**Лекция 12 Место проведения**

В этой лекции обсуждается **локализация** : процесс определения *местоположения* . Это особенно важно для Android, который в первую очередь предназначен как операционная система для *мобильных* устройств. Что делает телефоны и планшеты особенными и отличает их от настольных компьютеров, так это то, что они могут перемещаться и действительно перемещаются. И эта мобильность означает, что положение и местоположение устройства могут иметь *большое значение для того, как они используются; это основная часть того, что отличает функциональность приложений Android от любого другого компьютерного приложения. Действительно: локализация дает приложениям возможность создавать новые виды пользовательского опыта и настраивать свою функциональность в соответствии с контекстом* пользователя , поддерживая [контекстно-зависимые](https://en.wikipedia.org/wiki/Context_awareness) приложения.

* Классический пример контекстной осведомленности — это когда программное обеспечение определяет, находитесь ли вы дома, в офисе или в автобусе, и соответствующим образом меняет свои действия.
	+ Фактически, одним из победителей *первого* конкурса разработчиков Android (2008) стало приложение [Ecorio](http://web.archive.org/web/20100209012355/http%3A/www.ecorio.org/) , которое определяло, едете ли вы на машине, идете пешком или едете на автобусе, и на основе этого рассчитывало ваш углеродный след.
* Обратите внимание, что акцент на осведомлённости о контексте исходит из [повсеместных вычислений](https://en.wikipedia.org/wiki/Ubiquitous_computing) — дисциплины, которая рассматривает технологии, которые *распространены повсеместно* или везде, до такой степени, что они «вписываются в окружающую среду». Такова точка зрения автора на то, почему важна разработка телефонов: чтобы вы могли выполнять вычисления, не задумываясь об этом.

[Я настоятельно рекомендую вам прочитать оригинальную статью](http://dx.doi.org/10.1145/329124.329126) Марка Вайзера в Scientific American 1991 года . Это изящное видение, которое является основополагающим для многих современных исследований мобильных систем. Оно знаменует собой «официальное» начало области Ubicomp.

Короче говоря: локализация может дать нам информацию о ситуации пользователя (хотя местоположение мобильного телефона [не обязательно](http://dx.doi.org/10.1007/11853565_8) является косвенным показателем местоположения пользователя).

**12.1 Методы локализации**

Исследователи Ubicomp разрабатывают системы локализации уже *много лет* . Классическим примером является [обзорная статья](http://dx.doi.org/10.1109/2.940014) Джеффа Хайтауэра (который получал докторскую степень на кафедре CSE в Вашингтонском университете!)

* Ранний пример: система *Active Badge* (AT&T) имела именной бейдж, который излучал инфракрасное сообщение, которое улавливалось датчиками в комнате, чтобы определить, где находится владелец! Это точность на уровне комнаты, но улучшения и *триангуляция* (вычисление углов относительно того, что вы видите) снизили ее до *10 см* . Однако эта система требовала большой инфраструктуры для встраивания в конкретную комнату.

С Android мы обычно заинтересованы в более общей локализации. Мобильные устройства используют несколько различных видов локализации (как независимо, так и вместе).

**GPS**

**GPS** — это самая распространенная технология локализации общего назначения, и именно ее представляет себе большинство людей, когда они думают о локализации. [GPS](https://en.wikipedia.org/wiki/Global_Positioning_System) означает «Глобальная система позиционирования» — и да, она может работать в любой точке земного шара.

Функциональность GPS зависит от спутников: 24 спутника на высокой орбите (не геосинхронной) вокруг Земли. Спутники распределены таким образом, что от 4 до 12 из них видны из любой точки Земли в любое время, и их местоположение известно с высокой точностью. Каждый из этих спутников оснащен атомными синхронизированными часами, которые «тикают» каждую наносекунду. При каждом такте спутник передает свое текущее время и местоположение. Вы можете думать о них как *о очень* громких будильниках.

То, что вы называете «GPS» или «устройством GPS» в вашем телефоне (или в машине, или в часах), на самом деле является ***приемником****GPS* . Он способен прослушивать сообщения, передаваемые этими спутниками, и определять свое (устройства) местоположение на основе этой информации.

