



# Life cycle of a RenderObject

FlutterConnection, Paris, June 2, 2023



**Craig Labenz**

Developer Relations Engineer, Google

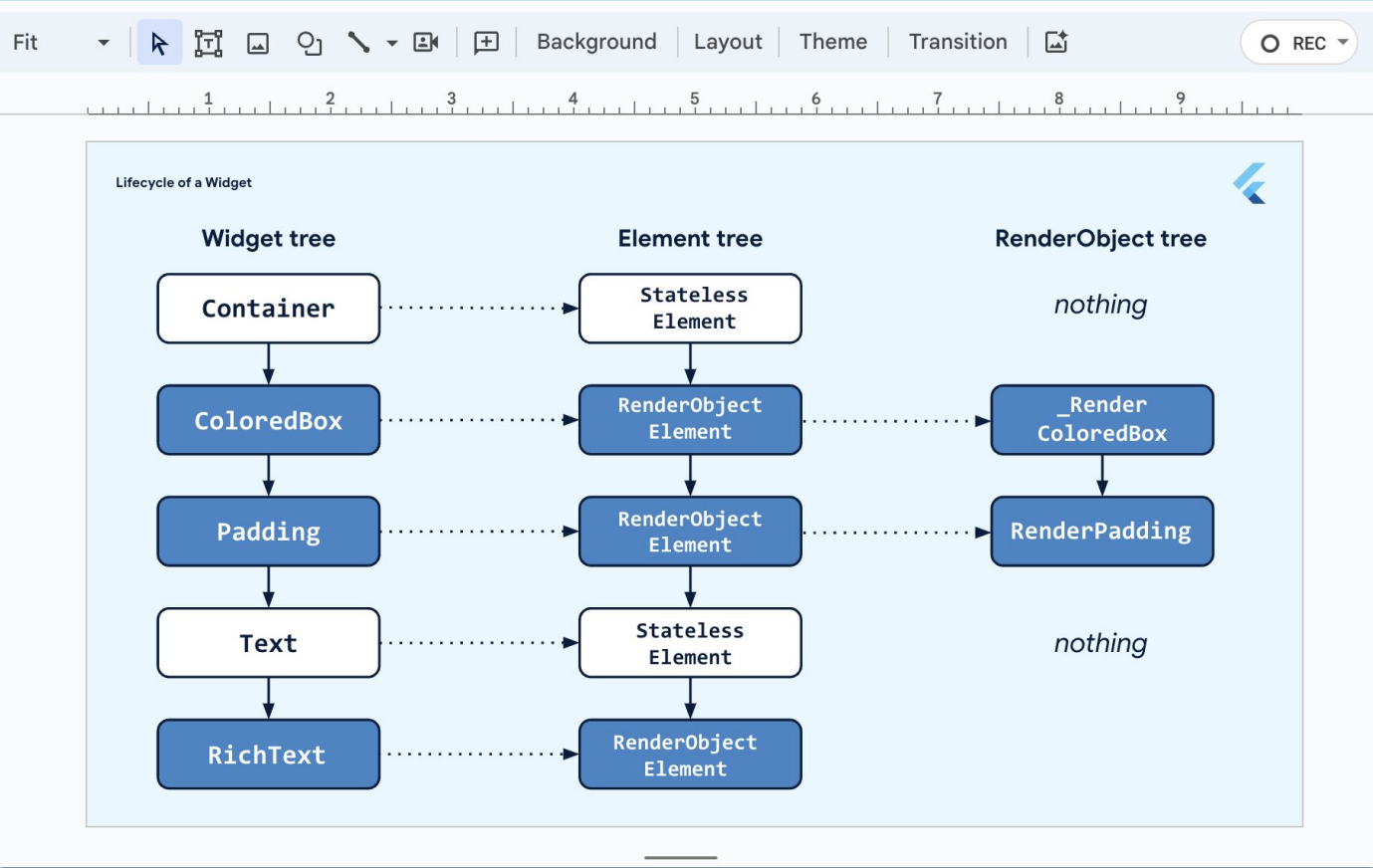


The screenshot shows a presentation slide titled "Lifecycle of a Widget". The slide has a light blue background and a white border. At the top, there is a toolbar with various icons for editing and presentation, including a "Fit" dropdown, a selection tool, a zoom tool, a pan tool, a lasso tool, a copy tool, a paste tool, a "Background" button, a "Layout" button, a "Theme" button, and a "Transition" button. A "REC" button is also visible in the top right corner. Below the toolbar is a horizontal timeline with markers numbered 1 through 9. The main content area of the slide contains the text "Lifecycle of a Widget" in the top left corner and a blue logo in the top right corner. In the center of the slide, there is a definition of a widget:

**Widget** [ wij-it ] (n): A discrete block of an app's UI containing:

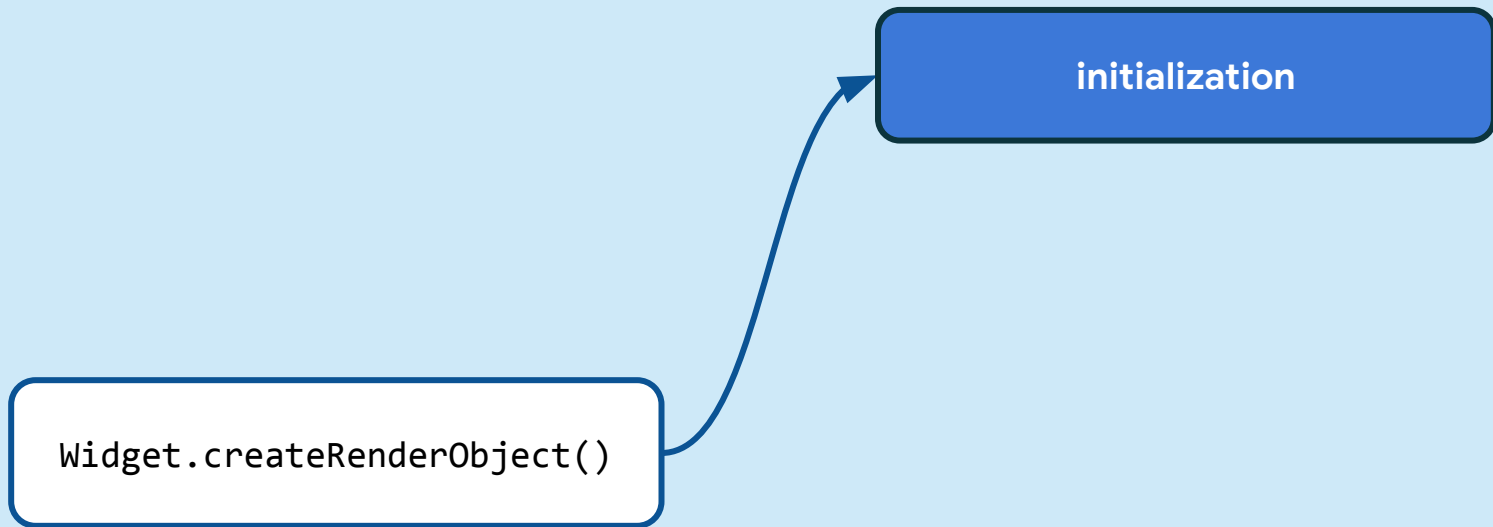
- a static configuration
- optionally, persistent state
- a **render()** method

A yellow arrow points from the text "optionally, persistent state" to the right.





# RenderObject





```
abstract class RenderObject extends AbstractNode {}
```



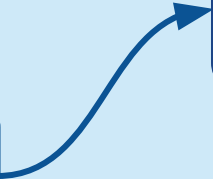
# Layout



initialization

flush  
( *layout* )

PipelineOwner.flush()







```
abstract class RenderObject extends AbstractNode {}
```



```
abstract class RenderObject extends AbstractNode {  
    void layout(Constraints constraints);  
}
```




```
abstract class RenderObject extends AbstractNode {  
    void layout(Constraints constraints);  
    void performLayout();  
}
```



```
abstract class RenderObject extends AbstractNode {  
    void layout(Constraints constraints);  
    void performLayout();  
    void markNeedsLayout();  
}
```



```
abstract class RenderObject extends AbstractNode {  
    void layout(Constraints constraints);  
    void performLayout();  
    void markNeedsLayout();  
}
```

A large yellow arrow with a black outline, pointing upwards and to the right, highlighting the 'void' keyword in the first method signature of the code block.



```
abstract class RenderBox extends RenderObject {  
    Size? size;  
  
    void performLayout() {  
        size = calculateSize();  
    }  
}
```



```
class RenderPadding extends RenderShiftedBox {  
  void layout(Constraints constraints) {  
    _constraints = constraints;  
    performLayout();  
  }  
  void performLayout() {  
    final innerConstraints = _constraints - padding;  
    child.layout(innerConstraints);  
    size = child.size + padding;  
  }  
}
```



```
class RenderPadding extends RenderShiftedBox {  
  void layout(Constraints constraints) {  
    _constraints = constraints;  
    performLayout();  
  }  
  void performLayout() {  
    final innerConstraints = _constraints - padding;  
    child.layout(innerConstraints);  
    size = child.size + padding;  
  }  
}
```

