЯ ПОЛИЭКРАННОГО ОТОБРАЖЕНИЯ РАСПРЕДЕЛЕННЫХ МУЛЬТИМЕДИЙНЫХ ДАННЫХ

Статья посвящена решению некоторых вопросов отображения распределенных мультимедийных данных на больших экранах. В частности, рассмотрены проблемы синхронизации процесса параллельной обработки и вывода фрагментов видеоизображения на соответствующие экранные модули полиэкранной системы отображения. Предложен алгоритм синхронизации и его программная реализация с использованием графических ускорителей, обеспечивающий визуальную непрерывность динамических сцен при отображении на полиэкранах. Рассмотрены вопросы управления распределенными входными потоками данных для полиэкранного отображения.

Ключевые слова: полиэкран, синхронизация, мультимедийные данные, реальное время, распределенный рендеринг.

Введение

В статье рассматриваются отдельные аспекты отображения на больших экранах (полиэкранах) распределенных мультимедийных данных и удаленного управления этими данными при создании многофункциональных систем отображения. Полиэкраны, представляющие собой набор экранных модулей (плазменных панелей, мониторов или проекторов), позволяют одновременно работать многим пользователям различных информационных центров подготовки и принятия решений, имеют высокое разрешение, что важно для визуализации научных данных. В связи с этим актуально высококачественное визуальное представление информации. При построении полиэкранных систем визуализации одной из важных задач является синхронизация вывода видеофрагментов, формируемых отдельными процессорами, на соответствующие экранные модули для обеспечения визуальной непрерывности полноэкранного изображения.

Из-за высокого разрешения, необходимого для генерации выходного изображения полиэкрана, ресурсов одного ПК в большинстве случаев оказывается недостаточно. Самым распространенным решением этой проблемы является использование распределенного рендеринга – подхода, когда множество объединенных в сеть ПК, образующих вычислительный кластер, параллельно обрабатывают отдельные фрагменты общего изображения с последующим выводом их на соответствующие модули полиэкрана. В зависимости от динамичности воспроизводимого изображения и конфигурации полиэкрана различия во временах отображения кадров на отдельных экранных модулях отрицательно сказываются на субъективном восприятии изображения зрителем уже при величинах от 15 до 30 мс [1]. В отсутствие синхронизации различные ПК кластера будут отображать кадры в разное время из-за

Городилов М. А., *Долговесов Б. С.*, *Храмцов И. Д.*, *Радостев А. Х.* Особенности построения систем для полиэкранного отображения распределенных мультимедийных данных // Вестн. Новосиб. гос. ун-та. Серия: Физика. 2015. Т. 10, вып. 2. С. 91–98.

непостоянства задержек в передаче сетевых сообщений и различий в нагрузке на ПК и заполненности буфера графического процессора. Кроме того, различия по частоте и фазе развертки экранных модулей также вносят трудно контролируемые временные погрешности. Это приводит к тому, что видеокадры, являющиеся частями общего изображения, будут отображаться на соответствующих экранных модулях в разное время. В связи с этим при построении полиэкранных систем визуализации одной из важных задач является синхронизация вывода видеофрагментов, формируемых отдельными ПК, на соответствующие экранные модули для обеспечения визуальной непрерывности изображения на полиэкране. Рассмотрены существующие методы синхронизации процесса полиэкранного отображения видеоданных, отмечены их недостатки.

Предлагаемый в данной статье метод синхронизации видеопотоков в вычислительном кластере позволяет обеспечить визуальную непрерывность изображения на полиэкране без использования дорогостоящих специализированных аппаратных средств.

Для информационных центров, кроме качественного отображения визуальной информации, возникает необходимость оперативного выбора источника информации для отображения. В ряде случаев выбор источника информации должен осуществлять оператор, находящийся вдали от аудиовизуальных комплексов обработки распределенных мультимедийных данных [2]. Для удаленного управления в статье предлагается эффективный метод организации удобного графического пользовательского интерфейса оператора.

