X11 PlatformEvent Migration

This Document is Public

Authors: nickdiego@igalia.com
Contributors: nickdiego@igalia.com
Reviewed by: rjkroege@chromium.org, sadrul@chromium.org
Revision number: 1
Last Updated: 2019-09-04

Summary

This document contains research notes about X11 PlatformEvent Migration - which is a sub-task of larger X11 Migration to Ozone. Such transition involves, among other things, getting rid of major differences in Ozone and non-Ozone X11 ports; PlatformEvent is one of those fundamental differences. So, the goal of this task is to refactor Aura/X11 to redefine PlatformEvent as the same type used in Ozone (currently ui::Event*).

Platforms

Linux

Team

Igalia team: *msisov@igalia.com* *nickdiego@igalia.com* *adunaev@igalia.com*

Launch bug

965991

Analysis

After analysing current **`PlatformEvent`** usage in Aura/X11, the task may be broken down as follows:

- Refactor **`PlatformEventDispatcher`** implementations to handle **`ui::Event`** (basically **`DesktopWindowTreeHostX11`**);
- Convert old **`PlatformEventDispatcher`** implementations which deal with X11-specific events into **`XEventDispatcher`**s instead;
- Cleanup **`ui::XWindow`** events processing;
- Figure out and implement missing features in **`X11EventSource`** as well as on **`X11EventSourceDefault`** delegate impl
    - Make **`XEvent => ui::Event`** translation code re-usable (so that tests can use it instead of **`ui::*Event`** native event ctors);
    - Make it possible to observe and override **`XEventDispatcher`**s
- Switch Aura/X11 to **`X11EventSourceDefault`** x11 event source delegate implementation
- Fix tests

Status

An exploratory migration CL has been started as proof-of-concept of the above analysis as well as a way to identify possible unforeseen issues. Some preliminary fixes/CLs have been landed as well as missing features and issues have been found.

Open issues

Below these issues are listed and detailed so that proper solutions can be proposed and discussed.

Native Event

`ui::Event` and its derived classes have constructors taking a native event as input (i.e: **`PlatformEvent`**). Platform-specific information is then extracted from that native event and converted into that **`Event`**'s counterparts and that **`PlatformEvent`** instance is made available to other components through the **`Event::native_event()`** method. Most ports (e.g: Aura/X11, Aura/Windows, CrOS) currently use this method to access native event specific info, i.e: Aura/X11 uses it to check some **`XEvent`** flags to decide what to do in certain situations.

In Ozone **`Event::native_event()`** returns a **`ui::Event*`** (**`PlatformEvent`** definition here), which does not provide any way to retrieve the real native event object (e.g: **`XEvent*`** for X11), so the issue here is to figure out what should be done when **`XEvent`** (or other native raw events, e.g: **`MSG`** in Ozone/Windows, etc) info access is required in Ozone or in this case where Aura/X11 is being migrated to use the same **`PlatformEvent`** as Ozone.

To better understand the issue, all the **Event::native_event()** references in the code base are listed below, grouped by platform:

Aura/X11

1. ~~GtkUi: **`KeyEvent`** => **`GdkEvent`** translation (here) => *Already has a non-USE_X11 ifdef (fixed)*~~
2. ~~Helper code here and here, mostly unused? => *Dead code, remove it*.~~
3. ~~Lazily Extract DomKey at KeyEvent (here)~~
4. ~~**`ui::Event`** => **`WebInputEvent`** conversion (here)~~
5. ~~Several parts of **`ui::Event`** implementation, especially in **`*Event`** initialization~~

CrOS

1. Aura: **`WindowTargeter::FindTargetForLocatedEvent()`** (here)
2. Ash: **`ExtendedMouseWarpController::WarpMouseCursor()`** (here)
3. Ash: **`UnifiedMouseWarpController::WarpMouseCursor()`** (here)

Windows

1. Several parts of **`WebInputEvent`** conversion code (e.g: this one)
2. **`ui::KeyEvent`** construction code (e.g: here)
3. ...

Possible solutions

1. Get rid of all **`Event::native_event()`** usages
    a. Feasible for X11? Could cause regressions?
2. Refactor Ozone, creating a real PlatformEvent (something like **`OzoneEvent`**, which could be extended? by each ozone backend exposing platform-specific metadata)
    a. Suggested by sky@ at crrev.com/c/1757487
    b. Large change?
3. Extend **`ui::Event`** API to store/expose native event as a runtime rather than compile-time defined **`Event::native_event()`** API (similar to runtime properties used in Aura/Views code)
    a. Proof of concept at crrev.com/c/1757487

4. Other suggestions? rjk@ sadrul@
5. Have the native_event() be null on ozone/drm (break the CrOS points above.) Then native_event() can be used only for a "native event that's not the ui::Event". So it can be !null with ozone/x11 or ozone/wayland or ozone/win etc.
6. Make ui::Event be immutable but able to wrap another ui::Event. So methods like Event::set_location() would become `const` Event& Event::MakeWithLocation().

Chosen solution

As per sadrul@'s suggestion, a variant of approach 1 has been selected, which consists of getting rid of **`Event::native_event()`** usage in Aura/X11 code only (at least for now). Below are listed the changes needed to achieve it:

1.1. Validate and fix up missing bits of GtkUi code that converts **ui::KeyEvent** => **GdkEvent** so that it does not require **native_event()**/**XEvent** anymore;
Status: Merged
   a. **GdkEventKey::group** is currently not set (as noted by sadrul@) which possibly affects kbd layout handling; fix it up.
   Solution adopted: to pass it through Event::Properties API if it's really needed?

1.2. Reuse (if possible) fixup 1 for KeyEvent=>GdkEventKey in X11/Gtk InputMethodContext impl
Status: Merged
   a. GdkEventKey::window is currently not accessible through ui::Event API.
   Solution adopted: set Event::target() as the root aura::Window to which that event is targeted. Some modifications at Aura level were necessary, mainly because InputMethod::DispatchKeyEvent() is called in PRE_DISPATCH event processing phase.

   b. Use gdk_keymap_translate_keyboard_state() to get *keyval* as in fixup 1 ?

1.3. Factor DOM Key extraction logic out of ui::KeyEvent into platform code
Status: Merged
   a. It would not happen "lazily" anymore? Where to move it into exactly? ***X11Window****? **X11EventSource**? Figure it out*.

1.4. Get rid of **XEvent** usage in **ui::MouseEvent** => **WebMouseEvent**
Status: Merged

   a. Fix it at **X11Window**/**X11EventSource** level / at event translation time (in Ozone)? Use **Event::Properties** to pass that flag?

1.5. Refactor ui::*Event initialization code which uses native_event() to use local object/pointer instead of native_event() function;
Status: CL 1 (Merged) | CL 2 (Merged)

1.6. Fix regressions;

Follow-up tasks

1. After getting 1.1 and 1.2 it was possible to use libgtkui to support IME in Ozone/X11 (as it does not use XEvent directly anymore. Merged.
2. Move **InputMethodContext** implementation out of GtkUi into lower level layer (closer/into platform code). E.g: //ui/base, //ui/ozone/platform/** ?
   a. Perhaps postpone this until X11 is fully migrated to Ozone?
   b. While there's Aura/X11 and Ozone/X11 it should be somewhere in //ui/base ?