Геолокация

Смартфоны и другие устройства на Android могут определять своё местоположение.

Чистый Android vs Google Play Services

Работа с геолокацией – это одна из тех областей, в которой у разработчика есть выбор: использовать встроенные в Android классы, или воспользоваться возможностями Google Play Services. Google в документации подталкивает разработчика ко второму варианту, ведь классы Google Play Services предлагают более удобные и компактные пути для решения задачи. Однако использование этих классов сделает невозможным запуск программы на устройствах без Google Play Services, что сузит потенциальный круг пользователей программы.

В данном курсе рассматриваются преимущественно встроенные Android-классы, без использования сервисов Google Play.

Запрос разрешения

Приложения, использующие геолокацию, должны запрашивать у пользователя разрешения на определение местоположения. В целом процедура аналогична запросам других разрешений.

Если приложение находится на экране, или запущен сервис переднего плана, то система считает, что приложение работает на переднем плане, и если пользователь предоставил приложению соответствующее разрешение, то система даёт приложению доступ к местоположению. Кроме того, доступ к геолокации из сервиса переднего плана должен быть заявлен в манифесте:

```
<service
    android:name="MyNavigationService"
    android:foregroundServiceType="location"
    ...>
</service>
```

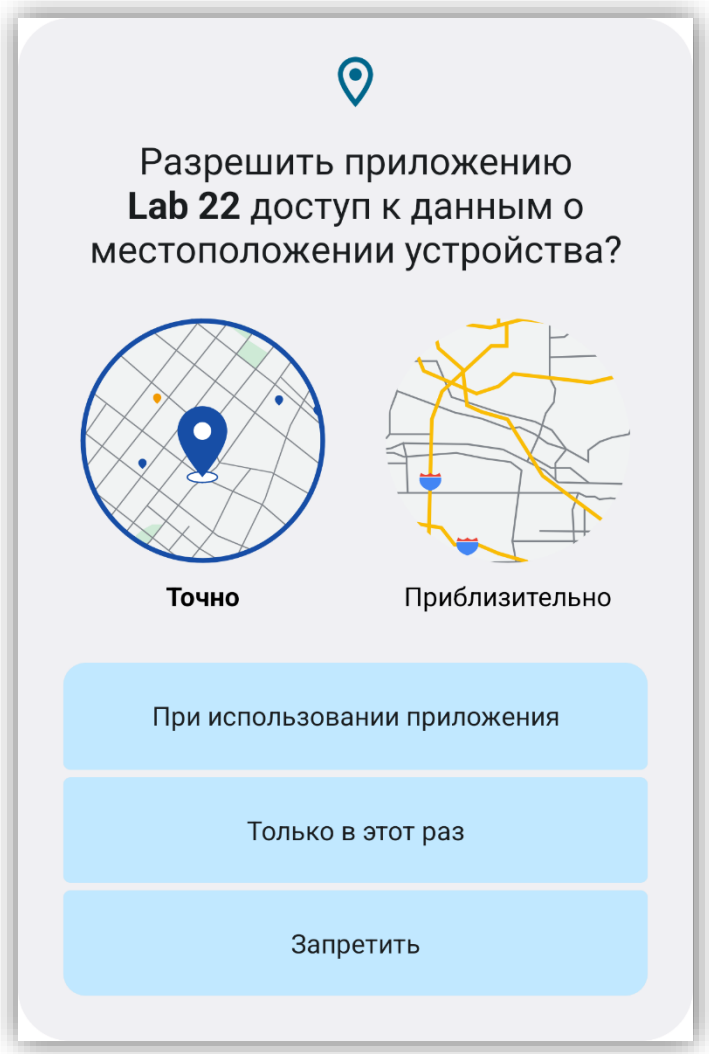
Если приложение работает в фоне (например, в виде сервиса) и желает получать информацию о местоположении пользователя, в манифест должно быть добавлено разрешение `ACCESS_BACKGROUND_LOCATION`, а для публикации такого приложения в Google Play нужно заполнить форму-запрос с разъяснениями необходимости такого функционала. Если модераторы Google не сочтут объяснения обоснованными, приложение не будет опубликовано в Google Play.

