

УДК 004.946

АЛГОРИТМ СИНХРОНИЗАЦИИ МУЛЬТИЭКРАННОЙ СИСТЕМЫ ДЛЯ МАКЕТА СТЕНДА ОБЪЕМНОЙ ВИЗУАЛИЗАЦИИ

© 2017 А.А. Сгибнев

Саратовский государственный технический университет имени Гагарина Ю.А.

Статья поступила в редакцию 20.03.2017

Описывается алгоритм синхронизации воспроизведения видеофайлов устройствами вывода в виде массива проекторов, которые используются в макете объемного дисплея.

Ключевые слова: *трехмерная визуализация, объемный дисплей, алгоритм, синхронизация, массив проекторов*

В настоящее время актуальным является вывод изображений на несколько устройств, например, мониторов или проекторов. Подобная схема используется расширения виртуального пространства, создания видеостен. Также увеличивается количество разработок так называемых объемных дисплеев. При этом объекты, формируемые объемным дисплеем, занимают реальное физическое пространство. Трехмерная визуализация – это восходящий тренд современных технологий, и она существенно влияет на многие сферы деятельности человека и инициирует создание и развитие целого ряда научных направлений и технологий, включая: навигацию, проектирование, медицину, компьютерные тренажеры и многое другое. Согласно анализу агентства Гартнер область волюметрических дисплеев относится к перспективным технологиям, которые находятся на этапе зарождения [1, 2].

Для получения 3D-изображения высокого качества обычно используются дисплеи с индивидуальными средствами просмотра и устройства дополненной реальности. Для демонстрации 3D-образов массовой аудитории, не вооруженной средствами стереонаблюдения, применяются относительно дорогие технические решения с большим объемом обрабатываемой информации. К ним относятся автостереоскопические планарные дисплеи с ограниченным числом точек наблюдения и с пока не удовлетворяющим потребителя качеством изображения. На рынке также нашли применение виртуальные витрины и «воздушные» дисплеи, воспроизводящие «подвешенные в воздухе», однако двухмерные образы.

В настоящее время существующими дисплеями не решены следующие проблемы: 1) демонстрация полноцветных трехмерных объектов, наблюдаемых массовой аудиторией вкруговую, с обзором в 360° без применения индивидуальных средств наблюдения и с разрешением Full HD; 2) организация объемных трансляций в режиме реального времени с передачей контента по сети. У аналогов поток информации в единицу времени, необходимый для воспроизведения 3D-изображения,

превышает пропускную способность существующих каналов связи. Предложен объемный дисплей [3], который базируется на запатентованной технологии совмещения оптической и компьютерной обработки изображения, позволяющей по любым двум опорным изображениям, разнесенным на определенный азимутальный угол, сформировать множество промежуточных изображений, плотно заполняющих сегмент, причем два смежных промежуточных изображения являются стереопарой.

Основные принципы предложенной технологии: использование оригинальной оптической системы для совмещения, ориентирования и комбинирования опорных изображений (вместо весьма трудоемкой компьютерной обработки). Кроме этого применяется многоэлементная система вывода опорных изображений, и на этой базе достигается повышение качества объемных образов на основе снижения удельной нагрузки на выходные элементы системы.

Цель работы: решение задачи синхронного вывода объекта визуализации на рабочую область [2].

Описание макета стенда объемной визуализации. Для валидации ряда гипотез нами осуществляются работы по созданию макета объемной визуализации, принцип функционирования которого заключается в следующем. Вокруг оси макета 1 (рис. 1) со скоростью 1200-1500 об/мин вращается непрозрачный барабан 2 со сквозными окнами 3, закрытыми прозрачным материалом. Внутри барабана установлен просветный лентикулярный экран 4, на который выводится изображение. На лицевой стороне стенда расположена прозрачная мультиэкранная панель 5, выполненная в виде части прямой многогранной призмы и покрытая пленкой обратной проекции, на которую выводится первичное изображение 6. Изображение формируется на всех гранях мультиэкранной панели с использованием массива проекторов 7, световые пучки 8 которых сфокусированы на слое пленки обратной проекции [4]. С использованием двух модулей оптической системы 9 и 10, сформированный световой пучок пересекает ось стенда и достигает поверхности зеркального многогранника 11, составленного из плоских поверхностных зеркал. Отразившись от

зеркального многогранника световой пучок достигает экрана 4, где и формирует итоговое изображение. Барабан на подшипниках 13 и приводится во вращение приводом 14. Вышеописанный макет является прототипом кругового объемного дисплея для визуализации трехмерных объектов и сцен.