Сначала приемник вычисляет *время прибытия* (TOA) на основе своих собственных часов и сравнения временных кодов со спутников. Затем он использует объявленное *время передачи* (TOT; то, что кричал спутник) для вычисления [времени полета](https://en.wikipedia.org/wiki/Time_of_flight) , или сколько времени потребовалось, чтобы сообщение спутника достигло приемника. Поскольку эти сообщения отправляются (в основном) со скоростью света, время *полета* эквивалентно расстоянию от спутника!

* Есть и другая работа по синхронизации, которая выполняется для того, чтобы все часы были одинаковыми, но мы не будем вдаваться в подробности.

И как только у него появятся расстояния до спутников, приемник может использовать [трилатерацию](https://en.wikipedia.org/wiki/Trilateration) для определения своего положения на основе спутников, которые он «видит». (Трилатерация похожа на триангуляцию, но основана на измерении расстояний, а не углов. По сути, вы строите три сферы заданных радиусов, а затем смотрите, где они пересекаются).

Точность GPS обычно составляет около 5 метров (15 футов); однако, многократно вычисляя положение приемника (поскольку спутники тикают каждую наносекунду), мы можем использовать *дифференциальное позиционирование* для экстраполяции положения с еще большей точностью, увеличивая точность до менее 1 метра! Частично это то, как Google может определить, где вы идете.

Хотя GPS вездесущ, масштабируем и достаточно точен, у него есть некоторые ограничения. Самая большая проблема с GPS заключается в том, что вам нужно иметь возможность видеть спутники! Это означает, что GPS часто не работает в помещениях, так как многие стены зданий блокируют сигналы. Аналогично, в густонаселенных районах (например, в центре Сиэтла) здания могут отражать сигнал и сбивать расчеты TOF, что затрудняет точное определение местоположения.

* Кроме того, приемникам требуется много энергии для постоянного прослушивания спутниковых сообщений. Это означает, что использование GPS может привести к значительному сокращению срока службы батареи устройства, что особенно важно для мобильных устройств!

**Локализация вышек сотовой связи**

Но ваш телефон также может дать вам приблизительную оценку вашего местоположения даже *без* GPS. Он делает это с помощью нескольких методов, таких как использование вышек сотовой связи, которые предоставляют услуги телефонной сети. Это также известно как [**локализация GSM**](https://en.wikipedia.org/wiki/Mobile_phone_tracking#Network-based) (глобальная система мобильной связи; стандарт для сотовой связи, используемый многими поставщиками услуг). Местоположение этих вышек известно, поэтому мы можем определить местоположение на их основе несколькими способами:

* Если вы подключены к вышке, вы должны находиться в ее зоне действия. Так что это сразу дает вам некоторую меру локализации. Хотя это не будет очень точной мерой (вы можете находиться *где угодно* в пределах этой зоны).
* Если вы видите несколько вышек (что важно для целей «передачи», чтобы ваш вызов не прерывался по мере вашего перемещения), вы можете трилатерировать положение между ними (например, найти возможную область перекрытия и выбрать место в ее середине). Это может дать точность в пределах 50 м в городских районах, а большее количество вышек обеспечивает лучшую точность.

**Локализация WiFi**

Но подождите, это еще не все! Какие еще типы антенн связи есть в вашем телефоне? **WiFi** ! Поскольку WiFi стал более популярным, были предприняты попытки определить *местоположение* точек доступа WiFi, чтобы их также можно было использовать для [трилатерации и локализации](https://en.wikipedia.org/wiki/Wi-Fi_positioning_system) .

Это часто делается через краудсорсинговые базы данных, с информацией, собранной с помощью [военного вождения](https://en.wikipedia.org/wiki/Wardriving) . Военное вождение подразумевает езду с GPS-приемником и ноутбуком, и просто запись того, какие маршрутизаторы WiFi вы видите в каких местах. Затем все это компилируется в базу данных, к которой можно делать запросы — учитывая, что вы видите *эти* маршрутизаторы, где вы должны быть?