Приложение может запросить приблизительное или точное местоположение:

- `ACCESS_COARSE_LOCATION` – приблизительное местоположение описывает позицию пользователя с точностью примерно 3 квадратных километра;
- `ACCESS_FINE_LOCATION` – точное местоположение, обычно в пределах 50 метров, а иногда и в пределах нескольких метров.

Пользователь, отвечая на запрос системы, может предоставить разрешение только на приблизительное местоположение, даже если было запрошено точное местоположение. В

этом случае система будет предоставлять приблизительное местоположение, а в некоторых случаях из-за багов может вообще не предоставить доступ к местоположению. Поэтому рекомендуется запрашивать одновременно и `ACCESS_FINE_LOCATION` и `ACCESS_COARSE_LOCATION`.



Определение местоположения

После того как разрешение получено, можно приступить к определению местоположения. Для этого используется класс `LocationManager`. Создавать его не нужно, экземпляр этого класса запрашивается у системы, например, в методе `onCreate`:

```
locationManager = getSystemService(LOCATION_SERVICE) as LocationManager
```

Сам этот объект местоположение не возвращает, но его можно использовать для запуска регулярного получения координат. Для этого используется метод `requestLocationUpdates`:

```
locationManager.requestLocationUpdates(
    LocationManager.GPS_PROVIDER, 1000, 10f, locationListener)
locationManager.requestLocationUpdates(
    LocationManager.NETWORK_PROVIDER, 1000, 10f, locationListener)
```

Метод принимает следующие параметры:

- `provider` – провайдер информации: спутники (`GPS`) или сеть (`NETWORK`):
    - `GPS_PROVIDER` получает данные со спутника; хоть он и называется одинаково с американской спутниковой системой навигации GPS, но на самом деле использует данные со всех спутников, с которыми умеет работать чип в телефоне – GPS, ГЛОНАСС, BeiDou, Галилео и других;
    - `NETWORK_PROVIDER` определяет местоположение по доступным мобильным сетям: метод триангуляции позволяет определить координаты по мощности сигнала с трёх ближайших вышек связи.
- `minTime` – минимальный интервал получения обновлений местоположения, обновления не будут присылаться чаще чем этот интервал.
- `minDistance` – минимальное расстояние, на которое должен переместиться смартфон, чтобы было вызвано обновление (10 метров). Если этот параметр равен 0, то он не учитывается.
- `listener` – ссылка на объект-слушатель, который будет принимать информацию при изменении местоположения или статуса сети.

Как видно из примера, можно запрашивать обновления сразу от двух провайдеров местоположения: и от спутника, и от сети. Информация от обоих провайдеров будет поступать к одному слушателю.

Слушатель представляет собой объект типа `LocationListener`, он должен содержать метод `onLocationChanged`, который и получает новые координаты:

```
private val locationListener = object : LocationListener {
    override fun onLocationChanged(location: Location) {
        // Что-то делаем с координатами location.latitude и location.longitude
    }
}
```

Если необходимость в определении местоположения отпала, то нужно отменить регулярное получение координат с помощью метода `removeUpdates`:

```
locationManager.removeUpdates(locationListener)
```

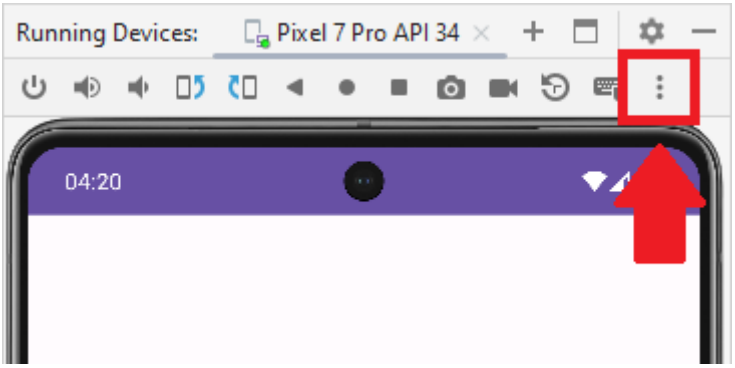
Этот метод в качестве параметра принимает объект-слушатель, который ранее подключался в методе `requestLocationUpdates`. Будут отменены все запросы, связанные с этим слушателем, то есть нет необходимости отдельно отменять прослушивание провайдера `GPS` и отдельно `NETWORK`.

