# SUMMARY

This doc describes Flutter’s updated and unified clip strategy for widgets. In short, Flutter defaults to not clip except for a few specialized widgets (e.g., ClipRect). To override the default no-clip, one shall explicitly set clipBehavior in widgets constructions.

**Author: Yuqian Li (liyuqian)**

**Go Link: flutter.dev/go/clip-behavior**

**Created:** 05/2020 / **Last updated:** 05/2020

# OBJECTIVE

The main goal of updating our clip strategy is to improve performance. Clipping is expensive, and in many cases a widget’s child content won’t exceed its boundary so such expense is wasted.

The proposed change will also give more flexibility to widgets that really need clipping: it allows specifying whether anti-aliasing is needed, or whether an offscreen layer is needed.

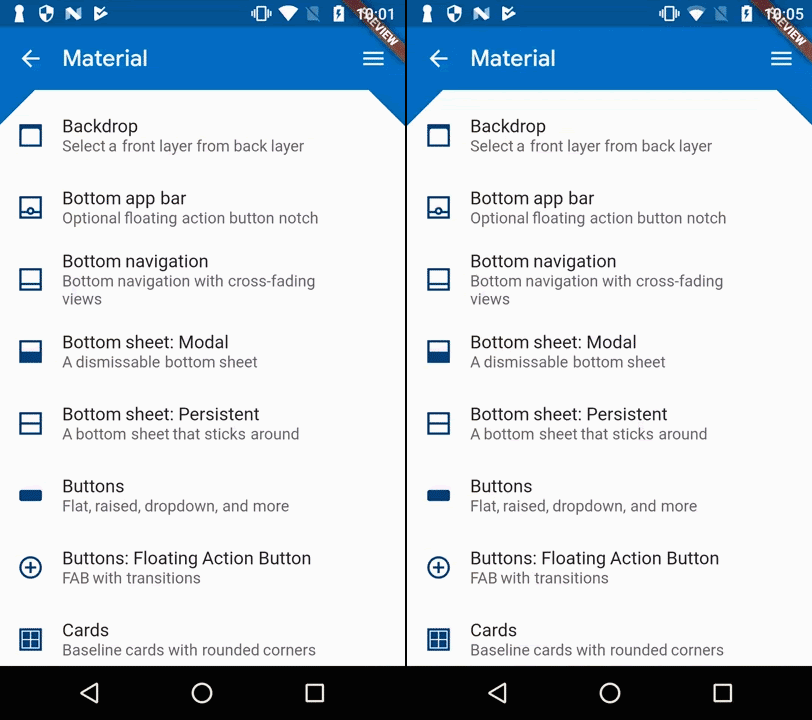
In addition to performance, this proposed change also unifies the clip control so we’ll always have the same clipBehavior configuration instead of multiple ones such as overflow and clipToSize.

# BACKGROUND

Flutter used to be slow because of clips. For example, the Flutter gallery app benchmark has an average frame rasterization time of about 35ms back in May 2018 whereas the budget for smooth 60fps rendering is 16ms. By removing unnecessary clips and their related operations, we see almost 2x speedup from 35ms/frame to 17.5ms/frame.



Here’s a comparison of transition with and without clips.



The biggest cost associated with clip at that time is that Flutter used to add a saveLayer after each clip (unless it’s a simple axis-aligned rectangle clip) to avoid the bleeding edge artifacts as described in <https://github.com/flutter/flutter/issues/18057>. Such behaviors were universal to material apps through widgets like Card, Chip, Button, etc., which will result in PhysicalShape and PhysicalModel that clip their content.

A saveLayer is especially expensive in older devices because it creates an offscreen render target, and a render target switch can sometimes cost about 1ms.

Even without a saveLayer, a clip is still expensive because it applies to all subsequent draws until it’s restored. Thus a single clip may slow down the performance on hundreds of draw operations.

In addition to performance issues, Flutter also suffered from some correctness issues as the clip was not managed and implemented in a single place. In several places, the saveLayer was inserted to the wrong place and thus it only increased the performance cost without fixing any bleeding edge artifacts.

