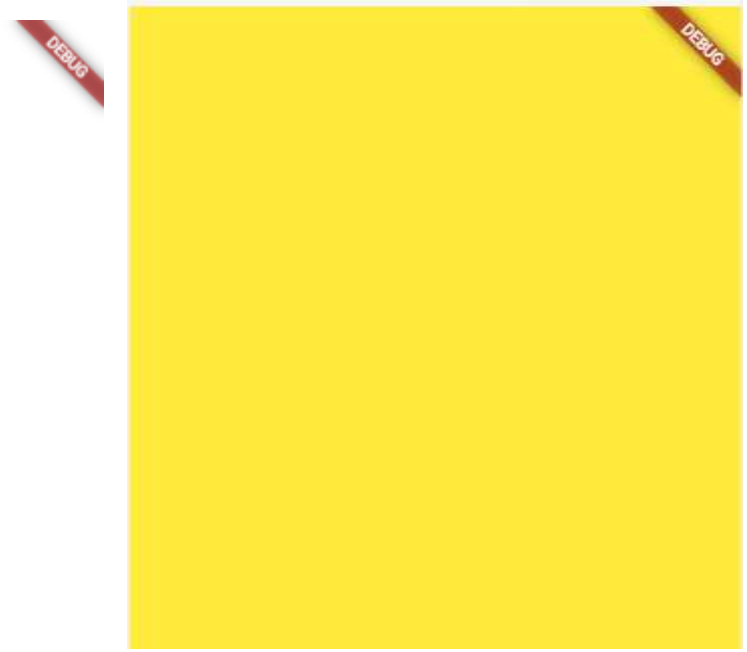


Example-3:

This example uses a GestureDetector that wraps a green Container and a second GestureDetector that wraps a yellow Container. The second GestureDetector is a child of the green Container. Both GestureDetectors define an onTap callback. When the callback is called it adds a red border to the corresponding Container.

When the green Container is tapped, it's parent GestureDetector enters the gesture arena. It wins because there is no competing GestureDetector and the green Container shows a red border. When the yellow Container is tapped, it's parent GestureDetector enters the gesture arena. The GestureDetector that wraps the green Container also enters the gesture arena (the pointer events coordinates are inside both GestureDetectors bounds). The GestureDetector that wraps the yellow Container wins because it was the first detector to enter the arena.

Code:

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

import 'package:flutter/material.dart';

/// Flutter code sample for [GestureDetector].

void main() {

  debugPrintGestureArenaDiagnostics = true;
```

```
  runApp(const NestedGestureDetectorsApp());
}
enum _OnTapWinner { none, yellow, green }
class NestedGestureDetectorsApp extends StatelessWidget {
  const NestedGestureDetectorsApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Nested GestureDetectors')),
        body: const NestedGestureDetectorsExample(),
      ),
    );
  }
}
class NestedGestureDetectorsExample extends StatefulWidget {
  const NestedGestureDetectorsExample({super.key});
  @override
  State<NestedGestureDetectorsExample> createState() =>
      _NestedGestureDetectorsExampleState();
}
class _NestedGestureDetectorsExampleState
    extends State<NestedGestureDetectorsExample> {
```

```
bool _isYellowTranslucent = false;
_OnTapWinner _winner = _OnTapWinner.none;
final Border highlightBorder = Border.all(color: Colors.red, width: 5);
@override
Widget build(BuildContext context) {
  return Column(
    children: <Widget>[
      Expanded(
        child: GestureDetector(
          onTap: () {
            debugPrint('Green onTap');
            setState(() {
              _winner = _OnTapWinner.green;
            });
          },
          onTapDown: (_) => debugPrint('Green onTapDown'),
          onTapCancel: () => debugPrint('Green onTapCancel'),
          child: Container(
            alignment: Alignment.center,
            decoration: BoxDecoration(
              border: _winner == _OnTapWinner.green ? highlightBorder : null,
              color: Colors.green,
            ),
```

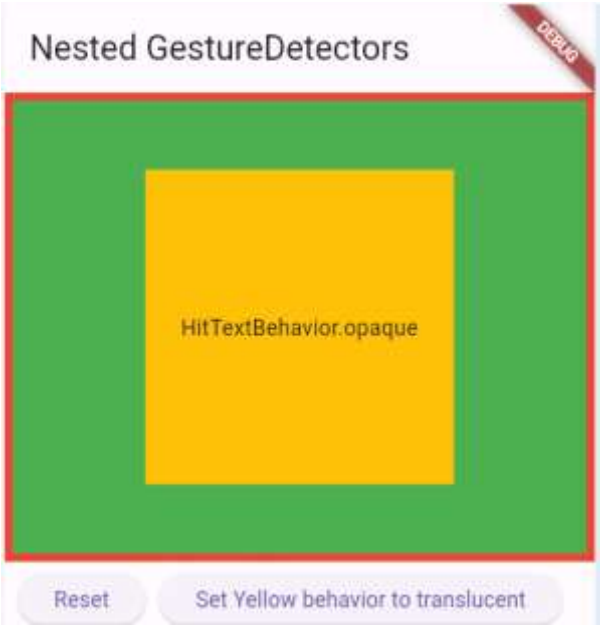
```
child: GestureDetector(
  // Setting behavior to transparent or opaque as no impact on
  // parent-child hit testing. A tap on 'Yellow' is also in
  // 'Green' bounds. Both enter the gesture arena, 'Yellow' wins
  // because it is in front.
  behavior: _isYellowTranslucent
      ? HitTestBehavior.translucent
      : HitTestBehavior.opaque,
  onTap: () {
    debugPrint('Yellow onTap');
    setState(() {
      _winner = _OnTapWinner.yellow;
    });
  },
  child: Container(
    alignment: Alignment.center,
    decoration: BoxDecoration(
      border:
          _winner == _OnTapWinner.yellow ? highlightBorder : null,
      color: Colors.amber,
    ),
    width: 200,
    height: 200,
```

```
            child: Text(
              'HitTextBehavior.${_isYellowTranslucent ? 'translucent' : 'opaque'}',
              textAlign: TextAlign.center,
            ),
          ),
        ),
      ),
    ),
  ),
),
Padding(
  padding: const EdgeInsets.all(8.0),
  child: Row(
    children: <Widget>[
      ElevatedButton(
        child: const Text('Reset'),
        onPressed: () {
          setState(() {
            _isYellowTranslucent = false;
            _winner = _OnTapWinner.none;
          });
        },
      ),
      const SizedBox(width: 8),
```

```
            ElevatedButton(
              child: Text(
                'Set Yellow behavior to ${_isYellowTranslucent ? 'opaque' : 'translucent'}',
              ),
              onPressed: () {
                setState(() => _isYellowTranslucent = !_isYellowTranslucent);
              },
            ),
          ],
        ),
      ),
    ],
  );
 }
 @override
 void dispose() {
  debugPrintGestureArenaDiagnostics = false;
  super.dispose();
 }
}
```

Output:


Nested GestureDetectors
DEBUG
HitTextBehavior.opaque
Reset
Set Yellow behavior to translucent


Nested GestureDetectors
DEBUG
HitTextBehavior.opaque
Reset
Set Yellow behavior to translucent


Nested GestureDetectors
DEBUG
HitTextBehavior.opaque
Reset
Set Yellow behavior to translucent