12. Лабораторная работа № 6 «Улучшение User Expirience/Usability с помощью gestures»

12.1. Цель лабораторной работы

Демонстрация процесса разработки практического приложения.

12.2. Инструкция по выполнению лабораторной работы

Основное устройство ввода на планшетном компьютере – сенсорный экран, он заменяет и «мышь», и клавиатуру. Однако его возможности гораздо шире, чем у «мыши», и их можно использовать для того, чтобы сделать интерфейс максимально удобным для пользователя. Одним из способов это сделать является создание и использование `gestures` (жестов).

Для этого нам будут необходимы базовые знания языка C++, знакомство с основами фреймворка Qt и библиотекой MeeGoTouch. Также необходим планшет или смартфон с сенсорным экраном, работающий под управлением MeeGo. Мы использовали MeeGo 1.1.99 Tablet на планшете 3Q TU1102T. На эмуляторе или виртуальной машине поддержки сенсорного экрана скорее всего не будет.

`Gesture` (жест) – фиксированная комбинация прикосновений пользователя к сенсорному экрану. Вообще говоря, для управления компьютером со стороны человека жесты могут быть использованы не только посредством сенсорного экрана. Сейчас существуют системы, способные выделять и опознавать обычные жесты человека с помощью видеокамер, разрабатывается «невидимые» клавиатуры, когда прикосновение к поверхности, например, стола в

определенном месте воспринимается компьютером как нажатие соответствующей клавиши. Давно известны «жесты» курсором мышки на экране обычного компьютера. Далее в этой работе мы будем употреблять термин gesture только для работы с сенсорным экраном.

`Gesture` – это не просто движение пальца по сенсорному экрану, а движение в какой-то мере формализованное, которое может запомнить пользователь и распознать система. Благодаря этому у пользователя складываются образцы поведения, накапливается опыт общения с устройством. С этой точки зрения наиболее выигрышны gesture, которые встроены в систему изначально и используются многими программистами во многих приложения. В то же время для отдельных приложений программист вполне может захотеть использовать возможности сенсорного экрана по максимуму и создать свою собственную gesture.

Таблица 1.

Gesture	Действие пользователя	Класс *Qt*
`Pan`	Прикоснуться, перетащить, отпустить	`QPanGesture`
`Pinch`	Прикоснуться двумя пальцами, переместить их и отпусить	`QPinchGesture`
`Swipe`	Прикоснуться, быстро перетащить, отпустить	`QSwipeGesture`
`Tap and hol`	Прикоснуться и удерживать	`QTapAndHoldGesture`
`Tap`	Прикоснуться и отпустить	`QTapGesture`

В табл. 1 представлены все пять встроенных в *Qt* `gestures`, описание действий пользователя и классов *Qt* для них. Их также можно использовать в MeeGoTouch. Не исключены ситуации, когда одно и то же движение пользователя воспринимается системой как несколько gestures одновременно. Некоторые виджеты MeeGoTouch имеют встроенную поддержку отдельных gestures, например, подклассы `MPannableWidget` или виджеты с контекстным меню.

Обрабатывать gestures можно как для отдельного виджета, так и для контейнера, такого, например, как страница приложения. Хотя для встроенных gestures выделены специальные обработчики,

соответствующие события можно также перехватывать внутри `event(QEvent *event)` или `gestureEvent(QGestureEvent *event)`. Проще всего, однако, просто переопределить обработчики для встроенных gestures.

Чтобы разрешить виджету получать информацию о gestures, необходимо вызвать для него метод `grabGesture(Qt::SwipeGesture)`, указав желаемый тип. Для `PinchGesture` делается исключение – поскольку она использует возможности мультисенсорного экрана, ее обработка несколько сложнее и для нее также необходимо вызвать `setAcceptTouchEvents(true)`. Это особенность реализации gestures в *Qt*.

