Multi-touch (MT) Protocol

 Henrik Rydberg <rydberg@euromail.se>

Introduction

In order to utilize the full power of the new multi-touch devices, a way to report detailed finger data to user space is needed. This document describes the multi-touch (MT) protocol which allows kernel drivers to report details for an arbitrary number of fingers.

Usage

Anonymous finger details are sent sequentially as separate packets of ABS events. Only the ABS_MT events are recognized as part of a finger packet. The end of a packet is marked by calling the input_mt_sync() function, which generates a SYN_MT_REPORT event. This instructs the receiver to accept the data for the current finger and prepare to receive another. The end of a multi-touch transfer is marked by calling the usual input_sync() function. This instructs the receiver to act upon events accumulated since last EV_SYN/SYN_REPORT and prepare to receive a new set of events/packets.

A set of ABS_MT events with the desired properties is defined. The events are divided into categories, to allow for partial implementation. The minimum set consists of ABS_MT_POSITION_X and ABS_MT_POSITION_Y, which allows for multiple fingers to be tracked. If the device supports it, the ABS_MT_TOUCH_MAJOR and ABS_MT_WIDTH_MAJOR may be used to provide the size of the contact area and approaching finger, respectively.

The TOUCH and WIDTH parameters have a geometrical interpretation; imagine looking through a window at someone gently holding a finger against the glass. You will see two regions, one inner region consisting of the part of the finger actually touching the glass, and one outer region formed by the perimeter of the finger. The diameter of the inner region is the ABS_MT_TOUCH_MAJOR, the diameter of the outer region is ABS_MT_WIDTH_MAJOR. Now imagine the person pressing the finger harder against the glass. The inner region will increase, and in general, the ratio ABS_MT_TOUCH_MAJOR / ABS_MT_WIDTH_MAJOR, which is always smaller than unity, is related to the finger pressure. For pressure-based devices, ABS_MT_PRESSURE may be used to provide the pressure on the contact area instead.

In addition to the MAJOR parameters, the oval shape of the finger can be described by adding the MINOR parameters, such that MAJOR and MINOR are the major and minor axis of an ellipse. Finally, the orientation of the oval shape can be describe with the ORIENTATION parameter.

The ABS_MT_TOOL_TYPE may be used to specify whether the touching tool is a finger or a pen or something else. Devices with more granular information may specify general shapes as blobs, i.e., as a sequence of rectangular shapes grouped together by an ABS_MT_BLOB_ID. Finally, for the few devices

that currently support it, the ABS_MT_TRACKING_ID event may be used to report finger tracking from hardware [5].

Here is what a minimal event sequence for a two-finger touch would look like:

```
ABS_MT_POSITION_X
ABS_MT_POSITION_Y
SYN_MT_REPORT
ABS_MT_POSITION_X
ABS_MT_POSITION_Y
SYN_MT_REPORT
SYN_REPORT
```

Here is the sequence after lifting one of the fingers:

```
ABS_MT_POSITION_X
ABS_MT_POSITION_Y
SYN_MT_REPORT
SYN_REPORT
```

And here is the sequence after lifting the remaining finger:

```
SYN_MT_REPORT
SYN_REPORT
```

If the driver reports one of BTN_TOUCH or ABS_PRESSURE in addition to the ABS_MT events, the last SYN_MT_REPORT event may be omitted. Otherwise, the last SYN_REPORT will be dropped by the input core, resulting in no zero-finger event reaching userland.

Event Semantics

The word "contact" is used to describe a tool which is in direct contact with the surface. A finger, a pen or a rubber all classify as contacts.

ABS_MT_TOUCH_MAJOR

The length of the major axis of the contact. The length should be given in surface units. If the surface has an X times Y resolution, the largest possible value of ABS_MT_TOUCH_MAJOR is sqrt(X^2 + Y^2), the diagonal [4].

ABS_MT_TOUCH_MINOR

The length, in surface units, of the minor axis of the contact. If the contact is circular, this event can be omitted [4].

ABS_MT_WIDTH_MAJOR

The length, in surface units, of the major axis of the approaching tool. This should be understood as the size of the tool itself. The orientation of the contact and the approaching tool are assumed to be the same [4].

ABS_MT_WIDTH_MINOR

The length, in surface units, of the minor axis of the approaching tool. Omit if circular [4].