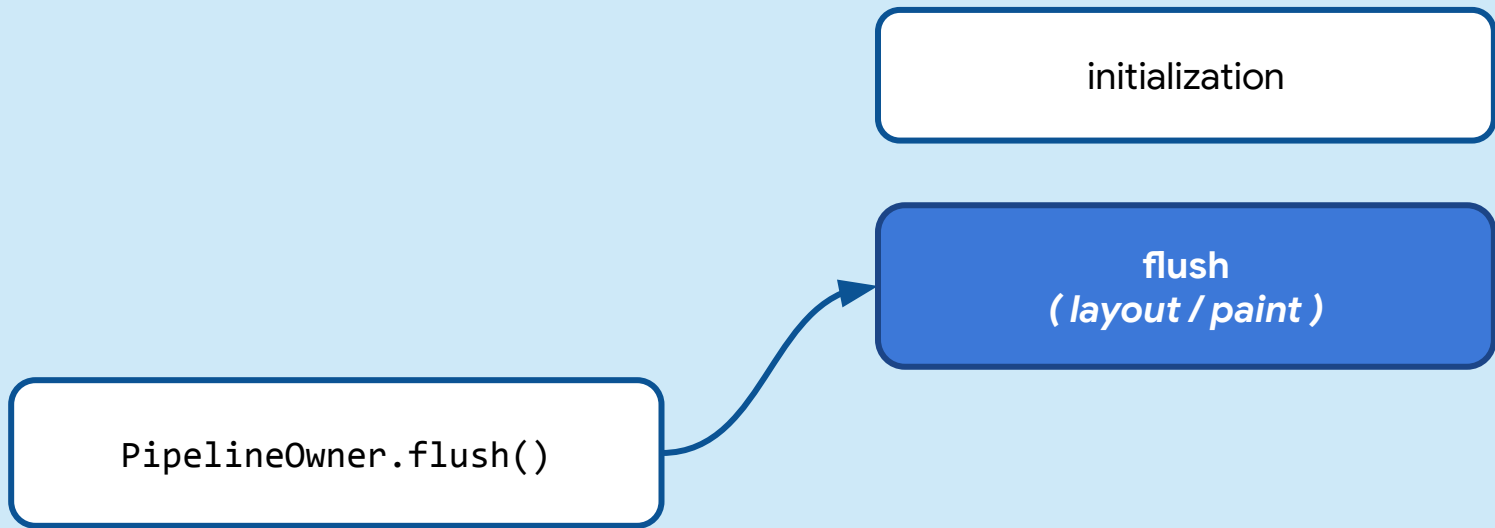




```
class RenderPadding extends RenderShiftedBox {  
  void layout(Constraints constraints) {  
    _constraints = constraints;  
    performLayout();  
  }  
  void performLayout() {  
    final innerConstraints = _constraints - padding;  
    child.layout(innerConstraints);  
    size = child.size + padding;  
  }  
}
```



# Painting





```
abstract class RenderObject extends AbstractNode {}
```



```
abstract class RenderObject extends AbstractNode {  
    void paint(PaintingContext context, Offset offset);  
}
```



```
abstract class RenderObject extends AbstractNode {  
    void paint(PaintingContext context, Offset offset);  
    void markNeedsPaint();  
}
```



```
abstract class RenderObject extends AbstractNode {  
    void paint(PaintingContext context, Offset offset);  
    void markNeedsPaint();  
}
```



```
abstract class RenderObject extends AbstractNode {  
    void paint(PaintingContext context, Offset offset);  
    void markNeedsPaint();  
}
```





```
abstract class RenderObject extends AbstractNode {  
    void paint(PaintingContext context, Offset offset);  
    void markNeedsPaint();  
}
```



```
context.canvas.drawRect(size, Paint());
```



```
// painting.dart
```

```
/// An interface for recording graphical operations.
```

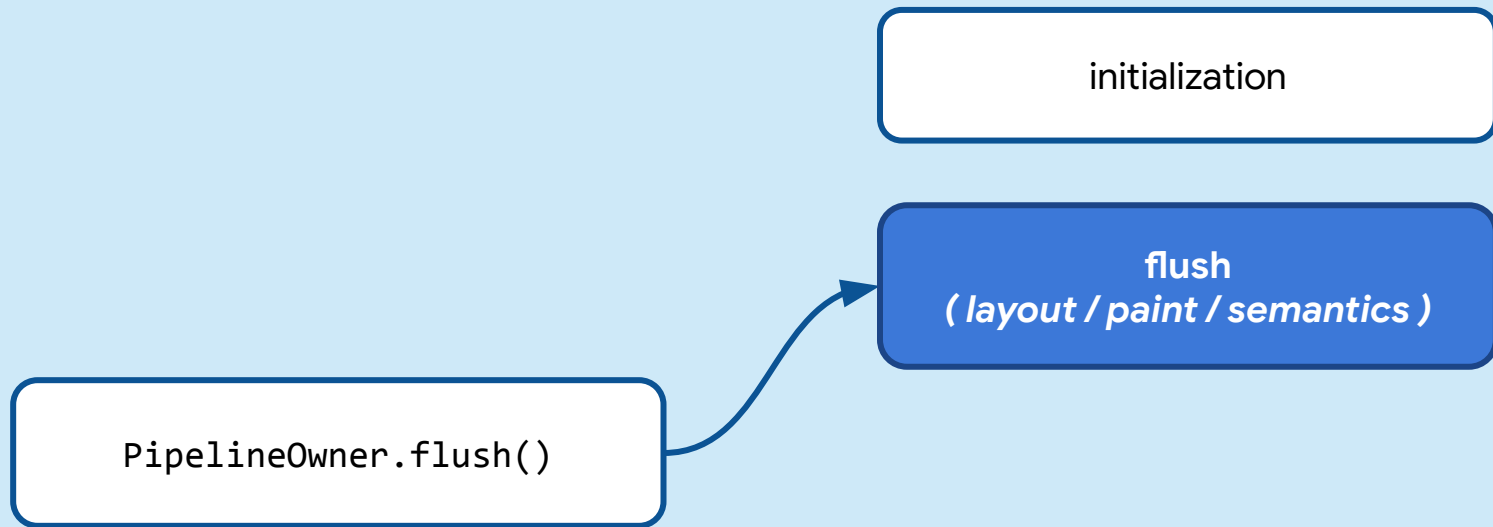
```
abstract class Canvas {}
```



```
class RenderTransform extends RenderProxyBox {  
  void paint(PaintingContext context, Offset offset) {  
    Layer layer = context.pushTransform(...);  
  }  
}
```



# Semantics





```
abstract class RenderObject extends AbstractNode {}
```



```
abstract class RenderObject extends AbstractNode {  
    void describeSemanticsConfiguration(  
        SemanticsConfiguration config,  
    );  
}
```





```
abstract class RenderObject extends AbstractNode {  
    void describeSemanticsConfiguration(  
        SemanticsConfiguration config,  
    );  
    void markNeedsSemanticsUpdate();  
}
```



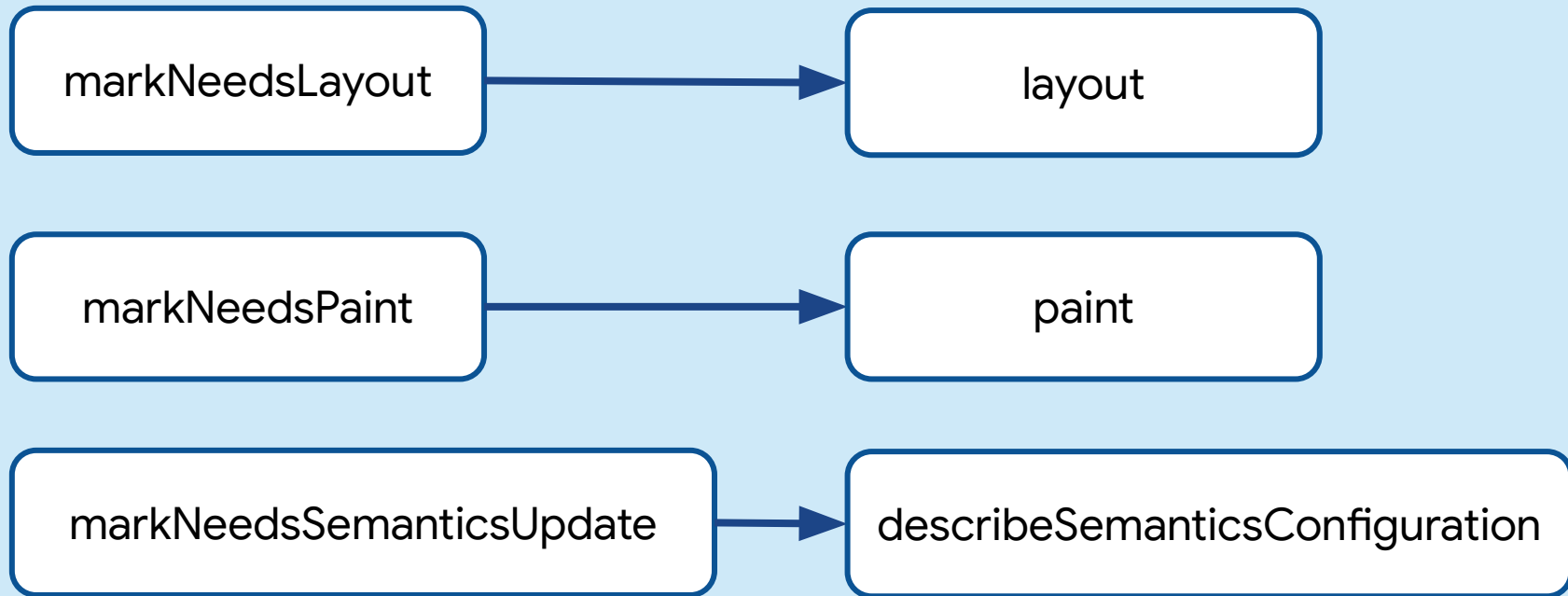
```
abstract class RenderObject extends AbstractNode {  
  void describeSemanticsConfiguration(  
    SemanticsConfiguration config,  
  ) {  
    config.isSemanticBoundary = true;  
    config.label = myTextValue;  
  }  
}
```



