Пользовательский контент в браузере дополненной реальности.

Баулин И.Н., Намиот Д.Е.

***Аннотация*— В данной статье изложены результаты магистерской диссертации, выполненной одним из авторов в Лаборатории ОИТ во время обучения в магистратуре на факультете ВМК МГУ им. М.В. Ломоносова. В работе развиваются идеи для представления пользовательского контента в системах дополненной реальности. В работе рассматривается спроектированная инструментальная система для создания подобного рода сервисов.**

***Ключевые слова*—дополненная реальность, контент, мэшап.**

I. Введение

Настоящая работа является дальнейшим развитием идей, изложенных в работе, посвященной созданию слоев для браузеров дополненной реальности [1]. Дополненная реальность (далее по тексту будут использоваться англоязычные термины Augmented Reality, AR) – это одна из разновидностей виртуальной реальности; представляет собой реальность, в которой виртуальные трехмерные объекты интегрируются в реальное трехмерное окружение (то есть реальное окружение дополняется виртуальными элементами). Главное отличие этих терминов состоит в том, что в виртуальной реальности пользователь полностью погружается в виртуальное окружение и перестает видеть реальные объекты вокруг себя, в дополненной же реальности реальные объекты, так или иначе продолжают быть доступны пользователю, но они интегрируются с виртуальными элементами. В идеале эта интеграция должна быть незаметна для человека. Существует два разных подхода к созданию дисплеев для реализации дополненной реальности - видеопрозрачные дисплеи и оптически прозрачные. В оптически прозрачных дисплеях реальный мир человек видит посредством своего собственного зрения, а для отображения виртуальных элементов используются специальные оптические комбинаторы. В видеопрозрачных дисплеях картинка реального мира обрабатывается в реальном времени видеокамерой, после чего каждый приходящий кадр дополняется виртуальными объектами, и итоговое комбинированное изображение показывается человеку. Дисплеи мобильных телефонов относятся как раз к типу видеопрозрачных дисплеев для дополненной реальности.

Стоит отметить, что понятие дополненной реальности относится не только к дополнению нашего зрения виртуальными элементами, но и в теории может относиться к любому из человеческих чувств восприятия информации. Например, дополненная реальность может быть использована при формировании звукового потока. Представим себе, что пользователь использует наушники, оснащенные внешним микрофоном. Внешние микрофоны фиксируют реальные звуки, поступающие из внешнего окружения, а наушники дополняют этот реальный звук синтетическим звуком. Кроме того, помимо дополнения реальной картинки виртуальными объектами, мы можем скрывать от пользователя реальные объекты, используя виртуальные эффекты. Например, что скрыть от человека стоящий перед ним стол, достаточно нарисовать перед этим столом виртуальную картинку, соответствующую тому, что находится за этим столом.

Для иллюстрации можно привести один из самых знаменитых примеров дополненной реальности, изображенной в кино. В кинофильме «Терминатор 2» зрения робота обладало возможностью выводить на экран всю информацию об окружающих объектах одновременно с реальной картинкой. (см. рис.1)

Рисунок 1. AR пример

Исторически для накладывания виртуальных объектов на реальное изображение использовались разнообразные

Статья получена 20 октября 2013.

Баулин А.Н. – выпускник магистратуры факультета Вычислительной Математики и Кибернетики Московского государственного университета им. Ломоносова.

Намиот Д.Е. – старший научный сотрудник лаборатории ОИТ факультета Вычислительной Математики и Кибернетики Московского государственного университета им. Ломоносова.

специальные устройства – шлемы дополненной реальности, видеоочки и т.д. Для того чтобы избежать привязки термина дополненной реальности к конкретным технологиям, классическое определение [2] предлагает рассматривать дополненную реальность как систему, которая:

1. Дополняет реальный мир виртуальными элементами
2. Дополнение происходит в реальном времени
3. Дополнение должно происходить в трехмерном пространстве

Стоит отдельно отметить, что в определении дополненной реальности крайне важна интерактивность происходящего. Таким образом, отснятый на видеокамеру материал, который был впоследствии совмещен с виртуальными элементами (такое повсеместно используется в кино) не может быть примером системы дополненной реальности.

Раньше технологии дополненной реальности были крайне сложными, дорогостоящими и были недоступны простым пользователям. Например, одной из самых востребованных областей, в которой применялись системы дополненной реальности, была военная сфера. Так летчикам на самолетах-истребителях уже давно доступна сложная система наведения на цель, в которой на реальную картинку накладывается виртуальная траектория, по которой может быть пущен снаряд. Также дополненная реальность была востребована в космических исследованиях и робототехнике.

В настоящий момент с развитием вычислительной техники и появлением новых технологий ситуация начинает меняться и дополненная реальность становится доступной простым пользователям. Системы дополненной реальности обычно обладают тремя основными компонентами – собственно само устройство, система позиционирования и система отображения. В качестве системы отображения в мобильных устройствах используется камера мобильного телефона. Для позиционирования мобильного устройства и построения некой виртуальной модели реального мира (для последующего дополнения этой модели виртуальными элементами) может использоваться как информация, поступающая с встроенного в телефон GPS датчика и G-сенсора, так и использоваться заранее предопределенные изображения-маркеры, которые распознаются мобильными устройствами посредством алгоритмов машинного зрения. Развитие современных мобильных устройств достигло такого уровня, что их вычислительная мощь позволяет реализовывать реальные прикладные AR-системы.

Целью данной работы было исследовать различные подходы, решения для построения AR-приложений и, как результат, предложить подход к применению популярной сейчас концепции пользовательского контента (в работе используются также англоязычные термины user generated content и user created content - UGC, UCC) в приложениях дополненной реальности.

Понятие User Generated Content тесно и неразрывно связано с понятиями Web 2.0 и participative web (дословно можно перевести как «общий» интернет). Понятие participative web основано на быстро развивающихся в интернете сервисах, которые дают конечным пользователям возможности способствовать созданию, оцениванию и распространению интернет контента, а также настройки веб-приложений под себя [2]. UGC дает большие преимущества онлайн сервисам по сравнению с традиционными веб-ресурсами, где контент обновляется исключительно администратором ресурса. В наполнении ресурса участвуют несравнимо большее число человек, каждый из которых как конечный пользователь сервиса заинтересован в том, чтобы выкладывать имеющийся у него контент (как медиаданные - видеоролики, фотографии, так и имеющиеся у человека сведения, знания). Кроме того, сами пользователи обычно имеют обычно возможность оценивать (обсуждать, комментировать и даже редактировать) имеющийся материал, что делает его лучше (актуальнее и т.д.). Подавляющее большинство из получивших недавно широкое распространение веб-сервисов как раз, так или иначе, реализуют у себя концепцию UGC. В качестве примера можно привести модные сейчас социальные сети (Facebook, MySpace, LinkedIn и т.д.), сервисы онлайн хранения фотографий (Flickr, Picasa Web Albums), Youtube, LiveJournal и, конечно, Wikipedia.

