Touchpad Pinch Zoom

Author: mcnee@chromium.org

Background

Previously, when the renderer received a touchpad pinch gesture event, it would generate a synthetic mouse wheel event which it would offer to the page. If the page did not cancel the event, the renderer would go on to apply the change to the page scale. The motivation for offering this wheel event was to allow pages to implement custom zoom behaviour in cases where the browser's native pinch zoom mechanism was not suitable (e.g. Google Maps). See issue 289887 for the discussion on offering wheel events for touchpad pinch.

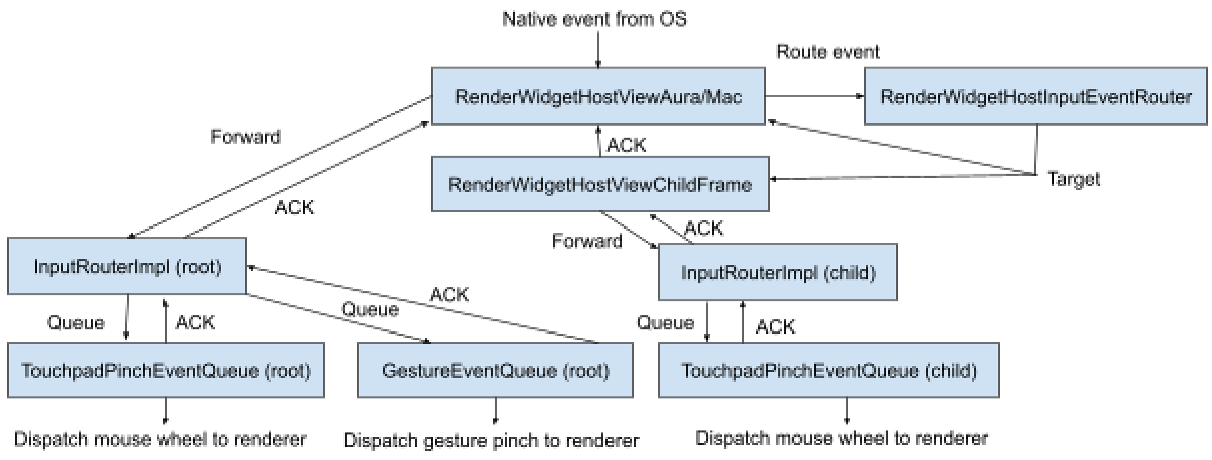
Note that for touchscreen gesture pinch events, there was no need for the renderer to generate such an event as touch events would have already been offered to the renderer before being recognized as a pinch gesture.

There are several problems with this approach to offering the synthetic wheel event. First, from a design perspective, having event generation done in the renderer is unusual. Other input methods all involve sending input events to the renderer and then generating the appropriate gesture events based on the ACKs from the renderer. Second, since the existence of a wheel handler would require the pinch events to go to the main thread, we must also handle the pinch zoom on the main thread. If it weren't for this approach, we would be able to handle pinch zoom entirely on the compositor thread. Third, this approach does not work in the presence of out-of-process iframes (OOPIFs). The gesture pinch may be routed to a child frame so that the wheel event can be offered to the frame. However, if the wheel goes unconsumed, the child has no way of updating the page scale in the main frame as it is in another renderer process.

Overview

We introduce the TouchpadPinchEventQueue to handle the sending of synthetic wheel events browser side. It accepts the touchpad gesture pinch events created from native events and sends the appropriate wheel events to the renderer. These synthetic wheel events may be consumed by the page (with preventDefault). The renderer's ACK of the event indicates to the browser whether the event was consumed. Once the renderer has acknowledged a wheel event, we offer the corresponding gesture pinch event back to the client of the TouchpadPinchEventQueue with the ACK of the wheel event. If the pinch has not been consumed by the page, the client may go on to send the actual gesture pinch event to the renderer which will perform the pinch zoom.