# Updates



# markNeeds\*





```
class RenderString extends RenderBox {  
  RenderString({required String value});  
}
```



```
class RenderString extends RenderBox {  
  RenderString({required String value}) : _value = value;  
  
  String _value;  
}
```



```
class RenderString extends RenderBox {  
  
  void set value(String newValue) {  
    if (newValue == _value) return;  
    _value = newValue;  
  
  }  
}
```



```
class RenderString extends RenderBox {  
  
  void set value(String newValue) {  
    if (newValue == _value) return;  
    _value = newValue;  
  
  }  
}
```





```
class RenderString extends RenderBox {  
  
  void set value(String newValue) {  
    if (newValue == _value) return;  
    _value = newValue;  
  
  }  
}
```



```
class RenderString extends RenderBox {  
  
  void set value(String newValue) {  
    if (newValue == _value) return;  
    _value = newValue;  
    markNeedsPaint();  
  }  
}
```



```
class RenderString extends RenderBox {  
  
  void set value(String newValue) {  
    if (newValue == _value) return;  
    _value = newValue;  
    markNeedsPaint();  
    markNeedsLayout();  
  }  
}
```



```
class RenderString extends RenderBox {  
  
  void set value(String newValue) {  
    if (newValue == _value) return;  
    _value = newValue;  
    markNeedsPaint();  
    markNeedsLayout();  
  }  
}
```



```
class RenderString extends RenderBox {  
  
  void set value(String newValue) {  
    if (newValue == _value) return;  
    _value = newValue;  
    markNeedsSemanticsUpdate();  
    markNeedsLayout();  
  }  
}
```



```
void markNeedsLayout() {  
    if (parent.isSameLayer) {  
        parent.markNeedsLayout();  
    }  
    owner!._nodesNeedingLayout.add(this);  
    owner!.requestVisualUpdate();  
}
```



```
void markNeedsLayout() {  
    if (parent.isSameLayer) {  
        parent.markNeedsLayout();  
    }  
    owner!._nodesNeedingLayout.add(this);  
    owner!.requestVisualUpdate();  
}
```



```
void markNeedsLayout() {  
    if (parent.isSameLayer) {  
        parent.markNeedsLayout();  
    }  
    owner!._nodesNeedingLayout.add(this);  
    owner!.requestVisualUpdate();  
}
```





```
void markNeedsLayout() {  
    if (parent.isSameLayer) {  
        parent.markNeedsLayout();  
    }  
    owner!._nodesNeedingLayout.add(this);  
    owner!.requestVisualUpdate();  
}
```



```
class StringWidget extends LeafRenderObjectWidget {  
  StringWidget({required this.value});  
  final String value;  
  
  void updateRenderObject(RenderString renderObject) {  
    renderObject.value = value;  
  }  
}
```



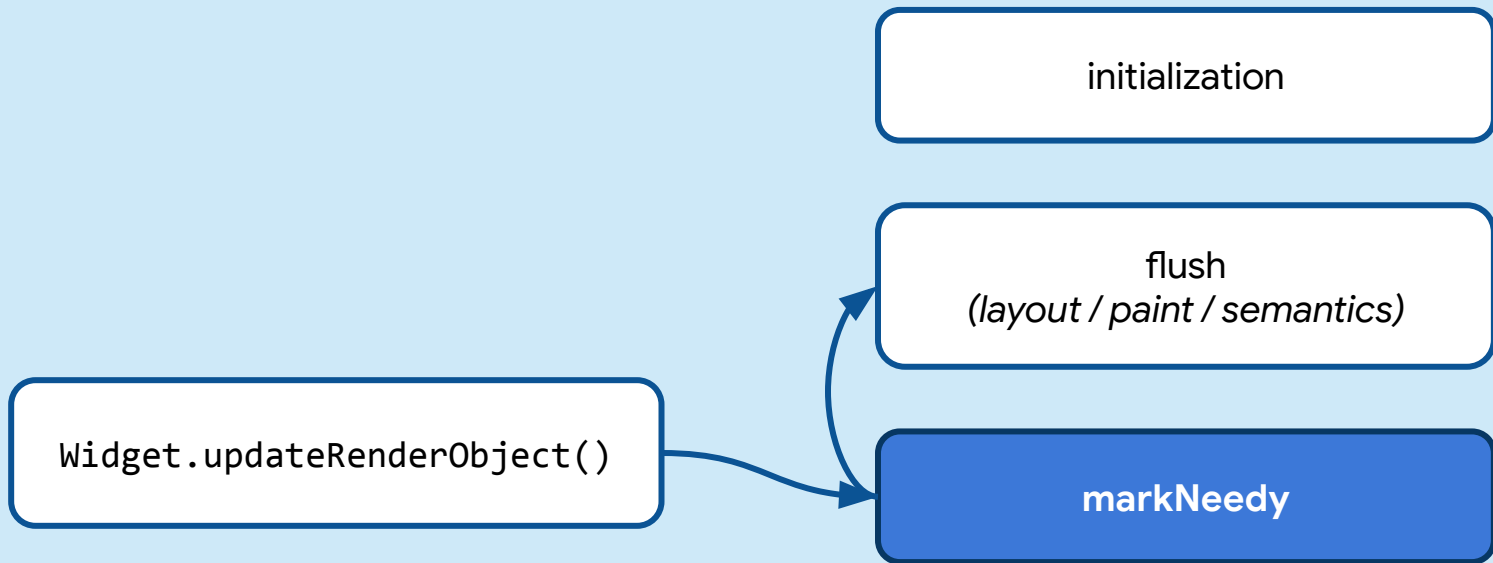
```
class StringWidget extends LeafRenderObjectWidget {  
  StringWidget({required this.value});  
  final String value;  
  
  void updateRenderObject(RenderString renderObject) {  
    renderObject.value = value;  
  }  
}
```



```
class StringWidget extends LeafRenderObjectWidget {  
  StringWidget({required this.value});  
  final String value;  
  
  void updateRenderObject(RenderString renderObject) {  
    renderObject.value = value;  
  }  
}
```



```
class RenderString extends RenderBox {  
  
  void set value(String newValue) {  
    if (newValue == _value) return;  
    _value = newValue;  
    markNeedsSemanticsUpdate();  
    markNeedsLayout();  
  }  
}
```





# Hit testing



```
abstract class RenderBox extends RenderObject {  
    bool hitTest(BoxHitTestResult result, {required Offset position}) {  
        if (!size.contains(position)) return false;  
        if (hitTestChildren(result, position) ||  
            hitTestSelf(result, position)) {  
            result.add(BoxHitTestEntry(this, position));  
            return true;  
        }  
        return false;  
    }  
}
```





```
abstract class RenderBox extends RenderObject {  
    bool hitTest(BoxHitTestResult result, {required Offset position}) {  
        if (!size.contains(position)) return false;  
        if (hitTestChildren(result, position) ||  
            hitTestSelf(result, position)) {  
            result.add(BoxHitTestEntry(this, position));  
            return true;  
        }  
        return false;  
    }  
}
```