Хотелось бы использовать эту привлекательную идею пользовательского контента и в приложениях дополненной реальности для мобильных устройств. Задача работы состоит в том, чтобы предложить подход к созданию таких систем/ приложений. Предлагаемая система должна обладать следующими характеристиками:

- Весь контент хранится у поставщиков контента и обновляется конечными пользователями на стороне поставщика контента
- Контент должен динамически запрашиваться у поставщиков контента по запросу от мобильного AR-приложения
- Архитектура системы должна давать возможность безболезненно интегрировать новых поставщиков контента
- Возможность выбора поставщиков контента при формировании запроса на выборку данных
- Возможность использования нескольких поставщиков контента одновременно (технически работа с данными, пришедшими от разных поставщиков контента, никак не должна различаться для мобильного приложения)

Кроме того в рамках работы необходимо рассмотреть возможность использования одно и того же сервиса предоставления контента разными мобильными приложениями дополненной реальности.

То есть хотелось бы рассмотреть возможность создания некоторой общей архитектурной модели, которая с одной стороны позволила бы суммировать информацию от нескольких поставщиков контента, а с другой стороны, позволила бы использовать один и тот

же сервис сбора контента несколькими мобильными AR приложениями одновременно.

II. Существующие решения

Как уже говорилось выше, одна из основных задач любого AR приложения - каким-то образом привязать виртуальные объекты к объектам реального мира, но для этого сначала необходимо получить представление о том, как выглядит реальный мир, извлечь необходимую метаинформацию об окружающем пользователя реальном окружении. Для этих целей существует два основных подхода - trackable surfaces и геопозиционирование.

Trackable surfaces [3] (общераспространенного перевода на русский данного термина пока нет, далее по тексту будет использоваться как перевод - отслеживаемые поверхности) - это подход получения информации об окружающем реальном мире посредством поиска в этом реальном мире заранее известных приложению отслеживаемых поверхностей. То есть мобильное приложение, используя алгоритмы машинного зрения, ищет в реальном мире одну из известных этому приложению картинок. В случае успеха поиска строит относительно этой картинки модельную систему координат, после чего дополняет ее виртуальными элементами. Подход trackable surfaces можно считать развитием идеи QR кодов - матричных черно-белых кодов, используемых для кодировании информации об объекте, которые можно прочитать сканирующим оборудованием, в том числе камерой мобильного телефона. Пример приведен на рисунке 2. Одной из самых известных разработок в этом направлении - программный набор для создания AR приложений от компании Qualcomm описан ниже.

Рисунок. 2 Пример QR-кода

Гео-позиционирование - подход к получению информации об окружающем реальном мире на основе данных о географическом местоположении пользователя и направлении камеры его мобильного телефона. Информация о гео-координатах получается посредством встроенного в телефон GPS датчика, а информация о направлении - с встроенного в телефон G-сенсора. Получив эту информацию мобильное приложение также способно выстроить модельную систему координат и дополнить эту систему координат виртуальными объектами, связанными с данным географическим местом. Такой подход реализован в двух самых известных на сегодняшний день браузерах дополненной реальности - Layar и Wikitude. Об этих браузерах будет рассказано ниже.

III. QCAR SDK

Qualcomm Augmented Reailty SDK (сокращенно QCAR SDK) представляет собой набор для разработки приложений дополненной реальности на основе платформы, предлагаемой Qualcomm [4]. Общая схема предлагаемой платформы изображена на рисунке 3.

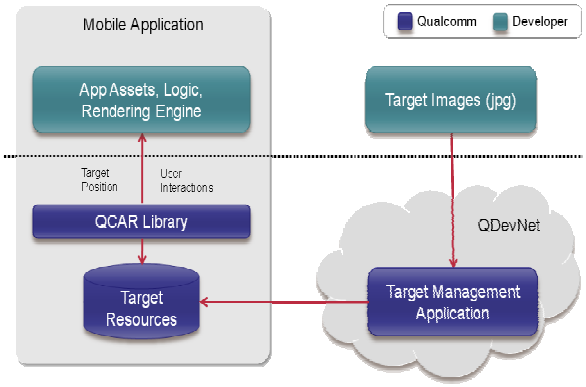

Рисунок 3. QCAR SDK

Платформа состоит из двух основных частей - библиотеки QCAR, которую может использовать разработчик при создании нового мобильного AR приложения и системы управления целями (Target Management System), которая находится на портале для разработчиков Qualcomm. Идея состоит в том, что разработчик нового приложения загружает на этот портал изображение, которое в дальнейшем будет использоваться мобильным приложением построения модельной системной координат в реальном мире. Загруженное изображение специальным образом обрабатывается и на выходе получается набор целевых ресурсов, которые разработчик должен связать со своим приложением. Таким образом, часть работы по созданию нового AR приложения лежит на разработчике этого приложения, а часть работы выполняется на стороне платформы QCAR. Соответствующее разделение работ выделено на схеме разным цветом.

QCAR предлагает на выбор три различных варианта отслеживаемых поверхностей - image targets, multi targets и рамочные маркеры.

Image targets (целевое изображение) представляет собой любое изображение, которое может отслеживать QCAR SDK. На деле это может быть любая нерегулярная текстура, что является большим преимуществом по сравнению с традиционными маркерами и QR кодами, для которых обязательно наличие специальных черно-белых областей или специальных кодов, для того чтобы сделать возможным распознавание. QCAR SDK использует продвинутые

алгоритмы обнаружения и отслеживания особенностей каждого изображения. На основе этих особенностей происходит сравнение картинки с камеры мобильного телефона с базой целевых ресурсов, которая заранее подготавливается разработчиком посредством Target Management System. Таким образом, особенности каждого изображения выявляются при загрузке этого изображения и сохраняются в базе данных, которая затем используется во время выполнения приложения. Как только целевое изображение обнаружено, QCAR SDK способно отслеживать это изображение до тех пор пока хотя бы часть изображения попадает в картинку, получаемую с камеры мобильного телефона. Одновременно могут отслеживаться до 5 целевых изображений, производительность приложения сильно зависит от загрузки центрального процессора и GPU.

Не каждое изображение способно стать хорошим целевым ресурсом. Подбираемое изображение должно иметь хорошую освещенность и яркость, а также насыщенный цвет. В изображении не должны встречаться повторяющиеся шаблоны. На рисунке 4 показано изображение, пригодное для использования как целевой ресурс, а также набор особенностей, который выдела Target Management System при загрузке данного изображения (рисунок. 5).