Простейший пример обработчика встроенной gesture:

```
void MyGestPage::swipeGestureEvent(QGestureEvent
*event,
                                QSwipeGesture
*gesture) {
    if (gesture->horizontalDirection() ==
QSwipeGesture::Left) {
        if (id > 0)
            ((MyApp*)qApp)->switchPage(id-1);
    } else if (gesture->horizontalDirection() ==
QSwipeGesture::Right) {
        if (id < 2)
            ((MyApp*)qApp)->switchPage(id+1);
    }
}
```

Обратите внимание на избыточность параметров. Для получения необходимой информации достаточно и `QEvent`, который надо будет привести к `QGestureEvent` и затем извлечь из него gesture, однако все это уже сделано для удобства прикладного программиста. Логика обработчика анализирует движения пользователя вдоль горизонтальной оси и в зависимости от направления движения переключает страницы приложения вперед или назад.

Направление движения привязано к физическому устройству, а не к его текущей ориентации в пространстве. То есть если перевернуть планшет «вверх ногами», то право и лево для

пользователя и gesture перестанут совпадать, несмотря на то, что пользовательский интерфейс среагирует на датчик ориентации. Этот факт надо учитывать при программировании реальных приложений.

Более сложной задачей является создание собственной gesture. Она решается в три этапа – описание распознавателя для gesture, регистрация его в системе и обработка сообщений от него.

Самое сложное – создание распознавателя, он реализуется в подклассе `QGestureRecognizer`. Как минимум необходимо переопределить метод `recognize()`:

```
Result MyGest::recognize(QGesture *gesture, QObject
*watched, QEvent *event) {
    if (event->type() == QEvent::TouchUpdate) {
        QTouchEvent *te =
static_cast<QTouchEvent*>(event);
        if (te->touchPoints().size() == 2) {
            gesture->setHotSpot(te-
>touchPoints().first().screenPos());
            return FinishGesture;
        }
    }
    return Ignore;
}
```

Приведенный пример – простейший, он считает gesture любое прикосновение к экрану двумя пальцами. К сожалению, сенсорный экран нашего планшета не поддерживает работу более двух пальцев одновременно. Более продвинутые алгоритмы могут учитывать координаты точек, другие сообщения от сенсорного экрана и т.п. Тем не менее, отметим два момента. Во-первых, в случае успешного распознавания метод возвращает `FinishGesture`, если событие не имеет к методу отношения, он возвращает Ignore. Также есть и другие возвращаемые значения, они описаны в документации. Во-вторых, gesture необходимо установить HotSpot, чтобы система могла отправить ее соответствующему виджету, и он, в свою очередь, смог ее обработать. В более сложных случаях также переопределяются методы `create()` и `reset()`.

Регистрация нового распознавателя осуществляется статическим методом

`QGestureRecognizer::registerRecognizer`, который возвращает идентификатор типа новой gesture. С помощью флага `Qt::CustomGesture` можно определить, самодельный это тип или встроенный.

Затем этот идентификатор следует использовать в обработчике, чтобы выделить события, относящиеся именно к этой gesture (естественно, специализированные обработчики уже нельзя использовать).

```
void MyGestPage::gestureEvent(QGestureEvent *event) {
    QGesture *gest = event->gesture(myGestType);
    if (gest) {
        setTitle(QString("Got %1").arg(myGestType));
        return;
    }
    MApplicationPage::gestureEvent(event);
}
```

В этом обработчике у события запрашивается gesture желаемого типа, и если он есть – выполняется какое-то действие. В противном случае вызывается метод базового класса для обработки других gesture.

Если gesture перестал быть актуальным, от него можно избавиться методом `unregisterRecognizer()`.

12.3. Задания для самостоятельной работы

1. Добавить в приложение слайдер или прогресс-бар и привязать его поведение к `PinchGesture`
2. Создать собственную gesture, которая будет распознавать начертание угловых скобок ('<' и '>') и организовать с ее помощью навигацию по страницам приложения.

12.4. Выводы

В этой лабораторной работе рассмотрен пример работы с сенсорным экраном при помощи gesture (жестов). Такой тип ввода информации позволяет легко управлять планшетным компьютером

пальцами без привлечения дополнительных устройств. Поддержка сенсорного экрана стоит на одном из первых мест при разработке приложений, т.к. возможность быстро и понятно «общаться» с устройством – это важный элемент при выборе пользователем именно вашего приложения.

Список литературы

1. Using Multi-Touch and Gestures with *Qt* http://www.slideshare.net/qtbynokia/using-multitouch-and-gestures-with-qt
2. Gestures and Multitouch http://apidocs.meego.com/1.1/platform/html/libmeegotouch/gestures.htm