**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**“Санкт-Петербургский государственный электротехнический университет “ЛЭТИ”**

**им. В.И. Ульянова (Ленина)” (СПбГЭТУ)**

|  |
| --- |
|  |

**Направление: 010500.62, «Прикладная математика и иноформатика»**

**Кафедра: МОЭВМ**

*К защите допустить:*

д.т.н., профессор Лисс Александр Рудольфович

ВЫПУСКНАЯ

КВАЛИФИКАЦИОННАЯ РАБОТА

БАКАЛАВРА

***Визуализация в Андроид***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Студент* |  | *Грибова Мария Андреевна* |  | */ /* |
| *Руководитель* |  | *Кринкин Кирилл Владимирович* |  | */ /* |
| *Зав. кафедрой* |  | *Ивановский Сергей Алексеевич* |  | */ /* |

Санкт-Петербург

2013 г.

**Аннотация.**

Бакалаврская работа посвящена разработке визуализатора тегов расширенной реальности для Android.

В ходе работы были изучены возможности гео-контекстных сервисов и принципы построения расширенной реальности, методы и инструменты визуализации в Android.

В результате работы было разработано приложение-визуализатор тегов для Android для LBS платформы Geo2tag [1], Программа была реализована на языке Java с использованием бесплатной библиотеки c открытым исходным кодом Mixare [2], для отрисовки объектов были использованы бесплатная библиотека OpenGL ES [3] и стандартный пакет android.graphics [4].

**Annotation.**

Bachelor work is devoted to the development of the application-visualizer of augmented reality geotags for Android.

During work process was investigated the possibility of geo-contextual services, principles of augmented reality techniques and visualization tools in Android.

As a result of work was developed application-visualizer of geotags for LBS platform Geo2tag [1]. The program was implemented in Java, using free open-source library Mixare [2], for drawing graphic objects was used free library OpenGL ES [3] and standard package android.graphics [4].

**Содержание.**

1.Введение………………………………………………………………………....7

2.Общие сведения о геолокационных системах и дополненной реальности…7

3.Цели и задачи данной работы………………………………………………….8

4.Дополненная реальность……………………………………………..…………9

4.1. Дополненная реальность и военная техника…...……………………10

4.2. Дополненная реальность и мобильные технологии…………………10

4.3. Дополненная реальность и игры……………………………………...14

5.Геолокационные системы……………………………………………………..15

5.1. Что такое LBS-сервисы………………………………………………..15

5.2. Основные методы определения местонахождения………………….16

6.Аналитический обзор………………………………………………………….18

6.1.Geo2tag………………………………………………………………….18

6.1.1.Основные возможности………………………………….……..18

6.1.2.Модель данных………………………………………………….19

6.2. LBS-сервисы в России………………………………………………...19

6.3. Известные LBS-сервисы………………………………………………20

6.4. Сравнение Geo2tag и его крупнейших аналогов…………………….21

7.Indoor-сервисы…………………………………………………………………24

7.1. Indor-сервисы, общие сведения……………………………………….24

7.2. Направления……………………………………………………………24

8.Обзор существующих приложений…………………………………………..27

8.1. Layar Reality Browser…………………………………………………..27

8.2.WikiTude Drive………………………………………………………….27

8.3.3D Compass+ (AR Compass)…………………………………………...29

8.4.Augement………………………………………………………………..30

9.Отличия разрабатываемого приложения от существующих……………….30

10.Особенности разработки AR-приложений под Android…………………...31

11.Существующие библиотеки и фреймворки для создания AR-приложений под Android……………………………………………………………………….33

12.Результаты работы, выводы…………………………………………………34

12.1.Описание работы приложения, интерфейс пользователя………….34

12.2.Архитектура…………………………………………………………...39

12.2.1.Предметная область…………………………………………...39

12.2.2.Прецеденты разрабатываемой системы……………………...39

12.2.3.Диаграмма прецедентов……………………………………….42

12.2.4. Схема взаимодействия Activity приложения………………..42

12.2.5.Описание взаимодействия с сервером……………………….43

12.2.6.Основные пакеты проекта, диаграмма классов……………...44

13. Направления развития……………………………………………………….45

14. Результаты работы…………………………………………………………...46

15.Используемая литература и интернет-ресурсы…………………………….46

16.Ссылки на ресурсы…………………………………………………………...47

**1.Введение.**

В настоящее время, в эпоху бурного технологического развития, большое распространение получили различные мобильные устройства, обладающие большой мощностью, в некоторых случаях даже превосходящей мощность многих компьютеров. Современный человек уже не представляет свою жизнь без смартфона или планшета. Просмотреть последние новости, узнать прогноз погоды, проверить почту и многое другое – все это возможно с помощью мобильных устройств, причем практически в любой точке земного шара, где бы ни находился пользователь. Не удивительно, что в последнее время активно развиваются приложения для данных устройств, особенно в области навигации, геолокации и дополненной реальности.

**2.Общие сведения о геолокационных системах и дополненной реальности.**

**Location-based service**, **LBS** (*Служба, основанная на местоположении*) — тип информационных и развлекательных услуг, основанных на определении текущего местоположения [мобильного](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD) устройства пользователя. Визуализационные возможности современных мобильных телефонов (в частности, [смартфонов](http://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD)) позволяют отображать на экране [электронные карты](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BA%D0%B0%D1%80%D1%82%D0%B0) достаточно высокого качества, что позволяет использовать [LBS](http://ru.wikipedia.org/wiki/LBS) для решения различных бизнес-задач, навигации и развлечений.

**Дополненная реальность** (*augmented reality*, AR), — термин, относящийся ко всем проектам, направленным на дополнение реальности любыми виртуальными элементами. Дополненная реальность — составляющая часть смешанной реальности (*mixed reality*), в которую также входит «дополненная виртуальность» (когда реальные объекты интегрируются в виртуальную среду). Самые распространённые примеры дополненной реальности — параллельная лицевой цветная линия, показывающая нахождение ближайшего полевого игрока к воротам при телетрансляции футбольных матчей, стрелки с указанием расстояния от места штрафного удара до ворот, нарисованная траектория шайбы во время хоккея и т. п.

Наибольшее распространиение геолокация и дополненная реальность получает в мобильных технологиях. Все большее количество людей приобретает современные смартфоны с возможностями использовать различные сервисы геолокации и дополненной реальности. Существует множество подобных приложений, однако многие пользователи отмечают, что интерфейс подобных приложений пока далек от совершенства: указатели занимают слишком много экранного пространства и накладываются друг на друга. Таким образом, данная область является очень перспективной на данный момент, областей развития множество: навигационные приложения (визуализация маршрута), туристические (отображение дополнительной информации об объектах), различные игры с дополненной реальностью и многое другое.

**3.Цели и задачи данной работы.**

Цель – разработка визуализатора тегов расширенной реальности для Android.

Для достижения данной цели необходимо решить следующие вспомогательные задачи:

* Изучить возможности гео-контекстных сервисов и принципов построения расширенной реальности
* Изучить методы и инструменты визуализации в Android
* Разработать мобильное приложение для визуализации гео-тегов

**4.Дополненная реальность.**

**Дополненная реальность** (*augmented reality*, AR), — термин, относящийся ко всем проектам, направленным на дополнение реальности любыми виртуальными элементами. Дополненная реальность — составляющая часть смешанной реальности (*mixed reality*), в которую также входит «дополненная виртуальность» (когда реальные объекты интегрируются в виртуальную среду).

Сам термин предположительно был предложен работавшим на корпорацию [Boeing](http://ru.wikipedia.org/wiki/Boeing) исследователем Томом Коделом в [1990 году](http://ru.wikipedia.org/wiki/1990_%D0%B3%D0%BE%D0%B4). Существует несколько определений дополненной реальности: исследователь Рональд Азума в [1997](http://ru.wikipedia.org/wiki/1997) году определил дополненную реальность как систему, которая:

1. Совмещает виртуальное и реальное
2. Взаимодействует в реальном времени
3. Работает в [3D](http://ru.wikipedia.org/wiki/3D)

В [1994](http://ru.wikipedia.org/wiki/1994) году Пол Милгром и Фумио Кисино описали Континуум Виртуальность-Реальность — пространство между реальностью и виртуальностью, между которыми расположены дополненная реальность (ближе к реальности) и дополненная виртуальность (ближе к виртуальности). Ещё одно определение: Дополненная реальность — добавление к поступающим из реального мира ощущениям мнимых объектов, обычно вспомогательно-информативного свойства.

Иногда используют в качестве синонимов названия «[расширенная реальность](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C)», «улучшенная реальность», «обогащённая реальность».

**4.1. Дополненная реальность и военная техника.**

В современных боевых самолетах и вертолетах часто используется [индикация на лобовом стекле или на шлеме пилота](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80_%D0%BD%D0%B0_%D0%BB%D0%BE%D0%B1%D0%BE%D0%B2%D0%BE%D0%BC_%D1%81%D1%82%D0%B5%D0%BA%D0%BB%D0%B5). Она позволяет пилоту получать наиболее важную информацию прямо на фоне наблюдаемой им обстановки, не отвлекаясь на основную приборную панель. Это позволяет сэкономить драгоценные секунды, например, во время маневренного воздушного боя. Также многие подобные системы позволяют производить целеуказание путем поворота головы или движений глазных яблок.

[Коллиматорный прицел](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B8%D0%BC%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%B8%D1%86%D0%B5%D0%BB) тоже можно отнести к простейшим механизмам дополненной реальности (Коллиматорные прицельные системы — это системы, использующие коллиматор для построения изображения прицельной метки, спроецированного в бесконечность. Коллима́тор — устройство для получения параллельных пучков лучей света или частиц.).

**4.2. Дополненная реальность и мобильные технологии.**

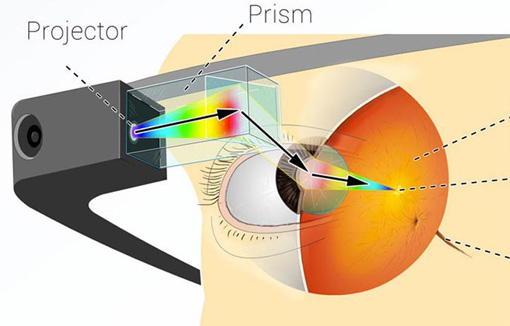
Существует множество программных решений для мобильных телефонов, которые позволяют при помощи дополненной реальности получить необходимые сведения об окружении. Это браузеры дополненной реальности и специализированные программы для отдельных сервисов, компаний или даже единственных моделей. Само распространение дополненной реальности и нарастающая известность технологии среди потребителей связано с тем, что вычислительная мощность и набор датчиков в аппаратных платформах для смартфонов и планшетов позволяют производить наложение любых цифровых данных на получаемое в реальном времени со встроенных в устройства камер изображение. Новой ступенью во взаимодействии мобильных устройств и дополненной реальности должно стать сотрудничество между разработчиком AR-браузера junaio [5] компанией Metaio [6] и производителем аппаратных платформ для смартфонов ST-Ericsson, направленное на встраивание в платформы разработанного Metaio чипа для дискретной обработки дополненной реальности. В самой Metaio ожидаемые результаты этого сотрудничества сравнивают с появлением дискретных видеоускорителей в персональных компьютерах, выведшее на новый уровень индустрию компьютерных игр.

В 2010 году компания AlterGeo [7] выпустила первое в истории России приложение с дополненной реальностью для iPhone. Будучи частью одноименного геосоциального сервиса, продукт позволял смотреть через камеру смартфона, в какой стороне и на каком расстоянии от пользователя расположены городские достопримечательности и заведения, а также где в данный момент находятся его друзья.

Часть решений в этой области воплощается в виде нательных компьютеров для постоянного контакта со средой дополненной реальности. В настоящее время Google работает над гарнитурой Project Glass (Google Glass) [8], а Vuzix — над Smart Glasses M100 [9].

Наверное, наибольший интерес на данный момент представляет Google Glass гарнитура (или нательный компьютер, что несколько ближе к функциональному набору устройства) для смартфонов на базе Android и iOS, разрабатываемая компанией [Google](http://ru.wikipedia.org/wiki/Google). В устройстве используется прозрачный дисплей, который крепится на голову (англ. HMD — head-mounted display) и находится чуть выше правого глаза, и камера, способная записывать видео высокого качества.

Тестирование продукта началось в апреле [2012 года](http://ru.wikipedia.org/wiki/2012_%D0%B3%D0%BE%D0%B4), а известная газета [The New York Times](http://ru.wikipedia.org/wiki/The_New_York_Times) сообщала о новинке ещё в конце февраля 2012 года. Прототипы гарнитуры модели Explorer Edition стоимостью $1500 были переданы разработчикам программного обеспечения на мероприятии Glass Foundry в феврале 2013 года, а первые потребительские экземпляры гарнитуры должны появиться на рынке в конце 2013 года по значительно более низкой цене.