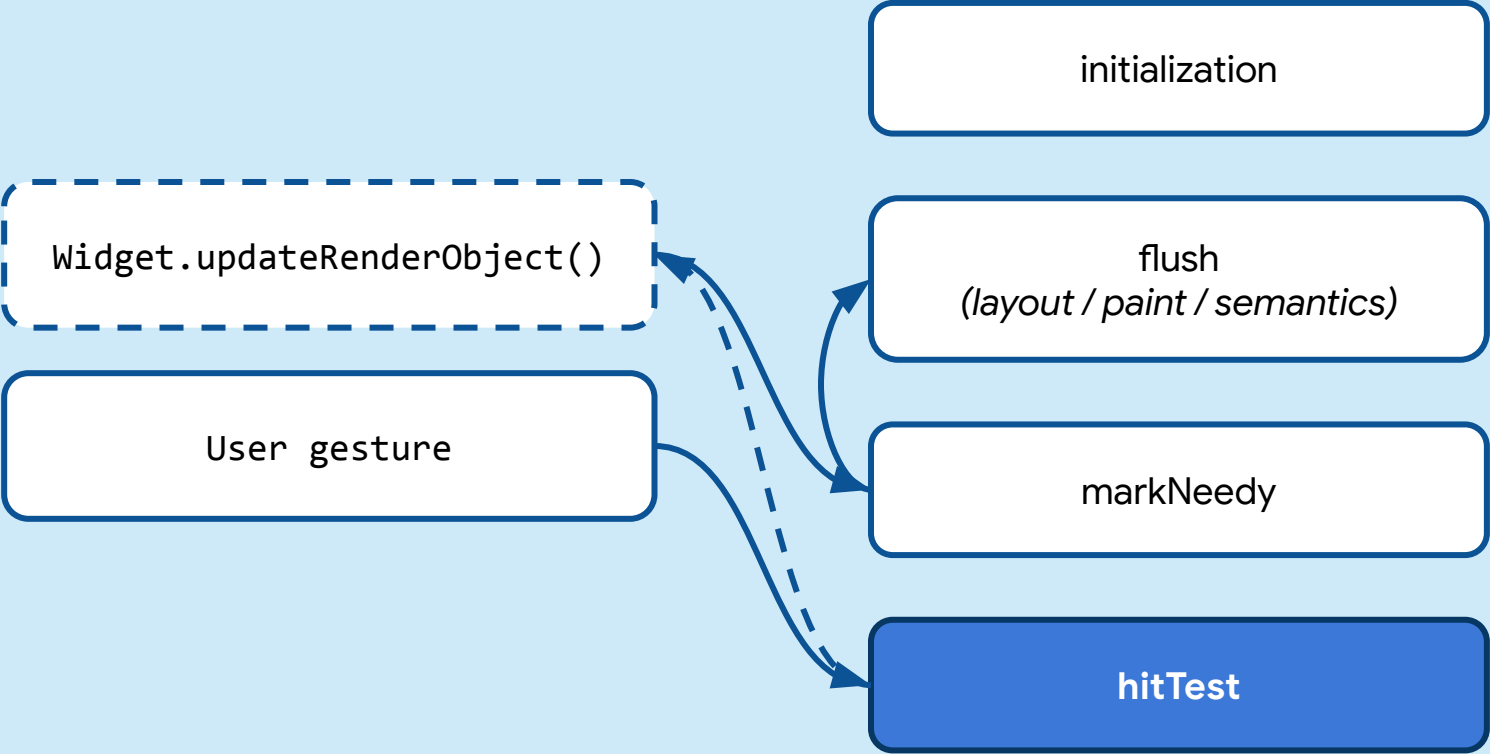
```
abstract class RenderBox extends RenderObject {  
  bool hitTest(BoxHitTestResult result, {required Offset position}) {  
    if (!size.contains(position)) return false;  
    if (hitTestChildren(result, position) ||  
        hitTestSelf(result, position)) {  
      result.add(BoxHitTestEntry(this, position));  
      return true;  
    }  
    return false;  
  }  
}
```



```
abstract class RenderBox extends RenderObject {  
  bool hitTest(BoxHitTestResult result, {required Offset position}) {  
    if (!size.contains(position)) return false;  
    if (hitTestChildren(result, position) ||  
        hitTestSelf(result, position)) {  
      result.add(BoxHitTestEntry(this, position));  
      return true;  
    }  
    return false;  
  }  
}
```



```
abstract class RenderBox extends RenderObject {  
  bool hitTest(BoxHitTestResult result, {required Offset position}) {  
    if (!size.contains(position)) return false;  
    if (hitTestChildren(result, position) ||  
        hitTestSelf(result, position)) {  
      result.add(BoxHitTestEntry(this, position));  
      return true;  
    }  
    return false;  
  }  
}
```

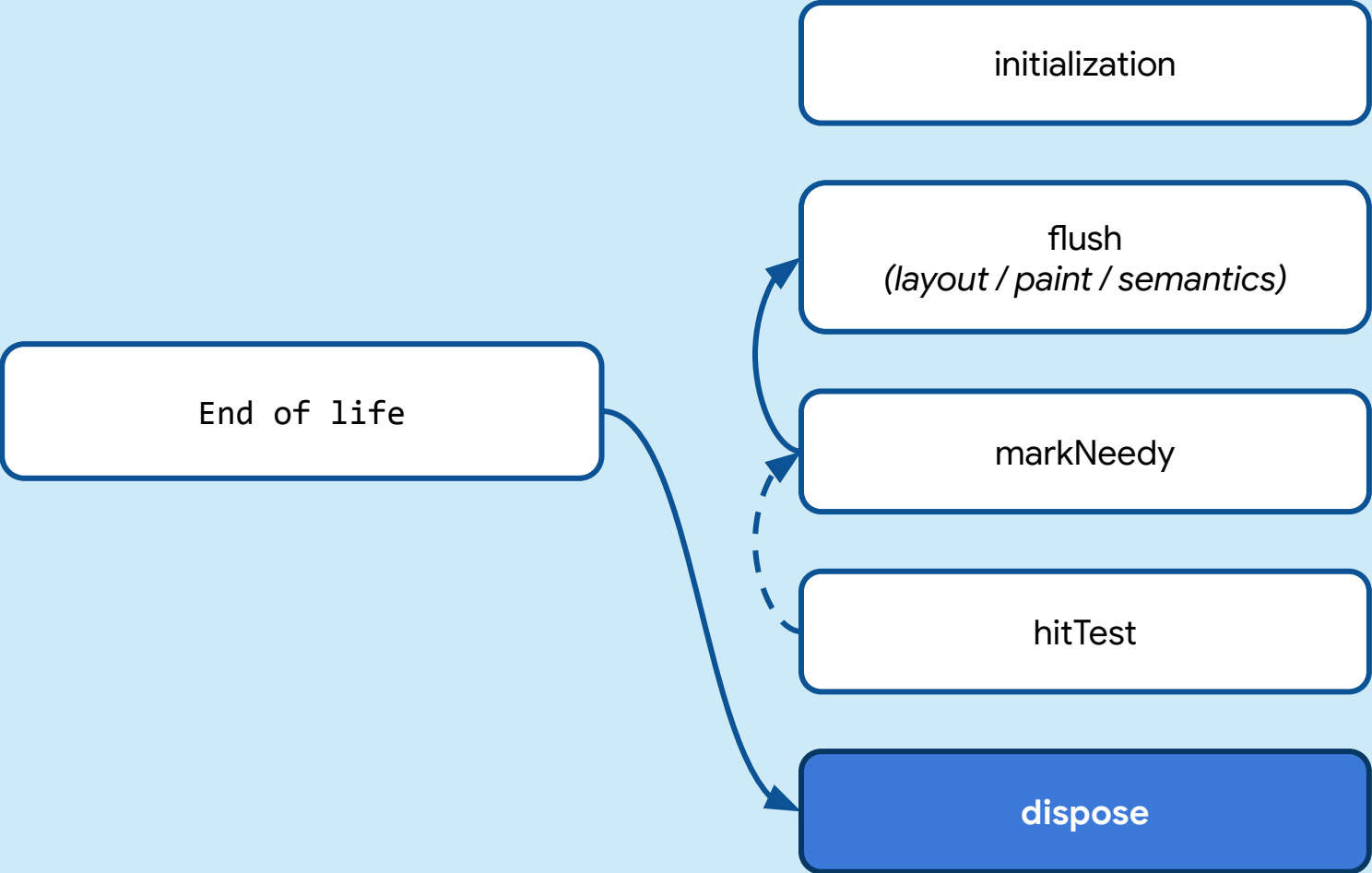




# Dispose



```
abstract class RenderObject extends AbstractNode {  
  void dispose() {  
    layer?.dispose();  
    textPainter?.dispose();  
    super.dispose();  
  }  
}
```







# ParentData



```
class RenderFlex extends RenderBox {  
  
  void addAll(List<Widget> children) {  
    Widget? previousChild;  
    for (final child in children) {  
      child.parentData = FlexParentData();  
      previousChild?.parentData.nextSibling = child;  
      previousChild = child;  
    }  
  }  
}
```





# Stack :: Positioned



**</ RenderObject>**



**</ RenderObject>**



# The Layer Tree



```
class RenderTransform extends RenderProxyBox {  
  void paint(PaintingContext context, Offset offset) {  
    context.canvas.drawRect(size, Paint());  
  }  
}
```