Особо следует отметить, что определение местоположения – это довольно ресурсоёмкая операция, которая при постоянном использовании расходует аккумулятор и уменьшает время работы устройства. Поэтому если речь идёт не о фоновом сервисе, а о приложении переднего плана, то рекомендуется использовать определение местоположения только при нахождении приложения на переднем плане – например, подключать прослушивание в методе `onResume` и отключать его в методе `onPause`.

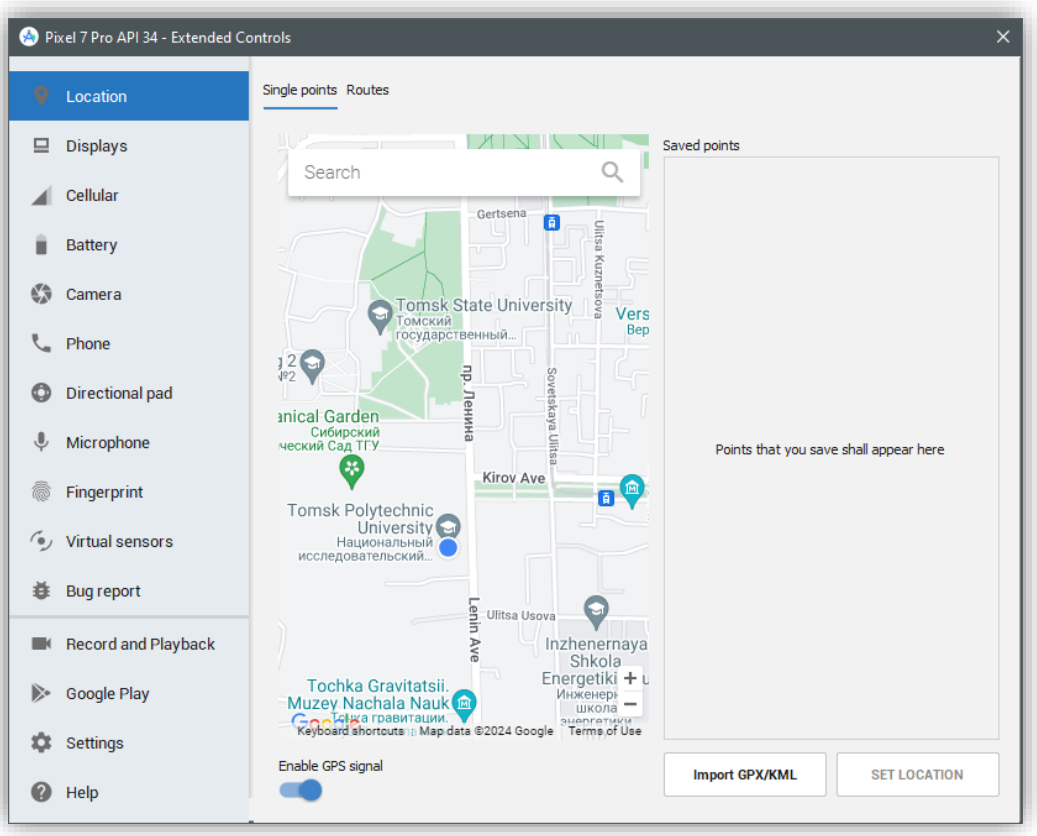
Тестирование приложения

Если программа запущена на физическом устройстве, то для получения координат устройство должно находиться в месте, где оно может получать информацию со спутников. Как правило, в зданиях железобетонные конструкции не пропускают спутниковый сигнал, поэтому для получения координат нужно подойти к окну и подождать некоторое время.