Рисунок 4. Пригодное изображение

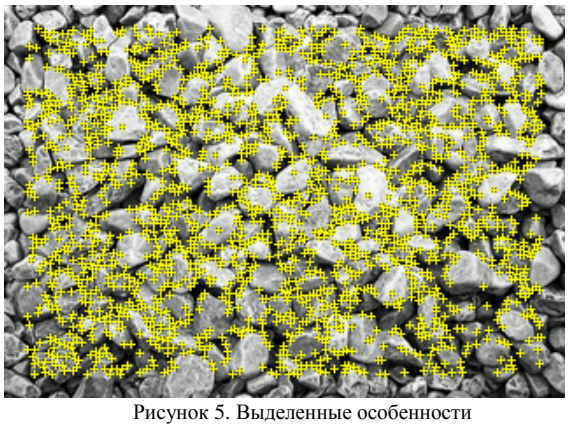
Рисунок 5. Выделенные особенности

Другой тип отслеживаемой поверхности - multi targets состоит из нескольких целевых изображений Image Targets, которые состоят в фиксированных пространственных отношениях. Как только одна из частей multi target обнаружена, все другие части могут быть отслежены только на основании этих пространственных отношений. До тех пор пока хотя бы одна часть изображения частична видна на изображении с камеры, QCAR способна отслеживать эту поверхность. Различие между использованием нескольких отдельных целевых изображений и одного multi target состоит в том, что в последнем случае поверхность представляет собой один единый объект отслеживания, а не несколько отдельных объектов со своей собственной информацией о местоположении в пространстве как в случае с несколькими отдельными Image Targets.

Multi targets также создаются при загрузке изображений через Target Management System. Особенности изображения выделяются при загрузке и сохраняются в базу данных. Пространственные отношения отдельных частей multi targets сохраняются в специальном конфигурационном xml файле. Другой способ создания multi target - использование набор специальных программных вызовов прямо во время выполнения приложения. Используя эти вызовы можно добавлять и удалять части multi target, а также менять информацию о пространственных отношениях. Эта гибкость может использоваться для создания единой информации о местоположении из нескольких целевых изображений или для отслеживания объектов, которые обладают меняющейся, но известной пространственной конфигурацией.

Последний тип возможных отслеживаемых поверхностей - рамочные маркеры сильно отличаются от остальных типов. При использовании этого типа не происходит предварительный анализ изображения для построения набора особенностей. Вместо этого специальный уникальный идентификатор рамочного маркера кодируется в бинарный шаблон вдоль границы маркируемого изображения. Внутри рамочного маркера может использоваться абсолютно любое изображение. Для распознавания данной поверхности необходимо чтобы вся рамка, в которой закодирована информация об идентификаторе маркера, была видна на картинке с камеры. Рамочные маркеры не генерируются посредством Target Management System, а распространяются отдельным архивом при установке QCAR SDK. С учетом того, что данный тип маркера требует меньшее количество вычислительных ресурсов, каждое приложение может использовать до 512 рамочных маркера одновременно. Пример рамочного маркера изображен на рисунке 6.

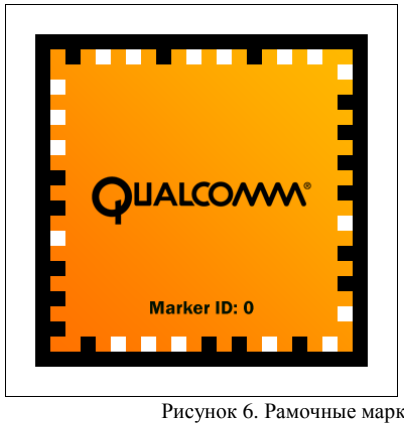

Рисунок 6. Рамочные маркеры.

На отслеживаемой поверхности могут быть определены так называемые виртуальные кнопки. Виртуальные кнопки представляют собой определенные разработчиком прямоугольные области на целевом изображении, которые при прикосновении (фактически перекрывании) инициируют некоторое программное событие. Виртуальные кнопки могут быть использованы для реализации таких событий как нажатие на кнопку или для определения того, что определенные области целевых изображений перекрываются другим объектом.

При выполнении приложения, написанного посредством QCAR SDK, каждый кадр, пришедший с камеры мобильного телефона, проходит через специальный конвейер, который изображен на рисунке 7 и описан ниже.

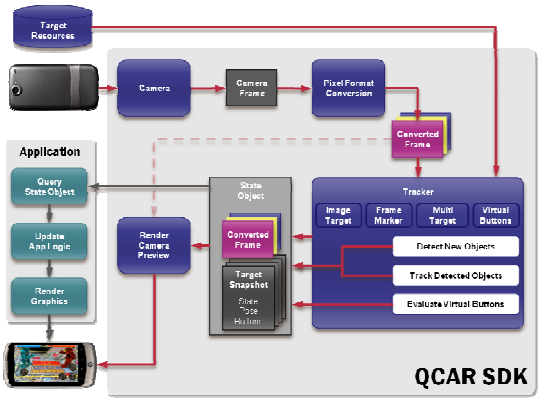

Рисунок 7. QCAR SDK

Каждый кадр, пришедший с камеры, поступает на конвейер в формате и размере, которые зависят от конкретного используемого мобильного устройства. Специальный преобразователь формата пикселей преобразует кадр из формата камеры (например, YUV12) в формат подходящий для рендеринга OpenGL ES (например, RGB565). Каждый приходящий кадр также сжимается для того чтобы иметь версию одного и того же кадра в разном разрешении.

Программный компонент под названием *Tracker* содержит алгоритмы машинного зрения, которые обнаруживают и отслеживают объекты реального мира в кадрах камеры. Как уже говорилось, для обнаружения определенных изображений используются целевые ресурсы этих изображений. Результат обнаружения целевых изображений, факта нажатия на виртуальные кнопки записывается в специальный объект состояния, который используется при фоновом рендеринге и доступен из кода приложения.

Фоновый рендерер ответственен за рендеринг пришедшего с камеры кадра, который хранится в объекте состояния. Производительность видео рендеринга оптимизируется под каждое конкретное устройство.

Задача разработчика приложения состоит в том, чтобы проинициализировать все вышеупомянутые компоненты и произвести три ключевых действия в коде своего приложения. Для каждого обработанного кадра разработчик приложения должен:

1. Запросить объект состояния, содержащий все вновь обнаруженные отслеживаемые поверхности, а также информацию об обновлении состояний существующих поверхностей.

2. Проанализировать новые данные согласно логике приложения.

3. Произвести рендеринг виртуальных объектов, которые будут дополнять реальную картинку, пришедшую с камеры.

Пример результата, который получается на выходе из конвейера, изображен на рисунке 8.

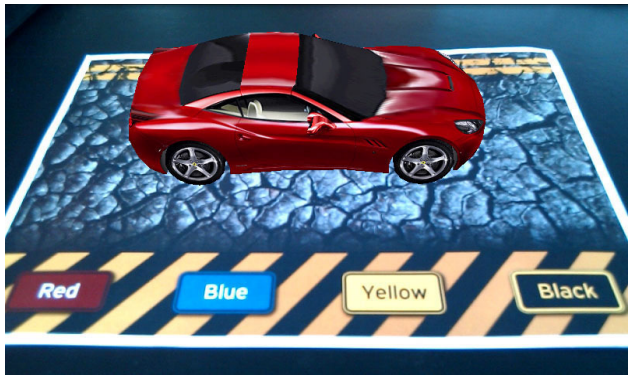

Рисунок 8. Выдача ковейера

IV. LAYAR

Сервис Layar [5] представляет из себя браузер дополненной реальности, который позволяет смотреть на окружающий мир, дополненный виртуальными элементами посредством, так называемых, слоев. Каждый слой содержит некоторую информацию, привязанную к географическому местоположению пользователя.