Алгоритм синхронизации видеоданных для полиэкранного отображения

Обзор методов синхронизации

Существующие подходы синхронизации видеопотоков для полиэкранных систем можно принципиально разделить на программные и аппаратные. Аппаратные решения используют специализированные внешние устройства для синхронизации развертки экранных модулей или обновления данных в буферах графических акселераторов. Программные решения реализуют тот или иной протокол синхронизации передачи данных между компьютерами кластера и времени их обработки. Это позволяет снизить различия во времени отображения кадров. В качестве примеров аппаратных решений можно рассмотреть устройства синхронизации *ATI FirePro S400* и *Nvidia Quadro Sync* со схожими характеристиками, объединяющие каждый до четырех графических ускорителей на одном ПК с возможностью синхронного вывода данных, в том числе и на полиэкран. Пример программного решения – алгоритм синхронизации для полиэкранного отображения на базе масштабируемой адаптивной графической среды SAGE (Scalable Adaptive Graphics Environment) [3]. Алгоритм использует два варианта синхронизации: с помощью сетевого барьера и с помощью синхронизации времени работы компьютеров. В первом варианте все компьютеры, входящие в состав кластера, осуществляют ожидание на сетевом барьере перед выводом данных на соответствующие экранные модули. Когда все фрагменты изображения готовы и потоки данных на всех компьютерах доходят до барьера, происходит одновременный вывод фрагментов изображения на все экранные модули полиэкрана. При таком подходе в случае возникновения задержки на одном из компьютеров кластера происходит задержка в работе всей системы. Во втором варианте время на всех компьютерах кластера синхронизируется с помощью протокола NTP (Network Time Protocol). Один из компьютеров является главным – на каждом новом кадре он рассылает всем остальным компьютерам время (Presentation Time, PT), за которое они должны вывести свою часть изображения на экранный модуль. Это время вычисляется путем прибавления к текущему времени главного компьютера некоторой добавки (Presentation Time offset, PTO). В описанной реализации PTO эмпирически задается константой времени отображения. Неудачный выбор этой константы может повлиять на работу алгоритма. Малое значение PTO приведет к тому, что некоторые компьютеры могут не успеть отобразить свои данные на экранном модуле к назначенному времени, а слишком большое – к излишнему их простою. В приведенных результатах тестирования максимальная рассинхронизация времени кадров

находится в пределах 1–2 мс. Однако оба варианта этого алгоритма применимы только в том случае, если используют предположение, что новые кадры поступают на компьютеры равномерно и без потерь и не пропускаются кадры при отображении. Подобные методы применяются и в других межплатформенных программных средствах для построения распределенных систем визуализации, например, Chromium и Equalizer. Данные программные продукты предоставляют сетевые барьеры, как основной примитив синхронизации кадров при полиэкранном отображении и, следовательно, имеют те же основные недостатки, что и в алгоритме, представленном в работе [3].

Описание алгоритма синхронизации видеоданных для полиэкранного отображения

В данной работе не рассматривается распределение видеоданных по компьютерам кластера. Если проигрывается видеофайл, считается, что каждый компьютер получает копию видео (либо только интересующий его фрагмент видео) перед началом воспроизведения. Для потокового видео (например, трансляции с видеокамеры) считается, что кадры разбиваются на фрагменты или передаются целиком в реальном времени.

Основная идея предлагаемого в данной статье алгоритма синхронизации заключается в том, что компьютеры кластера осуществляют буферизацию входящих кадров, затем на основе текущей заполненности буфера графического процессора определяется расчетное время, в которое очередной кадр будет отображен на экране. Важно отметить, что кадровая частота выходного видеопотока при этом совпадает с частотой вертикальной развертки экранного модуля (режим вертикальной синхронизации). Рассмотрим подробно цикл работы алгоритма для одного компьютера кластера и одного нового кадра.