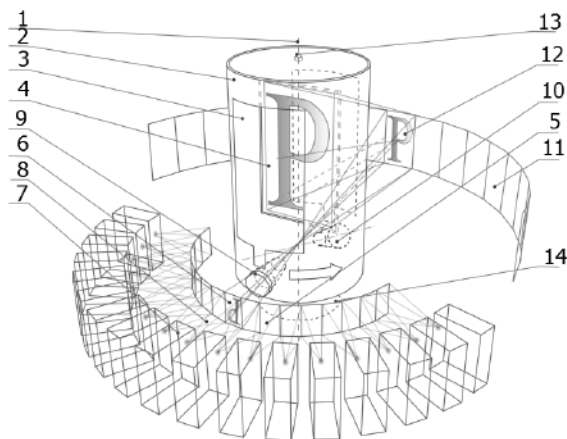


Рис. 1. Схема макета объемной визуализации

Для реализации первого макета используется три проектора, изображение которых разделяется на 18 частей и проецируется через систему из 36 зеркал на экран. Требуемое расположение зеркал определено на основе определенных вычислений, а затем соответствующее построение выполнено в программе 3ds max. Подобная схема реализуется для проверки оптической системы дисплея. Кроме метода формирования объема из совмещения изображений объекта, снятого в различных ракурсах, также проведено тестирование алгоритма, при котором конечное изображение создается из «срезов» исходной модели. Вышеописанные эксперименты направлены на исследование различных способов получения объема, оптимизации оптической системы и конструкции стенда в целом. Важным аспектом обеспечения функционирования макета и дисплея в целом является обеспечение синхронизации мультискранный системы при формировании

объемного изображения, решение этой задачи рассматривается ниже.

Выбор схемы построения программно-аппаратной части системы. В макете реализуется вывод изображений на несколько устройств, в нашем случае – проекторов. Аналогичная схема может использоваться в игровой индустрии для расширения виртуального пространства, создания видеостен из нескольких мониторов и др. При этом, как и в нашем случае, целесообразно создание стендов для проверки гипотез для определения характеристик процессов, протекающих при формировании объемного изображения. Кроме этого, при определении способов формирования и управления контентом изображений в создаваемых автостереоскопических дисплеях необходимо учитывать требования по быстродействию и гибкости соответствующих архитектур информационно-вычислительных средств [5-7], а также использовать апробированный математический аппарат для построения моделей трансформации изображений.

Для реализации стенда объемной визуализации необходимо формировать трехмерное изображение выводом подготовленного контента с разрешением 800x600 пикселей на массив проекторов в количестве 16 штук. Требуется сформировать объемное изображение, видимое в диапазоне 160 градусов по горизонтали (в дальнейшем планируется увеличить число видеовыходов и разрешение изображения). Для этого, в частности, необходимо апробировать проектные решения для программно-аппаратной реализации стенда объемной визуализации. После оценки различных способов формирования и управления контентом принято решение использовать довольно распространённый клиент-серверный способ подключения (рис. 2), т.к. он обладает достаточной гибкостью при масштабировании и приемлемой для поставленной задачи скоростью передачи данных. В основе этой схемы содержится использование локальной сети для рендеринга и отображения изображений. Компьютер клиента и N серверов объединяются в сеть. Каждый сервер соединяется с устройством вывода, отображающим необходимое изображение.

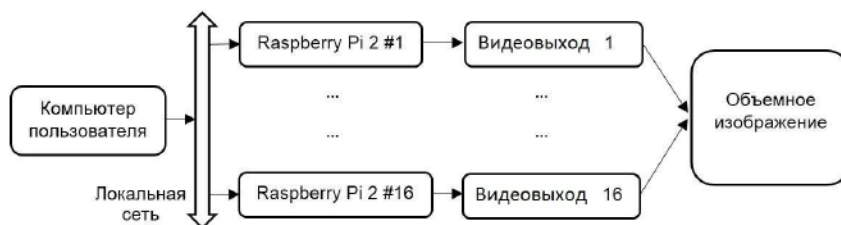


Рис. 2. Клиент-серверная схема подключения

В качестве серверов предлагается использовать одноплатные компьютеры Raspberry Pi 2 Model B. В процессе экспериментов выявлено, что наиболее подходящим для работы с VideoCore IV GPU процессора Broadcom BCM2836, поддерживающим максимальное вертикальное разрешение в 1080

пикселей (Full HD) является OMXPLAYER – специально разработанный для Raspberry проигрыватель с открытым исходным кодом. Лучший результат удалось достигнуть при использовании кода x264 [3]. Программа предназначена для синхронизации мультискранный системы автостерео-

скопического дисплея. При этом используются одноплатные компьютеры Raspberry Pi 2. Ниже рассматривается предложенный алгоритм для синхронизации функционирования мультиэкранной системы для выбранной клиент-серверной схемы подключения одноплатных компьютеров.