* Компания Google попала в [затруднительное положение](http://www.wired.com/2012/05/google-wifi-fcc-investigation/) из-за этого, когда она передвигалась по городу, делая снимки для Street-View.

Затем локализацию WiFi можно объединить с локализацией вышек сотовой связи, чтобы получить довольно точную оценку вашего местоположения даже без GPS.

И на самом деле Google предоставляет возможность автоматически использовать все эти различные методы, абстрагированные в один вызов метода!

Я хочу отметить, что, как и старые системы *Active Badge* , все эти системы локализации полагаются на какую-то существующую инфраструктуру: GPS требует спутников; GSM требует вышек сотовой связи, а WiFi требует базы данных маршрутизаторов. Все эти системы требуют и реагируют на окружающий мир, делая локализацию зависящей от фактического местоположения, а также от социальных и вычислительных систем!

**Представляющее местоположение**

Итак, как только у нас появится местоположение, как нам его представить?

Во-первых, обратите внимание, что существует философское различие между «местом» и «пространством». Пространство **—** это местоположение, но без каких-либо социальных коннотаций. Например, координаты GPS или декартовы xy-координаты будут указывать на «пространство». С другой стороны, **место — это что-то, что имеет социальное значение: Мэри Гейтс Холл; Вашингтонский университет; моя кухня. Пространство — это вычислительная конструкция; место — это человеческая конструкция. Когда мы говорим о локализации с помощью мобильного устройства, мы в основном будем говорить о***пространстве* . Но часто *место* — это то, что нас действительно интересует, и нам, возможно, придется конвертировать между ними (Google действительно предоставляет несколько способов конвертации между ними, например, с помощью своего [API Places](https://developers.google.com/places/) ).

Наши космические местоположения обычно указываются в виде двух координат: **широты** и **долготы** . ( **Высоту** также можно указать, но для нас это не очень важно).

* **Широта** («lat») — это ***угол*** между экваториальной плоскостью и линией, проходящей через точку и центр Земли — угол, под которым вам нужно подняться по поверхности Земли от экватора. Фактически, это мера «север/юг». Широта обычно измеряется в *градусах северной широты* , поэтому движение на юг от экватора дает отрицательную широту (хотя это может быть выражено положительно как «градусы южной широты»).
* **Долгота** («lng») — это ***угол*** между плоскостью нулевого меридиана и линией, проходящей через точку и центр Земли — угол, который вам нужно пройти по поверхности Земли от меридиана. Фактически, это мера «восток/запад». Широта измеряется в *градусах на восток* , то есть на восток от меридиана. Это означает, что западное полушарие имеет «отрицательную долготу» (хотя это можно выразить как положительные «градусы на запад»).

Например: [GPS-координаты UW](https://www.google.com/search?q=uw+gps+coordinates)[38](https://info448.github.io/location.html#fn38) — это северо-западные координаты, поэтому это будет выражено как N (положительные) и E (отрицательные).

Расстояние между градусами и милями зависит от того, где вы находитесь (особенно для долготы — кривизна Земли означает, что каждый градус имеет меньшее расстояние между собой по мере приближения к их «соединению» на полюсах). Однако, для очень грубого ощущения масштаба, на американском Северо-Западе 0,01 градуса соответствует расстоянию *около* мили (опять же: это не точное преобразование, и предназначено только для ощущения «единиц»).

**12.2 Местоположение Android**

Оставшаяся часть лекции будет посвящена обсуждению того, как реализовать приложение, которое может получить доступ к местоположению устройства. Это местоположение будет просто отображаться на данный момент; подключение местоположения к визуальному отображению (например, карте [)](https://info448.github.io/maps.html#maps) оставлено в качестве упражнения для читателя.

В этой лекции используется код, найденный по адресу <https://github.com/info448/lecture12-location> .

**Сервисы Google Play**

Для эффективного доступа к местоположению нам сначала нужно убедиться, что мы включили [Google Play Services](https://developers.google.com/android/guides/setup) . Это специальный набор библиотек (похожих на библиотеки поддержки), которые предоставляют дополнительные функции для Android. Эта функциональность будет включать в себя инструменты определения местоположения и картографии, которые нам интересны. (Большая часть этой функциональности изначально была встроена в ядро ​​Android, но с тех пор Google переместила ее в отдельное приложение, которое можно было бы легче распространять и обновлять!)