Приложение может работать в нескольких режимах, самый интересный из которых – режим дополненной реальности. В этом режиме картинка, получаемая с камеры мобильного устройства, дополняется виртуальными элементами, связанные с конкретным слоем. Каждый слой содержит свою собственную информацию. В самом простом случае это может быть слой, содержащий, к примеру, информацию обо всех ресторанах итальянской кухни в окрестности пользователя. Вся информация привязана к так называемым точкам интереса, которые изображены на рисунке в виде отдельных элементов (кругов) и связаны с конкретным географическим положением. Массив точек интереса, доступных в данный момент пользователю, определяется географическими координатами, получаемыми с GPS устройства, направлением и углом наклона камеры. Размер круга определяется расстоянием от пользователя до точки интереса.

Под картинкой, получаемой с камеры, отображается информация, связанная с выбранной точкой интереса. Эта информация включает в себя текстовое описание точки, изображение, и список действий, доступных пользователю. В качестве возможных действий пользователь может позвонить по телефону, отправить

sms/email, перейти на предложенный веб-сайт (при этом просматриваемый веб-сайт будет интегрирован в Layar). Если на устройстве установлены дополнительные приложения для навигации (например, Google Maps), то пользователь может посмотреть маршрут до выбранной точки интереса. С конкретной точкой интереса можно связать триггеры, запускающие определенные события при приближении к данной точке. Например, с приближением к точке можно связать воспроизведение некоторой мелодии или появление на картинке некоторого трехмерного объекта.

Каждую точку интереса и список действий с ней связанных определяет и настраивает разработчик слоя, к которому привязана данная точка. Таким образом, слой дополненной реальности можно определить как совокупность всех точек интереса данного слоя и некоторой статической метаинформации, связанной с данным слоем. При этом статическая метаинформация постоянно хранится на серверах самого сервиса Layar, а динамическая информация о массиве точек интереса получается в реальном времени от разработчика слоя.

В меню программы, в специальном репозитории доступно большое количество разнообразных слоев. Большая часть из этих слоев создана сторонними разработчиками и предоставляет пользователю самую разнообразную функциональность. Пользователь имеет возможность загрузить понравившийся слой и даже переключаться между слоями в реальном времени. Таким образом, слой может служить аналогом вкладок в обычном браузере, по которым пользователь переключается для получения различного контента.

Помимо режима дополненной реальности Layar может отображать информацию еще в двух режимах. В одном из этих режимов точки интереса выведены простым списком и отсортированы в порядке возрастания расстояния от пользователя до точки. В другом режиме точки интереса отображены на карте. При этом вся функциональность, доступная в режиме дополненной реальности, сохраняется.

Layar имеет многокомпонентную архитектуру, направленную на то, чтобы скрыть от разработчиков новых слоев детали работы с мобильным устройствам и максимально облегчить интеграцию существующих сервисов (в том числе сервисов Web 2.0) в существующую архитектуру этого AR-браузера.

Клиентская часть приложения запускается на мобильном устройстве. Мобильный клиент получает всю необходимую информацию от внешних устройств посредством собственных интерфейсов мобильного устройства. В частности, от GPS модуля клиент получает географические координаты пользователя (ширина, долгота). Помимо этого, для определения направления обзора пользователя используется G-сенсор. Для отображения окружающей местности в режиме дополненной реальности используется внутренняя камера устройства. Виртуальные объекты добавляются поверх картинки с камеры самим приложением. Таким образом, мобильный клиент инкапсулирует в себе всю работу по взаимодействию с аппаратными ресурсами мобильного устройства и освобождает разработчиков новой функциональности от обязанности вникать в детали реализации под конкретную мобильную платформу.

Для отображения местности в режиме карты сама карта подгружается, используя собственный программный интерфейс устройства. Данные предоставляются поставщиком карты местности по умолчанию.

Мобильный клиент чаще всего обращается за всей необходимой информацией к серверу Layar. Общение происходит посредством Layar Client API. Детали реализации этого программного интерфейса скрыты от конечных пользователей и разработчиков новых слоев.

На сервере Layar хранится только статическая информация. Она включает в себя информацию об используемом слое (его заголовок, описание, внешний вид и т.д.). Вся эта информация предварительно загружается сторонним разработчиком новой функциональности через специально созданный веб-сайт (Layar Provisioning WebSite).

Вся динамическая информация, в частности информация о списке точек интереса для данной местности подгружается динамически в тот момент, когда пользователь загрузил данный слой. Эта динамическая информация автоматически запрашивается клиентом автоматически каждые 5 минут или, если положение пользователя изменилось более чем на 100 метров.

В этом смысле сервер Layar работает как прокси-сервер к серверу разработчиков. Для реализации этого взаимодействия используется так называемый Layar Developer API. Этот программный интерфейс полностью открыт сторонним разработчикам [6].

Каждый раз, когда пользователь Layar выбирает новый слой, он ожидает, что точки интереса будут предоставлены ему как можно быстрее. Layar API предоставляет механизм разделения массива точек интереса на несколько отдельных массивов, для того чтобы ускорить процесс отображения этих точек. Обычно ожидается, что пользователь может наблюдать одновременно около 10 точек интереса на экране. В случае, если размер массива точек, удовлетворяющий поисковым критериям, больше этого числа, то разработчик в посланном JSON ответе, может указать специальный ключ, что оставшиеся точки могут быть запрошены отдельным запросом. Но в любом случае пользователю не будет показано более 50 точек интереса. Слишком большое число точек интересно очевидно перегружает внимание пользователя и препятствует обработке и восприятию поступающей информации

Фиксированная информация о слое загружается разработчиком слоя через специально созданный веб-интерфейс Layar Provisioning WebSite. Вся предоставленная разработчиком информация загружается на сервер Layar и, таким образом, становится доступна для мобильного клиента. К фиксированной информации в первую очередь относится информация о внешнем виде слоя. При создании собственного слоя можно изменять цвета всех

элементов управления, а также менять логотипы и баннеры в пользовательском интерфейсе мобильного клиента. Благодаря этому внешний вид различных слоев может достаточно сильно отличаться. Кроме того на веб-сайте определяются фиксированный набор фильтров, связанных со слоем.

Как уже говорилось, через веб-интерфейс на сервер Layar загружается только статическая информация о слое. Динамическая информация загружается в реальном времени с веб-сервиса, предоставленного разработчиком. Ссылку на веб-сервис разработчик также указывает при первоначальной загрузке статической информации о слое через веб-интерфейс.