Все новые кадры предварительно загружаются во входной буфер (FIFO-очередь). У каждого кадра из этого буфера есть временная метка T_i, определяющая, когда кадр должен быть отображен. Но поскольку практически во всех современных графических приложениях используется двойная или тройная буферизация при рендеринге изображения, то даже при отправке графическому процессору команды «Отобразить» для нового кадра, он не отобразится на экране в этот момент. Сначала он поступает в один из вторичных буферов (back buffer) и лишь после того, как все предыдущие кадры будут отображены, его буфер станет первичным (front buffer), и новый кадр будет отображен на экране. Таким образом, кадр, отправленный на отображение в момент времени t на самом деле будет отображен в момент времени

$$t' = t + \Delta T \cdot k + dt,$$

где ΔT – длительность одного кадра (в данном случае совпадающая с периодом развертки экранного модуля), k – количество кадров в буфере графического процессора, а dt – фазовая составляющая, определяющаяся разницей между временем поступления запроса на отображение и началом следующего цикла развертки монитора. Поскольку заполненность буфера графического процессора может зависеть от текущей нагрузки и значительно варьироваться в разные моменты времени на разных компьютерах, без дополнительной коррекции время отображения кадров также будет различаться более чем на ΔT. Это составляет около 16 мс для типичного монитора с частотой обновления 60 Гц.

Таким образом, предлагается, используя данные о времени отображения предыдущих кадров и пропорционально-интегральный регулятор (ПИ-регулятор), определять момент времени, в который новый кадр будет в действительности отображен на экране. Этот момент (оценочное время) вычисляется как

$$T_i' = T_{i-1}' + \Delta T + dt_k,$$

где T_{i-1}' – (время отображения предыдущего кадра) оценка того же момента для предыдущего кадра (для первого кадра – текущее системное время узла), ΔT – период вертикальной развертки монитора, а dt_k – корректировочный коэффициент. В процессе функционирования алгоритма хранится информация о последних N_{hist} кадрах: оценочное (T_i') и реальное (T_i) время отображения. С использованием этой информации значение коэффициента dt_k вычисляется как средняя разница между реальным и оценочным временем, умноженная на настроечный коэффициент K:

$$dt_k = K \cdot \frac{1}{N_{hist}} \sum_{i=1}^{N_{hist}} \left(T_i - T_i' \right).$$

До тех пор, пока хотя бы один из вторичных буферов графического процессора свободен для записи, компьютер кластера принимает решение о том, какой кадр отправить на отображение следующим. Компьютер сравнивает текущее время t с оценочным временем T_k' первого кадра во входной очереди. Если $t < T_k'$ или входная очередь свободна, то будет повторно выбран последний отображенный кадр, иначе первый кадр будет удален из очереди и отправлен на отображение.

Программная реализация алгоритма синхронизации

Для проверки работы алгоритма был разработан прототип системы синхронизации видеопотоков для отображения на полиэкране. За основу был взят мультимедийный фреймворк (framework) *Microsoft Media Foundation*. Поскольку его архитектура основана на слабо связанных компонентах, формирующих граф потоковой обработки медиаданных. Это позволяет разместить логику синхронизации процесса отображения в одном независимом модуле. В качестве такого модуля был создан компонент-приемник («sink» в терминологии Media Foundation) на основе *DirectX11 API.*

В предлагаемой архитектуре среди всех распределенных компьютеров кластера один играет роль *master*-компьютера, который управляет воспроизведением и является источником синхронизированного времени для остальных *slave*-компьютеров (ведомых). Временная синхронизация *начала*, *паузы* и *остановки* воспроизведения медиаданных между распределенными компьютерами кластера осуществляется путем рассылки *master*-компьютером управляющих сообщений всем *slave*-компьютерам. Синхронизация времени реализована программно согласно протоколу *PTP* (*Precision Time Protocol*), что позволило достичь точности синхронизации часов на распределенных компьютерах кластера менее 1 мс. Информация о времени отображения видеокадров, требуемая для реализации алгоритма, определяется средствами *DXGI 1.2*, включающими интерфейс *IDXGISwapChain* для доступа к очереди буферов графического процессора и метод сбора статистики кадра – *GetFrameStatistics*. Исходные данные о времени отображения кадров поступают вместе с кадрами от вышележащих компонентов графа воспроизведения через интерфейс *IMFSample*.