Design


Native event from OS
Route event
RenderWidgetHostViewAura/Mac
RenderWidgetHostInputEventRouter
Forward
ACK
RenderWidgetHostViewChildFrame
Target
ACK
Forward
ACK
InputRouterImpl (root)
InputRouterImpl (child)
ACK
Queue
Queue
ACK
Queue
ACK
TouchpadPinchEventQueue (root)
GestureEventQueue (root)
TouchpadPinchEventQueue (child)
Dispatch mouse wheel to renderer
Dispatch gesture pinch to renderer
Dispatch mouse wheel to renderer

Event Handling Flow

The event handling flow is as follows:

1. The root RenderWidgetHostView receives a native pinch event.
2. A GesturePinch Begin/Update/End event is created with needs_wheel_event = true
3. The gesture pinch event is routed to the target RWHV
4. The InputRouter receives the event and since needs_wheel_event = true, it queues the event in the TouchpadPinchEventQueue
5. When the TouchpadPinchEventQueue processes the event:
    a. If it is a GesturePinch Begin/End, immediately acknowledge the event to the RWHV as ignored, since these only represent phase information and don't correspond to a cancelable MouseWheel event.
    b. If it is a GesturePinch Update:
        i. Create a synthetic MouseWheel event and send it to the renderer
        ii. When the renderer acknowledges the MouseWheel event, acknowledge the corresponding GesturePinch Update to the RWHV
6. If the targeted RWHV is not the root RWHV, forward the ACK to the root RWHV

7. If a GesturePinch Update’s corresponding MouseWheel event was consumed or if the event is not intended to change the page scale (zoom_disabled = true), then we are done processing this event
8. Since the gesture pinch event has needs_wheel_event = true, the root RWHV resends the event with needs_wheel_event = false
9. The InputRouter receives the event and since needs_wheel_event = false, it queues the event in the GestureEventQueue
10. After the gesture pinch event is sent to the renderer and has been acknowledged, the root RWHV receives the ACK, and since needs_wheel_event = false, we are done processing this event

Event Handling in Renderer

The gesture pinch events are handled entirely on the compositor thread. The compositor thread is always capable of performing a pinch zoom (compare with scrolling where we may have scrollable areas which can only be scrolled on the main thread). Furthermore, after the browser receives an acknowledgement that a synthetic wheel has not been consumed and forwards the gesture pinch event to the renderer, we know that we will be performing a page scale change. Any wheel listeners have already seen the synthetic wheel and had the opportunity to cancel it. Hence, there is no need for the renderer to forward the gesture pinch event to the main thread. This allows us to avoid having to duplicate the pinch event handling logic between the two threads.

While the gesture pinch events themselves are handled on the compositor, we are still subject to performance issues due to the need to wait for the corresponding synthetic mouse wheel event to be acknowledged by the renderer. As an optimization, we could send the synthetic wheel events as non-blocking if the first synthetic wheel of a pinch sequence is not canceled.

Code Location

The TouchpadPinchEventQueue itself is in content/browser/renderer_host/input/. There is related code in the platform specific RenderWidgetHostViews in content/browser/renderer_host/ which handle the creation of gesture pinch events from the platform’s native events.

Test Plan

The TouchpadPinchEventQueue is covered by unit tests in content/browser/renderer_host/input/touchpad_pinch_event_queue_unittest.cc.

Browser tests have been added in content/browser/renderer_host/input/touchpad_pinch_browsertest.cc to verify the sending of synthetic wheel events to the page and the change in page scale. There are also several gesture pinch layout tests (in fast/events/pinch/ and synthetic_gestures/) that have been updated to use a compositor now that pinch events are no longer handled on the main thread.

In order to ensure touchpad pinch works with OOPIFs, the test SitePerProcessHitTestBrowserTest.TouchpadPinchOverOOPIF has been added to test that performing a touchpad pinch over an OOPIF offers the synthetic wheel events to the child and causes the page scale factor to change for the main frame.

Document History

May 23, 2018: Initial revision.
July 17, 2018: Add diagram and further describe event ACKs.