Hybrid Composition for iOS

How are UIViews embedded?

To support embedded native views, a new leaf node type **EmbeddedView** has been added.

EmbeddedViews are not painted by Flutter but rather contain a view id to a UIView that will be composited into the Flutter layer tree.

Composition

Flow (the Flutter Compositor) and Quartz (the iOS Compositor) work together to composite EmbeddedViews.

General strategy: Traverse the Flow layer tree until we encounter an EmbeddedView node.

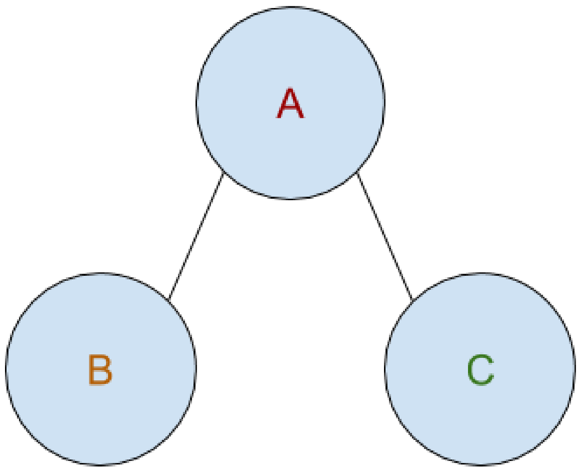
The parents on the EmbeddedView into one Quartz node with the UIView as a child node.

Flow Layer Tree / Quartz Layer Tree Examples

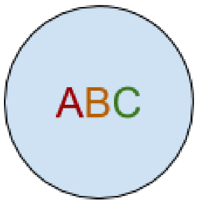
Example 1: No EmbeddedViews

When no EmbeddedViews are included, all of the Flow nodes can be painted on one canvas and result in a single Quartz node.

The left is the Flow layer tree, the right is the Quartz layer tree.



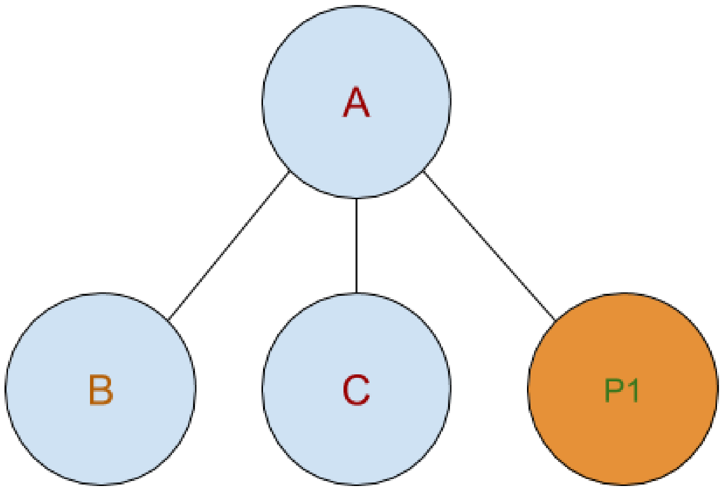
Flow Layer Tree



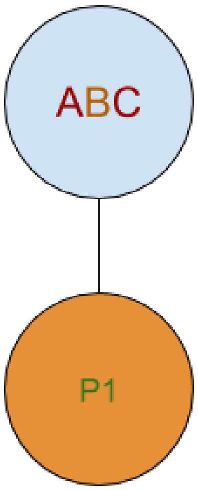
Quartz Layer Tree

Example 2: EmbeddedViews

Similar to the previous example, however an EmbeddedView is painted after all the pure Flutter layers. In this case, the pure Flutter nodes are still combined into a single Quartz node and the UIView is overlayed on-top of the single node.