Взаимодействие Glass с пользователем осуществляется через голосовые команды (базовой является команда «Ok, Glass», после которой должна идти просьба выполнить какую-либо функцию, кроме того через гарнитуру можно надиктовывать тексты), жесты, распознаваемые тачпадом, который расположен на дужке за дисплеем, и систему передачи звука с использованием костной проводимости. Интерфейс устройства был продемонстрирован Google в опубликованном в феврале 2013 года видео, а уже в марте компания использовала выставку SXSW Interactive [10] для показа первых приложений для Glass от сторонних разработчиков. Производитель работает над образцом гарнитуры для людей, носящих корректирующие зрение очки, который должен обладать «модульной» конструкцией с возможностью подбора необходимых линз и оправ.

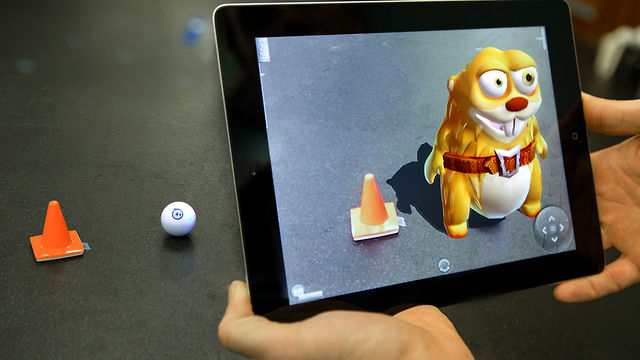
Концепция Google Glass в конечном счёте должна реализовывать одновременно три отдельные функции, сведя их воедино: [дополненную реальность](http://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C), [мобильную связь + интернет](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D0%BA%D0%BE%D0%BC%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8),  [лайфлогинг](http://en.wikipedia.org/wiki/Lifelog).

Однако подобные аксессуары с камерами, способные вести незаметную видео- и фотосъёмку, одним из которых и является Glass, вызывают опасения по поводу нарушений прав людей на личную жизнь. Во многих странах они могут попасть под действие законодательных запретов: к примеру, статья 376 [УК Республики Беларусь](http://ru.wikipedia.org/wiki/%D0%A3%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4%D0%B5%D0%BA%D1%81_%D0%A0%D0%B5%D1%81%D0%BF%D1%83%D0%B1%D0%BB%D0%B8%D0%BA%D0%B8_%D0%91%D0%B5%D0%BB%D0%B0%D1%80%D1%83%D1%81%D1%8C) «за незаконное изготовление, приобретение либо сбыт средств для негласного получения информации». В феврале 2013 пользователь [Google+](http://ru.wikipedia.org/wiki/Google%2B) [11]  [Юрий Чернявский (Yuriy Chernyavskyy)](https://plus.google.com/101984526270492961858) первый в своем посте сообщил о возможных сомнениях относительно того, что Google Glass можно будет легально использовать, покупать и продавать, согласно действующему законодательству в России и на Украине. Так, впервые было указано на то, что Google Glass можно считать «шпионским» устройством, поскольку с его помощью можно вести фото- и видеосъемку незаметно от находящихся поблизости людей. Позднее Служба безопасности Украины заявила, что по предварительному заключению, основанному на информации из Интернета, ничего шпионского в Glass не видит.

Возможность купить данное устройство в России может быть ограничена, в случае если оно будет признано «специальным техническим средством для негласного получения информации» с ограниченным оборотом (ч.3 ст.30 — ст.138.1 УК РФ).  В марте 2013 года в городе [Сиэттл](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%8D%D1%82%D1%82%D0%BB), США, владелец бара «The 5 Point» Дэйв Мэйнерт стал первым бизнесменом, который запретил ношение Glass на частной территории.

**4.3. Дополненная реальность и игры.**

Существуют компьютерные игры, производящие обработку видеосигнала с камеры и накладывающие на изображение окружающего мира дополнительные элементы. Например, в 2004 году была выпущена игра для мобильных телефонов с названием [Mosquitos](http://ru.wikipedia.org/w/index.php?title=Mosquitos_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0)&action=edit&redlink=1), отображающая на экране телефона изображение с расположенной позади него камеры, с наложенными на это изображение прицелом и огромными комарами, от которых «отстреливался» игрок.

В современном мире игры дополненной реальности получили широкое распространение на смартфонах и планшетах, а также игровых консолях. В качестве примеров можно привести массу проектов, наиболее известными из которых являются Box! Open Me [12] для PlayStation Vita, SpecTrek [13] для Android, AppTag [14] для Android и iOS, требующая оригинальных аксессуаров и Книга Заклинаний для PlayStation 3.

Большой интерес представляет игра Sharky the beaver [15]. Sharky – это роботизированный мячик, который выступает в качестве вращающегося маркера. Пользователь контролирует мячик через Bluetooth-устройство. В то время как мячик катается по полу, на экране пользователь видит вместо обычного мячика зверька Sharky, который ходит по комнате и поедает пирожные. Через 2 потока данных, создатели балансируют на грани виртуального и реального миров одновременно. Sharky доступен для разработчиков, как software development kit. Благодаря ему мы наблюдаем появление возможности создания целой библиотеки образов, которые люди контролируют при помощи маленького робота. Так, например, мебельные компании могут использовать данную технологию и дать своим пользователям шанс примерять в той или иной части новой комнаты виртуальный, шкаф, стол или кровать. Робот-мячик – Sphero [16], небольшой сферический робот из блестящего полупрозрачного поликарбоната.

Sphero – действительно инновационный гаджет. Это не только роботизированный шарик, но так же это самостоятельная игровая система управляемая смартфоном, и в то же время – контроллер для игры на смартфоне. Список поддерживаемых девайсов традиционный – iOS и Android, при этом количество игр в которых может использоваться Sphero – действительно впечатляет (*The Rolling Dead*— игра с возможностями расширенной реальности, где Sphero используется для стрельбы огненными шарами уничтожающими зомби, *Nyan Cat Space Party*— Sphero может использоваться в качестве контроллера, *Zombie Rollers*— Sphero может использоваться в качестве контроллера, *Blot*— то же самое, *Pass the Sphero* — мультиплеерная игра для вечеринки).

**5.Геолокационные системы.**

**5.1. Что такое LBS-сервисы.**

**Location-based service**, **LBS** ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Служба, основанная на местоположении*) — тип информационных и развлекательных услуг, основанных на определении текущего местоположения [мобильного устройства](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD) пользователя. Визуализационные возможности современных мобильных телефонов позволяют отображать на экране [электронные карты](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BA%D0%B0%D1%80%D1%82%D0%B0) достаточно высокого качества, что позволяет использовать [LBS](http://ru.wikipedia.org/wiki/LBS) для решения различных бизнес-задач, навигации и развлечений.

LBS не обязательно используют технологию [ГЛОНАСС](http://ru.wikipedia.org/wiki/%D0%93%D0%9B%D0%9E%D0%9D%D0%90%D0%A1%D0%A1) [17], [GPS](http://ru.wikipedia.org/wiki/GPS) [18] или пр. [спутниковые системы](http://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D1%83%D1%82%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BD%D0%B0%D0%B2%D0%B8%D0%B3%D0%B0%D1%86%D0%B8%D0%B8) для определения местоположения. Местоположение мобильного объекта, например, может быть определено с использованием заранее известной информации о расположении базовых станций сетей сотовой связи [GSM](http://ru.wikipedia.org/wiki/GSM) [19], [UMTS](http://ru.wikipedia.org/wiki/UMTS) [20] и др., а также посредством данных о расположении точек доступа [Wi-Fi](http://ru.wikipedia.org/wiki/Wi-Fi). В различных [стандартах радиосвязи](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%80%D0%B0%D0%B4%D0%B8%D0%BE%D1%81%D0%B2%D1%8F%D0%B7%D1%8C) точность может колебаться от многих десятков [километров](http://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BB%D0%BE%D0%BC%D0%B5%D1%82%D1%80) до 50 метров в [WCDMA](http://ru.wikipedia.org/wiki/UMTS) [21] и [LTE](http://ru.wikipedia.org/wiki/3GPP_Long_Term_Evolution) [22].

При этом во всех случаях используется одна и та же методика расчёта положения — [обратная геодезическая засечка](http://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%B3%D0%B5%D0%BE%D0%B4%D0%B5%D0%B7%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D1%81%D0%B5%D1%87%D0%BA%D0%B0&action=edit&redlink=1).

Под «поиском» в сотовой LBS подразумевается не определение местоположения собственно мобильного телефона, а привязка [его местоположения](http://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BA%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82%D1%8B) к ориентирам, нанесённым на электронную карту LBS-системы [оператором сотовой сети](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D1%81%D0%BE%D1%82%D0%BE%D0%B2%D0%BE%D0%B9_%D1%81%D0%B2%D1%8F%D0%B7%D0%B8) или поставщиком услуги и предоставляемая в простой и доступной для человека [форме](http://ru.wikipedia.org/wiki/%D0%95%D1%81%D1%82%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA).

**5.2. Основные методы определения местонахождения.**

* **Cell of Origin** — простейший метод, позволяющий вычислять местонахождение мобильного телефона по известному CellID (Cell identifier,  *идентификатор ячейки/соты*). Не требует модификации сетевого оборудования и клиентского терминала, достаточно установки программного комплекса и MLC (Mobile Location Center,  *центр мобильной локализации*). Координаты вычисляются на основе примерного знания расположения и радиуса ячеек сотовой сети, в которых мобильные телефоны обслуживаются конкретной базовой станцией. Точность определения местоположения зависит от густоты сети базовых станций, текущих местных радиоусловий и конфигурации сот. В центре крупного города точность обычно составляет несколько сотен метров, а на окраинах и в небольших городах — около километра. В сельской местности точность снижается до 35 км, а в режиме Extended Cell (ECell) до 120 км. За пределами зоны покрытия сотовой сети данный метод не работает и LBS недоступны.
* **TOA** (Time of Arrival,  *Оценка времени прибытия сигнала*) — основан на измерении и сравнении интервалов времени прохождения сигнала от мобильного телефона абонента до нескольких базовых станций. Требует модернизации оборудования сотовой сети. Точность может достигать 125 м. Базовые станции, принимающие сигнал мобильного телефона, должны быть оснащены LMU (Location Measurement Unit,  *блок определения местоположения*). По разности времени поступления сигнала управляющим компьютером сети сотовой связи с помощью алгоритма трилатерации рассчитывается местоположение передатчика. Полученные координаты передаются соответствующему сетевому приложению (серверу услуги) или клиенту.
* **OTD** (Observed Time Difference,  *Наблюдаемая разность времени прибытия сигнала*) — основан на измерении и сравнении интервалов времени прохождения сигналов от нескольких базовых станций до мобильного телефона абонента. Требует модернизации сетевого оборудования, однако такая модернизация значительно дешевле TOA. Управляющий контроллер мобильного телефона измеряет время прохождения сигнала от нескольких базовых станций, одна из которых оснащена блоком LMU. Для получения информации о своем местоположении абонент совершает звонок, при котором его телефон до установки речевого соединения посылает специальное сигнальное сообщение, MLC производит необходимые вычисления для расчета местоположения, после чего пакет данных с координатами местонахождения абонента пересылается на сотовый телефон.
* **A-GPS** (Assisted Global Positioning System,  *Вспомогательная Глобальная система позиционирования*) — основан на встраивании в мобильные телефоны модуля GPS и переносе части вычислительных функции на Mobile Location Center для снижения энергопотребления и ускорения определения местоположения.

**6.Аналитический обзор.**

**6.1.Geo2tag .**

**Geo2tag LBS Platform [23]** является открытой программной платформой для построения сервисов, основанных на местоположении. Платформа предоставляет все необходимые программные интерфейсы, структуры данных, запросы и фильтры, которые требуются для создания базовой функциональности сервисов, использующих местоположение. Любой разработчик может как скачать исходный код платформы, внести необходимые изменения и развернуть на своем хостинге, так и воспользоваться [готовым сервером](http://tracks.geo2tag.org/) для запуска своих сервисов. Последние стабильные версии платформы доступны на [странице закачек](http://geo2tag.org/index.php/Downloads); информацию о составе будущих выпусков можно найти на [странице с планами](http://geo2tag.org/index.php/Roadmap). Тем, кто хочет участвовать в процесе разработки стоит заглянуть в [трекер](http://jira.geo2tag.org:8080/secure/Dashboard.jspa) или написать письмо geo2tag-development (at) osll.spb.ru.

### 6.1.1.Основные возможности.

* предоставление хранилища для геотегов
* выполнение 2D/3D пространственно-временных запросов
* классификация тегов (по пользователям, по тематике,...)
* RESTfull API для доступа к данным

### 6.1.2.Модель данных.

Одной из главных задач **Geo2tag LBS Platform** является эффективный доступ к данным георазметки и выполнение запросов на них, поэтому используемая модель данных очень простая. Основными объектами (необходимыми и достаточными для построения практически любого сервиса) являются геотеги и каналы.

* **геотег** (или геометка) представляет из себя кортеж данных следующего вида {время, географические координаты, данные}. В качестве данных могут выступать либо URL/URI, либо произвольные текстовые данные. Как правило поле данных используется для записи ссылки на внешний сервис или ресурс.
* **канал** – атрибут геотега, определяющий основной классификационный признак. Каждому геотегу должен быть сопоставлен один или более каналов. Простейший пример использования – персональный канал пользователя в котором собраны его персональные геометки.