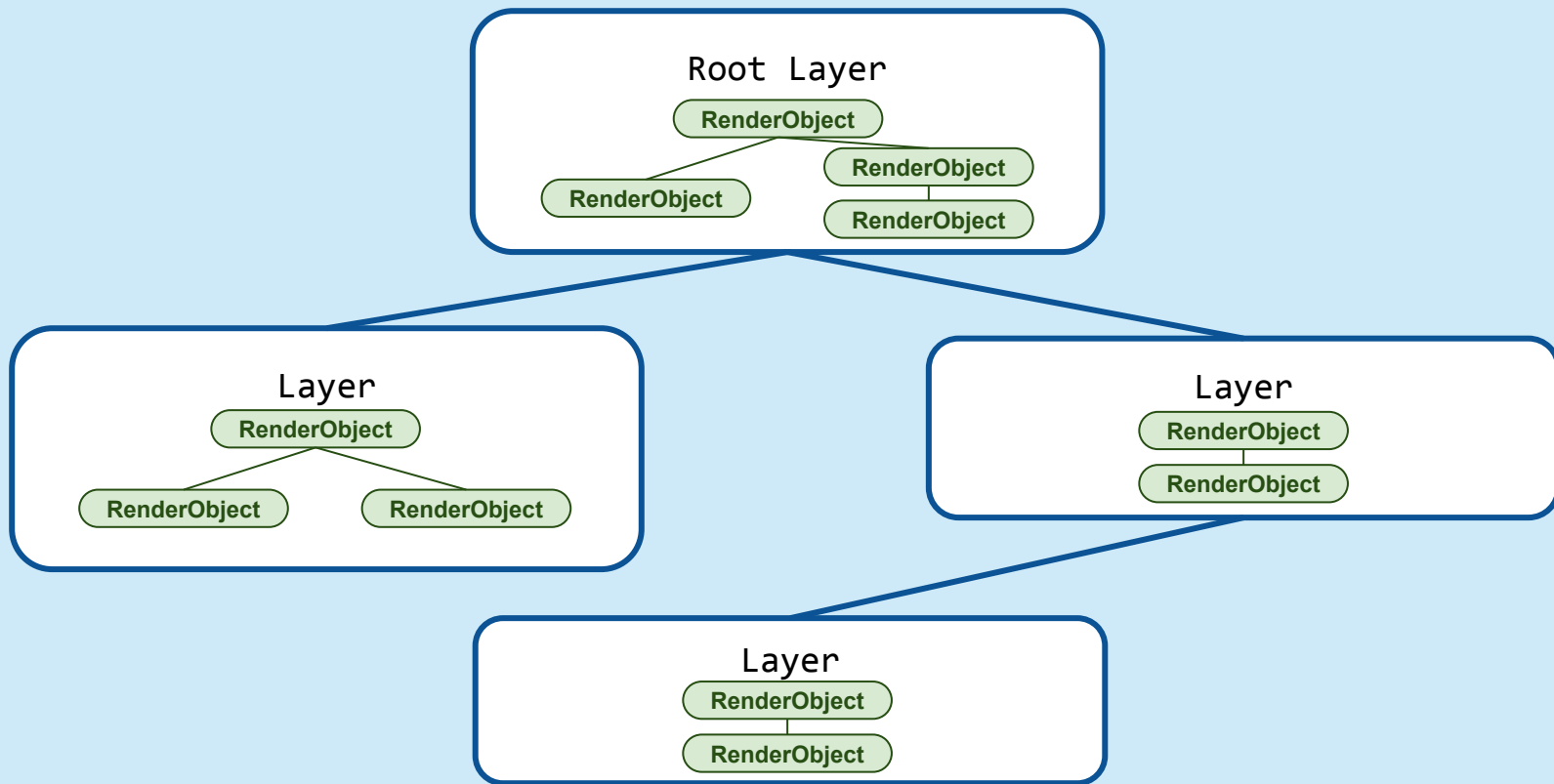
```
class RenderTransform extends RenderProxyBox {  
  void paint(PaintingContext context, Offset offset) {  
    context.canvas.drawRect(size, Paint());  
  }  
}
```



```
class RenderTransform extends RenderProxyBox {  
  void paint(PaintingContext context, Offset offset) {  
    context.canvas.drawRect(size, Paint());  
  }  
}
```



```
class RenderTransform extends RenderProxyBox {  
  void paint(PaintingContext context, Offset offset) {  
    layer = context.pushTransform(...);  
  }  
}
```





# Layers



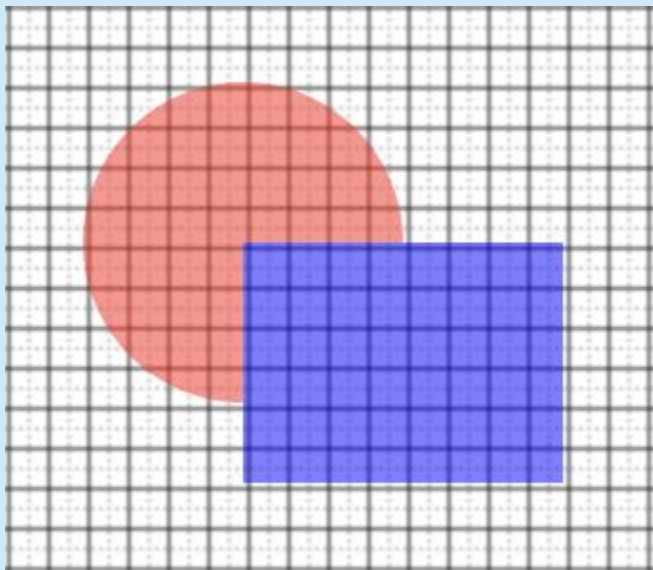
$$10 * 3 + 5 = 35$$

$$10 * (3 + 5) = 80$$



**Background + Offset + Opacity(Circle + Square)**

**Background + Offset + Opacity(Circle + Square)**

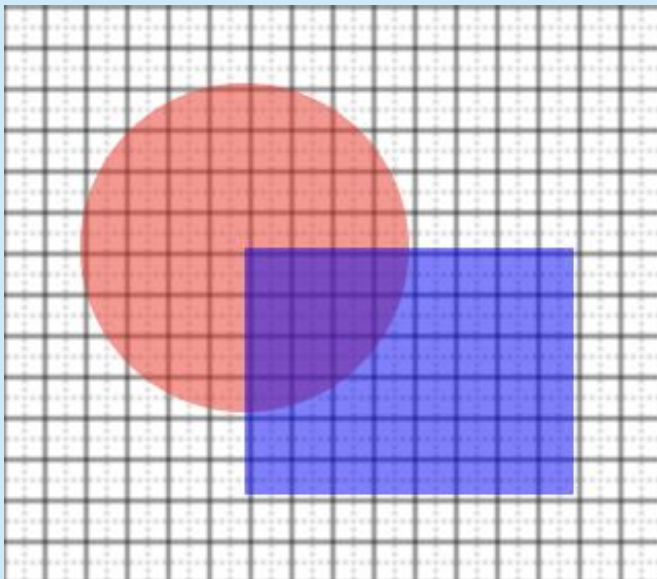






**Background + Offset + Opacity(Circle) + Opacity(Square)**

**Background + Offset + Opacity(Circle) + Opacity(Square)**





**Needing Layers is rare**



**</ Layers>**



```
class WidgetsFlutterBinding with RenderBinding {  
  ui.Layer rootLayer;  
  ui.SceneBuilder builder;  
  ui.FlutterView view;  
  
  void drawFrame() {  
    pipelineOwner.flush();  
    final ui.Scene scene = builder.build(rootLayer);  
    view.render(scene);  
  }  
}
```



```
class WidgetsFlutterBinding with RenderBinding {  
  ui.Layer rootLayer;  
  ui.SceneBuilder builder;  
  ui.FlutterView view;  
  
  void drawFrame() {  
    pipelineOwner.flush();  
    final ui.Scene scene = builder.build(rootLayer);  
    view.render(scene);  
  }  
}
```



```
class WidgetsFlutterBinding with RenderBinding {  
  ui.Layer rootLayer;  
  ui.SceneBuilder builder;  
  ui.FlutterView view;  
  
  void drawFrame() {  
    pipelineOwner.flush();  
    final ui.Scene scene = builder.build(rootLayer);  
    view.render(scene);  
  }  
}
```



```
class WidgetsFlutterBinding with RenderBinding {  
  ui.Layer rootLayer;  
  ui.SceneBuilder builder;  
  ui.FlutterView view;  
  
  void drawFrame() {  
    pipelineOwner.flush();  
    final ui.Scene scene = builder.build(rootLayer);  
    view.render(scene);  
  }  
}
```





```
class WidgetsFlutterBinding with RenderBinding {  
  ui.Layer rootLayer;  
  ui.SceneBuilder builder;  
  ui.FlutterView view;  
  
  void drawFrame() {  
    pipelineOwner.flush();  
    final ui.Scene scene = builder.build(rootLayer);  
    view.render(scene);  
  }  
}
```



```
class WidgetsFlutterBinding with RenderBinding {  
  ui.FlutterView view;  
  ui.Layer rootLayer;  
  ui.SceneBuilder builder;  
  
  void drawFrame() {  
    pipelineOwner.flush();  
    final ui.Scene scene = builder.build(rootLayer);  
    view.render(scene);  
  }  
}
```



```
class WidgetsFlutterBinding with RenderBinding {  
  ui.FlutterView view;  
  ui.Layer rootLayer;  
  ui.SceneBuilder builder;  
  
  void drawFrame() {  
    pipelineOwner.flush();  
    final ui.Scene scene = builder.build(rootLayer);  
    view.render(scene);  
  }  
}
```