Если программа выполняется в эмуляторе на виртуальном устройстве, то для задания координат нужно нажать кнопку с тремя точками в нижней части служебной панели:



Появится окно Extended Controls:



В разделе Location можно выбрать требуемую точку на карте (или найти её поиском) и нажать кнопку «SET LOCATION». В этот момент эмулируемое устройство получит новые координаты.

Если устройство в эмуляторе не получает данные о мобильной сети, это может быть связано с тем, что не завершена настройка операционной системы (в шторке уведомлений должно быть сообщение об этом). После завершения настройки данные должны появиться.

Адреса

Помимо определения текущего местоположения пользователя, часто встаёт задача определить *что* находится в месте с указанными координатами. Для этих целей используется класс `Geocoder`.

Метод `getFromLocation` возвращает массив с информацией об объектах, которые находятся по указанным координатам. До версии Android 13 использовался синхронный вызов метода – он мог занимать какое-то время, поскольку требуемая информация получается с серверов компании Google:

```
val addresses = geocoder.getFromLocation(lat, lon, 1)
```

В Android 13 такой способ был объявлен устаревшим, и теперь официально рекомендуемым является асинхронный вызов метод: выполнение потока продолжается, а когда данные готовы – они приходят в функцию обратного вызова:

```
geocoder.getFromLocation(lat, lon, 1, object : Geocoder.GeocodeListener {
    override fun onGeocode(addresses: MutableList<Address>) {
        //
    }
})
```

В обоих случаях в качестве результата возвращается объект `addresses`, содержащий список найденных в данной области объектов. Для описания каждого такого объекта в классе `Address` имеется ряд свойств, например, для широты 56.484602 и долготы 84.947465 будет возвращена следующая информация:

- `countryCode` / `countryName` – информация о стране (`RU` / `Россия`)
- `postalCode` – почтовый индекс (`634034`)
- `adminArea` – административная единица, например, регион в России или штат в Америке (`Томская область`)
- `subAdminArea` – муниципальное образование – город, посёлок и т.д. (`город Томск`)
- `thoroughfare` – улица (`проспект Ленина`)
- `feature` – номер дома (`30`)

Если удалось определить какая организация располагается по данному адресу, то для неё могут быть приведены дополнительные сведения, такие как телефон (`phone`) или адрес сайта (`url`). Если какие-то составляющие адреса определить не удалось, то в соответствующих полях будет `null`, это нужно учитывать при работе с объектом `Address`.

Другая задача, которая иногда встаёт – это определение той же информации, но не по координатам, а по текстовому адресу. Например, пользователь ввёл текстовый адрес, а программе требуется уточнить его, получить точную информацию. В этом случае можно использовать метод `getFromLocationName`, который полностью аналогичен рассмотренному выше методу `getFromLocation`, однако вместо координат принимает строку с адресом:

```
getFromLocationName("город Томск, проспект Ленина, дом 30", 1)
```

Результаты, возвращаемые этим методом, также полностью аналогичны, полезными также могут оказаться и координаты, которые определяются по адресу:

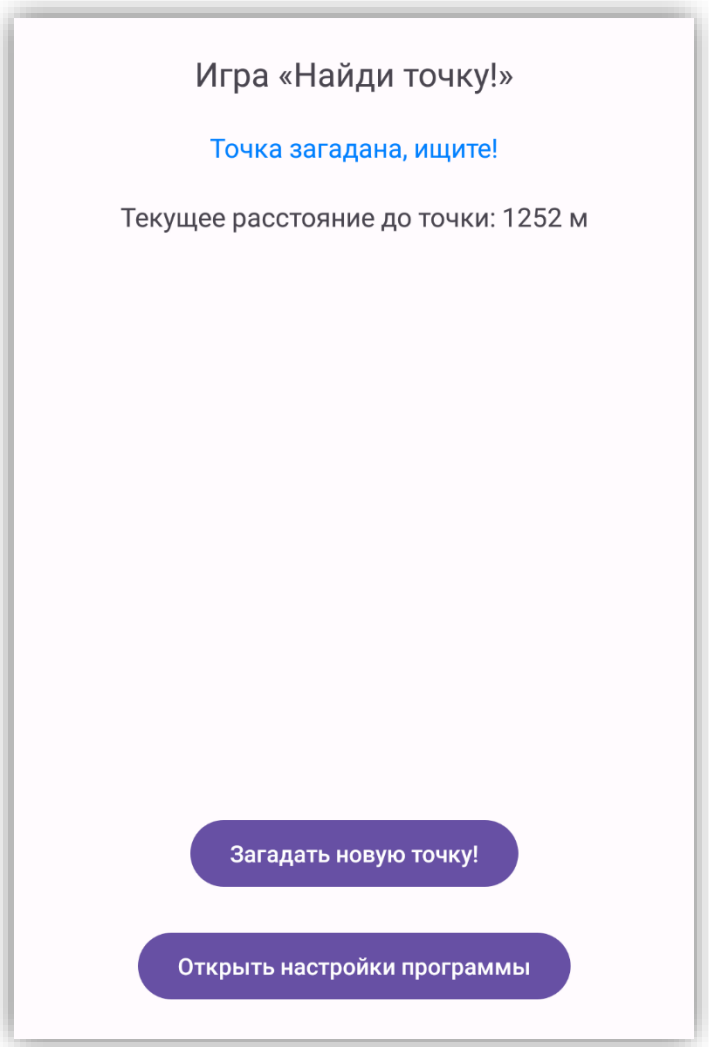
- `latitude` – широта (`56.4649206`)
- `longitude` – номер дома (`84.95040390000001`)

Вызываться метод `getFromLocationName` может как в синхронном варианте (в старых версиях Android), так и в асинхронном (начиная с Android 13):

```
val geocoder = Geocoder(this, Locale.getDefault())
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
    geocoder.getFromLocationName("адрес", 1, object : Geocoder.GeocodeListener {
        override fun onGeocode(addresses: MutableList<Address>) {
            // Делаем что-то с адресами...
        }
    })
}
else {
    val addresses = geocoder.getFromLocationName("адрес", 1)
    // Тоже делаем что-то с адресами, но в старых версиях Android...
}
```

Задание

Разработайте программу-игру «Найди точку!» Программа должна использовать средства геолокации и должна выглядеть примерно так:



При запуске программы, а также при нажатии кнопки «Загадать новую точку!» программа генерирует случайную точку в радиусе примерно 2.5 км от текущего местоположения пользователя и меняет статус на «Точка загадана, ищите!», цвет надписи синий. Когда местоположение пользователя меняется (реально или с помощью виртуального перемещения по карте в эмуляторе Android Studio) программа определяет расстояние от пользователя до загаданной точки, и выводит это расстояние на экран. Если местоположение пользователя оказывается в пределах 100 метров от загаданной точки, то статус меняется на «Ура, точка найдена!», цвет надписи зелёный.

Кнопка «Открыть настройки программы» позволяет открыть окно информации о программе, где можно посмотреть/изменить разрешения, в т. ч. доступ к геолокации.

Расстояние между точками определяется по следующей формуле:

$$D = d \cdot \arccos(\sin\varphi_a \cdot \sin\varphi_b + \cos\varphi_a \cdot \cos\varphi_b \cdot \cos(\lambda_a - \lambda_b))$$

Здесь:

- d – это радиус земного шара в метрах; поскольку форма Земли не идеально круглая, то можно взять примерное значение 6’371’000 метров;
- φ_a и λ_a – широта и долгота первой точки;
- φ_b и λ_b – широта и долгота второй точки;

Чтобы сгенерировать точку в требуемом радиусе от исходной, можно очень условно считать, что расстояние в 0.04 градуса соответствует 5 километрам – такое приближение работает в районе г. Томска, в других местах может быть другое значение. Точное попадание генерируемой точки в указанный радиус потребовало бы сложных вычислений, поэтому в задании это не требуется.