**6.2. LBS-сервисы в России.**

С [2002 года](http://ru.wikipedia.org/wiki/2002_%D0%B3%D0%BE%D0%B4) в [России](http://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D1%8F) операторы сотовых сетей применяют LBS в коммерческих целях. Основными направлениями являются:

* определение собственного местоположения мобильной станции (мобильного телефона);
* определение местоположения удалённой мобильной станции (ассоциируется с определением местоположения другого абонента).

Маркетинговые решения [VAS](http://ru.wikipedia.org/wiki/VAS) предлагают следующие виды услуг:

* услуги навигации и слежения;
* [ГИС](http://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) — прогнозы погоды в зоне местонахождения, поиск ближайших объектов инфраструктуры;
* службы общения и развлечений, например, многопользовательские игры, учитывающие местоположение игроков, или службы знакомств.

**6.3. Известные LBS-сервисы.**

* i-area — первый коммерческий геолокационный сервис, введённый японским оператором NTT DoCoMo Inс. в июле 2001 года.
* Google Latitude — геолокационный сервис от Google.
* Fire Eagle — геолокационный сервис от Yahoo.
* AlterGeo — самая крупная и известная геосоциальная сеть в России.
* BluePont — популярный американский геолокационный сервис.
* Foursquare — популярная американская геоконтекстная социальная сеть.
* Gowalla — популярная американская геоконтекстная социальная сеть.
* Услуга «Навигатор» ОАО «МегаФон»
* Услуга «Радар» ОАО «МегаФон»
* Портал «МТС-Поиск» ОАО «МТС»
* «Билайн Координаты» ОАО «ВымпелКом»
* «Локатор» от ОАО «Вымплеком»
* AlterGeo (экс-Wi2Geo) — самая крупная и известная геосоциальная сеть.
* ePythia (экс-Geoplan) — геопланирование задач.
* Advizzer — единый каталог мест досуга и сервиса, с подробной информацией о каждом заведении, с функцией геопозиционирования и социальной направленностью.
* GSmesh — построение и сохранение треков перемещения владельца сотового телефона.
* Встречер — мобильная социальная сеть с возможностью делиться информацией о своем местоположении, назначать встречи.
* CityRangers — городская мобильная программа лояльности и контроля качества услуг, предоставляемых заведениями и службами города.
* mobile-monitor.ru — помогает найти местоположение мобильного телефона и его владельца
* Пососедству.ру — помогает найти интересные места и события рядом.
* Pinpoint — персональный плейс-менеджер
* Geo2tag — Открытая платформа для создания LBS-сервисов
* Lokata — интернет-сервис, позволяющий искать товары в каталогах и брошюрах местных ритейлеров, находить ближайшие магазины и узнавать их часы работы.
* StreetPager — геолокационный сервис для обмена сообщениями между людьми, которые находятся в одном месте.

**6.4. Сравнение Geo2tag и его крупнейших аналогов.**

Идея создания программной платформы для разработки сервисов использующих геоинформацию не нова. Однако, **Geo2tag LBS Platform**, фактически является превой полностью открытой платформой такого рода. Более подробное сравнение можно найти на [странице аналогов](http://geo2tag.org/index.php/%D0%90%D0%BD%D0%B0%D0%BB%D0%BE%D0%B3%D0%B8_Geo2tag). Фокус платформы Geo2tag в обеспечении максимальной [производительности](http://geo2tag.org/index.php/Performance) при обработке запросов и [масштабируемости](http://geo2tag.org/index.php/Performance#.D0.93.D0.BE.D1.80.D0.B8.D0.B7.D0.BE.D0.BD.D1.82.D0.B0.D0.BB.D1.8C.D0.BD.D0.BE.D0.B5_.D0.BC.D0.B0.D1.81.D1.88.D1.82.D0.B0.D0.B1.D0.B8.D1.80.D0.BE.D0.B2.D0.B0.D0.BD.D0.B8.D0.B5_LBS-.D0.BF.D0.BB.D0.B0.D1.82.D1.84.D0.BE.D1.80.D0.BC.D1.8B).

Сравнительная таблица Geo2Tag и некоторых крупнейших геолокационных сервисов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Geo2Tag | Forsquare | AlterGeo | Google Latitude |
| Открытая платформа | + | - | - | - |
| Дата основания | 2011 | 2009 | 2010 | 2000 |
| Хранение тегов | + | + | + | + |
| Социальная сеть | - | + | + | + |
| Российский сервис | + | - | + | - |

**Forsquare [24]** –[социальная сеть](http://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C_%28%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82%29) с функцией [геопозиционирования](http://ru.wikipedia.org/wiki/Location-based_service), предназначенная в основном для работы с мобильных устройств. Данный сервис доступен пользователям не только с устройствами, которые оборудованы [GPS](http://ru.wikipedia.org/wiki/GPS)-навигацией, например пользователям [смартфонов](http://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82%D1%84%D0%BE%D0%BD), но и просто для работы с любым сотовым телефоном. Если мобильный телефон не оборудован GPS-навигацией, то местоположение определяется с помощью сервиса [LBS](http://ru.wikipedia.org/wiki/Location-based_service). Пользователи отмечаются («check-in») в различных заведениях с помощью мобильной версии веб-сайта, [SMS](http://ru.wikipedia.org/wiki/SMS)-сообщения или же специального приложения, разработанного под определённую ОС мобильного устройства. Каждая такая отметка позволяет пользователю зарабатывать foursquare-баллы, а в некоторых случаях и «значки».

Авторами сервиса, запущенного в 2009 году, являются Деннис Кроули и Навин Сельвадурай. Первый до этого уже имел опыт работы с похожей системой Dodgeball, в рамках его дипломного проекта в Нью-Йоркском университете. Впоследствии компания [Google](http://ru.wikipedia.org/wiki/Google) приобрела Dodgeball [27] в 2005, а спустя четыре года, в 2009, сервис был заменён приложением [Google Latitude](http://ru.wikipedia.org/w/index.php?title=Google_Latitude&action=edit&redlink=1). Пользователи могли использовать Dodgeball только с помощью технологии [SMS](http://ru.wikipedia.org/wiki/SMS)-сообщений.

В марте 2011 года, компания Foursquare Labs, Inc объявила о регистрации семимиллионного пользователя, а по состоянию на 21 апреля 2011 число зарегистрированных пользователей составляло более 8 миллионов с ежедневным темпом прироста около 35 000 пользователей.

**AlterGeo [25]** (до 2010 года — Wi2Geo) — многофункциональный [кросс-платформенный](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81-%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [геосоциальный сервис](http://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B5%D1%82%D0%B8), который позволяет [пользователям](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C), в зависимости от их текущего местоположения, находить поблизости и выбирать места для отдыха, делиться с [друзьями](http://ru.wikipedia.org/wiki/%D0%94%D1%80%D1%83%D0%B3) рекомендациями и [фотографиями](http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F), выигрывать виртуальные и реальные награды, а также получать [скидки](http://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%B8%D0%B4%D0%BA%D0%B0) и бонусы в городских заведениях.

В основном, AlterGeo предназначен для использования с мобильных устройств и доступен через бесплатные приложения для [iPhone](http://ru.wikipedia.org/wiki/IPhone), [Android](http://ru.wikipedia.org/wiki/Android), [Windows Phone 7](http://ru.wikipedia.org/wiki/Windows_Phone_7), а также на [вебсайте](http://altergeo.ru) его [мобильной версии](http://m.altergeo.ru).

По состоянию на весну 2012 года количество [пользователей](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C) AlterGeo составляло более 900 тысяч человек, преимущественно из [России](http://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D1%8F) и [Украины](http://ru.wikipedia.org/wiki/%D0%A3%D0%BA%D1%80%D0%B0%D0%B8%D0%BD%D0%B0). Его называют крупнейшим геосоциальным сервисом [Рунета](http://ru.wikipedia.org/wiki/%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%BE%D1%8F%D0%B7%D1%8B%D1%87%D0%BD%D1%8B%D0%B9_%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82).

**Google Latitude [26]** является геолокационным приложением, разработанным Google в качестве преемника своего SMS-сервиса Dodgeball [27]. Приложение позволяет пользователю мобильного телефона разрешить, чтобы определенные люди могли видеть его текущее местоположение. Через их собственную учетную запись Google местоположение пользователя отображается на карте Google Maps [28]. Пользователь может управлять точностью и данными, которые могуть видеть другие пользователи. В целях обеспечения конфиденциальности, видимость местоположения может быть отключена пользователем, или же местоположение можно ввести вручную.

**7.Indoor-сервисы.**

**7.1. Indor-сервисы, общие сведения.**

Indoor = “находящийся в помещении”, “происходящий в помещении”.

Согласно данным Strategy Analytics [29], в настоящий момент пользователи мобильных приложений, находящиеся внутри зданий, создают до 80% мобильного трафика. В помещениях привычный нам GPS не работает, поэтому определить местоположение с его помощью невозможно, но в больших помещениях (например, аэропортах или торговых центрах) это часто бывает очень полезно.

**7.2. Направления.**

#### Приложения для аэропортов и транспортных хабов.

Пассажиры получат возможность определить свое положение внутри незнакомого здания, проложить маршрут к необходимой им стойке регистрации или выходу к самолету. Аэропорт получит возможность производить push-уведомления пассажиров о начале и завершении регистрации на рейс, о начале посадки в самолет. В итоге выигрывает как пассажир — за счет оперативного получения требуемой информации,- так и аэропорт — за счет оптимизации потоков пассажиров внутри здания.

*Приложения для торговых центров.*

Представьте, что у продавца в торговом центре появилась возможность обнаруживать факт приближения потенциального покупателя с помощью установленного у пользователя мобильного приложения. Получив уникальный идентификатор клиента, маркетинговое ПО на основе своей базы данных сможет определить степень лояльности покупателя, проанализировать список предыдущих покупок и сделать покупателю уникальное предложение со скидкой на конкретные группы товаров при помощи push-уведомления в мобильное приложение. Покупатель получает систему ориентирования по торговому центру, уникальные персонифицированные скидки. Продавец получает колоссальный объем информации для маркетингового анализа, строит принципиально новые программы триггер-маркетинга, увеличивает лояльность к своему бренду, повышает продажи.

*Check-In приложения.*

Ряд таких хорошо известных проектов как [Foursquare](https://foursquare.com/), [Gowalla](http://gowalla.com/), [Facebook places](http://www.facebook.com/places) в настоящее время использует принцип Check-In, когда пользователь вручную указывает место своего пребывания. С появлением широко распространенных Indoor Location сервисов отпадет потребность в отвлечении внимания пользователя на дополнительные действия. Все будет происходить автоматически. Знание расположения друга внутри здания облегчит задачу его поиска.

#### Другие приложения.

Приложения для оптического распознавания объектов смогут увеличить производительность за счет уменьшения зоны поиска. В недалеком будущем из календаря мобильного устройства будет возможно почерпнуть информацию о том где, когда и как долго Вы были.

*Техническая реализация.*

Существует ряд технологических платформ и алгоритмов для отслеживания положения объектов в реальном времени. Часть из них применима к системам определения положения внутри зданий.

* WiFi локация
* Bluetooth локация
* GSM локация

Для разработчиков мобильных приложений есть потенциальная возможность использовать три этих технологических платформы в силу того что данные технологии поддерживаются почти всеми современными мобильными устройствами. Также известно о попытках использовать ультразвуковые маяки.

**8.Обзор существующих приложений.**

**8.1. Layar Reality Browser [30].**

Layar Reality Browser – самое известное и популярное приложение, сочетающее в себе геолокацию, и информацию по интересным местам.



**8.2.WikiTude Drive [31].**

WikiTude Drive – навигатор с дополненной реальностью под Андроид, который рисует навигационный путь поверх изображения с камеры.  


## Wikitude World Browser [32].



Если в вашем компьютере и телефоне есть онлайн энциклопедия, такую же вы можете иметь и в дополненной реальности, установив на устройство Android Wikitude World Browser. Приложение даст вам дополнительную информацию о различных местах, которые вы просматриваете на экране телефона.

Если вы ищете, где остановиться, или не знаете, в каком ресторане перекусить, откройте приложение Wikitude и просканируйте окружение. Прямо на своём экране вы увидите содержание дополненной реальности. Это приложение покажет вам места, где можно остановиться, рестораны, где можно утолить голод, специальные предложения гостиниц и многое другое.

С помощью этого приложения вы также можете просматривать контент ваших друзей, отмеченный геотегами. Кроме того, вам будут доступны обновления ваших друзей в Twitter о самом месте.

Если в вашем любимом японском ресторане ещё нет контента для дополненной реальности, не переживайте, приложение позволит вам создать свой собственный и поделиться им со всем миром.

**8.3. 3D Compass+ (AR Compass) [33].**



Смартфоны на базе Android – идеальный инструмент путешественника, ведь в одном устройстве можно найти всё необходимое: карты, навигационные приложения, поисковики местных заведений различного типа, и это далеко не полный список. Компас – также необходимая вещь для туристов и исследователей. Этот инструмент ведёт вас в правильном направлении. Приложение 3D Compass+ (AR Compass) – ваш компас в дополненной реальности.