Чтобы включить библиотеку Play Services, выполните несколько шагов:

1. Убедитесь, что файл ***уровня проекта*** build.gradle включает ссылку на [репозиторий Maven от Google](https://developer.android.com/studio/build/dependencies.html#google-maven) :
2. allprojects {
3. repositories {
4. google()
5. jcenter()
6. // If you're using a version of Gradle lower than 4.1, you must instead use:
7. maven {
8. url 'https://maven.google.com'
9. }
10. }

}

*Убедитесь, что вы поместили это под allprojects, а неbuildscripts* !

1. Убедитесь, что устройство поддерживает эти сервисы (например, что это устройство Google, а не Amazon). Для эмулятора перейдите в AVD Managerи убедитесь, что *целевая* платформа включает Google APIs.
2. Измените свой build.gradleфайл так, чтобы вы могли получить доступ к классам Location. В файле ***уровня модуля*** build.gradle , под dependenciesadd

implementation 'com.google.android.gms:play-services-location:16.0.0'

Это загрузит службы определения местоположения (но не другие службы воспроизведения, которые занимают дополнительное место и могут потребовать дополнительных ключей API). Обратите внимание, что вы можете указать другую версию служб, если она больше, чем 11.6.0.

Кроме того, вам нужно будет запросить разрешение на доступ к местоположению устройства. Мы можем запросить два уровня разрешений: ACCESS\_COARSE\_LOCATION(для точности уровня GSM/WiFi) и ACCESS\_FINE\_LOCATION(для точности уровня GPS). Мы воспользуемся последним, поскольку нам нужна точность уровня GPS.

Это **опасное** разрешение, поэтому нам нужно убедиться, что мы запрашиваем разрешение во время выполнения! Подробности смотрите [в лекции о разрешениях .](https://info448.github.io/files-and-permissions.html#permissions)

Мы собираемся использовать Google Play Services для доступа к местоположению устройства. API Google предоставляют хороший набор методов для [доступа к местоположению](https://developer.android.com/training/location/retrieve-current.html) (без необходимости указывать источник этой локализации, GPS или GSM), и это рекомендуемый API для использования.

* Существует встроенный android.locationAPI (например, для устройств Android на базе сторонних продуктов), но его использование не рекомендуется и оно сложнее в использовании.

**Доступ к местоположению**

Google предоставляет доступ к местоположению через [FusedLocationProviderClient](https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderClient). Это «унифицированный» интерфейс для доступа к местоположению. Он объединяет все различные способы получения местоположения, предоставляя тот, который лучше всего подходит для наших конкретных потребностей. Вы можете думать о нем как о «обертке» вокруг более подробных служб определения местоположения.

* Это позволит нам указать на высоком уровне, хотим ли мы пожертвовать точностью ради энергопотребления, вместо того, чтобы нам было нужно явно это указывать. И это будет принимать решения о том, как лучше всего получить местоположение, учитывая наши заявленные потребности и другую контекстную информацию.
* Получить доступ к (одиночному) клиенту можно, вызвав метод класса LocationServices:

**val** fusedLocationClient = LocationServices.getFusedLocationProviderClient(**this**)

Мы собираемся указать это «высокоуровневое» требование с помощью объекта [39](https://info448.github.io/location.html#fn39) , который представляет детали нашего запроса (например, как мы хотим, чтобы наш телефон искал свое местоположение).[LocationRequest](https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest)

*//java*

LocationRequest request = **new** LocationRequest();

request.setInterval(10000);

request.setFastestInterval(5000);

request.setPriority(LocationRequest.PRIORITY\_HIGH\_ACCURACY);

*//kotlin*

**val** locationRequest = LocationRequest().apply {

 interval = 10000

 fastestInterval = 5000

 priority = LocationRequest.PRIORITY\_HIGH\_ACCURACY

}

* Мы создаем объект, затем указываем «интервал», который мы хотим проверять на наличие обновлений. Мы также можем указать «самый быстрый» интервал, который является максимальной частотой, с которой мы хотим получать обновления (предполагая, что они доступны). Это немного похоже на минимум и максимум. 5–10 секунд хорошо для навигации в реальном времени. Подробнее см. в разделе [Настройка запроса местоположения .](https://developer.android.com/training/location/change-location-settings#location-request)

В качестве альтернативы можно получить доступ только к [текущему местоположению](https://developer.android.com/training/location/retrieve-current) , если вам не нужны регулярные обновления.