Слой может быть доступен в двух режимах – тестовом и опубликованном. Тестовый режим используется только на время разработки, создаваемый слой доступен только разработчику слоя по специально созданному ключу. После публикации слой появляется в репозитории Layar и становится доступным для использования всем пользователям.

Динамическая информация об уровне подгружается в реальном времени с сервера разработчика. Интерфейс взаимодействия сервера Layar с веб-сервисом, который должен предоставить разработчик уровня, открыт и хорошо документирован. Для запроса информации о точках интереса, связанных с текущей локацией, используется протокол HTTP. Например, HTTP GET запрос *GetPointsOfInterest*. Все вызовы веб-сервиса должны придерживаться архитектуры REST.

В качестве формата ответа от сервера разработчика слоя Layar API в данный момент понимает только формат JSON.

V. WIKITUDE

Одним из главных конкурентов Layar в области мобильных браузеров дополненной реальности является Wikitude [7]. В отличие от Layar, который представляет собой единое приложение, Wikitude состоит из нескольких отдельных ветвей. Wikitude World Browser представляет из себя мобильное приложение дополненной реальности, которое обрабатывает информацию с GPS, компаса и акселерометра и дополняет получаемую с фотокамеры картинку виртуальными элементами. Wikitude также может отображать информацию как в режиме камеры, так и в режиме карты. Пример работы wikitude в режиме дополненной реальности изображен на рисунке 9.

Существует несколько способов добавить информацию о собственных точках интереса на Wikitude. Если пользователь хочет отметить отдельную точку интереса, он может это сделать в online-режиме через веб-сайт wikitude.me. На веб-сайте доступен простой интерфейс, который позволяет пользователю отметить своё любимое место посредством интерфейса Google Maps. Информация о точке интереса загружается в базу данных wikitude и становится доступна из мобильного браузера. Самое интересное, что посредством веб-сайта пользователь может редактировать точки интереса, созданные другими пользователями, и даже удалять их. При этом существует специальная команда модераторов, которая отслеживает все изменения в базу данных точек интереса. Весь этот механизм очень похож на механизм функционирования Википедии.

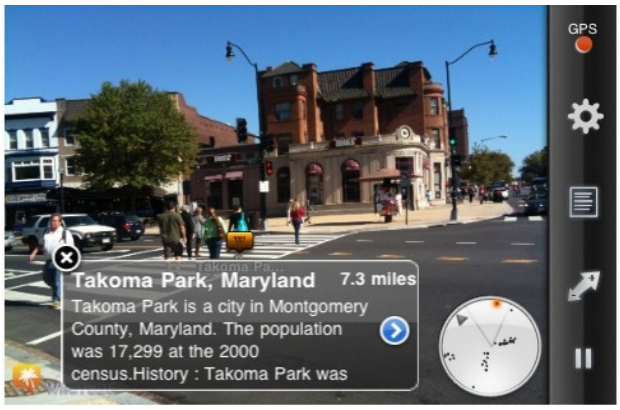

Рисунок 9. Wikitude

Понятно, что одиночная загрузка точек интереса может быть интересна только конечному пользователю, который просто хочет увидеть информацию о своем любимом отеле/ресторане/баре в браузере дополненной реальности. В большинстве случаев ручная загрузка каждой отдельной точки слишком трудоемка и неприменима. Например, если мы хотим создать большой путеводитель по основным достопримечательностям города. Если пользователь владеет большим набором точек интереса, он может потоково загрузить всю информацию о них (в том числе картинки и логотипы) в базу данных Wikitude, используя форматы KML и ARML. На том же веб-сайте wikitude.me пристутсвует веб-нитерфейс для загрузки файлов этих форматов.

KML (Keyhole Markup Language) [8] предствляет из себя основанный на XML язык разметки для представления трехмерных геопространственных данных в программе Google Earth. KML связывает некоторую информацию о географическом местоположении с тем как эта информация отображается в Google Earth (в виде простой пометки, картинки или трехмерного изображения). При этом сам KML файл можно получить двумя способами – экспортировать его из Google Earth или написать вручную.

ARML(Augmented Reality Markup Language) [9] представляет из себя все тот же KML, только дополненный функциональностью, специфичной для приложений дополненной реальности. Создатели спецификации ARML позиционируют свой формат как попытка создания единого стандарта для представления точек интереса.[3] По задумке, все браузеры дополненной реальности будут уметь обрабатывать точки интереса, заданные в этом формате, и это позволит избежать привязке точек интереса к конкретному браузеру дополненной реальности. ARML позволяет связывать точки интереса с ссылками на веб-сайты, телефонными номерами, адресами e-mail и т.д.

Функциональность, доступная на всех браузерах дополненной реальности, определена в пространстве имен ar. Функциональность, специфичная для каждого конкретного браузера дополненной реальности, определена в собственном пространстве имен. Например, для Wikitude это пространство wikitude.

ARML документ также можно создать несколькими способами. Во-первых, его можно написать вручную. Во-вторых, его можно получить из существующего KML файла, опять же добавив необходимую информацию вручную. В ближайшем будущем разработчики обещают создание пользовательского интерфейса для удобной генерации этих ARML файлов. Стоит отметит, что информация, загружаемая из ARML файла очень похоже на информацию о массиве точек, которая в Layar загружается в реальном времени посредством вызова веб-сервиса разработчика.

В Wikitude есть возможность, аналогично Layar, получать информацию о массиве точек интереса с веб-сервиса поставщика контента. Схема взаимодействия мобильного клиента, сервера Wikitude и сервера поставщика данных изображена на рисунке 10.

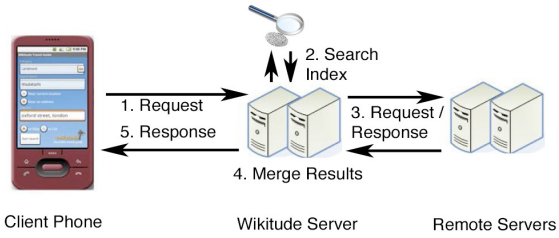

Рисунок 10. Wikitude сервисы

Когда мобильный клиент передает на сервер Wikitude информацию о своем местоположении, на сервере Wikitude запускается два параллельных процесса – локальный поиск по внутренней базе данных самого Wikitude (т.е. фактически по информации, загруженной через wikitude.me) и поиск информации о массиве точек интереса на удаленных серверах. При этом каждый сторонний веб-сервис обязан послать в ответ все миры (аналог слоев в Layar), которые он обрабатывает. Время ответа сервера разработчика жестко ограничено 3-5 секундами. Если сервер wikitude за это время не получил ответа от веб-сервиса, результаты с этого в веб-сервиса не попадут в окончательную выборку. После этого результаты удаленного и локального поиска сливаются и предоставляются мобильному клиенту.

Технически дата запрашивается от веб-сервера посредством HTTP GET запроса. В ответ сервер Wikitude получает данные в описанном выше формате ARML. Стоит отметить, что в основе ARML лежит XML, более высокоуровневый и тяжеловесный формат обмена данными, чем JSON, используемый в Layar. Теоретически это может привести к некоторому падению производительности при обмене данными с удаленными серверами, но зато дает сообществу разработчиков систем дополненной реальности возможность унифицировать формат передачи массива точек интереса и таким образом в перспективе облегчить им жизнь.