Приложение главным образом работает как компас в вашем телефоне. С помощью камеры оно показывает изображение того места, куда вы идёте, и компас с дополненной реальностью вверху. Вы можете превратить компас в розу ветров, морской компас, а также искусственный вертикальный компас. Приложение также использует GPS в телефоне и показывает на экране карту, делая ваш телефон прекрасным инструментом для навигации. Адрес вашего месторасположения также отображается в приложении. Прямо со страницы с приложением вы можете изменить яркость экрана.

**8.5.Augement [34].**



Приложение Augment позволяет вам видеть собственные объекты кубической формы (картины, мобильные телефоны, большие коробки, аквариумы, большие экраны, компьютер). Удивите своих гостей картинами и мебелью в дополненной реальности. Можно даже заполнить комнату предметами в дополненной реальности. Чтобы воспользоваться приложением, вам для начала нужно приобрести маркер. Маркер – это лист бумаги с чёрной и белой маркировкой. Он используется для того, чтобы показать объект дополненной реальности на экране мобильного устройства.

Как только вы приобрели маркеры, можете просматривать объекты дополненной реальности. Можно также посетить галерею приложения и использовать объекты, созданные другими пользователями. Кроме того, вы можете создавать свои собственные объекты дополненной реальности кубической формы. Чтобы узнать, как создавать собственные объекты дополненной реальности, посмотрите видео, расположенное ниже.

**9.Отличия разрабатываемого приложения от существующих.**

На данный момент не существует популярных приложений такого рода, которым пользовалось бы большое количество людей. Самое популярное подобное - forsquare, однако там нет возможности просмотра тегов в режиме дополненной реальности, нет возможности создания своих каналов с тегами и подписки на эти каналы и т.д. Хотелось бы сделать подобное приложение и развивать его в будущем.

**10.Особенности разработки AR-приложений под Android.**

Что нужно для создания приложения с AR:

* Камера
* OpenGL ES – библиотека для отрисовки объектов
* Гироскоп/акселерометр – для определения изменений положения мобильного устройства в пространстве
* LocationManager – для определения координат
* Вспомогательная библиотека/фреймворк

Когда в смартфоны пришёл акселерометр и гироскоп, пришла идея соединить их с камерой и OpenGL ES – так родилось множество игр, навигационных помощников, но больше всего бюджета в этом направлении расходуется на маркетинговые и промоушн-приложения. Например, вырезав из журнала бумажный каркас для часов, одев его на руку и посмотрев через камеру телефона, пользователь может “примерить” любую марку часов из тех, что рекламирует журнал.   
Теперь чисто техническая часть, те небольшие проблемы, с которыми программист столкнётся при разработке на платформе Android.

Превью камеры, точнее SurfaceView, можно поместить во View любых размеров. Однако мы сами должны найти подходящий размер превью (список размеров иногда большой, и может различаться на девайсах от разных производителей, да что там, даже от одного производителя). В поисках оптимального разрешения и пропорции придется перебирать все размеры из возможных вариантов и сравнивать их с размером и пропорцией View, куда мы хотим поместить это превью в рантайме. Размер превью не всегда будет соответствовать размеру SurfaceView, так что для соблюдения пропорций картинки и получения подходящего размера превью придется делать свой ViewGroup, размещать там SurfaceView, и делать расчеты, что и как размещать в методе onLayout.   
  
Также в Android стандартные UI элементы можно рисовать поверх превью (SurfaceView), то 3D модели в GLSurfaceView нужно размещать под превью. При этом нужно выполнить ряд действий:  
- для SurfaceView вызвать метод getHolder().setType(SurfaceHolder.SURFACE\_TYPE\_PUSH\_BUFFERS), без этого вызова превью не работает и приложение завершается;

- в GLSurfaceView нужно указать, чтобы она была прозрачной — getHolder().setFormat(PixelFormat.TRANSLUCENT). Без этого вызова на некоторых устройствах может всё работать (работало на HTC Desire S), а может и не работать (не работало на Google GalaxyNexus). Возможно, это из-за различий GPU на разных аппаратах. Так что лучше не пренебрегать данным методом.

Определение местоположения и положения устройства в пространстве определяется с помощью класса LocationManager.

**11.Существующие библиотеки и фреймворки для создания AR-приложений под Android.**

**11.1.Vuforia SDK**

developer.vuforia.com

**11.2.AndAR**

code.google.com/p/andar

**11.3. Mixare**

code.google.com/p/mixare

**11.4.Look!**

www.lookar.net

**11.5.LibreGeoSocial**

www.libregeosocial.org

**11.6.android-augment-reality-framework**

code.google.com/p/android-augment-reality-framework

**11.7.droidar**

code.google.com/p/droidar

**11.8.Metaio sdk**

www.metaio.com/software/mobile-sdk

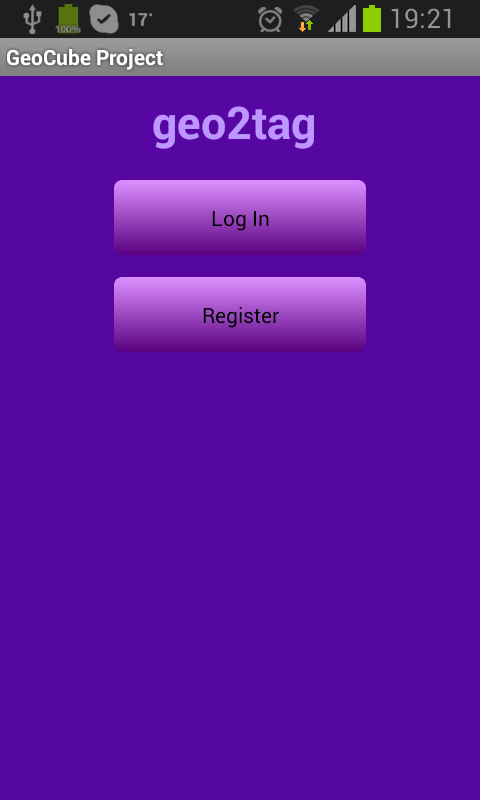
Для данного проекта было решено использовать Mixare – open sourse проект, представляет клиент-приложение, которое отображает теги из Википедии и Твиттера в режиме дополненной реальности.

Выбран был в связи с тем, что проект свободный, достаточно простой для понимания, имеет всю необходимую основу для разработки данного проекта.

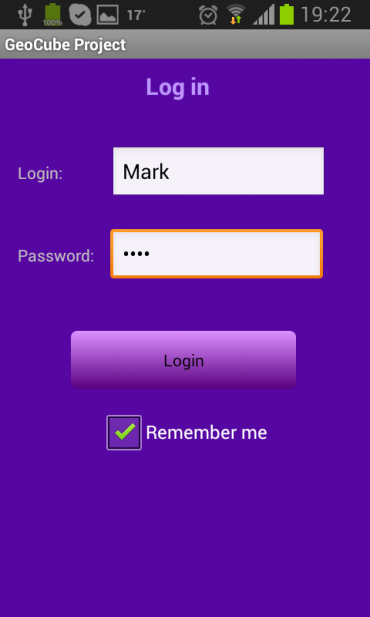
**12.Результаты работы, выводы.**

**12.1.Описание работы приложения, интерфейс пользователя.**

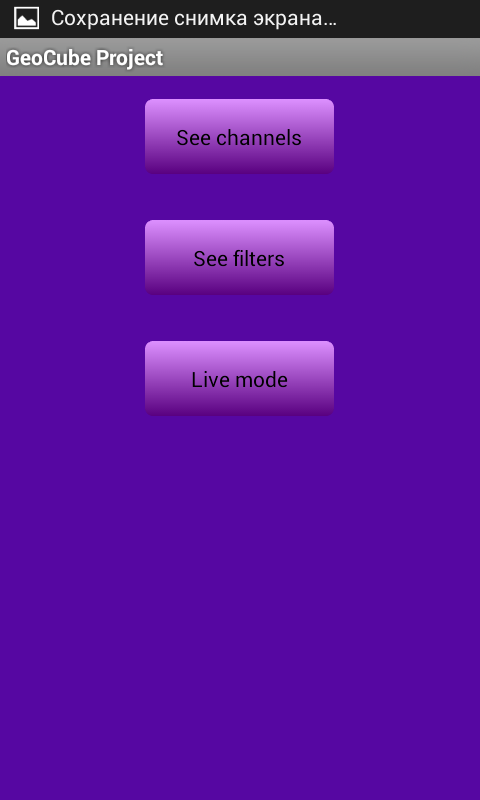
Стартовый экран с переходом на экраны входа или регистрации:



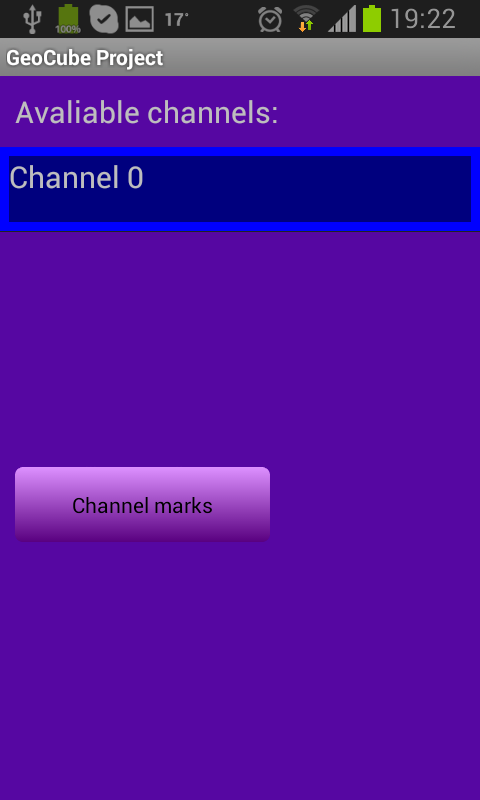
Экран входа в систему:



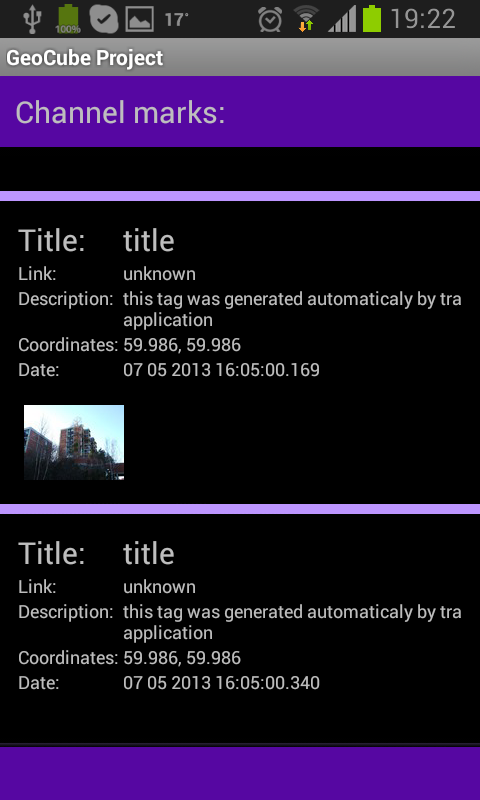
После успешного входа пользователь видит следующий экран с возможностью перейти на экран со списком доступных каналов, экран фильтров или режим отображения тегов на местности с помощью камеры:



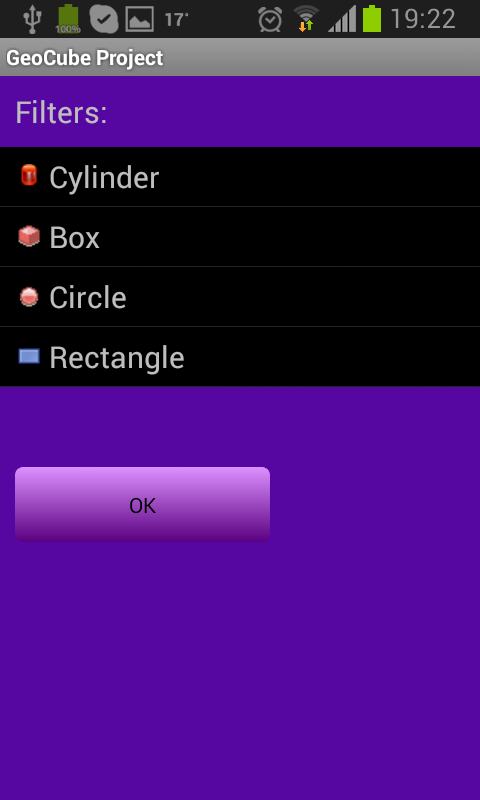
Экран со списком доступных каналов:



