Support ui controls for ozone

msisov@igalia / Draft: 2020-09-21
WIP patch - https://crrev.com/c/2424316

Objective

Enable interactive_ui_tests on linux-ozone-rel bot.

Background

In the Ozone project, we have made the Chromium browser support both X11 and Wayland backends that can be chosen at runtime. Our recent effort was to make Linux builds compile ozone by default and let users enable Ozone if they want.

Our final goal is to make Ozone the default on Linux and remove the old non-Ozone/X11 path (see use_x11 gn arg in ui.gni). In order to achieve that, several steps need to be taken - ensure feature parity between ozone layer and non-ozone/x11, ensure linux-ozone-rel runs the same set of test suites that linux-rel bot runs and fix any issues we face while enabling tests.

The very first issue that is going to be fixed is the UIControlsAura implementation for Ozone.

Overview

Currently, Ozone has UIControlsOzone that is a subclass of UIControlsAura. Its purpose is to receive data, which it uses to create synthesized mouse/keyboard/touch events, parse that and send events with that data to ui::EventSource.

However, that implementation lacks some logic that is required to have all the tests in the interactive_ui_tests pass. For example, the UIControlsOzone is unable to reroute events based on a state of native capture. That is, events might need to be rerouted from one WindowTreeHost to another one based on the state of native capture including translation of location of the event in question and forwarding it further.

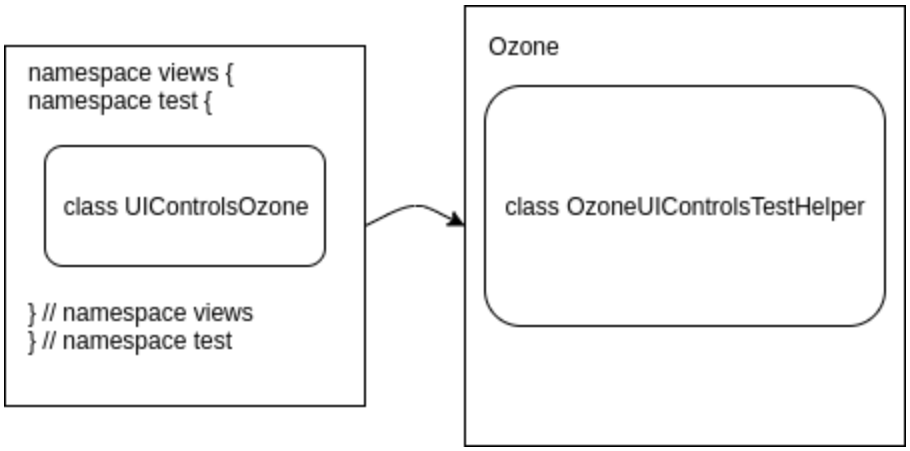
Designing and implementing that logic inside the UIControlsOzone could be a solution, but there is a drawback - the underlying PlatformWindow implementation of an ozone platform in question (let it be X11 or Wayland) will not be part of the tests. For example, both X11Window and WaylandWindow have differing logic that translates and reroutes events based on the state of native capture.

In contrast, non-Ozone/X11 uses UIControlsDesktopX11 that is able to create XEvents and send them to X11 Chrome-client via x11:SendEvent. The events are processed by X11Window (this processing includes translation of XEvents to ui::Events, checking native capture and rerouting the events based on that knowledge) and then sent further to aura via PlatformWindowDelegate for processing.

Thus, UIControls implementation for Ozone must follow the same approach.

Detailed Design

UIControlsOzone; OzoneUIControlsTestHelper



UIOzoneControlsOzone will communicate with the interface called OzoneUIControlsTestHelper that Ozone platform, which is interested in running interactive_ui_tests, can implement.

The OzoneUIControlsTestHelper mustn't be created via OzonePlatform and it must be part of the "test_support" Ozone gn target that can be used by external clients. That is, "//ui/ozone" could provide an additional "//ui/ozone:test_support" target that external clients can depend on. This is needed because of some internal test components (such as MockPlatformWindowDelegate) that ozone platforms use for tests and should not leak outside. Instead, the helper will be created via a factory method called ui::CreateOzoneUIControlsTestHelper. The factory methods will be generated by the generate_constructor_list.py script following the same approach (see CreateOzoneUIControlsTestHelper in the WIP patch) as `ClientNativePixmapFactory` uses.

This is required to ensure this code won't be part of the production code. It's also worth noting that stub implementation for platforms that won't support this factory will be provided.

The OzoneUIControlsTestHelper interface will have the following design:

```
class OzoneUIControlsTestHelper {
  // Returns current button down mask.
  virtual unsigned ButtonDownMask() const = 0;
  // Sends key press event and executes |closure| when done.
  virtual void SendKeyPressEvent(gfx::AcceleratedWidget widget,
                                 ui::KeyboardCode key,
                                 bool control,
                                 bool shift,
                                 bool alt,
                                 bool command,
                                 base::OnceClosure closure) = 0;
  // Sends mouse motion notify event and executes |closure| when done.
  virtual void SendMouseMotionNotifyEvent(gfx::AcceleratedWidget widget,
                                          const gfx::Point& mouse_loc,
                                          const gfx::Point& mouse_root_loc,
                                          base::OnceClosure closure) = 0;
  // Sends mouse event and executes |closure| when done.
  virtual void SendMouseEvent(gfx::AcceleratedWidget widget,
                              ui_controls::MouseButton type,
                              int button_state,
                              int accelerator_state,
                              const gfx::Point& mouse_loc,
                              const gfx::Point& mouse_root_loc,
                              base::OnceClosure closure) = 0;
  // Sends touch event and executes |closure| when done.
  Virtual void SendTouchEvents(int action,
                                        int id,
                                        int x,
                                        int y,
                                        base::OnceClosure task) = 0;
  // Executes closure after all pending ui events are sent.
  virtual void RunClosureAfterAllPendingUIEvents(base::OnceClosure closure) = 0;
};
```

