Clutter and Multitouch

Contents

Clutter and Multitouch

Introduction

This document explains Collabora's design about several issues related to the main UI toolkit used in Apertis: Clutter.

Multi-touch

This section describes the support for multi-touch (MT) events and gestures in the Apertis middle-ware. It will be explained which requirements Collabora will support and the general architecture of MT on Linux and X.Org.

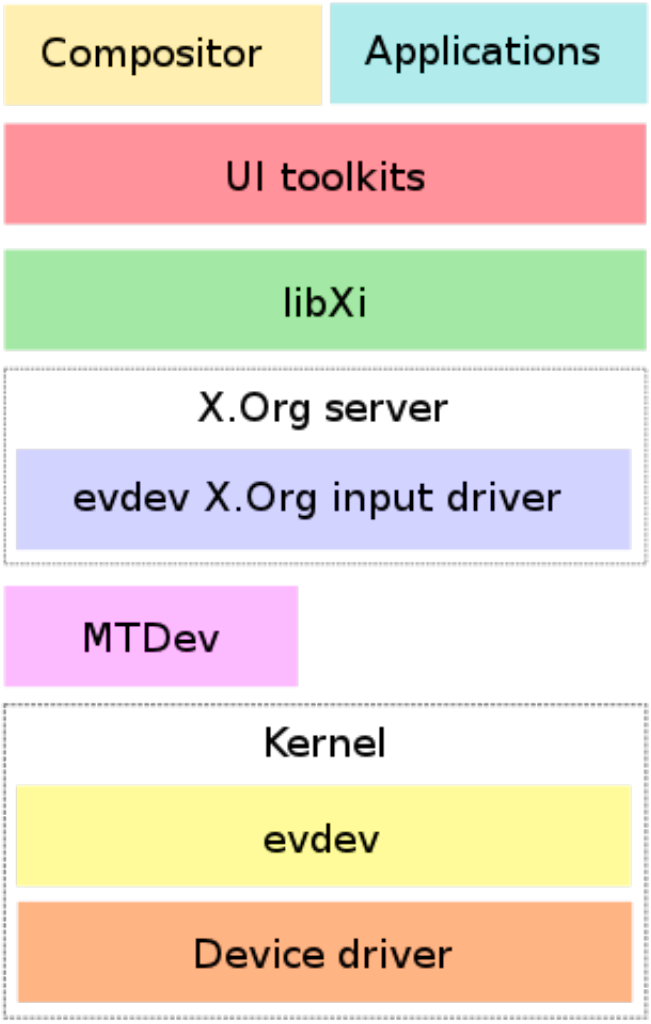
When we mention MT in this document, we refer to the ability of applications to react to multiple touch event streams at the same time. By gestures, we refer to the higher-level abstraction that groups individual touch events in a single meaningful event.

At this point of time (Q1 2012), MT and gesture functionality is implemented in several consumer products but is only starting to be available in frameworks of general availability such as X.Org and HTML. As will be explained later, this is reflected in X.Org just being released with MT functionality[1], Clutter not having MT support in a release yet, and the lack of high level gesture support in X.Org-based toolkits. In the same vein, MT is not yet standardized in the web. This design will discuss the challenges posed by these facts and ways of overcoming them.

The multi-touch stack

For the purposes of this document, the MT stack on X.Org is layered as follows:

[1] http://lists.x.org/archives/xorg-announce/2012-March/001846.html



Compositor

The compositor has to be able to react to gestures that may happen anywhere in the display. The X server usually delivers events to the window where they happen, so the compositor overrides this behavior by telling the X server that it has interest in all events regardless of where they happen (specifically, it does so by registering a passive grab on the root window).

The compositor will receive all events and decide for each whether it should be handled at this level, or let it pass through to the application to which the underlying window belongs.

For touch events, this isn't done for individual events but rather for touch sequences. A touch sequence is the series of touch events that starts when a finger is placed on the screen, and finished when it is lifted. Thus, a touch sequence belongs to a single finger, and a gesture can be composed by as many touch sequences as fingers are involved.