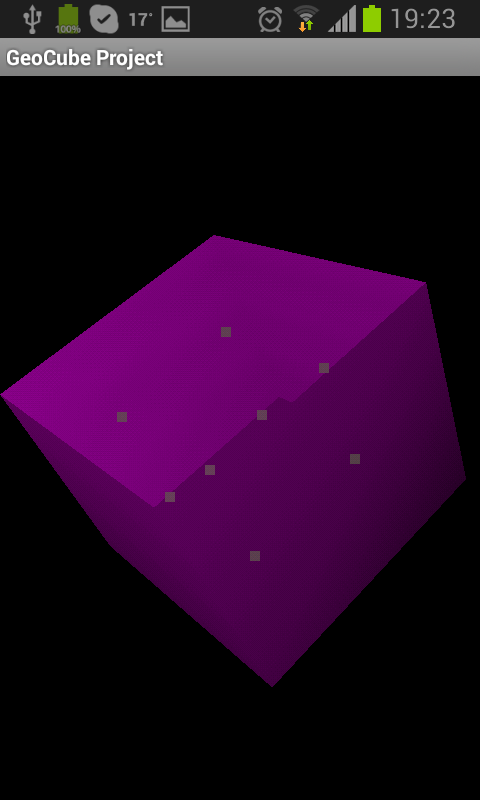
Теги выбранного канала:



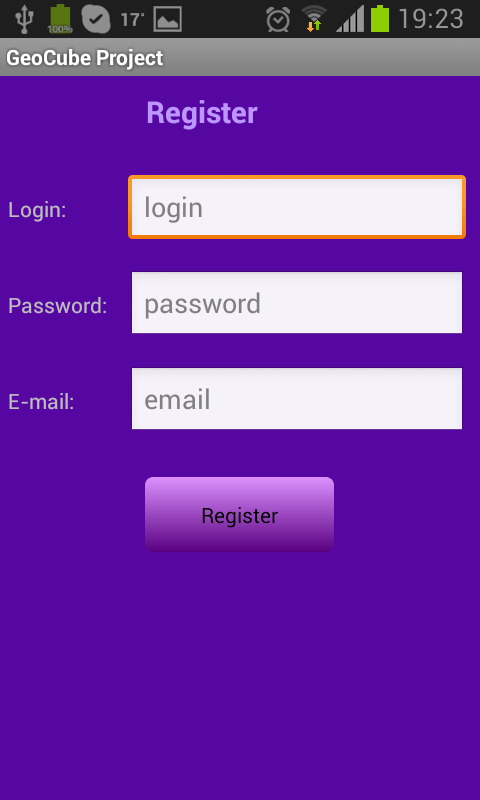
Экран доступных фильтров:



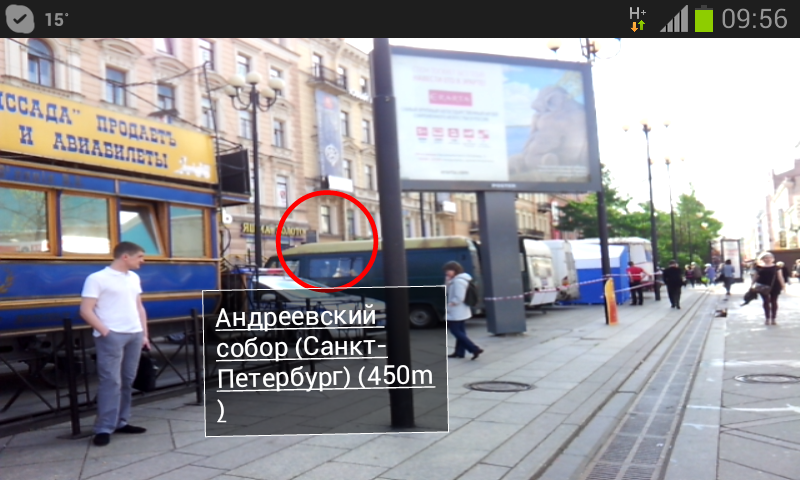
Пример фильтра (кубический фильтр):

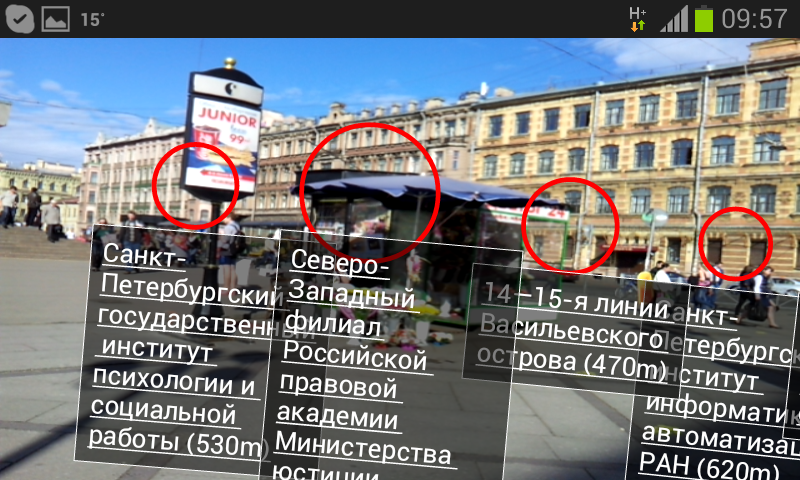


Экран регистрации:



Экран режима отображения тегов на местности через камеру:





**12.2.Архитектура.**

**12.2.1.Предметная область.**

* Геотег (геометка) – кортеж данных вида: {время, географические координаты, данные}.
* Канал – атрибут геотега, определяющий основной классификационный признак.
* Фильтр – 3D/2D область пространства с заключенными в ней геотегами.

**12.2.2.Прецеденты разрабатываемой системы.**

**Прецедент 1. Просмотр доступных каналов.**

Участники и их требования:

Пользователь приложения, который хочет просмотреть список доступных каналов.

Сценарий:

1. Пользователь посылает запрос на сервер, сервер возвращает пользователю список доступных каналов либо, в случае неудачи, сообщение об ошибке. Если подключение к интернету мобильного устройства отсутствует, пользователь получает предупреждение и предлагается повторить запрос.

2.В случае успеха пользователь видит список доступных каналов.

**Прецедент 2.Просмотр тегов каналов.**

Участники и их требования:

Пользователь, который хочет просмотреть доступные теги выбранного канала.

Сценарий:

1.Повторяется прецедент 1.

2.В случае успеха (пользователь видит список доступных каналов), пользователь может выбрать один из доступных каналов для просмотра его тегов.

3.Посылается запрос на сервер. В случае неудачи пользователь получает сообщение об ошибке, в случае успеха пользователь видит список доступных тегов данного канала.

**Прецедент 3.Просмотр доступных фильтров.**

Участники и их требования:

Пользователь, который хочет просмотреть один из доступных фильтров и теги, соответствующие этому фильтру.

Сценарий:

1.Пользователь выбирает один из фильтров из списка.

2.Посылается запрос на сервер. В случае неудачи, пользователь получает сообщение об ошибке, в случае успеха пользователь видит изображение фильтра с тегами, заключенными внутри фильтра.

**Прецедент 4.Просмотр тегов в «живом» режиме.**

Участники и их требования:

Пользователь, который хочет просмотреть теги на местности в «живом» режиме (с помощью камеры).

Сценарий:

1.Посылается запрос доступных каналов на сервер.

2.В случае неудачи пользователь получает сообщение об ошибке, в случае успеха пользователь видит теги на изображении камеры.

**Прецедент 5. Подписка на каналы.**

Участники и их требования:

Пользователь, который хочет иметь доступ к каналам и их тегам в оффлайн режиме.

Сценарий:

1.Повторяется прецедент 1.

2.Загруженные каналы с тегами сохраняются в память телефона.

3.В дальнейшем пользователь в оффлайн режиме может просмотреть список сохраненных каналов вместе с тегами.

**12.2.3.Диаграмма прецедентов.**

Просмотр доступных фильтров

Подписка на каналы

Просмотр тегов в «живом» режиме

Просмотр тегов каналов

Просмотр доступных каналов

**12.2.4. Схема взаимодействия Activity приложения.**

Стартовая страница

Логин

Регистрация

Главная страница пользователя

Просмотр каналов

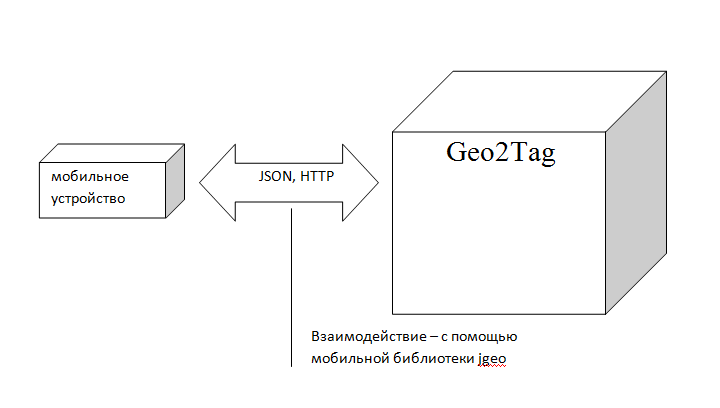
Просмотр тегов в «живом» режиме

Просмотр тегов канала

Просмотр фильтров

Просмотр тегов фильтра

**12.2.5.Описание взаимодействия с сервером.**

****

Взаимодействие с сервером geo2tag происходит с помощью мобильной библиотеки jgeo. Протокол передачи данных – HTTP, данные передаются в формате JSON.

Пример запроса на подписку каналов:

{

"description": "Subscribe to channel",

"type": "object",

"properties": {

"auth\_token": {"type":"string"},

"channel":{"type":"string"}

}

}

Пример ответа на этот запрос:

{

"description": "response",

"type": "object",

"properties": {

"errno": {"type":"number"}

}

}

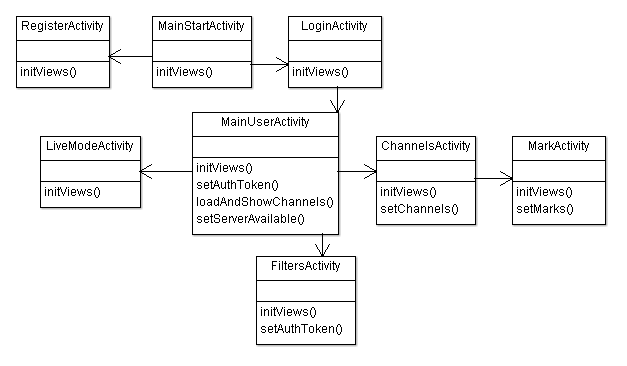
**12.2.6.Основные пакеты проекта, диаграмма классов.**

**Org.geo2tag.geocube** – пользовательские классы, использующиеся для отображения интерфейса, обращения к серверу, обработки запросов и тп.

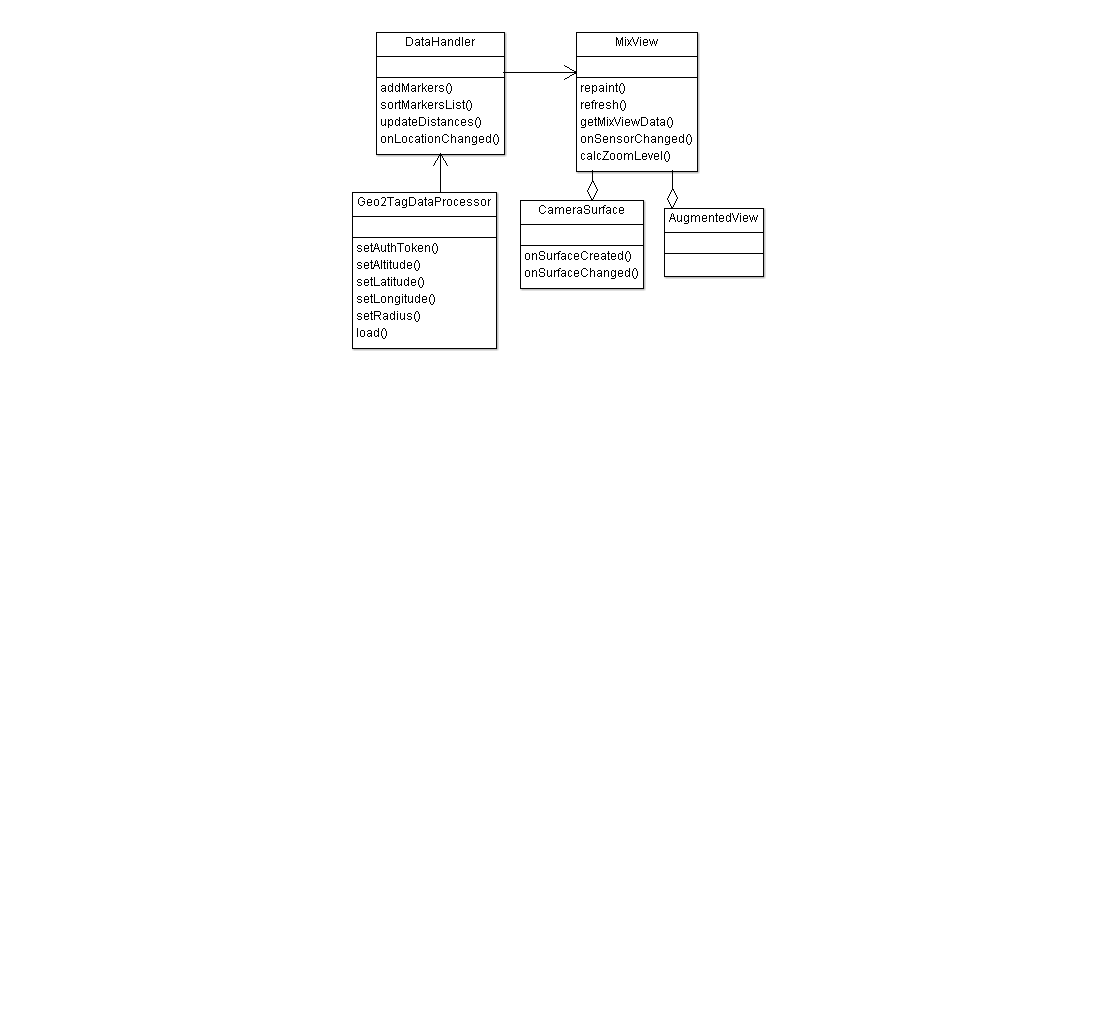
**Org.geo2tag.mixare** – классы, использующиеся для отображения тегов на местности с помощью камеры.