Данная архитектура достаточно сильно отличается от архитектуры Layar. Сервер Layar хранит только фиксированную информацию об используемом слое, вся же динамическая информация подгружалась с одного конкретного сервера разработчика данного конкретного слоя. В Wikitude же при регистрации веб-сервиса разработчик указывает информацию, о каких мирах предоставляет информацию его веб-сервис. Таким образом, сервер Wikitude вынужден для каждого запроса клиента обращаться за информацией ко всем серверам разработчиков, которые заявили, что владеют информацией о массиве точек интереса для этого мира. Помимо этого сервер Wikitude вынужден запускать локальный поиск для отражения точек интереса, загруженных через веб-интерфейс. Такая колоссальная загрузка на сервер вынуждает разработчков Wikitude вводить определенные ограничения. Например, на сервере Wikitude реализованы сложные алгоритмы, которые определяют, есть ли у определенного мира точки интереса в данной определенной местности. Если система определила, что этот мир в этой области не используется, то сервер Wikitude какое-то время не будет запрашивать информацию об этом мире в этой области, для того чтобы снизить нагрузку на сервер.

VI. User Generated Content в AR

Общая идея предлагаемого подхода к реализации User Generated Content в AR приложениях состоит в том, чтобы расширить функции серверного приложения (серверного посредника), с которого мобильный клиент, так или иначе, загружает необходимый ему контент. Серверное приложение должно выступать как некий программный посредник, с которым можно легко интегрировать различных поставщиков контента. Предполагается, что данные поставщики контента наполняются пользовательским контентом. То есть, серверное приложение должно работать с несколькими поставщиками контента, брать на себя и скрывать от мобильного приложения все детали программных вызовов, специфичных для каждого отдельного поставщика контента. Серверное приложение должно быть способно динамически (без перекомпиляции) подключать и отключать поставщиков контента или по запросу мобильного клиента (если мы предоставляем пользователю сведения об источниках контента), или по желанию администратора системы. То есть, в случае пришедшего от мобильного клиента запроса на выдачу контента, задачей предлагаемого программного посредника является обработка пришедшего запроса, формирование новых запросов к используемым поставщикам контента в известных им терминах и объединение пришедших результатов в один единый ответ мобильному клиенту.

Второй важной характеристикой серверного посредника – возможность работы с ним одновременно нескольких браузеров дополненной реальности. На самом деле массив информации, запрашиваемый

браузерами дополненной реальности очень похож. Семантически это обычно одна и та же информация (некоторый ресурс, ссылка на него, текстовое описание, географическое положение). Синтаксически же одна и та же семантика скрывается за вызовами разных программных интерфейсов. Задача программного посредника:

- понимать синтаксис разных программных интерфейсов разных браузеров дополненной реальности
- обеспечивать возможность выбора некоторой группы поставщиков контента из списка доступных (или всех)
- запрашивать у поставщиков контента необходимый набор точек интереса (независимо от того с какого браузера дополненной реальности пришел вызов)
- формировать корректный серверный ответ в том формате, в каком его ждет соответствующий браузер дополненной реальности.
- позволять легко добавлять в систему новых поставщиков контента и новые браузеры дополненной реальности

Идея программного посредника с расширенным функционалом применима для любого браузера дополненной реальности, где присутствует возможность запрашивать требуемый контент с удаленных серверов поставщиков данных. В той или иной форме такой возможностью обладают и Layar, и Wikitude.

Общая схема предлагаемого подхода изображена на рисунке 11. На схеме отсутствуют упоминания о каких-то конкретных браузерах дополненной реальности, протоколах или поставщиках данных. Данная схема универсальна и может быть использована при работе с различными системами.

Для иллюстрации предлагаемого подхода было создано демонстрационное приложение, использующее идею описанную выше. Демонстрационное приложение представляет собой новый слой для браузера дополненной реальности Layar. Предлагаемый контент для слоя - фотографии старой Москвы. То есть пользователь, выбравший соответствующий слой в каталоге слоев, при наведении камеры своего мобильного устройства на какой-то географический или архитектурный объект (здание, площадь, парк и т.п.) должен получить возможность увидеть фотографии этого объекта (или того, что было на его месте) в прошлом.

В качестве поставщиков контента для создаваемого приложения выступают сервисы хранения фотографий с открытым API: Flickr и Picassa. Задачей программного посредника, созданного в рамках демонстрационного приложения, было извлечение фотографий из баз фотографий Flickr и Picasa посредством вызовов необходимых программных интерфейсов.

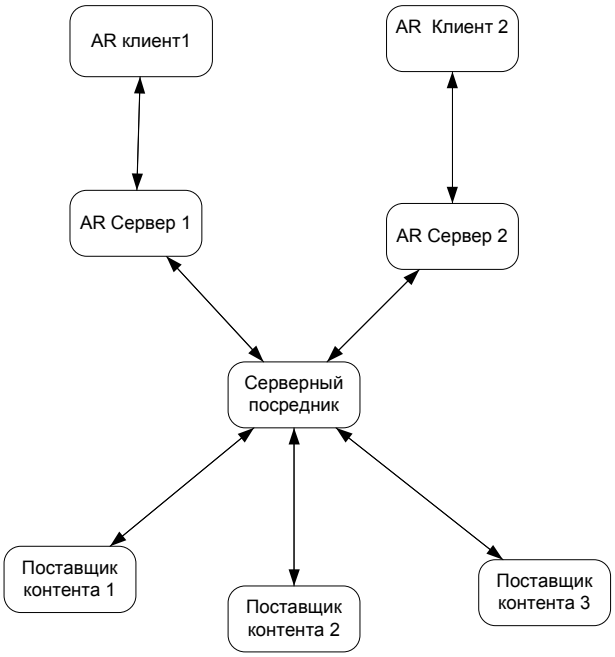

Рисунок 11. Интеграция UGC

Таким образом, программный посредник будет суммировать информацию о точках интереса с двух поставщиков контента. Архитектура программного посредника создана таким образом, что добавить нового поставщика контента (например, oldmos.ru) не составит большого труда.

Структура самого демонстрационного приложения, реализованного в качестве прототипа для Layar была подробно описано в работе [1]. Поэтому ниже остановимся на общей архитектуре системы.