Shader A



Shader B

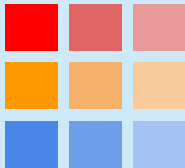


Shader C

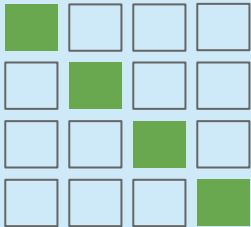




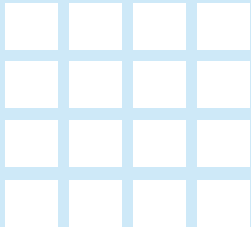
Shader A



Shader B

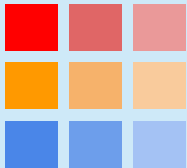


Shader C

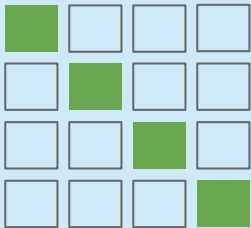




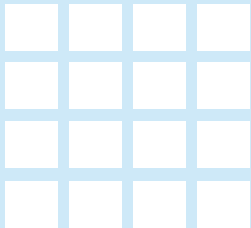
Shader A 



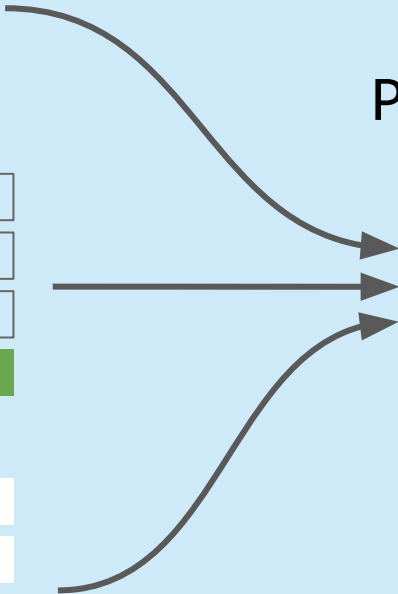
Shader B 



Shader C 

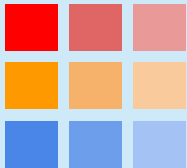