**ru.spb.osll** – библиотека jgeo для «общения» с сервером geo2tag.

12.2.6.1.org.geo2tag.geocube – диаграмма классов.



12.2.6.2. org.geo2tag.mixare – диаграмма классов.



**13. Направления развития**

* Ускорение скорости работы (отрисовки геотегов)

В настоящее время теги для отрисовки обновляются каждые 5 секунд, то есть каждые 5 секунд посылается запрос на сервер. Быстродействие планируется увеличить засчет сохранения тегов в памяти телефона и обновления списка каждый час либо любой другой более длительный интервал времени.

* Возможность задания расстояний, размеров фильтров

Радиус сканирования доступных тегов по умолчанию – 20 километров. В дальнейшем планируется предоставить пользователю возможность изменять это расстояние, а также задавать параметры фильтров (их размеров и координат вершин).

* Отображение тегов на изображении карты

Было бы удобно увидеть отображение тегов не только в режиме камеры, но и на изображении камеры.

* Поиск по тегам

Планируется добавить поиск по ключевым словам в списке полученных тегов.

**14. Результаты работы.**

* Разработан визуализатор тегов расширенной реальности для платформы Android для геолокацонной системы Geo2Tag.
* Исследованы методы отрисовки объектов в Android.
* Изучены области применения геолокации и дополненной реальности.

**15.Используемая литература и интернет-ресурсы.**

1) Satya Komatineni, Dave McLean “Pro Android”, 2012

2) Mark Murphy “The busy coder’s guide to advanced android development”, 2009

3) Голощапов А.Л. “Google Android. Программирование для мобильных устройств”, издательство БХВ-Петербург, 2011

4) Json Morris “Android user interface development guide”, 2011

5) developer.android.com

6) habrahabr.ru

7) ru.wikipedia.org

**16. Ссылки на ресурсы.**

[1] Geo2tag: geo2tag.org

[2] Mixare: mixare.org

[3] OpenGL ES: developer.android.com/training/graphics/opengl

[4] android.graphics: developer.android.com/reference/android/graphics

[5] AR-браузера junaio: www.junaio.com

[6] компанией Metaio: metaio.com

[7] AlterGeo: altergeo.ru

[8] Project Glass (Google Glass): www.google.com/glass/start

[9] Smart Glasses M100 (Vizux): www.vuzix.com/consumer/products\_m100.html

[10] выставку SXSW Interactive: sxsw.com/interactive

[11] Google+ : plus.google.com/up/search

[12] Box! Open Me (PlayStation Vita): www.giantbomb.com/box-open-me

[13] SpecTrek для Android: en.wikipedia.org/wiki/Spectrek

[14] AppTag для Android и iOS: www.hex3.co/products/apptag

[15] Sharky the beaver: www.gosphero.com/featured%20videos/sphero-augmented-reality-sharky-the-beaver/

[16] Робот-мячик – Sphero: www.gosphero.com/

[17] ГЛОНАСС: ru.wikipedia.org/wiki/ГЛОНАСС

[18] GPS: ru.wikipedia.org/wiki/GPS

[19] GSM: ru.wikipedia.org/wiki/GSM

[20] UMTS: ru.wikipedia.org/wiki/UMTS

[21] WCDMA: ru.wikipedia.org/wiki/W-CDMA

[22] LTE: ru.wikipedia.org/wiki/LTE\_Advanced

[23] Geo2tag LBS Platform: geo2tag.org

[24] Foursquare: ru.wikipedia.org/wiki/Foursquare

[25] AlterGeo: altergeo.ru

[26] Google Latitude: latitude.google.com/latitude

[27] Dodgeball: en.wikipedia.org/wiki/Dodgeball

[28] Google Maps: maps.google.ru

[29] Strategy Analytics: www.strategyanalytics.com

[30] Layar Reality Browser: www.layar.com

[31] WikiTude Drive: www.wikitude.com/drivebeta

[32] Wikitude World Browser: wikitude-world-browser.en.softonic.com

[33] 3D Compass+ (AR Compass): 3d-compass-beta-ar-compass.android.informer.com

[34] Augement: augmentedev.com

**17.Приложение (код основных классов).**

**Пакет org.geo2tag.geocube.**

public class FiltersActivity extends MainUserActivity {

private ArrayList<FilterItem> filterItems;

private ListView filtersList;

private FilterListAdapter mArrayAdapter;

private Button okFilter;

private Location location;

private static String authToken;

double latitude;

double longitude;

double radius = 500;

double deltaX = 30;

double deltaY = 30;

String timeFrom = ProjectConstants.TIME\_FROM;

String timeTo = new Date().toString();

public static void setAuthToken(String authToken) {

FiltersActivity.authToken = authToken;

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_NOSENSOR);

setContentView(R.layout.filters\_activity);

location = MyLocationManager.getLocation(this);

latitude = location.getLatitude();

longitude = location.getLongitude();

initViews();

}

private View lastSelectedView = null;

public void clearSelection()

{

if(lastSelectedView != null) lastSelectedView.setBackgroundColor(Color.BLACK);

}

private void initViews() {

filtersList = (ListView) findViewById(R.id.filtersList);

filtersList.setFocusableInTouchMode(true);

filtersList.setChoiceMode(ListView.CHOICE\_MODE\_SINGLE);

filtersList.setOnItemClickListener(new AdapterView.OnItemClickListener() {

public void onItemClick(AdapterView<?> myAdapter, View myView, int myItemInt, long mylng) {

clearSelection();

lastSelectedView = myView;

myView.setBackgroundColor(Color.BLUE);

((FilterListAdapter) myAdapter.getAdapter()).setItemSelected(myItemInt);

}

});

filterItems = new ArrayList<FilterItem>();

FilterItem itemCylinder = new FilterItem("Cylinder", R.drawable.cylinder\_icon);

FilterItem itemBox = new FilterItem("Box", R.drawable.cube\_icon);

FilterItem itemCircle = new FilterItem("Circle", R.drawable.circle\_icon);

FilterItem itemRectangle = new FilterItem("Rectangle", R.drawable.rectangle\_icon);

filterItems.add(itemCylinder);

filterItems.add(itemBox);

filterItems.add(itemCircle);

filterItems.add(itemRectangle);

mArrayAdapter = new FilterListAdapter(this, filterItems);

filtersList.setAdapter(mArrayAdapter);

filtersList.setTextFilterEnabled(true);

mArrayAdapter.setNotifyOnChange(true);

okFilter = (Button) findViewById(R.id.okFilter);

okFilter.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

FilterItem selectedItem = filterItems.get(mArrayAdapter.getCurrentSelectedItem());

double x = latitude - deltaX;

double x1 = latitude + deltaX;

double y = longitude - deltaY;

double y1 = longitude + deltaY;

double z = location.getAltitude();

double z1 = z + radius;

if (selectedItem.filterName.equals("Box")) {

JsonFilterBoxRequest req = new JsonFilterBoxRequest(authToken, x, x1, y, y1, z, z1, timeFrom, timeTo, ProjectConstants.SERVER\_URL);

JSONObject ob = req.doRequest();

CubeActivity.setCoordinates(x, x1, y, y1, z, z1);

Intent in = new Intent(FiltersActivity.this, CubeActivity.class);

startActivity(in);

} else if (selectedItem.filterName.equals("Cylinder")) {

CylinderGLRenderer.setCoordinateForCylinder(radius / 2.0f, radius);

CylinderGLRenderer.setCoordinatesForPoints(x, y, z, x1, y1, z1);

Intent in = new Intent(FiltersActivity.this, CylinderActivity.class);

startActivity(in);

} else if (selectedItem.filterName.equals("Circle")) {

} else if (selectedItem.filterName.equals("Rectangle")) {

}

}

});

}

}

public class LiveModeActivity extends MainUserActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

}

}

public class LoginActivity extends MainStartActivity {

private EditText loginText;

private EditText passText;

private Button loginButton;

private CheckBox rememberMe;

private SharedPreferences settings;

private String loginSaved;

private String passSaved;

String authToken;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_NOSENSOR);

setContentView(R.layout.login\_activity);

initViews();

}

public void initViews() {

settings = getSharedPreferences(PREFS\_SETTINGS\_LOGIN\_NAME, MODE\_PRIVATE);

loginSaved = settings.getString("login", "");

passSaved = settings.getString("pass", "");

authToken = settings.getString("token", "");

loginText = (EditText) findViewById(R.id.loginText);

passText = (EditText) findViewById(R.id.passText);

loginButton = (Button) findViewById(R.id.loginButton);

rememberMe = (CheckBox) findViewById(R.id.rememberMeCheck);

rememberMe.setChecked(true);

loginText.setText(loginSaved);

passText.setText(passSaved);

loginButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String login = loginText.getText().toString();

String pass = passText.getText().toString();

if (!authToken.equals("") && loginSaved.equals(login) && passSaved.equals(pass)) {

MainUserActivity.setAuthToken(authToken);

MainUserActivity.setServerAvaliable(true);

} else {

StrictMode.ThreadPolicy tp = StrictMode.ThreadPolicy.LAX;

StrictMode.setThreadPolicy(tp);

JsonLoginRequest loginReques = new JsonLoginRequest(login, pass, SERVER\_URL);

JSONObject r = loginReques.doRequest();

JsonLoginResponse responce = new JsonLoginResponse();

if (r != null) {

responce.parseJson(r);

MainUserActivity.setAuthToken(responce.getAuthString());

MainUserActivity.setServerAvaliable(true);

} else {

MainUserActivity.setAuthToken(UUID.randomUUID().toString());

MainUserActivity.setServerAvaliable(false);

}

if (rememberMe.isChecked()) {

SharedPreferences.Editor editSettings = settings.edit();

editSettings.putString("login", login);

editSettings.putString("pass", pass);

editSettings.putString("token", authToken);

editSettings.commit();

}

}

Intent in = new Intent(LoginActivity.this, MainUserActivity.class);

startActivity(in);

}

});

}

}

public class RegisterActivity extends MainStartActivity {

private Button register;

private EditText pass;

private EditText login;

private EditText email;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_NOSENSOR);

setContentView(R.layout.register\_activity);

initViews();

}

public void initViews() {

register = (Button) findViewById(R.id.registerButton);

pass = (EditText) findViewById(R.id.passText);

login = (EditText) findViewById(R.id.loginText);

email = (EditText) findViewById(R.id.emailText);

register.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String emailString = email.getText().toString();

String passString = pass.getText().toString();

String loginString = login.getText().toString();

StrictMode.ThreadPolicy tp = StrictMode.ThreadPolicy.LAX;

StrictMode.setThreadPolicy(tp);

JsonRegisterUserReguest regRequest = new JsonRegisterUserReguest(emailString, loginString,

passString, ProjectConstants.SERVER\_URL);

JSONObject ob = regRequest.doRequest();

JsonRegisterUserResponse resp = new JsonRegisterUserResponse();

resp.parseJson(ob);

if (resp.getErrno() == 0) {

Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);

startActivity(intent);

} else {

new AlertDialog.Builder(RegisterActivity.this)

.setTitle("Error!")