There is a period of time during which the compositor inspects the touch events as they come and decides whether it should handle this particular gesture, or if it should be ignored and passed to the application. If the compositor decides the later, the events that had been already delivered to the compositor will be replayed to the application.

The period of time that the compositor needs to decide whether a gesture should be handled by applications should be as small as possible, in order to not disrupt the user experience.

An application can tell the X server that it doesn't want to have to wait until the compositor has let the touch sequence pass. In that case, either the gesture shouldn't cause any visible effects in the UI, or should be reverted in case the compositor ends up deciding to handle the touch sequence by itself.

Applications

Widgets inside applications can react to MT events in a way similar to how they react to single-touch events. Additionally, some toolkits provide additional functionality that make it easier to react to gestures. Widgets can either react to a few predefined gestures (tap, panning, pinch, etc.), or they can implement their own gesture recognizers by means of the lower level MT events.

UI toolkits

As mentioned before, UI toolkits provide API for reacting to MT events and usually also functionality related to gestures. Because MT is so new to X.Org, UI toolkits based on X.Org don't implement yet everything that applications would need regarding MT and gestures, so for now additional work needs to happen at the application level.

libXi

This library allow toolkits to communicate with the XInput extension in the X.Org server. Communication with the X.Org server is asynchronous and complex, so having a higher level library simplifies this interaction.

X.Org server

The X.Org server delivers input events to each application based on the coordinates of the event and the placement of the application windows.

evdev X.Org input driver

This input driver for X.Org uses udev to discover devices and evdev to get input events from the kernel and posts them to the X.Org server.

If it is needed to apply a jitter-reduction filter and it's impossible to do so in the kernel device driver, then we recommend to patch the evdev X.Org input driver.

MTDev

For device drivers that use the legacy MT protocol as opposed to the new slots protocol, the X.Org input driver will use libmtdev to translate events from the old protocol (type A) to the new one (type B).

> See http://www.kernel.org/doc/Documentation/input/multi-touch-protocol.txt

Kernel event driver (evdev)

This kernel module will emit input events in a protocol that the evdev X.Org input driver can understand.

Kernel device driver

This kernel module is hardware-dependent and will interface with the hardware and pass input events to the evdev event driver.

Requirements

Multitouch event handing in Clutter applications

Clutter will have APIs for reacting to multi-touch input events and for recognizing gestures.

The Apertis middleware will have the support that Clutter requires to provide MT and gestures functionality, as described later in this document.

Though it is expected that Clutter will eventually provide support for recognizing a few basic gestures such as taps and panning, more advanced gestures will have to be implemented outside Clutter. In the Apertis case, recognizing additional gestures will have to be done by the applications themselves or by the SDK API.

New gestures will be developed using the gesture framework in Clutter, regardless of whether the gesture recognition is implemented in the SDK, compositor or applications. In other words, new gestures can be developed making use of the Clutter API, but the code that implements them can belong to applications, compositors or libraries. No modifications to Clutter are required to implement new gestures.

Full-screen event handing in applications

Applications should be able to handle events anywhere in the screen even if their windows don't cover the whole of it. For example, there may be windows belonging to the system such as a launcher panel or a status bar in the screen, but a gesture that starts in one of the auxiliary windows should be handled by the focused application.

Multi-touch event handling in Mutter

The compositor based on Mutter will be able to react to multi-touch input events and recognize gestures using the same Clutter API as applications.

The compositor will be able to register for MT sequences before applications get them, so it can claim ownership over them in case a system-wide gesture is detected. Even then, applications will be able to get all events that happen in their windows though care needs to be taken in case the compositor ends up claiming ownership.

Multitouch event handing in web applications

Although there are no approved specifications yet on how browsers should expose MT events to web content, some browsers have started already to provide experimental APIs and some websites are using them. Most notable are the Safari browser on iOS and websites that target specifically iPhone and iPad devices, though nowadays other WebKit-based browsers implement MT events and more websites are starting to use them.