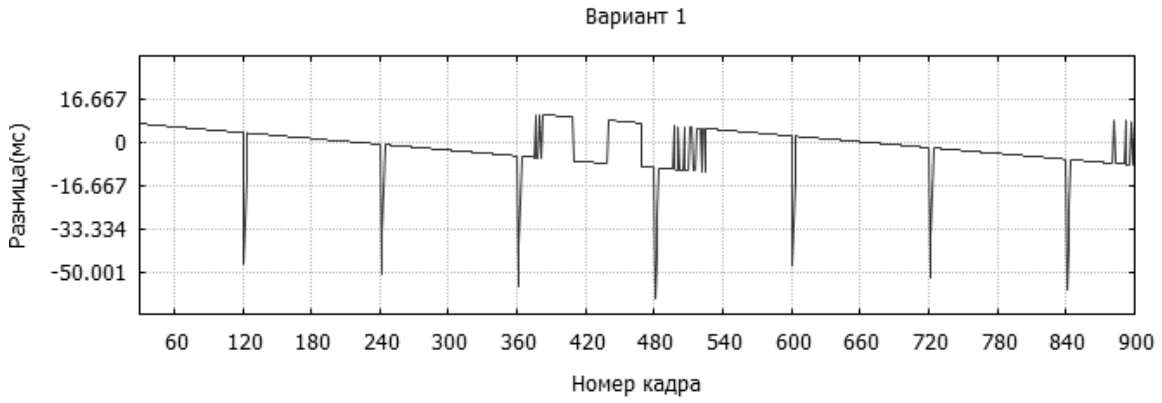
Результаты тестирования

Разработанный прототип системы синхронизации апробирован на системе из двух компьютеров, соединенных через локальную сеть *Ethernet* и представляющих вычислительный кластер. Один из них исполнял роль *master*-компьютера, другой – *slave*-компьютера. Тестовый видеофайл был заранее разбит на фрагменты и размещен на соответствующих компьютерах в конфигурации:

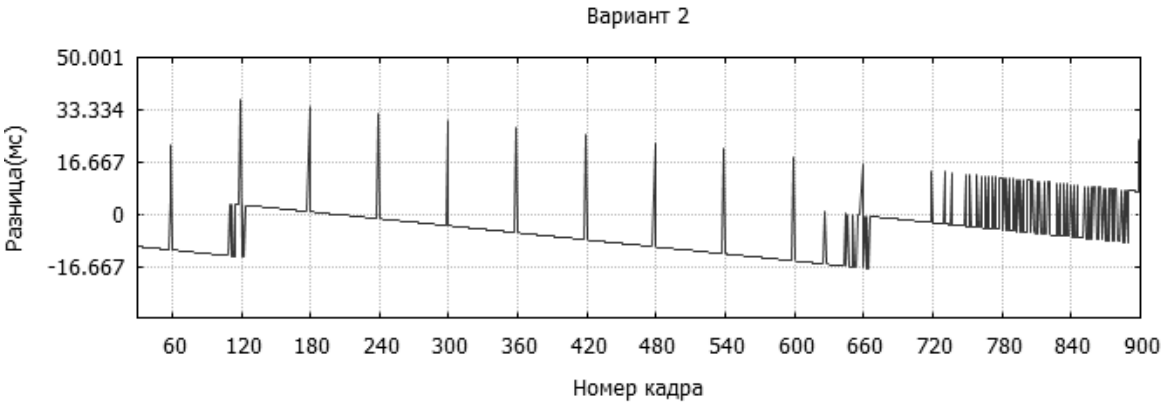
- процессор Intel Core 2 Duo, 3 ГГц;
- оперативная память 4 Гб;
- графический процессор NVidia GeForce 8800 GTX.;
- монитор LG Flatron L246WH, частота развертки 60 Гц (в качестве модуля полиэкрана).