### Glossary

* **AA, anti-alias** - A technique to reduce jigsaw artifacts and make vector graphics look smoother. A common way of doing that is by allowing a pixel to paint a fraction of the geometry’s color based on how much the geometry intersects with the pixel.
* **saveLayer** - A Skia canvas operation that not only saves the current transformation matrix and clip stack, but also allocates a new offscreen bitmap. Before restore, all future draw operations will be grouped and composited into that offscreen bitmap. When a restore is called, that bitmap is composited back as a whole.
* **PR, pull request** - A patch of code changes on Github.

# OVERVIEW

We propose to introduce a Clip enum and add a clipBehavior configuration of that enum type to all widgets that can clip. For most widgets, that clipBehavior is default to Clip.none and thus no clip is provided. For widgets that need to default to clip a rectangle, we’ll set the default clipBehavior to Clip.hardEdge to avoid anti-aliasing cost. For other widgets that default to clip (e.g., ClipRRect), we’ll set the default clipBehavior to Clip.antiAlias. Flutter’s old default behavior Clip.antiAliasWithSaveLayer is not going to be the default clipBehavior for any widget anymore.

### Non-goals

We do not propose to change the default behaviour of Canvas drawing, which currently has anti-aliasing enabled by default. That is: the anti-alias bit in the clip flag only affects whether the clip is anti-aliased or not; it does not affect whether the path inside the widget is anti-aliased or not.

# DETAILED DESIGN/DISCUSSION

We introduce Clip enum with 3 values:

1. Clip.hardEdge  
   (fastest clip without anti-aliasing, reasonable fidelity)
2. Clip.antiAlias  
   (a little slower, smoother edges, better fidelity)
3. Clip.antiAliasWithSaveLayer  
   (very slow, highest fidelity and slightly different semantics)

By default, all widgets that can clip has a default configuration of clipBehavior = Clip.none except

* ClipPath
  + Defaults to Clip.antiAlias
* ClipRRect
  + Defaults to Clip.antiAlias
* ClipRect
  + Defaults to Clip.hardEdge
* Stack
  + Defaults to Clip.hardEdge
* EditableText
  + Defaults to Clip.hardEdge
* ListWheelScrollView
  + Defaults to Clip.hardEdge
* SingleChildScrollView
  + Defaults to Clip.hardEdge
* NestedScrollView
  + Defaults to Clip.hardEdge
* ShrinkWrappingViewport
  + Defaults to Clip.hardEdge
* ListView
  + Defaults to Clip.hardEdge
* GridView
  + Defaults to Clip.hardEdge
* PageView
  + Defaults to Clip.hardEdge

The Clip.antiAliasWithSaveLayer is discouraged but we still provide such option because

1. saveLayer is very costly in performance, but we don’t see many of those bleeding edge artifacts in real applications. (In fact, in studying this, we discovered that we had a bug that would have caused these artifacts, but despite that, we haven't heard any complaints about them.)
2. With or without a saveLayer has different semantics so we shouldn’t automatically add a saveLayer. See the following fiddles and comment on issue 18057 for illustrations:
   1. <https://fiddle.skia.org/c/83ed46ceadaf90f36a4df3b98cbe1c35>
   2. <https://fiddle.skia.org/c/704acfa049a7e99fbe685232c45d1582>
   3. <https://github.com/flutter/flutter/issues/18057#issuecomment-394197336>.

In case Clip.antiAliasWithSaveLayer is used, we’ll correctly put the saveLayer after the clip to avoid the artifact in <https://github.com/flutter/flutter/issues/18057>.

For Stack widget, we removed its overflow configuration and will instead use clipBehavior to control whether the overflowed content is clipped or not.

For ListWheelViewport widget, we removed its clipToSize configuration and will instead use clipBehavior to control whether the overflowed content is clipped or not.

All related RenderObjects of those widgets will be updated accordingly to pass the clipBehavior through to the canvas.