A spec[2] is being drafted by the W3C on base on the WebKit implementation, but attention must be paid to the fact that because it's still a draft, it may change in ways that are incompatible with WebKit's implementation and the later may not always be in sync with the spec.

Support two separate touchscreens

The Apertis middleware will be able to drive two screens, each with multi-touch support.

Support out-of-screen touch events

The reactive area of the touchscreen may extend past the display and the user will be able to interact with out-of-screen UI elements.

Actors with bigger reactive area

So the UI is more tolerant to erratic touch events (caused for example by a bumpy road), some actors will be reactive past the boundaries of their representation in the screen.

[2] http://dvcs.w3.org/hg/webevents/raw-file/tip/touchevents.html

Approach

Multitouch event handing in Clutter applications

MT support in X.Org is very recent, so Collabora will have to update several components (mainly belonging to X) in the stack because Precise is not going to ship with the versions that are needed.

Clutter 1.8 had support for single-touch gestures. In 1.10, support for multi-touch event handling landed, and it is expected that for 1.12 (August 2012), support for multi-touch gestures will be added. It's also planned to have support for rotation and pinch gestures in Clutter 1.12[3].

Full-screen event handing in applications

In order for applications to be able to handle events anywhere in the screen even if their windows do not cover the whole of it, applications will have to set grabs on the other visible windows, even if they don't belong to the application process.

So in the case that the currently-focused application is displayed along with a launcher panel and a status bar, the application process should set a grab on those windows for the events that it is interested in. When another application becomes focused, the first one releases the grab and the second takes it.

In order to make sure that the second application takes the grab as soon as possible, it should try calling XIGrabTouchBegin repeatedly until it stops failing with BadAccess (this will happen once the previously focused application has removed its grab).

So the compositor can still handle system-level gestures as explained in 2.3.3, it will have to set a grab on an invisible window that is the parent of each additional window.

This is so complex because this scenario is a bit removed from the design of event delivery in X. In Wayland, as the compositor is also the display server and has total control over event delivery, it could redirect events to application windows instead of to the panels.

Multi-touch event handling in Mutter

Current releases of Mutter use XLib for event management, which doesn't support multi-touch. To Collabora's knowledge, there aren't as of yet any Mutter-based products that make use of system-wide multi-touch gestures.

> See https://wiki.gnome.org/GnomeOS/Design/Whiteboards/Touchscreen#Mutter_problems

Collabora has modified[4] Mutter to allow plugins to register for touch events and

[3] http://wiki.clutter-project.org/wiki/ClutterRoadMap#1.12
[4] http://blog.tomeuvizoso.net/2012/09/multi-touch-gestures-in-gnome-shell.html