Описание алгоритма синхронизации мультиэкранной системы. Процедура синхронизации, следующая: от Host-компьютера образы изображений (кадры видеофайлов) принимаются последовательно, далее Host-компьютер передает первый образ первому экранному компьютеру и по завершении передачи выполняет передачу второго образа второму экранному компьютеру, затем следующему и так до последнего. Время передачи всего фрейма образа не регламентируется и зависит от технических характеристик канала связи. Далее полученные изображения синхронно выводятся по сигналу с GPIO портов плат (рис. 3).



Рис. 3. Блок-схема алгоритма синхронизации

Ключевые моменты работы программы заключаются в следующем. Создан класс, который предназначен для инкапсуляции работы с синхронизирующими сигналами. Они могут быть собраны программой из управляющего соединения через сеть Интернет со специально предназначенных для синхронизации аппаратных GPIO портов или захватывая синхронизирующие сигналы с терминального соединения. Класс формирует в результате абстрактное событие, которое сохраняется в хранилище. Далее по указателю располагается глобальное хранилище, которое представляет контейнер, содержащий очереди событий синхронизации и сжатых фреймов, требующих их декодирования аппаратным способом, а также очередь подготовленных для рендеринга кадров. Объект-хранилище создается и инициализируется однажды с использованием паттерна Singleton, и далее все компоненты обращаются к его единственной копии. В работе программы используются флаги, позволяющие выполняться соответствующим потокам-обработчикам. Флаги могут быть настроены отдельно специальными командами, запуская такими образом процесс считывания синхронизирующих сигналов с GPIO пина, из управляющих пакетов по сети или по событиям в терминале. Также заданы флаги, необходимые для запуска только одного экземпляра потоков, выполняющих соответствующие обработки синхронизирующих событий. Они запрещают повторное создание созданного потока. Для синхронизации используются специфичные члены класса. В случае синхронизации по сети – это номер порта, IP адрес машины, формирующей синхронизирующие пакеты, сокет, установленный для соединения с этой машиной, а также необходимые функции, которые создают, закрывают сокет и переходят в режим связи с удаленной синхронизирующей машиной. Для захвата событий с одноплатного компьютера используется номер GPIO контакта и функция, проверяющее наступление события аппаратной синхронизации. При этом, реализованы функции, которые запускаются в отдельных потоках исполнения. Каждая функция реализует собственный, специфичный метод определения события синхронизации. По аппаратному сигналу с GPIO контакта, по событию, пришедшему по соединению с удаленной машиной по Интернет, по событию в терминале. Несмотря на различные способы синхронизации, функции формируют абстрактные события, которые затем помещаются в очередь в глобальное хранилище. После этого, другие компоненты системы, используя соответствующий API объекта-хранилища, могут получить доступ к событиям и реагировать на них. Особенность подхода в том, что все компоненты системы работают в параллельных потоках исполнения и имеют общую точку – глобальное хранилище, включающее очереди для обмена событиями, фреймами, кадрами. Описанный класс выполняет роль компонента, от которого требуется формировать поток абстрактных синхронизирующих событий, которые могут быть восприняты всеми другими

компонентами системы, скрывая при этом реализацию детектирования этих событий от других модулей системы.

В конструкторе создаются отдельные потоки исполнения для каждого метода детектирования события синхронизации. В процессе создания потока динамически выделяется память, которую затем требуется освободить, чтобы не было «утечки» памяти. Также на основе паттерна Singleton организуется доступ к единственному экземпляру хранилища. Аналогичным образом получают доступ к хранилищу и остальные модули системы. В деструкторе типовым способом уничтожаются потоки, освобождается динамически выделенная под них память, а указатели на потоки исполнения обнуляются. Кроме этого, в деструкторе компонента вызывается метод, который восстанавливает параметры терминала, т.к. они изменены в процессе выполнения модуля.

В функции инициализации компонента происходит запуск трех отдельных потоков исполнения, каждый из которых выполняет специфические задачи для детектирования моментов синхронизации аппаратным способом, через сеть Интернет или по событиям в терминале. В потоке будет исполняться процесс детектирования событий в случае, когда такое поведение включено, т.е. установлен соответствующий флаг, используя API компонент. В противном случае, поток будет простаивать. При первом исполнении цикла детектирования событий, настраивается терминал. Переводится в неблокирующее состояние, отключается режим echo, отключается буферизация в ОС и деление потока на элементы по символам-разделителям.