Pixel Buffer

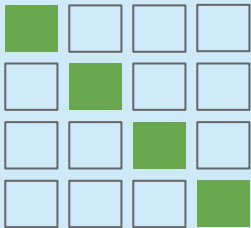




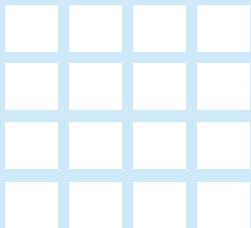
Shader A →



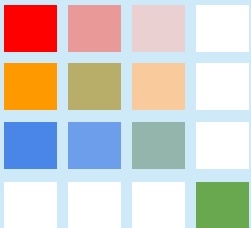
Shader B →



Shader C →



Pixel Buffer





# **Pixels on the screen!**

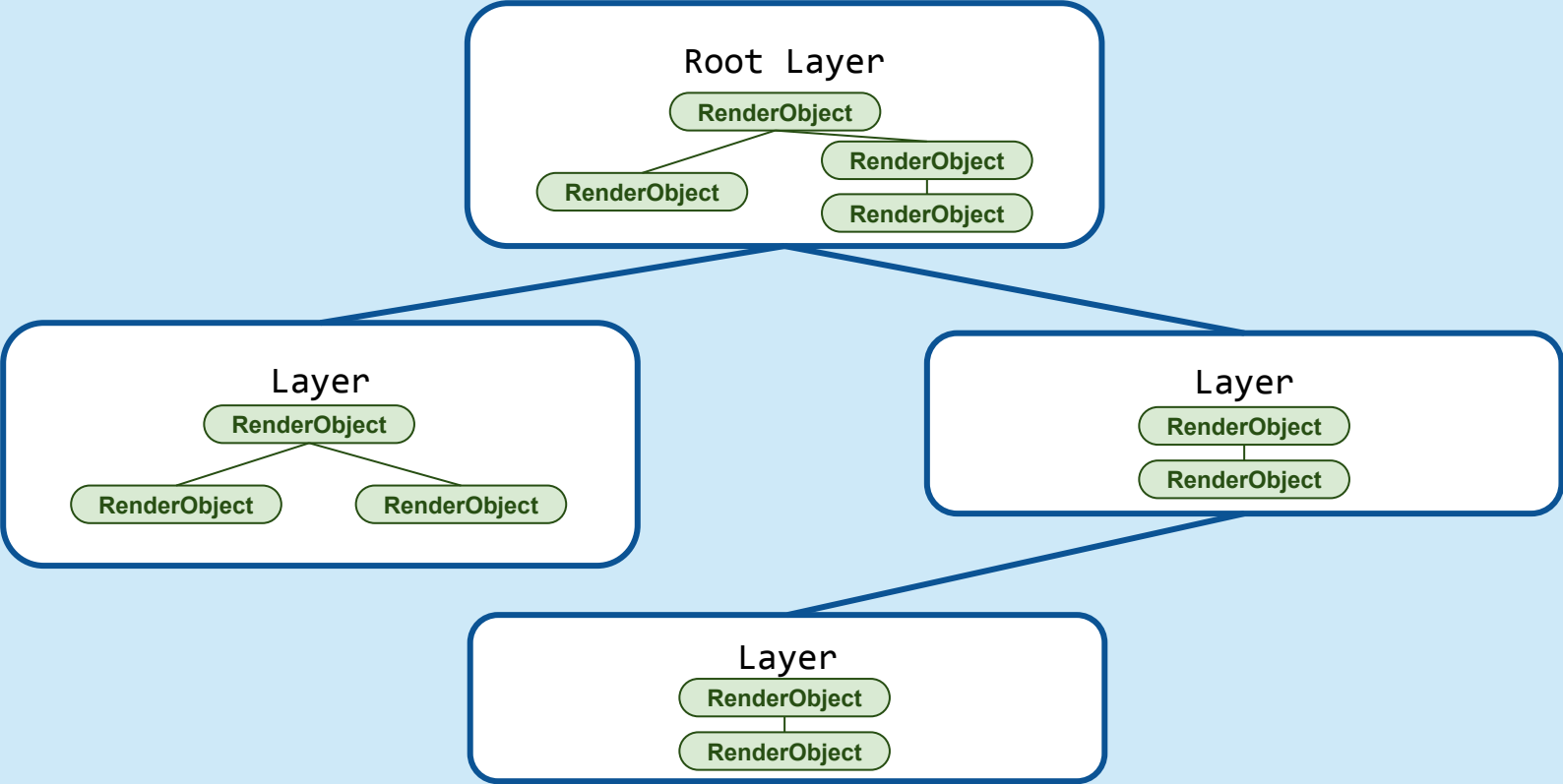


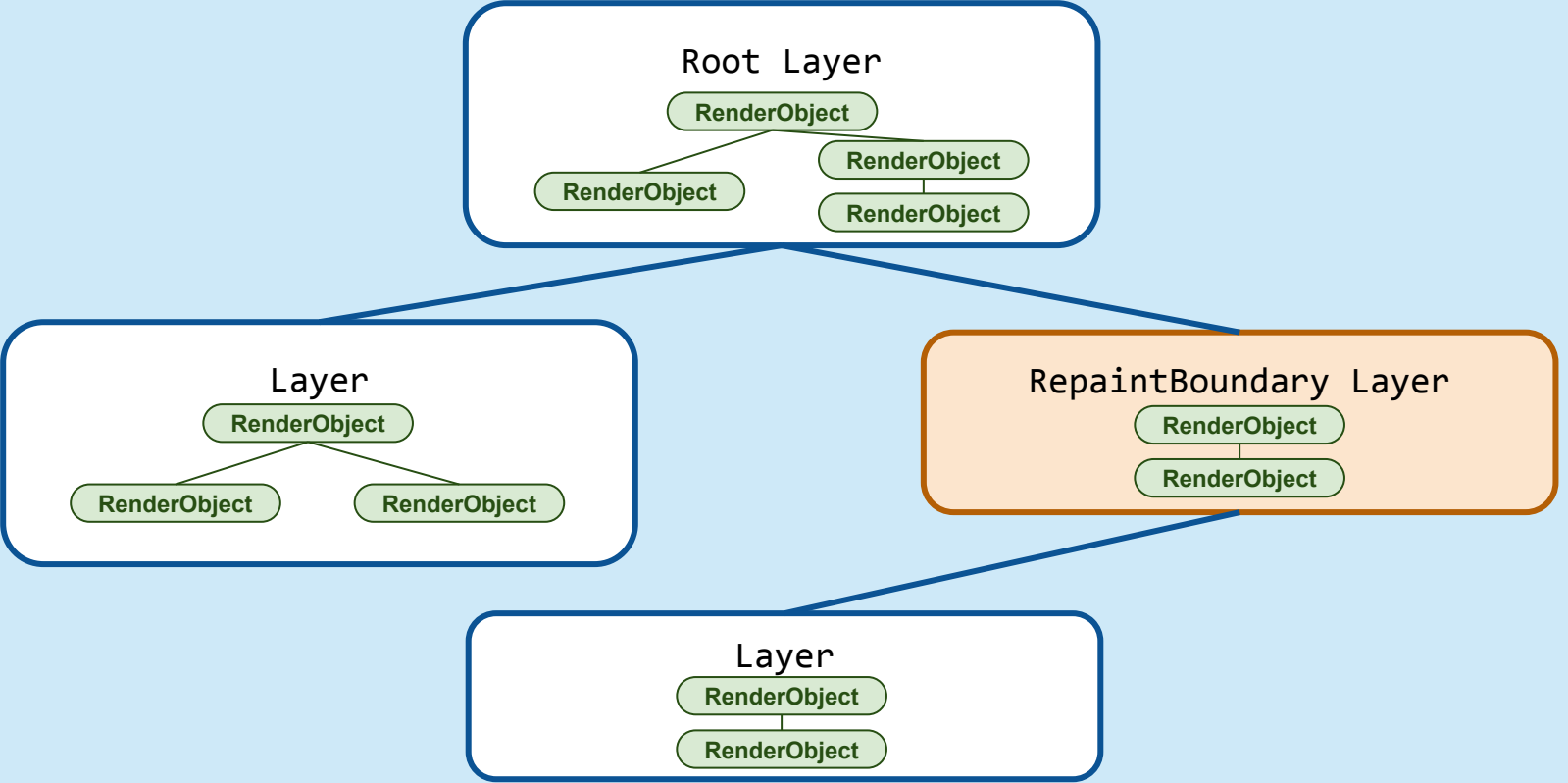


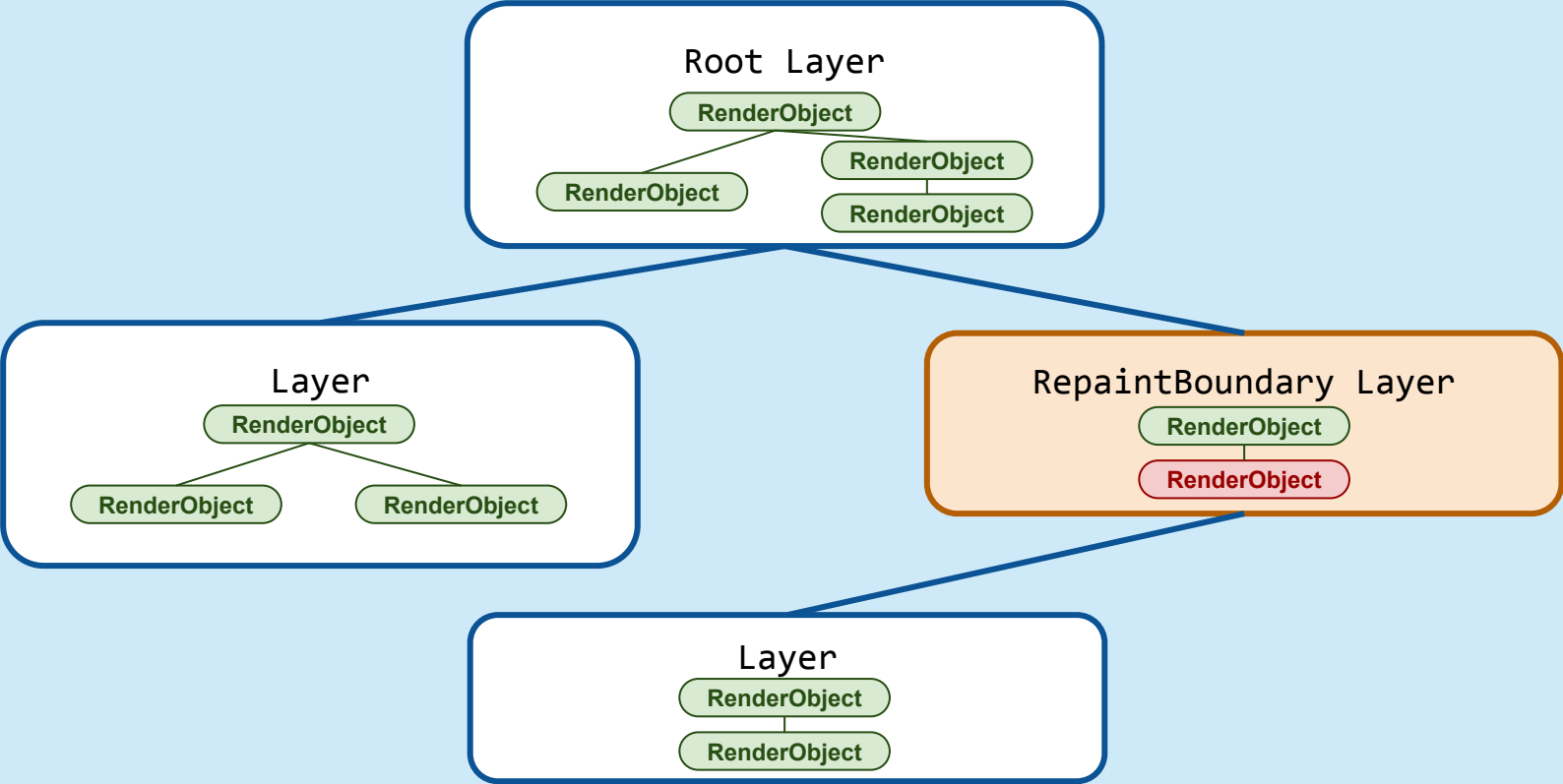
# **1. What happens when nothing is happening?**