.setMessage(ErrorMessages.getInstance().get(resp.getErrno()))

.setNeutralButton("Ok", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog,

int which) {

}

}).show();

}

}

});

}

}

public class MainStartActivity {

private Button registerButton;

private Button loginButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_NOSENSOR);

setContentView(R.layout.main\_start\_activity);

initViews();

}

public void initViews() {

registerButton = (Button) findViewById(R.id.registerButton);

loginButton = (Button) findViewById(R.id.loginButton);

registerButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent in = new Intent(MainStartActivity.this, RegisterActivity.class);

startActivity(in);

}

});

loginButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent in = new Intent(MainStartActivity.this, LoginActivity.class);

startActivity(in);

}

});

}

}

public class MainUserActivity extends MainStartActivity {

private static String authToken;

private Button scanChannnels;

private Button scanFilters;

private Button liveMode;

Location location;

double latitude;

double longitude;

double radius = 500;

private static boolean isServerAvaliable = false;

public static void setServerAvaliable(boolean serverAvaliable) {

isServerAvaliable = serverAvaliable;

}

public static void setAuthToken(String authToken) {

MainUserActivity.authToken = authToken;

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_NOSENSOR);

location = MyLocationManager.getLocation(this);

latitude = location.getLatitude();

longitude = location.getLongitude();

Log.d("Auth token: ", authToken);

setContentView(R.layout.main\_user\_activity);

initViews();

if (isServerAvaliable) {

Toast.makeText(this, "Server is avaliable", 3000);

} else {

Toast.makeText(this, "Server is not avaliable", 3000);

}

}

public void initViews() {

scanChannnels = (Button) findViewById(R.id.channelButton);

scanChannnels.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

loadAndShowChannels();

}

});

scanFilters = (Button) findViewById(R.id.filterButton);

scanFilters.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

FiltersActivity.setAuthToken(authToken);

Intent in = new Intent(MainUserActivity.this, FiltersActivity.class);

startActivity(in);

}

});

liveMode = (Button) findViewById(R.id.liveModeButton);

liveMode.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Geo2TagDataProcessor.setAuthToken(authToken);

Geo2TagDataProcessor.setLatitude(latitude);

Geo2TagDataProcessor.setLongitude(longitude);

Geo2TagDataProcessor.setRadius(radius);

Intent in = new Intent(MainUserActivity.this, MixView.class);

startActivity(in);

}

});

}

public void loadAndShowChannels() {

JsonLoadTagsRequest request = new JsonLoadTagsRequest(authToken, latitude, longitude, radius, ProjectConstants.SERVER\_URL);

JSONObject ob = request.doRequest();

if (ob != null) {

JsonLoadTagsResponse resp = new JsonLoadTagsResponse();

resp.parseJson(ob);

ArrayList<Channel> c = (ArrayList<Channel>) resp.getChannels();

ChannelsActivity.setChannels(c);

} else {

ArrayList<Channel> c = new ArrayList<Channel>();

ChannelsActivity.setChannels(c);

}

Intent in = new Intent(MainUserActivity.this, ChannelsActivity.class);

startActivity(in);

}

}

**Пакет org.geo2tag.mixare.**

public class Geo2TagDataProcessor extends DataHandler implements DataProcessor {

private static String authToken;

private static double latitude;

private static double longitude;

private static double radius;

public static void setAuthToken(String authToken) {

Geo2TagDataProcessor.authToken = authToken;

}

public static void setLatitude(double latitude) {

Geo2TagDataProcessor.latitude = latitude;

}

public static void setLongitude(double longitude) {

Geo2TagDataProcessor.longitude = longitude;

}

public static void setRadius(double radius) {

Geo2TagDataProcessor.radius = radius;

}

@Override

public String[] getUrlMatch() {

String[] str = {"geo2tag"};

return str;

}

@Override

public String[] getDataMatch() {

String[] str = {"geo2tag"};

return str;

}

@Override

public boolean matchesRequiredType(String type) {

if(type.equals(DataSource.TYPE.GEO2TAG.name())){

return true;

}

return false;

}

@Override

public List<Marker> load(String rawData, int taskId, int colour) throws JSONException {

List<Marker> markers = new ArrayList<Marker>();

ArrayList<Channel> channels = new ArrayList<Channel>();

radius = 20;

JsonLoadTagsRequest request = new JsonLoadTagsRequest(authToken, latitude, longitude, radius, ProjectConstants.SERVER\_URL);

JSONObject ob = request.doRequest();

if (ob != null) {

JsonLoadTagsResponse resp = new JsonLoadTagsResponse();

resp.parseJson(ob);

channels = (ArrayList<Channel>) resp.getChannels();

}

if (!channels.isEmpty()) {

for (Channel ch : channels) {

ArrayList<Mark> marks = (ArrayList<Mark>) ch.getMarks();

for (Mark m : marks) {

Marker ma = new POIMarker(

m.getTitle(),

HtmlUnescape.unescapeHTML(m.getDescription(), 0),

m.getLatitude(),

m.getLongitude(),

m.getAltitude(),

m.getLink(),

taskId, colour);

markers.add(ma);

}

}

}

return markers;

}

}

public class MixView extends Activity implements SensorEventListener, OnTouchListener {

private CameraSurface camScreen;

private AugmentedView augScreen;

private boolean isInited;

private static PaintScreen dWindow;

private static DataView dataView;

private boolean fError;

private MixViewDataHolder mixViewData;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

try {

handleIntent(getIntent());

final PowerManager pm = (PowerManager) getSystemService(Context.POWER\_SERVICE);

getMixViewData().setmWakeLock(pm.newWakeLock(PowerManager.SCREEN\_BRIGHT\_WAKE\_LOCK, "My Tag"));

killOnError();

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

maintainCamera();

maintainAugmentR();

if (!isInited) {

setdWindow(new PaintScreen());

setDataView(new DataView(getMixViewData().getMixContext()));

/\* set the radius in data view to the last selected by the user \*/

setZoomLevel();

isInited = true;

}

} catch (Exception ex) {

doError(ex);

}

}

public MixViewDataHolder getMixViewData() {

if (mixViewData==null){

mixViewData = new MixViewDataHolder(new MixContext(this));

}

return mixViewData;

}

@Override

protected void onPause() {

super.onPause();

try {

this.getMixViewData().getmWakeLock().release();

try {

getMixViewData().getSensorMgr().unregisterListener(this,

getMixViewData().getSensorGrav());

getMixViewData().getSensorMgr().unregisterListener(this,

getMixViewData().getSensorMag());

getMixViewData().setSensorMgr(null);

getMixViewData().getMixContext().getLocationFinder().switchOff();

getMixViewData().getMixContext().getDownloadManager().switchOff();

if (getDataView() != null) {

getDataView().cancelRefreshTimer();

}

} catch (Exception ignore) {

}

if (fError) {

finish();

}

} catch (Exception ex) {

doError(ex);

}

}

/\*\*

\* {@inheritDoc}

\* Mixare - Receives results from other launched activities

\* Base on the result returned, it either refreshes screen or not.

\* Default value for refreshing is false

\*/

protected void onActivityResult(final int requestCode,

final int resultCode, Intent data) {

try {

if (data.getBooleanExtra("RefreshScreen", false)) {

if (data.getBooleanExtra("RefreshScreen", true)) {

repaint();

}

} catch (Exception ex) {

// do nothing do to mix of return results.

}

}

@Override

protected void onResume() {

super.onResume();

try {

this.getMixViewData().getmWakeLock().acquire();

killOnError();

getMixViewData().getMixContext().doResume(this);

repaint();

getDataView().doStart();

getDataView().clearEvents();

getMixViewData().getMixContext().getDataSourceManager().refreshDataSources();

float angleX, angleY;

int marker\_orientation = -90;

int rotation = Compatibility.getRotation(this);

// display text from left to right and keep it horizontal

angleX = (float) Math.toRadians(marker\_orientation);

getMixViewData().getM1().set(1f, 0f, 0f, 0f,

FloatMath.cos(angleX),

-FloatMath.sin(angleX), 0f,

FloatMath.sin(angleX),

FloatMath.cos(angleX));

angleX = (float) Math.toRadians(marker\_orientation);

angleY = (float) Math.toRadians(marker\_orientation);

if (rotation == 1) {

getMixViewData().getM2().set(1f, 0f, 0f, 0f,

FloatMath.cos(angleX),

-FloatMath.sin(angleX), 0f,

FloatMath.sin(angleX),

FloatMath.cos(angleX));

getMixViewData().getM3().set(FloatMath.cos(angleY), 0f,

FloatMath.sin(angleY), 0f, 1f, 0f,

-FloatMath.sin(angleY), 0f,

FloatMath.cos(angleY));

} else {

getMixViewData().getM2().set(FloatMath.cos(angleX), 0f,

FloatMath.sin(angleX), 0f, 1f, 0f,

-FloatMath.sin(angleX), 0f,

FloatMath.cos(angleX));

getMixViewData().getM3().set(1f, 0f, 0f, 0f,

FloatMath.cos(angleY),

-FloatMath.sin(angleY), 0f,

FloatMath.sin(angleY),

FloatMath.cos(angleY));

}

getMixViewData().getM4().toIdentity();

for (int i = 0; i < getMixViewData().getHistR().length; i++) {

getMixViewData().getHistR()[i] = new Matrix();

}

getMixViewData()

.setSensorMgr((SensorManager) getSystemService(SENSOR\_SERVICE));

getMixViewData().setSensors(getMixViewData().getSensorMgr().getSensorList(

Sensor.TYPE\_ACCELEROMETER));

if (getMixViewData().getSensors().size() > 0) {

getMixViewData().setSensorGrav(getMixViewData().getSensors().get(0));

}

getMixViewData().setSensors(getMixViewData().getSensorMgr().getSensorList(

Sensor.TYPE\_MAGNETIC\_FIELD));

if (getMixViewData().getSensors().size() > 0) {

getMixViewData().setSensorMag(getMixViewData().getSensors().get(0));

}

getMixViewData().getSensorMgr().registerListener(this,

getMixViewData().getSensorGrav(), SENSOR\_DELAY\_GAME);

getMixViewData().getSensorMgr().registerListener(this,

getMixViewData().getSensorMag(), SENSOR\_DELAY\_GAME);

try {

GeomagneticField gmf = getMixViewData().getMixContext().getLocationFinder().getGeomagneticField();

angleY = (float) Math.toRadians(-gmf.getDeclination());

getMixViewData().getM4().set(FloatMath.cos(angleY), 0f,

FloatMath.sin(angleY), 0f, 1f, 0f,

-FloatMath.sin(angleY), 0f,

FloatMath.cos(angleY));

} catch (Exception ex) {

ex.printStackTrace();

}

getMixViewData().getMixContext().getDownloadManager().switchOn();

getMixViewData().getMixContext().getLocationFinder().switchOn();

} catch (Exception ex) {

doError(ex);

try {

if (getMixViewData().getSensorMgr() != null) {

getMixViewData().getSensorMgr().unregisterListener(this,

getMixViewData().getSensorGrav());

getMixViewData().getSensorMgr().unregisterListener(this,

getMixViewData().getSensorMag());

getMixViewData().setSensorMgr(null);

}

if (getMixViewData().getMixContext() != null) {

getMixViewData().getMixContext().getLocationFinder().switchOff();

getMixViewData().getMixContext().getDownloadManager().switchOff();

}

} catch (Exception ignore) {

}

}

if (getDataView().isFrozen() && getMixViewData().getSearchNotificationTxt() == null) {

getMixViewData().setSearchNotificationTxt(new TextView(this));

getMixViewData().getSearchNotificationTxt().setWidth(

getdWindow().getWidth());

getMixViewData().getSearchNotificationTxt().setPadding(10, 2, 0, 0);

getMixViewData().getSearchNotificationTxt().setText(

getString(R.string.search\_active\_1) + " "

+ DataSourceList.getDataSourcesStringList()

+ getString(R.string.search\_active\_2));

getMixViewData().getSearchNotificationTxt().setBackgroundColor(

Color.DKGRAY);

getMixViewData().getSearchNotificationTxt().setTextColor(Color.WHITE);

getMixViewData().getSearchNotificationTxt().setOnTouchListener(this);

addContentView(getMixViewData().getSearchNotificationTxt(),

new LayoutParams(LayoutParams.FILL\_PARENT,

LayoutParams.WRAP\_CONTENT));

} else if (!getDataView().isFrozen()

&& getMixViewData().getSearchNotificationTxt() != null) {

getMixViewData().getSearchNotificationTxt().setVisibility(View.GONE);

getMixViewData().setSearchNotificationTxt(null);

}

}

/\*\*

\* {@inheritDoc}

\* Customize Activity after switching back to it.

\* Currently it maintain and ensures view creation.

\*/

protected void onRestart (){

super.onRestart();

maintainCamera();

maintainAugmentR();

}

public void repaint() {

//clear stored data

getDataView().clearEvents();

setDataView(null); //It's smelly code, but enforce garbage collector

//to release data.

setDataView(new DataView(mixViewData.getMixContext()));

setdWindow(new PaintScreen());

}

/\*\*

\* Checks camScreen, if it does not exist, it creates one.

\*/

private void maintainCamera() {

if (camScreen == null){

camScreen = new CameraSurface(this);

}

setContentView(camScreen);

}

/\*\*

\* Checks augScreen, if it does not exist, it creates one.