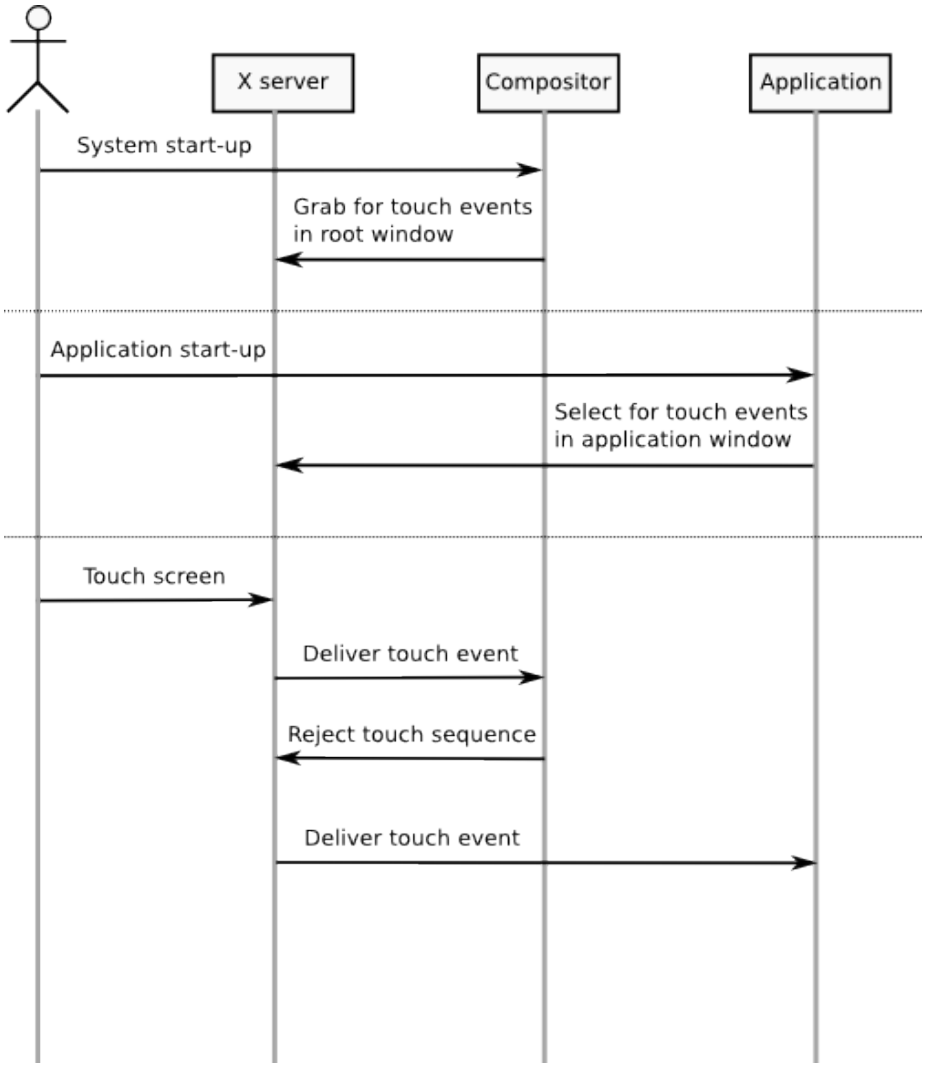
to pass them to Clutter so subclasses of ClutterGestureAction can be used.

Though applications are able to start receiving MT events even before the compositor rejects ownership, Collabora recommends that applications don't do that and instead that all efforts are directed towards having the compositor recognize gestures as fast as possible.

Otherwise, it will be very hard to avoid glitches in the user experience when the compositor decides to handle a gesture and the application already started to react to it.

By limiting system-wide gestures to those with 4 or 5 fingers as iOS 5 does (what Apple calls *multitasking gestures*), the compositor should be able to decide whether to take ownership of a MT sequence with a minimal delay and without applications having to do anything on their side. Gestures with less than 4 fingers will be considered as intended for applications and the compositor will decline ownership immediately.

This diagram illustrates how the different components interact when handling touch events:



If there is a window that gets placed on top of the others (specifically, the "busy" indicator animation) and it shouldn't get any events, it can be marked as such with the XShapeCombineRegion call, passing an empty region and the ShapeInput destKind.

See http://article.gmane.org/gmane.comp.kde.devel.kwin/19992

This way, any events that the compositor doesn't intercept will be delivered to the currently-focused application.

Multi-touch event handing in web applications

Collabora will implement the port specific bits of MT in WebKit-Clutter to en-

sure that it has state-of-the-art WebKit MT support, but won't be changing the behavior of the generic WebKit implementation nor working on the specification level.

Collabora won't be implementing any high-level gesture support because they are far from being specified and its support in browsers is very experimental.

Support two separate touchscreens

There is support in X.Org for arbitrarily matching input devices to screens, though the configuration isn't straightforward[5]. Collabora will test the simultaneous support of 2 touch-screens and produce documentation about how to configure X.Org in this regard during the development phase of the project.