За основную характеристику, определяющую качество синхронизации изображения, была выбрана разница между временем воспроизведения компьютерами видеокадров, являющихся частями общего изображения полиэкрана. Чем ближе эта величина к нулю, тем лучше синхронизированы изображения на отдельных мониторах. Для проверки влияния неравномерности поступления входных кадров на работу системы синхронизации была введена периодическая задержка T в модуль, осуществляющий прием кадров, т. е. каждые N кадров работа процесса приостанавливалась на T мс. Таким образом была смоделирована ситуация, когда один из компьютеров кластера работает с переменной задержкой относительно других либо когда кадры изображения поступают на компьютеры неравномерно. Размер входной очереди кадров и количество вторичных буферов графического акселератора были выбраны равными четырем. Тестовый видеофайл с частотой 30 кадров/с был последовательно воспроизведен в трех вариантах синхронизации.

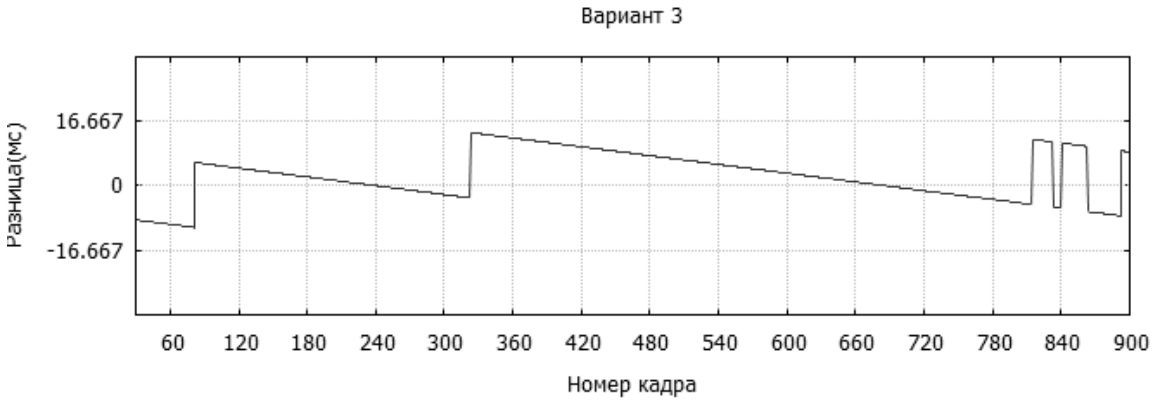
1. Вариант без буферизации кадров. Каждый компьютер, находясь в режиме ожидания, с приходом момента времени для очередного кадра отправляет его на отображение. Как показывают результаты измере-



а



б



в

Разница во времени отображения кадров: *а* – для $N = 120$, $T = 50$ мс; *б*, *в* – для $N = 60$, $T = 120$ мс

ний, возникающая при задержках разница во времени кадров совпадает с величиной задержки T (см. рисунок, *а*).

2. Вариант с буферизацией кадров без коррекции времени. В данном варианте при выводе кадров была использована буферизация с 4 вторичными буферами. Работа компьютеров осуществлялась по алгоритму, описанному выше, за исключением того, что заполненность буфера не учитывалась при принятии решения, в какой момент времени отправлять очередной кадр на отображение. Результаты измерений см. на рисунке (*б*). В данном случае использование буфериза-

ции позволяет снизить разницу во времени воспроизведения, возникающую из-за задержек поступления кадров, до 30 мс.

3. Вариант с буферизацией кадров и коррекцией времени**.** В данном варианте, как и в предыдущем, используется буферизация кадров с теми же параметрами. Работа компьютеров осуществлялась по описанному алгоритму. Результаты измерений показаны на рисунке (*в*). В данном варианте разница находится в пределах ±16 мс (величина задержки, как в варианте 2), что для монитора с частотой развертки 60 Гц составляет менее одного кадра.