Flow Layer Tree

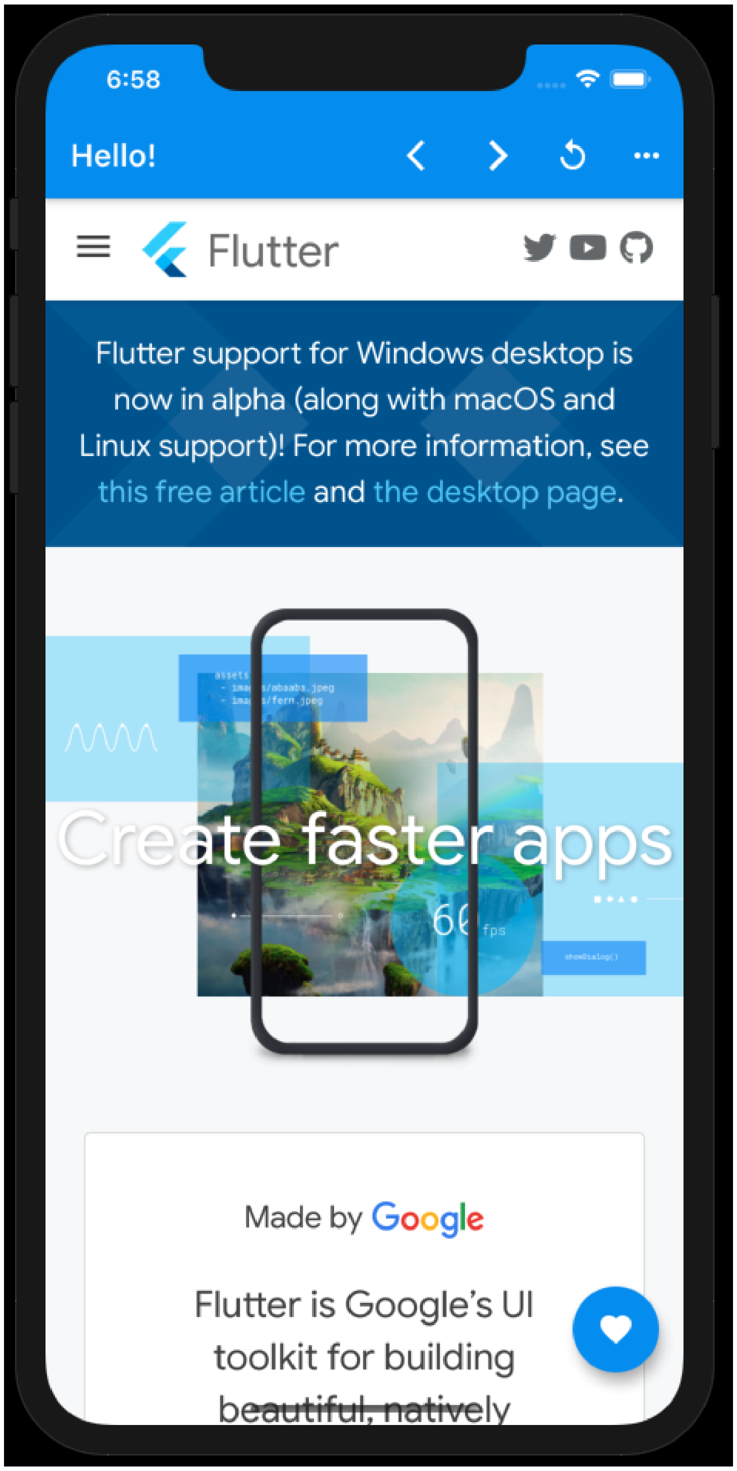


Quartz Layer Tree

Example 3: Practical Example

For this example we will cover the layer trees for the following app.

In this case, the pure Flutter button layer is drawn after the EmbeddedView (Webview) and cannot be combined into a single Quartz node with the rest of the pure Flutter layers, button is drawn after the Webview on it's own CALayer.

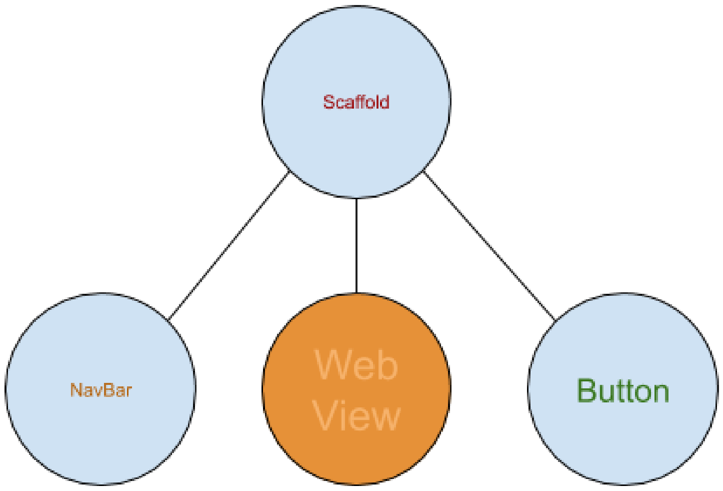


Flutter support for Windows desktop is now in alpha (along with macOS and Linux support)! For more information, see this free article and the desktop page.