Finally, this proposed change started before Flutter 1.0 and the creation of this Flutter design doc template. Therefore, some pull requests related to this proposal have already been landed:

* Remove unnecessary saveLayer #[5420](https://github.com/flutter/engine/pull/5420)
* Add Clip enum to Material and related widgets #[18576](https://github.com/flutter/flutter/pull/18576)
* Remove saveLayer after clip from dart #[18616](https://github.com/flutter/flutter/pull/18616)
* Add ClipMode to ClipPath/ClipRRect and PhysicalShape layers #[5647](https://github.com/flutter/engine/pull/5647)
* Add anti-alias switch to canvas clip calls #[5670](https://github.com/flutter/engine/pull/5670)
* Rename clip mode to clip behavior #[5853](https://github.com/flutter/engine/pull/5853)
* Rename clip to clipBehavior in compositing.dart #[5868](https://github.com/flutter/engine/pull/5868)
* Call drawPaint instead of drawPath if there's clip #[5937](https://github.com/flutter/engine/pull/5937)
* Call drawPath without clip if possible #[5952](https://github.com/flutter/engine/pull/5952)
* Set default clipBehavior to Clip.none and update tests #[20205](https://github.com/flutter/flutter/pull/20205)
* Expose clipBehavior to more Material Buttons #[20538](https://github.com/flutter/flutter/pull/20538)
* Add customBorder to InkWell so it can clip ShapeBorder #[20751](https://github.com/flutter/flutter/pull/20751)
* Set the default clip to Clip.none again. #[20752](https://github.com/flutter/flutter/pull/20752)
* Add default-no-clip tests to more buttons #[21012](https://github.com/flutter/flutter/pull/21012)
* Default clipBehavior of ClipRect to hardEdge. #[21703](https://github.com/flutter/flutter/pull/21703)
* Missing default hardEdge clip for ClipRectLayer #[21826](https://github.com/flutter/flutter/pull/21826)

An old breaking change proposal and migration instructions were earlier sent to the flutter-dev mailing list: <https://groups.google.com/d/msg/flutter-dev/XMHE0XdsXxI/nonrNQpbBAAJ>

As the time of this writing, we have one more pending PR

* Add clipBehavior to widgets with clipRect #[55977](https://github.com/flutter/flutter/pull/55977)

It will bring clipBehavior and related tests to the following widgets.

1. Flex
2. FittedBox
3. UnconstrainedBox
4. Stack
5. Wrap
6. EditableText
7. ListWheelScrollView
8. SingleChildScrollView
9. NestedScrollView
10. ShrinkWrappingViewport

# TESTING PLAN

For correctness, we add unit tests for all Widgets and RenderObjects that have been changed. Additionally, we’ll also rely on our golden tests to test the final rendering results. The list of tests is probably too long to list here.

For performance, we mainly rely on multiple variants of Flutter gallery transition perf tests such as

* flutter\_gallery\_ios32\_\_transition\_perf
* flutter\_gallery\_ios\_\_transition\_perf
* flutter\_gallery\_\_transition\_perf

# MIGRATION PLAN

To migrate the code, you have 4 choices:

1. Leave your code as is if your content does not need to be clipped (i.e., none of the widgets’ children expand outside their parent’s boundary). This will probably have a positive impact on your overall performance.
2. Add “clipBehavior: Clip.hardEdge” if you need clipping, and clipping without anti-alias is good enough for your (and your clients’) eyes. This will be the common case when you clip rectangles or shapes with very small curved areas (such as the corners of rounded rectangles).
3. Add “clipBehavior: Clip.antiAlias” if you need anti-aliased clipping. This will give you smoother edges at a slightly higher cost. This will be the common case when dealing with circles and arcs.
4. Add “clip.antiAliasWithSaveLayer” if you want the exact same behavior as before (May 2018). Be aware that it’s very costly in performance. This will be only rarely needed. One case where you might need this is if you have an image overlaid on a very different background color. In these cases, consider whether you can avoid overlapping multiple colors in one spot (e.g. by having the background color only present where the image is absent).

For Stack widget specifically, if you previously used `overflow: Overflow.visible`, you shall replace it with `**clipBehavior: Clip.none**`.

For `ListWheelViewport` widget, if you previously specified `clipToSize`, replace it with the corresponding `clipBehavior`: `Clip.none` for `clipToSize = false` and `Clip.hardEdge` for `clipToSize = true`.