Особенности организации пользовательского интерфейса удаленного управления отображением распределенных данных

Для оперативной смены сценариев отображения удаленный оператор должен иметь возможность просмотра данных от всех входных источников и результата их микширования. Многие аппаратные решения предоставляют подобную возможность (в частности, видеопроцессоры Creston и Extrone). Одним из подходов является передача видеопотока с возможностью переключения. Выбирается один просматриваемый видеопоток, который интересует удаленного пользователя, и передается по сети [4]. Достоинство метода – хорошее качество передаваемого изображения. Недостатки: переключение на другой источник данных занимает время; отсутствие возможности одновременного просмотра предварительных данных нескольких источников. Просмотр видеоданных одновременно возможен, когда исходные данные всех источников передаются с низким разрешением [5]. Недостатки: нагрузка на сеть зависит от количества источников, потенциально делая ее непостоянной; отсутствует синхронизация первичных данных между собой.

Интерфейс управления может быть организован так, что в исходном разрешении отображаются видеоданные нескольких выбранных источников, а остальные – с уменьшенным разрешением в виде миниатюр (preview) для удобства выбора оператором входного источника. Сформированный таким образом видеопоток с изменяемым разрешением видеоданных передается от вычислительного сервера к компьютеру оператора.

Так как вычислительные ресурсы компьютера оператора ограничены, то невозможно одновременно обрабатывать видеоданные всех входных источников в исходном разрешении для отображения в интерфейсе управления. Для оптимального решения данной задачи предлагается метод формирования на вычислительном сервере композиции фиксированного размера, включающей необходимые видеопотоки в уменьшенном масштабе для миниатюр, и передавать ее в сжатом виде по локальной сети. На компьютере оператора этот поток разжимается и разделяется на отдельные видеопотоки, в соответствии с информацией о расположении исходного видео в кадре. При этом поток данных о расположении миниатюр должен быть синхронизирован с видеопотоком от сервера. В этом случае передача видеопотоков от вычислительного сервера к компьютеру оператора осуществляется с изменяемым разрешением для представления оператору видеоминиатюр, соответствующих входным источникам данных. Требуемое разрешение для видеоминиатюр пользовательского интерфейса устанавливается оператором и передается на вычислительный сервер.

Процесс обработки одного кадра изображения видеоминиатюры включает следующие операции:

- прием от компьютера оператора информации о размере миниатюр и их позиции на экране оператора;
- генерация миниатюр и дополнительной информации о расположении и размерах кадров для входных источников;
- кодирование мозаики;
- отправка сжатых данных по сети посредством сетевого протокола на компьютер оператора и их декодирование;
- интерпретация дополнительной информации компьютером оператора.

Для реализации метода разработан алгоритм, формирующий графические окна в соответствии с параметрами, определяющие размер и позицию видеоминиатюры на экране интерфейса. Окна помещаются в видеоконтейнер по очереди в порядке убывания размера. При сравнении размеров окон и видеоконтейнеров используется функция сортировки, учитывающая соотношение сторон исходного видеоконтейнера. При

этом должно выполняться условие: окно, не помещающееся в видеоконтейнер, будет уменьшено до необходимого размера, соотношение сторон будет сохранено. В результате работы алгоритма формируется набор окон, содержащих координаты и размеры кадров миниатюр для отображения. Эти данные передаются в видеоконтейнер в качестве дополнительной информации.

Для тестирования алгоритма формирования потока кадров фиксированного размера, содержащего мозаику кадров изображений всех источников, использовались: библиотека *LibAV*, формат видеоконтейнера *MP4*, видеокодек *H264*, протокол передачи данных *RTMP*. Для передачи дополнительной информации о мозаике кадров использовался отдельный поток пакетов с данными. Для реализации удаленного вызова процедур использовался протокол *JSON-RPC*. В результате тестирования выявлено, что на границах стыковок кадров при динамических изменениях присутствуют артефакты в виде небольших искажений квадратной формы. Это обусловлено тем, что кодек *H264* кодирует информацию об изображении небольшими блоками (например, 16×16 пикселей). В таком случае, если сделать размеры кадров изображения кратными 16 пикселям, можно добиться исчезновения данного искажения.