The above four values can be used to derive additional information about the contact. The ratio ABS_MT_TOUCH_MAJOR / ABS_MT_WIDTH_MAJOR approximates the notion of pressure. The fingers of the hand and the palm all have different characteristic widths [1].

ABS_MT_PRESSURE

The pressure, in arbitrary units, on the contact area. May be used instead of TOUCH and WIDTH for pressure-based devices or any device with a spatial signal intensity distribution.

ABS_MT_ORIENTATION

The orientation of the ellipse. The value should describe a signed quarter of a revolution clockwise around the touch center. The signed value range is arbitrary, but zero should be returned for a finger aligned along the Y axis of the surface, a negative value when finger is turned to the left, and a positive value when finger turned to the right. When completely aligned with the X axis, the range max should be returned. Orientation can be omitted if the touching object is circular, or if the information is not available in the kernel driver. Partial orientation support is possible if the device can distinguish between the two axis, but not (uniquely) any values in between. In such cases, the range of ABS_MT_ORIENTATION should be [0, 1] [4].

ABS_MT_POSITION_X

The surface X coordinate of the center of the touching ellipse.

ABS_MT_POSITION_Y

The surface Y coordinate of the center of the touching ellipse.

ABS_MT_TOOL_TYPE

The type of approaching tool. A lot of kernel drivers cannot distinguish between different tool types, such as a finger or a pen. In such cases, the event should be omitted. The protocol currently supports MT_TOOL_FINGER and MT_TOOL_PEN [2].

ABS_MT_BLOB_ID

The BLOB_ID groups several packets together into one arbitrarily shaped contact. This is a low-level anonymous grouping, and should not be confused with the high-level trackingID [5]. Most kernel drivers will not have blob capability, and can safely omit the event.

ABS_MT_TRACKING_ID

The TRACKING_ID identifies an initiated contact throughout its life cycle [5]. There are currently only a few devices that support it, so this event should normally be omitted.

Event Computation

The flora of different hardware unavoidably leads to some devices fitting better to the MT protocol than others. To simplify and unify the mapping, this section gives recipes for how to compute certain events.

For devices reporting contacts as rectangular shapes, signed orientation cannot be obtained. Assuming X and Y are the lengths of the sides of the touching rectangle, here is a simple formula that retains the most information possible:

```
ABS_MT_TOUCH_MAJOR := max(X, Y)
ABS_MT_TOUCH_MINOR := min(X, Y)
ABS_MT_ORIENTATION := bool(X > Y)
```

The range of ABS_MT_ORIENTATION should be set to [0, 1], to indicate that the device can distinguish between a finger along the Y axis (0) and a finger along the X axis (1).

Finger Tracking

The kernel driver should generate an arbitrary enumeration of the set of anonymous contacts currently on the surface. The order in which the packets appear in the event stream is not important.

The process of finger tracking, i.e., to assign a unique trackingID to each initiated contact on the surface, is left to user space; preferably the multi-touch X driver [3]. In that driver, the trackingID stays the same and unique until the contact vanishes (when the finger leaves the surface). The problem of assigning a set of anonymous fingers to a set of identified fingers is a euclidian bipartite matching problem at each event update, and relies on a sufficiently rapid update rate.

There are a few devices that support trackingID in hardware. User space can make use of these native identifiers to reduce bandwidth and cpu usage.

Gestures

In the specific application of creating gesture events, the TOUCH and WIDTH parameters can be used to, e.g., approximate finger pressure or distinguish between index finger and thumb. With the addition of the MINOR parameters, one can also distinguish between a sweeping finger and a pointing finger, and with ORIENTATION, one can detect twisting of fingers.

Notes

In order to stay compatible with existing applications, the data

reported in a finger packet must not be recognized as single-touch
events. In addition, all finger data must bypass input filtering,
since subsequent events of the same type refer to different fingers.

The first kernel driver to utilize the MT protocol is the bcm5974 driver,
where examples can be found.

[1] With the extension ABS_MT_APPROACH_X and ABS_MT_APPROACH_Y, the
difference between the contact position and the approaching tool position
could be used to derive tilt.
[2] The list can of course be extended.
[3] Multitouch X driver project: http://bitmath.org/code/multitouch/.
[4] See the section on event computation.
[5] See the section on finger tracking.