VII. Реализация

Общая схема приложения изображена на рис. 12

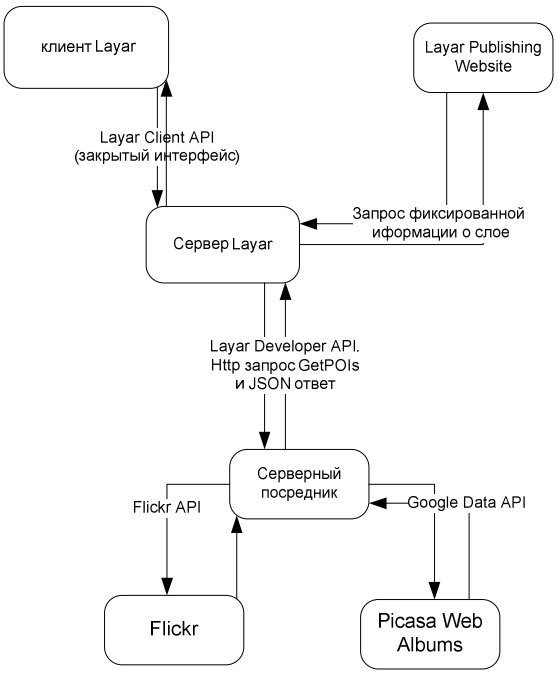

Рисунок 12. Схема приложения

Тестовая реализация была выполнена для браузера Layar. На сервере Layar хранится статическая информация о слое, загруженная туда разработчиком слоя посредством Layar publishing website. Layar publishing website – это веб-интерфейс, позволяющий разработчикам управлять существующиими слоями, публиковать и тестировать новые. При регистрации слоя на этом веб сайте, на сервер Layar загружается вся необходимая фиксированная информация о слое:

- URL обработчика Layar Developer API запросов для предоставления динамической информации о точках интереса
- данные о внешнем виде слоя (набор используемых цветов, возможность загрузки собственных иконок для отображения точек интереса)
- текстовое описание слоя и тэги, по которым этот слой можно будет найти в каталоге
- версия Layar API, которая этим слоем поддерживается
- географические области, для которых доступен этот слой
- задание набора фильтров, которые будут доступны пользователю
- прочие настройки (например, признак того, является ли этот слой бесплатным или нет)

Главная из всех этих настроек – это URL адрес обработчика http запросов на стороне разработчика слоя, который будет обрабатывать веб-запросы от сервера Layar. Для запроса ресурсов от стороннего сервера сервер Layar использует один HTTP GET запрос определенного формата. В этом запросе веб-серверу разработчика передается вся необходимая информация от пользователя – географическое положение (широта, долгота), название запрашиваемого слоя, а также фильтры, установленные на стороне пользователя (в частности максимальная удаленность запрашиваемых точек интереса).

Получив всю необходимую информацию, обработчик http запросов должен предоставить в ответ все необходимые данные. Задача серверного приложение состоит в том, чтобы запросить поставщиков контента (в примере - Flickr и Picasa) о всех фотографиях и метаданных к ним, удовлетворяющие заданному набору географических координат и тэгированных одним из специальных тэгов - «oldmoscow», «oldmos», «старая москва» и т.п.

Серверное приложение, получив запрос о предоставлении точек интереса, формирует API запросы к серверам Flickr и Picasa, запрашивая все фотографии, удовлетворяющие заданному диапазону GPS координат и имеющим соответствующий тэг. Поиск осуществляется исключительно по тегам и гео-координатам. Такая система позволяет всем пользователям Flickr/Picasa учувствовать в наполнении базы архивных фотографий. Это классический пример UGC системы.

Получив от Flickr/Picasa необходимый массив информации (несколько ссылок на одну и ту же фотографию различных размеров, текстовое описание фото), задача серверного приложения – сформировать корректный ответ Layar серверу. В качестве формата ответа используется JSON. JSON ответ от сервера включает в себя всю необходимую информацию о точках интереса – их географические координаты, текстовое описание, удаленность от текущего географического местоположения пользователя и собственно ссылки на фотографии (представлены ссылки на фотографии в разных размерах).

Клиент Layar, получив информацию о массиве точек интереса, сам загружает с Flickr/Picasa необходимые фотографии (размер загружаемой фотографии зависит от удаленности выбранной точки интереса от позиции пользователя). Фотографии рисуются как двумерные изображения поверх картинки с камеры. Все изображения масштабируются в соответствие с расстоянием до точек интереса, так чтобы эти изображения максимально естественно накладывались на реальные объекты. Само масштабирование настраивается разработчиком слоя и передается отдельно для каждой точки интереса при формировании JSON ответа от сервера.

Схематически структура серверного посредника изображена на рисунке 13. Сплошными линиями показан поток команд, а пунктирными – поток данных.

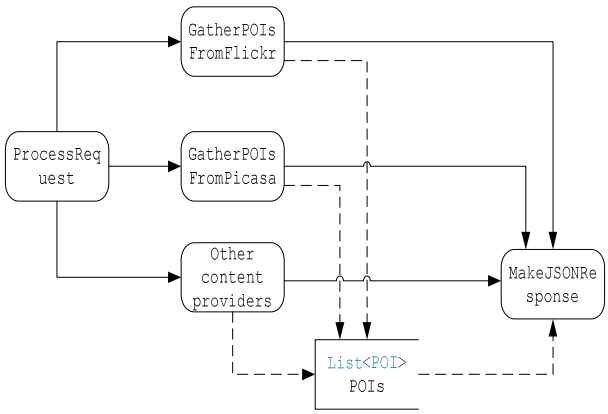

Рисунок 13. Серверный посредник

Первый этап работы посредника – получение HTTP запроса. HTTP запрос обрабатывается в методе *ProcessRequest*, являющегося входной точкой в логику серверного приложения и представляющего собой реализацию соответствующего метода интерфейса IHttpHandler. Одна из задач метода – инициализировать динамический список точек интереса, который будет использоваться при подготовке JSON ответа серверу Layar.

Каждая точка интереса (point of interest) должна быть представлена соответствующим объектом класса POI. Как ясно из названия, класс POI инкапсулирует в себе внутреннюю структуру точки интереса. Эта структура соответствует описанию структуры точки интереса в спецификации Layar Developer API [10]. Доступ к внутренним полям объекта осуществляется посредством вызовов специальных методов, как того требует объектно-ориентированная парадигма. POIs –

представляет из себя динамический список точек интереса, то есть его длина заранее неизвестна, он будет заполняться по мере извлечения новых точек интереса с серверов поставщиков контента.

После завершения инициализации управление передается методам, ответственным извлечение точек интереса с серверов контент провайдера. Пока таких методов два – GatherPOIsFromFlickr и GatherPOIsFromPicasa. Задача этих методов – используя вызовы соответствующего API, запросить сервера поставщиков контента на предмет подходящих точек интереса и, в случае успеха, создать новый объект класса POI для каждой извлеченной точки интереса и добавить этот объект в контейнер POIs. Эти методы абсолютно не зависят друг от друга – они могут выполняться параллельно в разных потоках. Наличие или отсутствие одного из методов никак не влияет на работу другого. При необходимости информацию о включении/исключении определенного поставщика контента из логики серверного посредника можно вынести во внешний конфигурационный файл. Таким образом, можно добавлять и удалять поставщиков контента (из существующего списка) из логики обработки веб-запроса без перекомпиляции серверного приложения.