Заключение

Предложенный алгоритм синхронного вывода распределенных видеофрагментов обеспечивает высококачественное отображение в реальном времени крупномасштабных сцен на полиэкранах. Рассмотренные существующие методы синхронизации применимы при предположении, что новые кадры изображения должны поступать на процессоры кластера, соответствующие экранным модулям, равномерно и без потерь, а при отображении не должны пропускаться кадры. Для выполнения данных условий необходимы дополнительные вычислительные ресурсы и временные затраты, что нежелательно для системы отображения в реальном времени. В отличие от существующих решений предлагаемый алгоритм устойчив к неравномерности частоты поступления видеокадров за счет буферизации кадров на входе и двойной/ тройной – на выходе. Это позволяет избежать нежелательных артефактов (эффект «замирания», потеря кадров, разрывы изображения и др.), особенно характерных для распределенного отображения динамических сцен на полиэкранах. Алгоритм не накладывает ограничений на число компьютеров в кластере, поскольку вычислительные затраты на синхронизацию на одном компьютере составляют незначительную часть (менее 1 %) от общего объема вычислений и не увеличиваются с ростом числа компьютеров.

Для удаленного управления отображением видеоданных предложен эффективный метод организации удобного пользовательского интерфейса. Для отображения миниатюр (превью-раскладок) интерфейса, соответствующих входным источникам данных, формируется ограниченный поток данных для передачи по сети, не зависящий от количества источников. Метод обеспечивает в реальном времени предпросмотр входных источников и оперативный вывод требуемых видеоданных для отображения на полиэкранных средствах. Кроме того, данный подход не накладывает существенных ограничений на производительность компьютера оператора и не требует дополнительных аппаратных средств, что позволяет использовать планшетные компьютеры.

Список литературы

1. *Deshpande S., Daly S.* Quality of Experience for Large Ultra-High-Resolution Tiled Displays with Synchronization Mismatch // EURASIP Journal on Image and Video Processing, 2011. http://jivp.eurasipjournals.com/content/2011/1/647591

2. *Морозов Б. Б., Долговесов Б. С., Мазурок Б. С., Городилов М. А.* Построение распределенной мультимедийной виртуальной среды с многоканальной визуализацией медиаданных на графических акселераторах // Программирование. 2014. № 4. С. 52–62.

3. *Nam S., Deshpande S., Vishwanath V., Jeong B., Renambot L., Leigh J.* Multi-application inter-tile synchronization on ultra-high-resolution display walls // Proceedings of the first annual ACM SIGMM conference on Multimedia systems, 2010. P. 145–156.

4. *Gibson J.* Concurrent autonomous video-lecturing system applicability study. Theseus, Helsinki Metropolia University of Applied Sciences, 2013.

5. *Binotto A. P. D.* A Real-time Collaborative Tele-ultrasonography // Potentials, IEEE 2012. Vol. 31. Iss. 6.

Материал поступил в редколлегию 10.04.2015

M. A. Gorodilov, **B. S. Dolgovesov**, **I. D. Khramtsov**, **A. H. Radostev**

Institute of Automation and Electrometry SB RAS
1 Koptyug Ave., *Novosibirsk*, *630090*, *Russian Federation*

gorodilovm@gmail.com, *bsd@iae.nsk.su*, *khramtsov9203@ya.ru*
idargerogue@gmail.com

FEATURES OF MULTI-SCREEN DISPLAY OF DISTRIBUTED MULTIMEDIA DATA

This article is devoted to solving some of the issues of distributed multimedia data display on large screens. In particular, we consider the problem of synchronization of the process of parallel rendering and output of video fragments to the appropriate modules of multiscreen display systems. Proposed synchronization algorithm and its implementation using graphics accelerators, provides visual continuity of dynamic scenes when displayed on a multiscreen. The problems of managing distributed input data stream for multiscreen display are considered.

Keywords: multiscreen, synchronization, multimedia, real-time, distributed rendering.