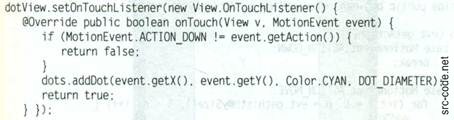
Слушание событий касания – программирование Android

Добавлено [Vadik](http://src-code.net/author/vadik/) дата August 7, 2013

[**0**](http://src-code.net/slushanie-sobytij-kasaniya-programmirovanie-android/#respond)

Как вы уже, конечно же, догадались, чтобы модифицировать демонстрационное приложение для реагирования на прикосновения к экрану, нужно просто добавить к коду обработчик касаний. Код из примера 7.8 дополняет приложение так, чтобы в DotView, там, где происходит прикосновение к экрану, появлялась голубая точка. Этот код нужно добавить в демонстрационное приложение (см. пример 7.7) в начале функции onCreate, прямо после вызова к ее родительскому методу. Обратите внимание: поскольку код, отображающий координаты х и у последней добавленной точки, подключен только к модели, он продолжает работать правильно, независимо от того, как именно вид добавляет точку.

Пример 7.8. Точки от прикосновений



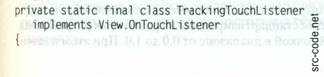
Событие MotіonEvent, переданное обработчику, имеет несколько других свойств, кроме информации о месте того касания, которое вызвало появление этой точки. Из примера видно, что в нем также содержится информация о типе события. Таких типов четыре: DOWN, UP, MOVE или CANCEL. Обычное касание генерирует одно событие DOWN и одно событие UP. Прикосновение и перетаскивание генерирует событие DOWN, несколько событий MOVE и, наконец, событие UP.

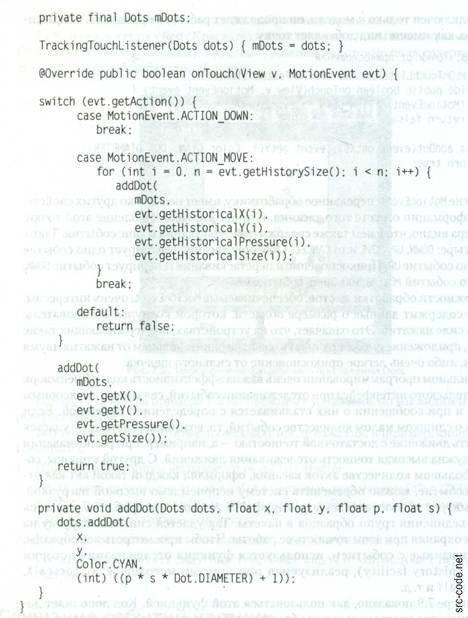
Возможности обработки жестов, обеспечиваемые MotіonEvent, очень интересны. Событие содержит данные о размере области, которой коснулся пользователь, а также о силе нажатия. Это означает, что на устройствах, поддерживающих такие функции, приложение сможет отличать нажатие одним пальцем от нажатия двумя пальцами, либо очень легкое прикосновение от сильного щелчка.

В мобильном программировании очень важна эффективность кода. Фреймворк пользовательского интерфейса при отслеживании событий, связанных с сенсорным экраном, и при сообщении о них сталкивается с определенной дилеммой. Если сообщать о слишком малом количестве событий, то, возможно, системе не удастся проследить движение с достаточной точностью – а, например, при распознавании почерка нужна высокая точность отслеживания движений. С другой стороны, сообщая о большом количестве актов касания, оформляя каждый такой акт как отдельное событие, можно обременить систему неприемлемо высокой нагрузкой. Во фреймворке пользовательского интерфейса Android эта проблема решается путем объединения групп образцов в пакеты. Так удается снизить нагрузку на систему, сохраняя при этом точность ее работы. Чтобы просмотреть все образцы, ассоциированные с событием, используется функция отслеживания истории событий (history facility), реализуемая при помощи методов getHistorical X, getHistoricalY и т. д.

В примере 7.9 показано, как пользоваться этой функцией. Код дополняет демонстрационную программу таким образом, чтобы можно было отслеживать жесты пользователя, который касается сенсорного экрана. Фреймворк передает образцы координат х и у методу onTouch объекта, установленного как OnTouchListener для DotView. Во всех полученных координатах этот метод отображает голубую точку.

Пример 7.9. Отслеживание движений

ц



На рис. 7.6 показано, как могла бы выглядеть дополненная версия приложения после нескольких щелчков и перетаскиваний.

В данной реализации для определения диаметра точки, которую следует отобразить, используются показатели размера той площади, к которой приложено давление, и силы этого давления. К сожалению, эмулятор Android не имитирует площади и силы нажатия, поэтому в нем все точки имеют одинаковый диаметр. Значения площади и силы нажатия стандартизированы на различных устройствах и имеют вид чисел с плавающей точкой в диапазоне от 0,0 до 1,0. При этом в зависимости от калибровки экрана возможны значения, превышающие 1,0. Эмулятор всегда воспринимает силу и площадь нажатия как равные нулю, то есть имеющие минимальное значение.

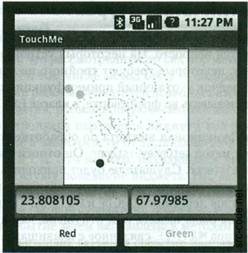


Рис. 7.6. Результат, достигаемый после продолжительной работы демонстрационной программы Dots

Цикл обработки событий ACTI0N\_M0VE работает с событиями из зафиксированной истории, собранными в пакеты. Если касания сменяются быстрее, чем фреймворк может об этом сообщать, то он объединяет их в единое событие. Метод getHi storySize, относящийся к MotіonEvent, возвращает количество образцов в пакете, а различные методы getHistory дают информацию о специфике подсобытий.

Устройства с трекболами (шариковыми манипуляторами) также генерируют события MotіonEvent при перемещении трекбола. Эти события подобны тем, что возникают при касаниях сенсорного экрана, но обработка их происходит иначе. События трекбола MotionEvent передаются в вид View при вызове метода dispatchTrackbal 1 Event, а не dispatchTouchEvent, применяемого для обработки касаний. Притом что dispatchTrackbal 1 Event передает событие методу onTrackbal 1 Event, он перед этим не передает данное событие слушателю! Генерируемые трекболом события MotionEvent не только невидимы для обычного механизма обработки касаний, более того – для отклика на них виджет должен создать подкласс от View и переопределить метод onTrackbal Event.

Характерные особенности обработки событий MotionEvent, генерируемых трекболом, этим не ограничиваются. Если эти события не потребляются (об этом процессе – чуть ниже), то превращаются в события крестовины. Это вполне объяснимо, ведь на большинстве устройств есть или трекбол, или крестовина, а вот одновременно эти элементы почти не встречаются. Без такого преобразования было бы невозможно генерировать события крестовины на устройстве, где есть только трекбол. Конечно, все это подразумевает, что приложение, обрабатывающее события трекбола, должно делать это настолько аккуратно, чтобы не нарушить преобразования.

Источник: Android. Программирование на Java для нового поколения мобильных устройств