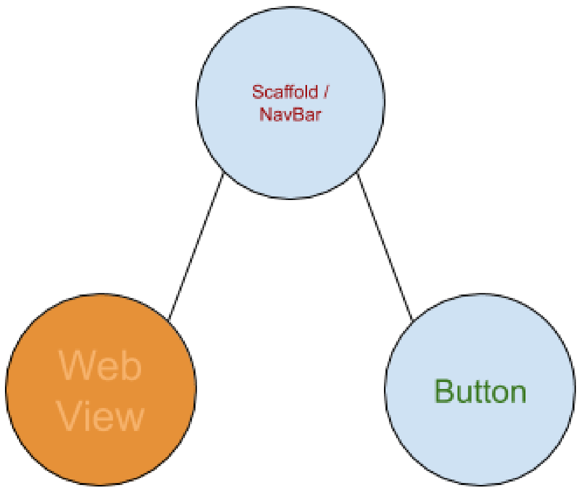
Made by Google

Flutter is Google's UI toolkit for building beautiful, natively

Figure 1: image



Flow Layer Tree



Quartz Layer Tree

Compositing Unobstructed Platform Views

Example 1

The following image illustrates an example where the EmbeddedView is overlayed above a single background canvas.

Example 2

There are two platform views, the algorithm successfully determines that it only needs to allocate an additional UIView for the upper left corner of the FAB.

Applying mutations

For non platform view widgets, mutations are saved to the Skia canvas. Since UIViews are not painted by Skia unlike pure Flutter widgets, the mutations applied to the Skia canvas will not directly apply to the UIView. Mutations for EmbeddedViews must be recorded and applied separately. To do this, we introduce the mutator stack which tracks the mutations specifically for