* Мы также указываем [приоритет](https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest#constant-summary) , который является индикатором для FusedLocationApi о том, какой тип точности нам нужен. HIGH\_ACCURACYПо сути, это GPS (обмен мощности на точность!)

Нам также нужно будет указать [LocationCallback](https://developers.google.com/android/reference/com/google/android/gms/location/LocationCallback)объект, который представляет собой «функцию обратного вызова», которая будет выполняться при получении каждого обновления местоположения. Обычно это определяется как *анонимный объект* :

*//kotlin*

locationCallback = **object** : LocationCallback() {

 **override** **fun** onLocationResult(locationResult: LocationResult?) {

 **if**(locationResult != **null**) {

 *//do something with result!*

 }

 }

}

Обратный onLocationResult()вызов будет передан объекту, который является [«пакетной коллекцией»](https://developer.android.com/guide/topics/location/battery#batch) обновлений местоположения (что может быть полезно для экономии батареи); доступ к locationсвойству для получения *списка* объектов Location. Каждый Locationсодержит широту/долготу местоположения. Затем мы можем использовать это местоположение (например, отобразить его). Мы можем получить доступ к широте и долготе с помощью геттеров:

```java

//java

textLat.setText("" + location.getLatitude());

textLng.setText("" + location.getLongitude());

```

Наконец, вы можете отправить запрос на обновление местоположения с помощью FusedLocationProviderClient:

fusedLocationClient.requestLocationUpdates(locationRequest, locationCallback, **null**)

(Третий параметр — a [Looper](https://developer.android.com/reference/android/os/Looper)), который используется для выполнения асинхронных операций в фоновом потоке.

Перед фактической отправкой запроса проверьте разрешения на выполнение! Не забудьте реализоватьonRequestPermissionResult

**Важно отметить** , что для экономии заряда батареи вам следует прекратить *запрашивать* обновления местоположения, когда приложение не используется, например, когда оно приостановлено или остановлено:

*//kotlin*

**override** **fun** onStop() {

 **super**.onStop()

 fusedLocationClient.removeLocationUpdates(locationCallback) *//stop the updates that go to this callback*

}

Это можно проверить (даже в помещении) с помощью эмулятора. Хотя эмулятор на самом деле не имеет приемника GPS, можно задать ему «поддельное» местоположение с помощью боковой панели параметров эмулятора (откуда мы ранее отправляли SMS-сообщения). Это позволяет нам «отправлять» телефону местоположение, почти как если бы мы, люди, были приемником GPS!

* Вы можете протестировать, указав эмулятору координаты UW (47.6550 N, -122.3080 E), и наблюдать за его обновлением!
	+ Обратите внимание, что вам может потребоваться запустить Mapsприложение, чтобы убедиться, что настройки местоположения устройства включены и верны. Посмотрите [здесь,](https://developer.android.com/training/location/change-location-settings.html) как мы можем сделать запрос самостоятельно (это много повторяющегося кода, поэтому оставим это в качестве упражнения для читателя).
* Класс FusedLocationApiтакже имеет setMockLocation()метод, который позволяет программно определять местоположения (например, можно сделать кнопку, которая меняет ваше местоположение). Это может быть полезно для тестирования и отладки.

Эта функция была добавлена ​​в Google Play Services v12 и *значительно*FusedLocationProviderApi упрощает процесс определения местоположения !

1. <https://www.google.com/search?q=uw+gps+coordinates>[↩](https://info448.github.io/location.html#fnref38)
2. <https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest>[↩](https://info448.github.io/location.html#fnref39)