Support out-of-screen touch events

Regarding out-of-screen events support, we propose writing a daemon that listens for events from the touchscreen kernel driver and that, based on its configuration, translates those to key presses for special key codes that correspond to each button.

As the X evdev driver will also get those events, it has to be configured to ignore touch events outside the screen area.

In the SDK, a wrapper around Xephyr will be provided that synthesizes key events when buttons around the Xephyr screen are pressed, simulating the ones in the real hardware.

Actors with bigger reactive area

Clutter actors tell the Clutter framework in which area of the screen they are sensitive to pointer events. This area usually matches the area that the actor uses to display itself, but the actor could choose to mark a bigger area as its reactive area.

Though the Clutter maintainer has recommended this approach, he warns that the optimization that culls actors based on their paint volumes might get in the way in this case.

Collabora will verify that this works and communicate with the upstream community in case any problem is discovered.

Risks

The DDX[6] driver provided by the hardware vendor should support having a frame-buffer that is bigger than the actual display resolution, for the out-of-screen touch events.

[5] http://www.x.org/wiki/XInputCoordinateTransformationMatrixUsage
[6] http://dri.freedesktop.org/wiki/DDX

Smooth panning

This section proposes an improvement to the kinetic scrolling functionality in Mx so that panning is smooth even when the input device's resolution is very low. This is going to affect only to Clutter applications that use MxKineticScrollView as the container of scrollable areas.

The problem with input devices with low resolution is that as the finger moves during panning, the motion events received by the application are similarly low-resolution (i.e., occurring infrequently). Given that Mx currently updates the position in the scrolled view to match the position of the finger, the panning movement appears "jumpy".

Requirements

When panning, the position in the scrolled view will smoothly interpolate to the last finger position, instead of jumping there straight away. The visual effect would be that of the scroll position lagging slightly behind the finger. The lower the resolution of the touch screen, the bigger the lag.

Approach

Collabora will rewrite the part of the MxKineticScrollView[7] widget that tracks the finger position when panning, ideally using the function mx_adjustment_interpolate[8] to animate the movement along the path between the current position and the finger position. The time function (ClutterAlpha[9]) used in the interpolation animation will be configurable, as well as the speed at which the scrolling position will follow the finger.

Risks

Upstream could decide to reject this feature when it is proposed for inclusion because of the substantial added complexity to a widget (MxKineticScrollView) that is already pretty complex. However, preliminary discussions with the Mx maintainers show that they are interested in gaining this functionality.

Another risk is Intel not funding all the work in Clutter that it has committed to. In that case, Collabora may need to do the work.

Documentation

The following items have been identified for future documentation work later in the project:

[7] http://docs.clutter-project.org/docs/mx/stable/MxKineticScrollView.html

[8] http://docs.clutter-project.org/docs/mx/stable/MxAdjustment.html#mx-adjustment-interpolate

[9] http://developer.gnome.org/clutter/stable/ClutterAlpha.html

- Add a section to the Clutter Cookbook about implementing a ClutterGestureAction subclass.
- Best practices about MT gestures in user experience, both system-wide and application gestures. Compare these guidelines with the equivalent ones in iOS and Android.
- Best practices about performance with Clutter and Mx (including how to write containers which are responsive independently of the number of children and how to drag actors across the screen).
- Best practices about using and writing reusable UI components (including Mx widgets), and explicitly these subjects (to be specified further at a later stage):
  - Panning actors
  - Finger moves outside the "active area" of an actor (e.g., moving button of timeline in video very fast)
  - Snap forward/backward to final position
  - Expand/shrink of groups (sliding)
- Best practices about writing applications in which functionality and UI are separate, so derivatives can be written by replacing the UI and without having to modify the rest of the application.

Design notes

The following items have been identified for future investigation and design work later in the project and are thus not addressed in this design:

- Support two separate touchscreens
- Support out-of-screen touch events
- Implement jitter reduction (there already has an algorithm), taking into account that the input driver may be a binary blob
- Implement zoom via pitch gestures in Google Earth without having access to its source code