Thus, UIControlsOzone will create OzoneUIControlsTestHelper via CreateOzoneUIControlsTestHelper method and delegate the functionality of the ui controls to ozone implementations.

Ozone and non-Ozone X11

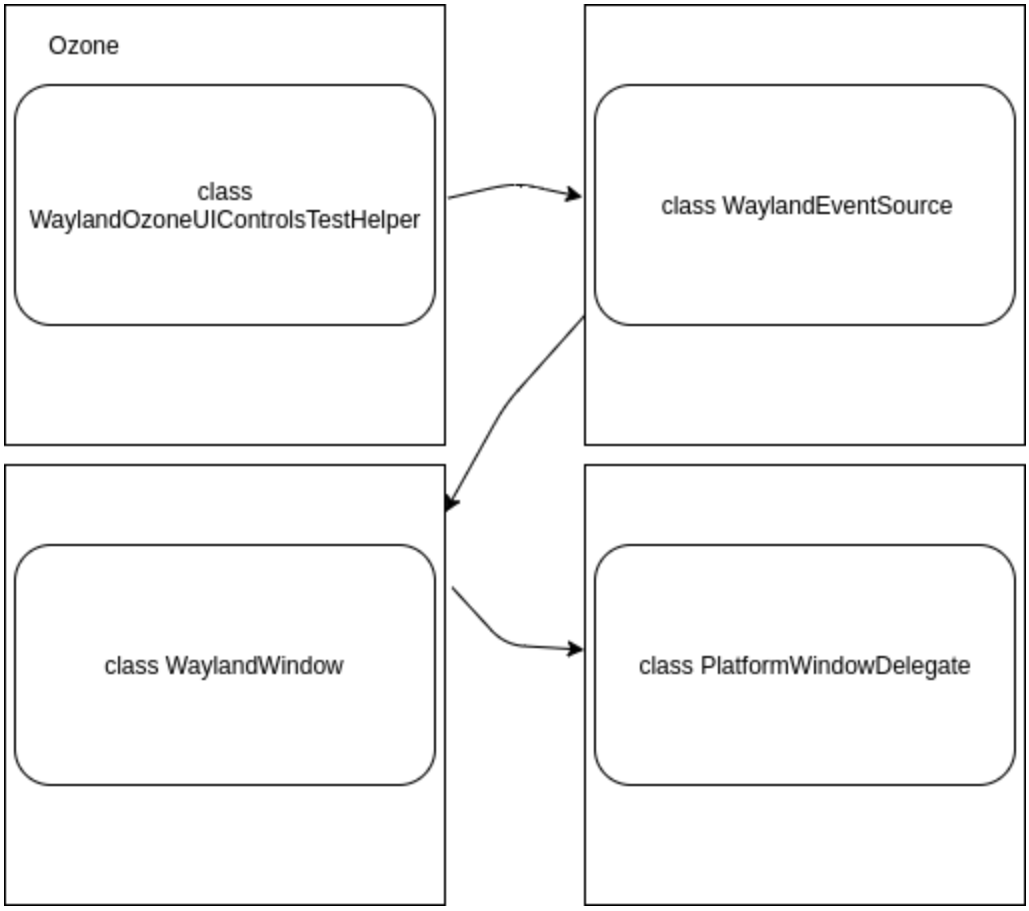
In case of Ozone/X11, a common UIControls implementation will be located at //ui/base/x, which both non-Ozone/X11 and Ozone/X11 will be able to access.

In case of non-Ozone/X11, the call flow will be the following - UIControlsDesktopX11 -> X11UIControlsTestHelper (located at //ui/base/x) that uses x11::SendEvent and sends native XEvents to the Chromium/X11 client.

In case of Ozone/X11, the call flow will be the following - UIControlsOzone -> X11OzoneUIControlsTestHelper (subclass of OzoneUIControlsTestHelper) -> X11UIControlsTestHelper (located at //ui/base/x) that uses x11::SendEvent and sends native XEvents to the Chromium/X11 client.

Wayland

In the case of Wayland, sending native events is impossible now, a library called libei is under development at the moment. This library can be considered at some point and Wayland can potentially take the same approach as X11 uses in the future, but another approach has to be used in the short term.



WaylandOzoneUIControlsTestHelper will be a subclass of OzoneUIControlsTestHelper. It will directly inject data for events into the WaylandEventSource that usually gets this kind of data from WaylandPointer, WaylandTouch, and WaylandKeyboard. The WaylandEventSource will process the data as usual and send events to WaylandWindows.

The WaylandOzoneUIControlsTestHelper will also need to manipulate the focus state of each WaylandWindow through the WaylandEventSource “emulating” WaylandPointer, WaylandTouch, and WaylandKeyboard objects.