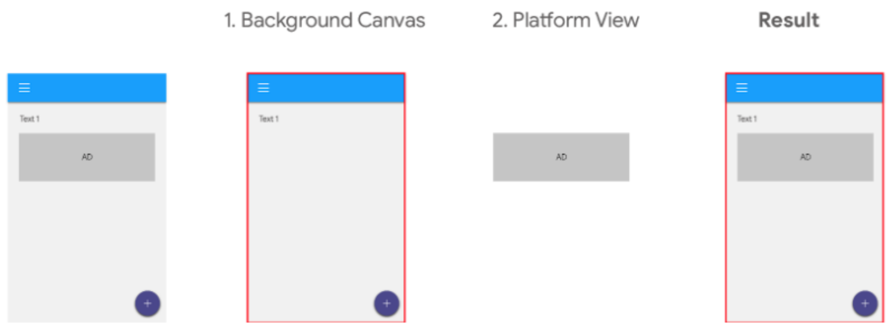


Figure 2: image

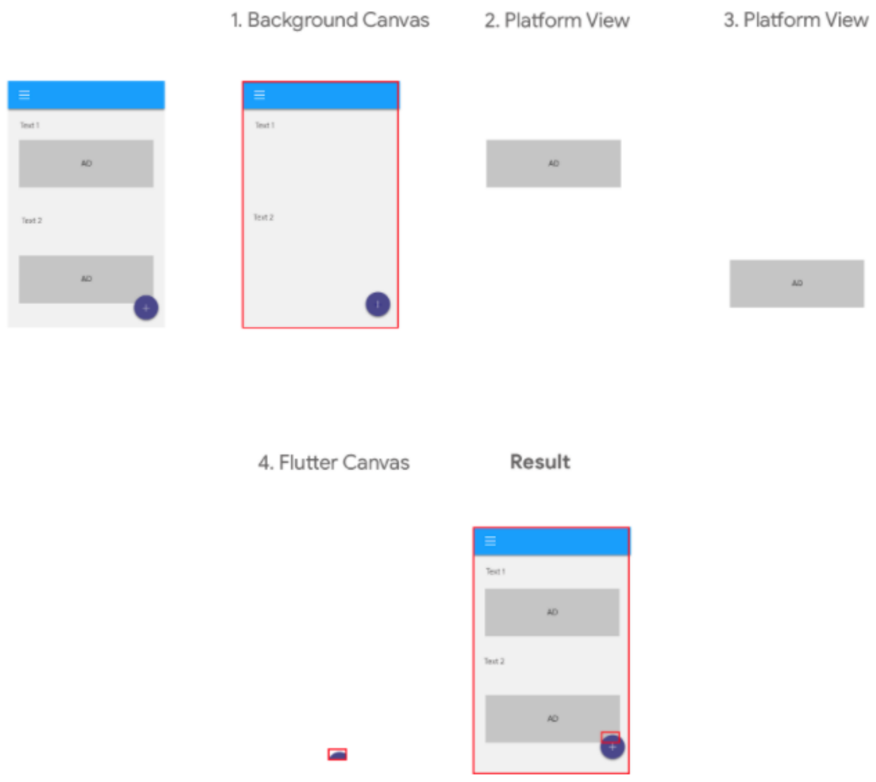


Figure 3: image

EmbeddedViews and ensures iOS applies the mutations when the UIView is drawn.

The following example illustrates the mutator stack during the layer tree traversal. When leaving any node that pushed a mutation onto the stack, the mutation is popped off the stack.

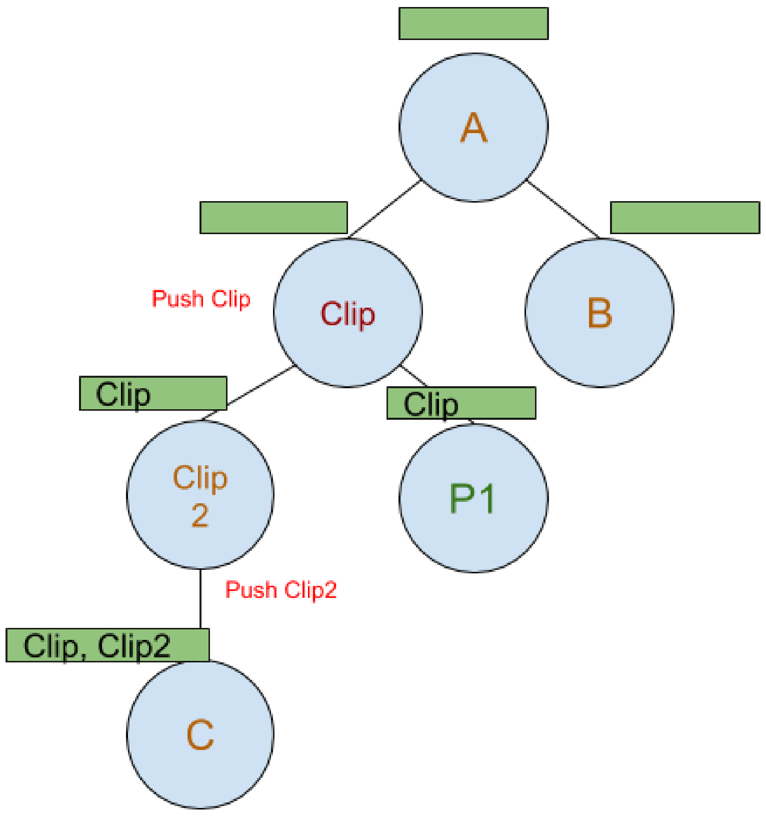


Figure 4: Mutator Stack

Handling Touch

How do EmbeddedViews handle touch events?

Problem: Impossible to synchronously determine whether Flutter or Native iOS should handle the touch event - may need to see multiple events before decision.

Solution: Custom UIGestureRecognizer for views. Prevent other gesture recognizers from recognizing event until UIGestureRecognizer fails. Then Flutter can make async decision whether to use the UIGestureRecognizer.