Flutter InfiniteScroll

SUMMARY

Adding InfiniteScroll support to Scrollbars

Author: Kate Lovett (@Piinks)
Go Link: flutter.dev/go/infinite-scroll
Created: 11/2021 / **Last updated:** 01/2022
Issues:

- https://github.com/flutter/flutter/issues/41434
- https://github.com/flutter/flutter/issues/25652

Prototype PR:

- https://github.com/flutter/flutter/pull/96825

OBJECTIVE

This document will lay out planned changes to Scrollbar in order to support two visual and interactive (and 1 not) behaviors when a ScrollView is infinitely long as well as when the user would like to assume the ScrollView is infinitely long for smoother scrollbar behavior.

BACKGROUND

Scrollbars represent the current position in a viewport relative to the full extent of that viewport. When a ScrollView is infinitely long, there is no way to compute this relative position. Users expect to be able to provide a scrollbar in this case that not only visually represents a position, but is still interactive, meaning it can be dragged or tapped in order to manipulate the position of the ScrollView. Infinite scroll has become a common term in referring to the behavior and representation of these types of ScrollViews, and there are a few patterns that are typically used in implementing them.

One of these is a page loading effect, where a scrollbar thumb reaching the end of a ScrollView will then load more content. This can be seen on desktop, web, and mobile in applications like Twitter or Reddit.

TODO: From comment below, if the user drags the scrollbar to the edge where it triggers adding more depth, the adjustment does not occur until the user releases the gesture. This can be seen in the case of Twitter, observation of Reddit implementation does not support dragging the scrollbar.

- View page loading infinite scroll (Desktop web - Twitter)
- View page loading infinite scroll (Mobile - Reddit)

The other is an amortized, or continuous, extent. In this case, the scrollbar thumb will smoothly travel down the scrollbar track at a gradually slower and slower rate, never reaching the actual end of the track. An example of this is the stdout of a LUCI build while it is running.

Lastly, in some cases, a scrollbar will not be included at all if the ScrollView does not have a finite extent. This can be seen in apps like Facebook, Instagram, and Pinterest. This is currently the default behavior for Flutter, rendering no scrollbar at all if the extent is infinite. This default behavior is confusing for users, PMs have reported customers have a difficult time knowing when to expect a scrollbar, and finding stackoverflow inquiries regarding scrollbars in infinite lists is not difficult to find (A, B). For these reasons, this proposal also seeks to enable a new default behavior to resolve the confusion of show-no-show scrollbars.

- View no infinite scroll (Mobile - Pinterest)

This proposal adds support for all of the above behaviors, as well as the option for finite ScrollViews to assume they are infinite in length. This last aspect has been considered as part of this design as it has the opportunity to add consistency to Scrollbar behavior in ScrollViews that lazily build variable-sized children. These types of ScrollViews estimate the maximum extent of the ScrollView, an estimate that will update as more and more children are lazily built during scrolling. This can create a jittery Scrollbar thumb, moving up and down the track slightly as the estimated extent is updated. This same behavior is seen in the wild (See Twitter example above), but the ability to opt-in to a smooth scrollbar thumb in these cases would be a nice polished experience for users that prefer it.

Glossary

- Scrollbar - indicates which portion of a ScrollView is actually visible.
- InfiniteScroll - A web design technique where, as the user scrolls down a page, more content automatically and continuously loads at the bottom,

eliminating the user's need to click to the next page. The idea behind infinite scroll is that it allows people to enjoy a frictionless browsing experience.

- ScrollBehavior - Describes how Scrollable widgets should behave.

OVERVIEW

Sample Code

```dart
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
 const MyApp({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     // This app configures the InfiniteScrollBehavior
     // using theme.
     theme: ThemeData(
       scrollbarTheme: const ScrollbarThemeData(
         infiniteBehavior: InfiniteScrollBehavior.none,
         isAlwaysShown: true,
       )
     ),
     home: const MyStatefulWidget(),
   );
 }
}

class MyStatefulWidget extends StatelessWidget {
 const MyStatefulWidget({Key? key}) : super(key: key);

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(
       title: const Text('InfiniteScrollBehavior.none'),
     ),
     body: ListView.builder(
       // itemCount: 1000,
       itemBuilder: (_, index) {
         return Text('Item $index');
       }
     )
   );
 }
}
```

This proposal adds an enum, InfiniteScrollBehavior, to

- RawScrollbar
- Scrollbar
- CupertinoScrollbar
- ScrollbarPainter
- ScrollbarTheme

So that users can configure the behavior where they would expect to in regards to other scrollbar behaviors.
The InfiniteScrollBehaviors are as listed above, allowing for page, continuous and none behaviors.

The behaviors of page and continuous rely on an assumption of scrollable extent that is updated as the user scrolls to always maintain an amount of distance along the scrollbar track.

InfiniteScrollBehavior.page

When using page behavior, the assumed scrollable extent is only updated when the scrollbar thumb approaches the end of the track, representing the currently assumed scrollable extent.
Currently, the assumed extent and the amount of appended extent is not configurable by the user. This can be exposed as part of this change, or considered in a follow-up change if it is requested by users. The PR makes mention of this here.
This behavior supports interacting (dragging & tapping) on the scrollbar and track.

View InfiniteScrollBehavior.page in Flutter

InfiniteScrollBehavior.continuous

Similar to paging, the assumed scrollable extent is updated as the user scrolls, but more regularly rather than waiting until the thumb reaches the end. This always maintains an extent ahead of the scrollbar thumb, keeping the movement smooth and consistent along the track.
The assumed leading extent is also not exposed in the proposal's current state, but is notated as a possible feature users will want.
This behavior is expected as the new default behavior for infinite ScrollViews.
This behavior supports interacting (dragging & tapping) on the scrollbar and track.

View InfiniteScrollBehavior.continuous in Flutter

InfiniteScrollBehavior.none

This would maintain the current behavior, rendering no scrollbar in infinite ScrollViews. This would no longer be the default though. Adding a Scrollbar widget should always result in there being a scrollbar. The current behavior only creates confusion.
This behavior does not support interacting (dragging & tapping) on the scrollbar and track, as there would be none to interact with.

View InfiniteScrollBehavior.none in Flutter

Another approach

Another approach for configuring InfiniteScrollBehavior was considered through the inherited ScrollBehavior all Scrollable widgets refer to. ScrollBehavior provides ScrollPhysics, defines valid PointerDeviceKinds for scrolling, and applies decorations including the Scrollbar and GlowingOverscrollIndicator to Scrollable widgets. This would reduce the footprint of the change by not adding the property to so many classes. It also would have the benefit of being accessible beyond the scope of Scrollbar widgets in case there are more uses for InfiniteScroll in the framework in the future. Ultimately, users seem to expect this to be configurable on the Scrollbar, or ScrollbarTheme, and so the above approach was chosen over this one.

Another proposal

Another contributor @xu-baolin has also proposed similar infinite functionality, PR available here. Invited to collaborate on the final solution here. :)

OPEN QUESTIONS

- Is a simulated effect sufficient for the desired user experience?
- Are there other use cases for an InfiniteScrollBehavior in the framework, such that it should be defined more broadly?
    - Currently it is defined alongside the scrollbar class in the widgets library
- Should the simulated extents be configurable by the user?
- Dynamic response to different input types
- Deferred loading for high speeds

MIGRATION PLAN

TBD
I have not run customer tests against the current proposal. Will update after initial feedback if found to be breaking.
The opportunity for this to be a breaking change lies in the new default behavior

from no scrollbar to having a scrollbar in infinite ScrollViews.