\*/

private void maintainAugmentR() {

if (augScreen == null ){

augScreen = new AugmentedView(this);

}

addContentView(augScreen, new LayoutParams(

LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT));

}

public void refresh(){

dataView.refresh();

}

public float calcZoomLevel(){

int myZoomLevel = 500;

float myout = 5;

if (myZoomLevel <= 26) {

myout = myZoomLevel / 25f;

} else if (25 < myZoomLevel && myZoomLevel < 50) {

myout = (1 + (myZoomLevel - 25)) \* 0.38f;

} else if (25 == myZoomLevel) {

myout = 1;

} else if (50 == myZoomLevel) {

myout = 10;

} else if (50 < myZoomLevel && myZoomLevel < 75) {

myout = (10 + (myZoomLevel - 50)) \* 0.83f;

} else {

myout = (30 + (myZoomLevel - 75) \* 2f);

}

return myout;

}

/\* \*\*\*\*\*\*\*\* Operators - Sensors \*\*\*\*\*\* \*/

public void onSensorChanged(SensorEvent evt) {

try {

if (evt.sensor.getType() == Sensor.TYPE\_ACCELEROMETER) {

getMixViewData().getGrav()[0] = evt.values[0];

getMixViewData().getGrav()[1] = evt.values[1];

getMixViewData().getGrav()[2] = evt.values[2];

augScreen.postInvalidate();

} else if (evt.sensor.getType() == Sensor.TYPE\_MAGNETIC\_FIELD) {

getMixViewData().getMag()[0] = evt.values[0];

getMixViewData().getMag()[1] = evt.values[1];

getMixViewData().getMag()[2] = evt.values[2];

augScreen.postInvalidate();

}

SensorManager.getRotationMatrix(getMixViewData().getRTmp(),

getMixViewData().getI(), getMixViewData().getGrav(),

getMixViewData().getMag());

int rotation = Compatibility.getRotation(this);

if (rotation == 1) {

SensorManager.remapCoordinateSystem(getMixViewData().getRTmp(),

SensorManager.AXIS\_X, SensorManager.AXIS\_MINUS\_Z,

getMixViewData().getRot());

} else {

SensorManager.remapCoordinateSystem(getMixViewData().getRTmp(),

SensorManager.AXIS\_Y, SensorManager.AXIS\_MINUS\_Z,

getMixViewData().getRot());

}

getMixViewData().getTempR().set(getMixViewData().getRot()[0],

getMixViewData().getRot()[1], getMixViewData().getRot()[2],

getMixViewData().getRot()[3], getMixViewData().getRot()[4],

getMixViewData().getRot()[5], getMixViewData().getRot()[6],

getMixViewData().getRot()[7], getMixViewData().getRot()[8]);

getMixViewData().getFinalR().toIdentity();

getMixViewData().getFinalR().prod(getMixViewData().getM4());

getMixViewData().getFinalR().prod(getMixViewData().getM1());

getMixViewData().getFinalR().prod(getMixViewData().getTempR());

getMixViewData().getFinalR().prod(getMixViewData().getM3());

getMixViewData().getFinalR().prod(getMixViewData().getM2());

getMixViewData().getFinalR().invert();

getMixViewData().getHistR()[getMixViewData().getrHistIdx()].set(getMixViewData()

.getFinalR());

getMixViewData().setrHistIdx(getMixViewData().getrHistIdx() + 1);

if (getMixViewData().getrHistIdx() >= getMixViewData().getHistR().length)

getMixViewData().setrHistIdx(0);

getMixViewData().getSmoothR().set(0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f);

for (int i = 0; i < getMixViewData().getHistR().length; i++) {

getMixViewData().getSmoothR().add(getMixViewData().getHistR()[i]);

}

getMixViewData().getSmoothR().mult(

1 / (float) getMixViewData().getHistR().length);

getMixViewData().getMixContext().updateSmoothRotation(getMixViewData().getSmoothR());

} catch (Exception ex) {

ex.printStackTrace();

}

}

@Override

public boolean onTouchEvent(MotionEvent me) {

try {

killOnError();

float xPress = me.getX();

float yPress = me.getY();

if (me.getAction() == MotionEvent.ACTION\_UP) {

getDataView().clickEvent(xPress, yPress);

}//TODO add gesture events (low)

return true;

} catch (Exception ex) {

// doError(ex);

ex.printStackTrace();

return super.onTouchEvent(me);

}

}

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

try {

killOnError();

if (keyCode == KeyEvent.KEYCODE\_BACK) {

if (getDataView().isDetailsView()) {

getDataView().keyEvent(keyCode);

getDataView().setDetailsView(false);

return true;

} else {

//TODO handle keyback to finish app correctly

return super.onKeyDown(keyCode, event);

}

} else if (keyCode == KeyEvent.KEYCODE\_MENU) {

return super.onKeyDown(keyCode, event);

} else {

getDataView().keyEvent(keyCode);

return false;

}

} catch (Exception ex) {

ex.printStackTrace();

return super.onKeyDown(keyCode, event);

}

}

public void onAccuracyChanged(Sensor sensor, int accuracy) {

if (sensor.getType() == Sensor.TYPE\_MAGNETIC\_FIELD

&& accuracy == SensorManager.SENSOR\_STATUS\_UNRELIABLE

&& getMixViewData().getCompassErrorDisplayed() == 0) {

for (int i = 0; i < 2; i++) {

Toast.makeText(getMixViewData().getMixContext(),

"Compass data unreliable. Please recalibrate compass.",

Toast.LENGTH\_LONG).show();

}

getMixViewData().setCompassErrorDisplayed(getMixViewData()

.getCompassErrorDisplayed() + 1);

}

}

@Override

public boolean onTouch(View v, MotionEvent event) {

getDataView().setFrozen(false);

if (getMixViewData().getSearchNotificationTxt() != null) {

getMixViewData().getSearchNotificationTxt().setVisibility(View.GONE);

getMixViewData().setSearchNotificationTxt(null);

}

return false;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\* Handlers \*\*\*\*\*\*\*\*\*\*\*\*\*/

public void doError(Exception ex1) {

if (!fError) {

fError = true;

setErrorDialog();

ex1.printStackTrace();

}

try {

augScreen.invalidate();

} catch (Exception ignore) {

}

}

public void killOnError() throws Exception {

if (fError)

throw new Exception();

}

private void handleIntent(Intent intent) {

if (Intent.ACTION\_SEARCH.equals(intent.getAction())) {

String query = intent.getStringExtra(SearchManager.QUERY);

doMixSearch(query);

}

}

private void doMixSearch(String query) {

DataHandler jLayer = getDataView().getDataHandler();

if (!getDataView().isFrozen()) {

MixListView.originalMarkerList = jLayer.getMarkerList();

MixMap.originalMarkerList = jLayer.getMarkerList();

}

ArrayList<Marker> searchResults = new ArrayList<Marker>();

Log.d("SEARCH-------------------0", "" + query);

if (jLayer.getMarkerCount() > 0) {

for (int i = 0; i < jLayer.getMarkerCount(); i++) {

Marker ma = jLayer.getMarker(i);

if (ma.getTitle().toLowerCase().indexOf(query.toLowerCase()) != -1) {

searchResults.add(ma);

/\* the website for the corresponding title \*/

}

}

}

if (searchResults.size() > 0) {

getDataView().setFrozen(true);

jLayer.setMarkerList(searchResults);

}

else Toast.makeText(this, getString(R.string.search\_failed\_notification),

Toast.LENGTH\_LONG).show();

}

/\* \*\*\*\*\*\*\* Getter and Setters \*\*\*\*\*\*\*\*\*\* \*/

/\*\*

\* @return the dWindow

\*/

static PaintScreen getdWindow() {

return dWindow;

}

/\*\*

\* @param dWindow

\* the dWindow to set

\*/

static void setdWindow(PaintScreen dWindow) {

MixView.dWindow = dWindow;

}

/\*\*

\* @return the dataView

\*/

static DataView getDataView() {

return dataView;

}

/\*\*

\* @param dataView

\* the dataView to set

\*/

static void setDataView(DataView dataView) {

MixView.dataView = dataView;

}

private void setZoomLevel() {

float myout = calcZoomLevel();

getDataView().setRadius(myout);

mixViewData.setZoomLevel(String.valueOf(myout));

getMixViewData().getMixContext().getDownloadManager().switchOn();

}

/\*\*

\* @author daniele

\*

\*/

class CameraSurface extends SurfaceView implements SurfaceHolder.Callback {

MixView app;

SurfaceHolder holder;

Camera camera;

CameraSurface(Context context) {

super(context);

try {

app = (MixView) context;

holder = getHolder();

holder.addCallback(this);

holder.setType(SurfaceHolder.SURFACE\_TYPE\_PUSH\_BUFFERS);

} catch (Exception ex) {

}

}

public void surfaceCreated(SurfaceHolder holder) {

try {

if (camera != null) {

try {

camera.stopPreview();

} catch (Exception ignore) {

}

try {

camera.release();

} catch (Exception ignore) {

}

camera = null;

}

camera = Camera.open();

camera.setPreviewDisplay(holder);

} catch (Exception ex) {

try {

if (camera != null) {

try {

camera.stopPreview();

} catch (Exception ignore) {

}

try {

camera.release();

} catch (Exception ignore) {

}

camera = null;

}

} catch (Exception ignore) {

}

}

}

public void surfaceDestroyed(SurfaceHolder holder) {

try {

if (camera != null) {

try {

camera.stopPreview();

} catch (Exception ignore) {

}

try {

camera.release();

} catch (Exception ignore) {

}

camera = null;

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {

try {

Camera.Parameters parameters = camera.getParameters();

try {

List<Camera.Size> supportedSizes = null;

// On older devices (<1.6) the following will fail

// the camera will work nevertheless

supportedSizes = Compatibility.getSupportedPreviewSizes(parameters);

// preview form factor

float ff = (float) w / h;

Log.d("Mixare", "Screen res: w:" + w + " h:" + h

+ " aspect ratio:" + ff);

// holder for the best form factor and size

float bff = 0;

int bestw = 0;

int besth = 0;

Iterator<Camera.Size> itr = supportedSizes.iterator();

// we look for the best preview size, it has to be the closest

// to the

// screen form factor, and be less wide than the screen itself

// the latter requirement is because the HTC Hero with update

// 2.1 will

// report camera preview sizes larger than the screen, and it

// will fail

// to initialize the camera

// other devices could work with previews larger than the screen

// though

while (itr.hasNext()) {

Camera.Size element = itr.next();

// current form factor

float cff = (float) element.width / element.height;

// check if the current element is a candidate to replace

// the best match so far

// current form factor should be closer to the bff

// preview width should be less than screen width

// preview width should be more than current bestw

// this combination will ensure that the highest resolution

// will win

Log.d("Mixare", "Candidate camera element: w:"

+ element.width + " h:" + element.height

+ " aspect ratio:" + cff);

if ((ff - cff <= ff - bff) && (element.width <= w)

&& (element.width >= bestw)) {

bff = cff;

bestw = element.width;

besth = element.height;

}

}

Log.d("Mixare", "Chosen camera element: w:" + bestw + " h:"

+ besth + " aspect ratio:" + bff);

// Some Samsung phones will end up with bestw and besth = 0

// because their minimum preview size is bigger then the screen

// size.

// In this case, we use the default values: 480x320

if ((bestw == 0) || (besth == 0)) {

Log.d("Mixare", "Using default camera parameters!");

bestw = 480;

besth = 320;

}

parameters.setPreviewSize(bestw, besth);

} catch (Exception ex) {

parameters.setPreviewSize(480, 320);

}

camera.setParameters(parameters);

camera.startPreview();

} catch (Exception ex) {

ex.printStackTrace();

}

}

}

class AugmentedView extends View {

MixView app;

public AugmentedView(Context context) {

super(context);

try {

app = (MixView) context;

app.killOnError();

} catch (Exception ex) {

app.doError(ex);

}

}

@Override

protected void onDraw(Canvas canvas) {

try {

app.killOnError();

MixView.getdWindow().setWidth(canvas.getWidth());

MixView.getdWindow().setHeight(canvas.getHeight());

MixView.getdWindow().setCanvas(canvas);

if (!MixView.getDataView().isInited()) {

MixView.getDataView().init(MixView.getdWindow().getWidth(),

MixView.getdWindow().getHeight());

}

MixView.getDataView().draw(MixView.getdWindow());

} catch (Exception ex) {

app.doError(ex);

}

}

}

public class DataHandler extends MixView {

// complete marker list

private List<Marker> markerList = new ArrayList<Marker>();

public void addMarkers(List<Marker> markers) {

for(Marker ma:markers) {

if(!markerList.contains(ma))

markerList.add(ma);

}

}

public void sortMarkerList() {

Collections.sort(markerList);

}

public void updateDistances(Location location) {

for(Marker ma: markerList) {

float[] dist=new float[3];

Location.distanceBetween(ma.getLatitude(), ma.getLongitude(), location.getLatitude(), location.getLongitude(), dist);

ma.setDistance(dist[0]);

}

}

public void updateActivationStatus(MixContext mixContext) {

Hashtable<Class, Integer> map = new Hashtable<Class, Integer>();

for(Marker ma: markerList) {

Class<? extends Marker> mClass=ma.getClass();

map.put(mClass, (map.get(mClass)!=null)?map.get(mClass)+1:1);

boolean belowMax = (map.get(mClass) <= ma.getMaxObjects());

//boolean dataSourceSelected = mixContext.isDataSourceSelected(ma.getDatasource());

ma.setActive((belowMax));

}

}

public void onLocationChanged(Location location) {

updateDistances(location);

sortMarkerList();

for(Marker ma: markerList) {

ma.update(location);

}

}

/\*\*

\* @deprecated Nobody should get direct access to the list

\*/

public List<Marker> getMarkerList() {

return markerList;

}

/\*\*

\* @deprecated Nobody should get direct access to the list

\*/

public void setMarkerList(List<Marker> markerList) {

this.markerList = markerList;

}

public int getMarkerCount() {

return markerList.size();

}

public Marker getMarker(int index) {

return markerList.get(index);

}

}