При необходимости добавить в логику приложения интеграцию с новым поставщиком контента достаточно лишь описать соответствующий метод, который будет на основании пришедших параметров запрашивать массив точек интереса у этого нового поставщика контента и помещать созданные им объекты класса POI в коллекцию точек интереса.

После завершения выполнения всех методов, ответственных за извлечение контента с внешних серверов, управление передается методу MakeJSONResponse, задача которого состоит в том, чтобы сформировать корректный ответ в необходимом формате для Layar. Для этого метод проходит по каждому элементу коллекции POIs и строит на основании этих данных корректный серверный ответ.

В случае если необходимый формат ответа изменится, или сервер начнет требовать представление данных в каком-то еще формате, например, XML, то достаточно будет написать новый метод, который будет строить корректный ответ на основании коллекции точек интереса. На процесс сбора данных для коллекции введение нового формата никак не повлияет.

Важный момент заключается в том, что идею серверного посредника легко расширить и на несколько браузеров дополненной реальности. Все что для этого надо сделать:

- Определить класс POI так, чтобы он содержал максимум возможных атрибутов точек интереса хотя бы для одного из сервисов
- Изменить метод, принимающий HTTP запросы так, чтобы он смог извлекать необходимые параметры из разных форматов запросов, соответствующих вызовам разных программных интерфейсов
- Изменить метод, формирующий серверный ответ так, чтобы собранные данные оборачивались в корректный формат, отвечающий вызову конкретного программного интерфейса

Другой важный момент состоит в том, что методы сбора точек интереса с разных контент провайдеров никак не меняются. Фактически, предлагаемая архитектура программного посредника позволяет легко интегрировать друг с другом несколько различных поставщиков ресурсов и несколько браузеров дополненной реальности, для которых этот контент поставляется.

VIII. Дальнейшее развитие

Было бы интересно увидеть продолжение этих работ в рамках лаборатории ОИТ ВМК МГУ им. М.В. Ломоносова [11]. В качестве весьма интересного направления можно предложить интеграцию дополненной реальности и контекстно-зависимых вычислений. Примеры можно найти, например, в работах [12] и [13]. В качестве технологической основы, развиваемой в лаборатории, можно упомянуть проект SpotEx [14,15], который позволяет связать контент для мобильных пользователей с элементами сетевой инфраструктуры. Так вот этим контентом вполне может быть AR уровень. В качестве областей применения можно было бы указать, например, позиционирование в помещениях [16]. Например, как показано в [17], гео-координаты могут быть заменены информацией о сетевой близости. Это позволяет строить приложения, которые будут работать как традиционные LBS системы, но без доступа к гео-позиционной информации. AR уровень мог бы быть использован для показа контекстной информации в процессе перемещения. Другое возможное направление исследований – это построение специализированного распознавателя QR-кодов, на базе имеющегося прототипа [18], который мог бы быть использован для показа AR контента.

Библиография

[1] Баулин И. Н. Слои для Браузера Дополненной Реальности //International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 7. – С. 16-23.

[2] Azuma, Ronald T. "A survey of augmented reality." Presence 6.4 (1997): 355-385.

[3] Hill A. et al. Kharma: An open kml/html architecture for mobile augmented reality applications //Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on. – IEEE, 2010. – С. 233-234.

[4] Craig A., McGrath R. E., Gutierrez A. Technical Note: Augmented Reality Software Kits for Smart Phones //Institute for Computing in Humanities, Arts, and Social Science (I‐CHASS), University of Illinois, Urbana‐Champaign, May. – 2011.

[5] Vaughan-Nichols S. J. Augmented Reality: No Longer a Novelty? //Computer. – 2009. – Т. 42. – №. 12. – С. 19-22.

[6] Madden L. Professional augmented reality browsers for smartphones: programming for junaio, layar and wikitude. – Wiley. com, 2011.

[7] Chapman R. J., Riddle D. L., Merlo J. L. Techniques for Supporting the Author of Outdoor Mobile Multimodal Augmented Reality //Proceedings of the Human Factors and Ergonomics Society Annual Meeting. – SAGE Publications, 2009. – Т. 53. – №. 27.

[8] Visser A. Survey of XML Languages for Augmented Reality Content //Proc. of AR Standardization Forum, Barcelona. – 2011.
[9] Lechner M. Arml-augmented reality markup language //Ginzkeyplatz. – 2010. – Т. 11. – С. 5020.
[10] Belimpasakis P., You Y., Selonen P. Enabling rapid creation of content for consumption in mobile augmented reality //Next Generation Mobile Applications, Services and Technologies (NGMAST), 2010 Fourth International Conference on. – IEEE, 2010. – С. 1-6.
[11] Намиот, Д., & Сухомлин, В. (2013). О проектах лаборатории ОИТ. International Journal of Open Information Technologies, 1(5), 18-21
[12] Voss G. B. et al. Context-Aware Virtual Laboratory for Teaching Computer Networks: A Proposal in the 3D OpenSim Environment //Proceedings of the 2013 XV Symposium on Virtual and Augmented Reality. – IEEE Computer Society, 2013. – С. 252-255.
[13] Zhu J., Ong S. K., Nee A. Y. C. An authorable context-aware augmented reality system to assist the maintenance technicians //The International Journal of Advanced Manufacturing Technology. – 2013. – С. 1-16.
[14] Namiot, D., & Sneps-Sneppe, M. Local messages for smartphones. Future Internet Communications (CFIC), 2013 Conference on, pp.1-6, IEEE, DOI: 10.1109/CFIC.2013.6566322
[15] Namiot D., Sneps-Sneppe M. Proximity as a service //Future Internet Communications (BCFIC), 2012 2nd Baltic Congress on. – IEEE, 2012. – С. 199-205.
[16] Абдрахманова А., Намиот Д. Использование двумерных штрихкодов для создания системы позиционирования и навигации в помещении // Прикладная информатика. — 2013. — Т. 43, № 1. — С. 31–39.
[17] Dmitry Namiot and Manfred Sneps-Sneppe. “Geofence and Network Proximity”. Internet of Things, Smart Spaces, and Next Generation Networking, Lecture Notes in Computer Science Volume 8121, 2013, pp. 117-127, DOI: 10.1007/978-3-642-40316-3_11
[18] Namiot D., Sneps-Sneppe M., and Skokov O. Context-Aware QR-Codes // World Applied Sciences Journal. — 2013. — Vol. 25, no. 4. — P. 554–560. DOI: 10.5829/idosi.wasj.2013.25.04.11360

User Generated Content in Augmented Reality Browser.

Baulin I.N., Namiot D.E.

Abstract — This article presents the results of the master's thesis, completed by one of the authors in the Laboratory of Open Information Technologies during the master degree study at the Faculty of Computational Mathematics and Cybernetics Lomonosov Moscow State University. The paper advances the idea of user-generated content for presentations in augmented reality systems. The paper presents a new middleware system designed to create augmented reality mashups with user generated content.

Keywords — augmented reality, content, mashup.