При возникновении события, требующего завершения приложения, создается абстрактное событие и помещается в конец очереди в объекте-хранилище. Это событие особого характера. О том, что появился запрос на прекращение работы системы, «узнают» все компоненты, несмотря на количество событий в очереди. После этого компоненты прекратят штатный цикл работы и перейдут к завершению, освободив при этом динамически выделенную память. Кроме этого, выставляется флаг, требующий завершения потока исполнения, т.е. выхода из цикла обработки. Также происходит вызов прерывания основного потока в системе, который запускал все компоненты, затем перешел в режим «сна», а теперь необходимо его «разбудить» по прерыванию, чтобы он перешел в режим ожидания завершения всех компонентов, после этого освободил динамически выделенную память и завершил основную программу.

Определяются системно-зависимые переменные для работы с потоками, в которых заносится информация о потоке, его настройке, ключи и т.д. Также в каждом потоке исполнения находится общий мьютекс, т.к. он является членом класса, а не конкретного потока. В результате имеется возможность блокировать работу всех потоков

одновременно, если это требуется для работы одного из них.

Так как все компоненты системы работают параллельно, то может возникнуть ситуация одновременного чтения и записи общего ресурса. Для контроля над этим состоянием используется техника с мьютексом. Однако работая непосредственно с мьютексом, имеется возможность перейти в состояние вечной блокировки (deadlock), когда выход из функции произошел до снятия защиты. Для упрощения работы с мьютексами реализован класс, который закрывает мьютекс в собственном конструкторе и открывает в деструкторе. Соответственно, в критических функциях доступа к общему ресурсу достаточно сразу после входа в функцию создать объект класса. При выходе из функции объект автоматически уничтожится и мьютекс будет разблокирован.

Каждый поток исполнения имеет имя. Также для работы отдельного потока определено, что будет выделяться в памяти 20 страниц памяти. Одна страница имеет размер в 4КБ. Это достаточно для полноценной работы потока. Также устанавливается таймаут, который можно взвести, если требуется заставить поток быть неактивным некоторое время. После истечения установленного таймаута, поток вновь продолжает исполнение.

Выводы: рассмотрен предложенный алгоритм синхронизации видеовыходов с нескольких плат одноплатных компьютеров Raspberry Pi 2 для использования в макете стенда объемной визуализации. Реализующая его программа используется при создании перспективного автостереоскопического дисплея для получения 3D-изображения высокого качества. Его отличительными характеристиками является многоэлементная система вывода опорных изображений, что позволяет достигнуть повышения качества объемных образов на основе снижения удельной нагрузки на выходные элементы системы.

Работа выполнена при поддержке Фонда содействия инновациям.

СПИСОК ЛИТЕРАТУРЫ:

1. *Blundell, B. Volumetric Three-Dimensional Display Systems / B.G. Blundell, J. Adam // Schwarz, pp. 330.*
2. *Nagano, K. An Autostereoscopic Projector Array Optimized for 3D Facial Display / K. Nagano, A. Jones et al. Debevec // SIGGRAPH 2013 Emerging Technologies, 2013.*
3. *Большаков, А.А. Объемный дисплей и способ формирования трехмерных изображений / А.А. Большаков, А.В. Никонов // (19) RU (11) 2526901 (13) C1 // Заявка: 2013103443/28, 25.01.2013; Опубликовано: 27.08.2014; Бюл. № 21.*
4. *Большаков, А.А. Разработка объемного дисплея для решения задач визуализации и диспетчеризации / А.А. Большаков, А.А. Сгибнев, М.А. Железов, А.В. Мельников // Автоматизация в промышленности. 2016. №7. С. 31-34.*

5. Jones, A. An Automultiscopic Projector Array for Interactive Digital Humans / A. Jones, J. Unger, K. Nagan et al. // In SIGGRAPH 2015. ACM Press. 2015.
6. Annen, T. Distributed rendering for multiview parallax displays / T. Annen, W. Matusik, H. Pfister et al. // Proc. SPIE 6055, Stereoscopic Displays and Virtual Reality Systems XIII. 2006.
7. Большаков, А.А. Выбор и реализация программно-аппаратного комплекса для вывода массива изображений в стенде объемной визуализации / А.А. Большаков, А.А. Сгибнев // Вестник Саратовского государственного технического университета. 2015. Т. 4, №1(81). С. 120-125.

SYNCHRONIZATION ALGORITHM OF MULTISCREEN SYSTEM FOR THE VOLUMETRIC VISUALIZATION TESTING UNIT

© 2017 A.A. Sgibnev

Saratov State Technical University named after Yu.A. Gagarin

In the paper there is described the algorithm of video file synchronization for output devices such as array of projectors that are used in the volumetric display testing unit.

Key words: *three-dimensional visualization, volumetric display, algorithm, synchronization, array of projectors*