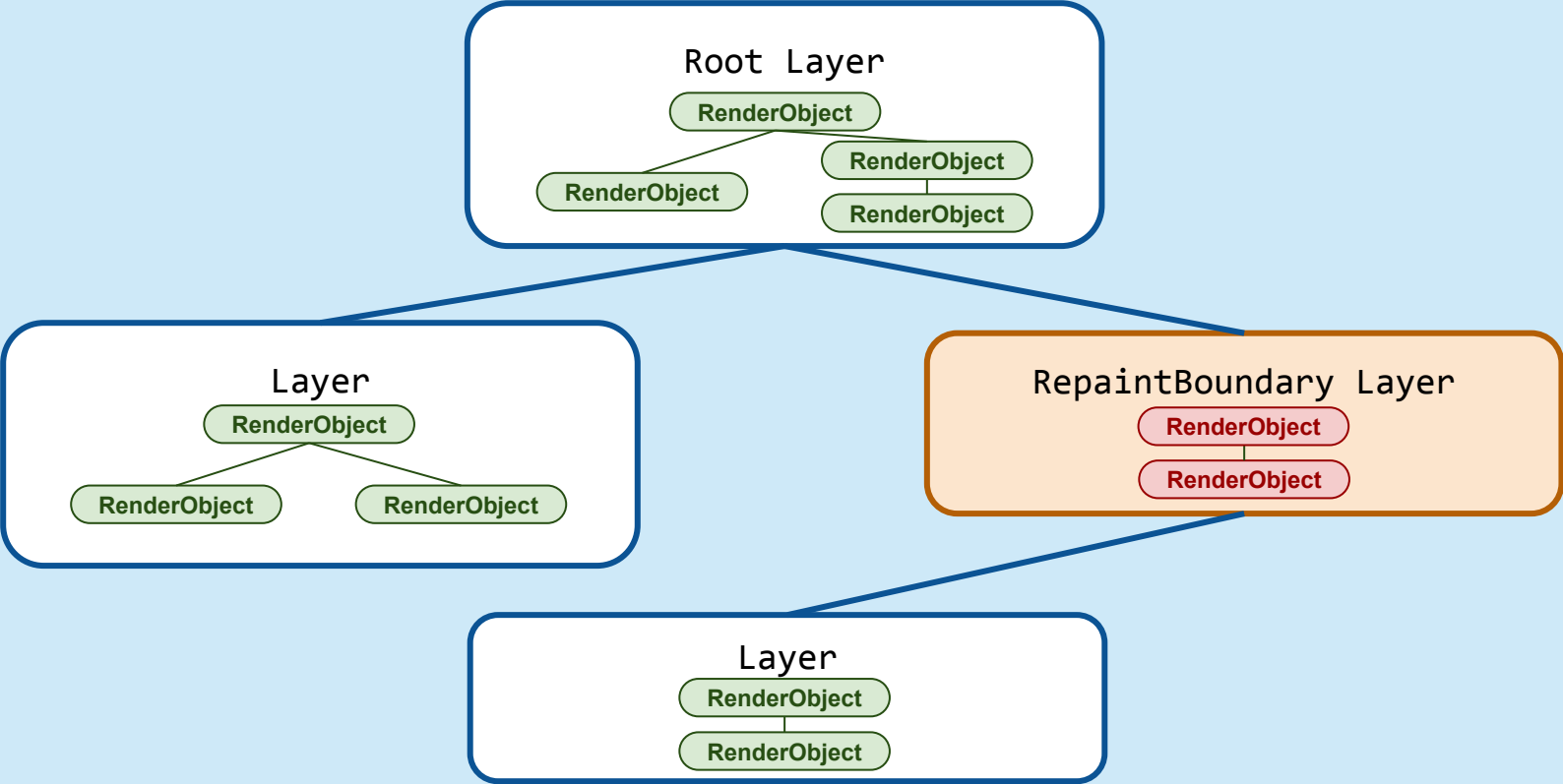


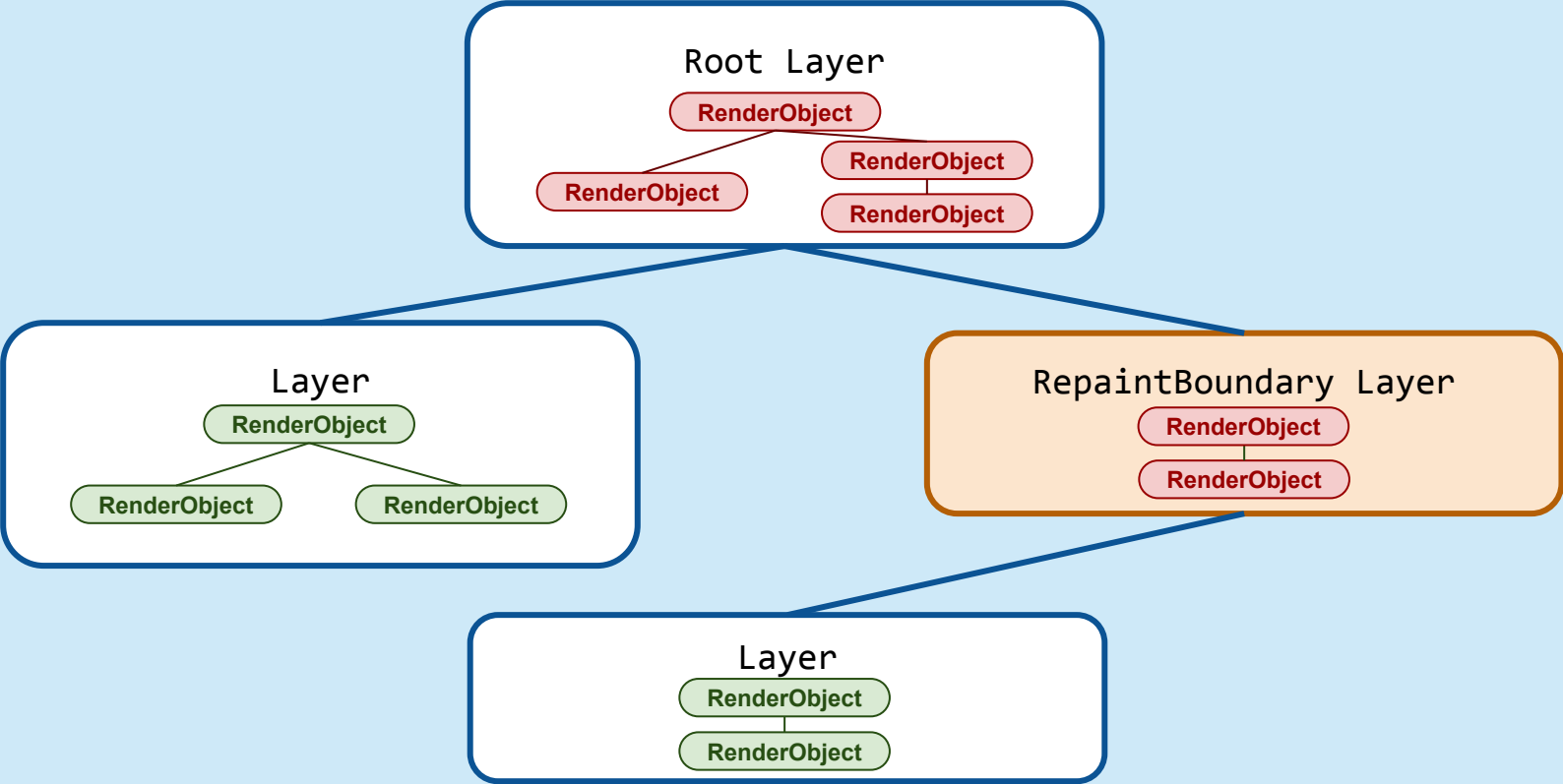
## 2. RepaintBoundary













# Scrolling items





# Scrolling item is animating



```
GestureDetector(  
  onTap: () => setState(() => myColor = getNextColor()),  
  child: ColoredBox(  
    color: myColor  
    child: RepaintBoundary(  
      child: ColoredBox(color: myColor, child: ...),  
    ),  
  ),  
)
```



```
GestureDetector(  
  onTap: () => setState(() => myColor = getNextColor()),  
  child: ColoredBox(  
    color: myColor  
    child: RepaintBoundary(  
      child: ColoredBox(color: myColor, child: ...),  
    ),  
  ),  
)
```



```
GestureDetector(  
  onTap: () => setState(() => myColor = getNextColor()),  
  child: ColoredBox(  
    color: myColor  
    child: RepaintBoundary(  
      child: ColoredBox(color: myColor, child: ...),  
    ),  
  ),  
)
```



```
GestureDetector(  
  onTap: () => setState(() => myColor = getNextColor()),  
  child: ColoredBox(  
    color: myColor  
    child: RepaintBoundary(  
      child: ColoredBox(color: myColor, child: ...),  
    ),  
  ),  
)
```



```
GestureDetector(  
  onTap: () => setState(() => myColor = getNextColor()),  
  child: ColoredBox(  
    color: myColor  
    child: RepaintBoundary(  
      child: ColoredBox(color: myColor, child: ...),  
    ),  
  ),  
)
```



```
GestureDetector(  
  onTap: () => setState(() => myColor = getNextColor()),  
  child: ColoredBox(  
    color: myColor  
    child: RepaintBoundary(  
      child: ColoredBox(color: myColor, child: ...),  
    ),  
  ),  
)
```



Layout Explorer    Widget Details Tree    ⌵ Expand all    ✕ Collapse to selected

📁 [root] > ... > ⌕ Expanded > Ⓢ ScrollingPa... > ⓕ Focus > 📋 ListView

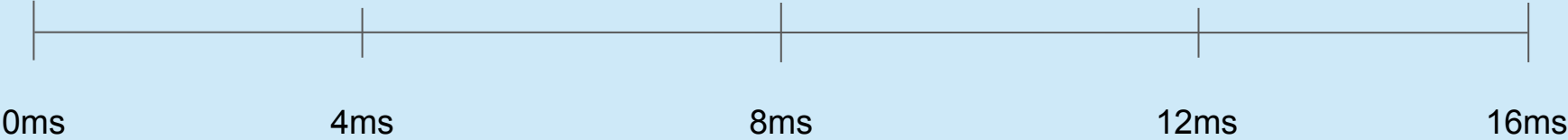
▼ Ⓡ RepaintBoundary

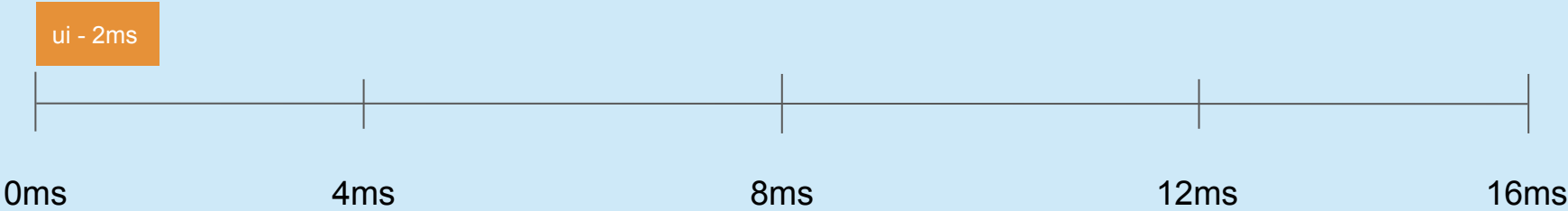
- ▼ renderObject: RenderRepaintBoundary#d3b07 layoutBoundary=up2
  - needs compositing
  - parentData: <none> (can use size)
  - constraints: BoxConstraints(w=250.0, 0.0<=h<=500.0)
  - layer: OffsetLayer#7aab2
  - size: Size(250.0, 500.0)
  - metrics: 99.3% useful (1 bad vs 150 good)
  - diagnosis: this is an outstandingly useful repaint boundary and should definitely be kept
- ▼ Ⓛ Listener
  - listeners: signal
  - behavior: deferToChild
  - > renderObject: RenderPointerListener#e28ab layoutBoundary=up3



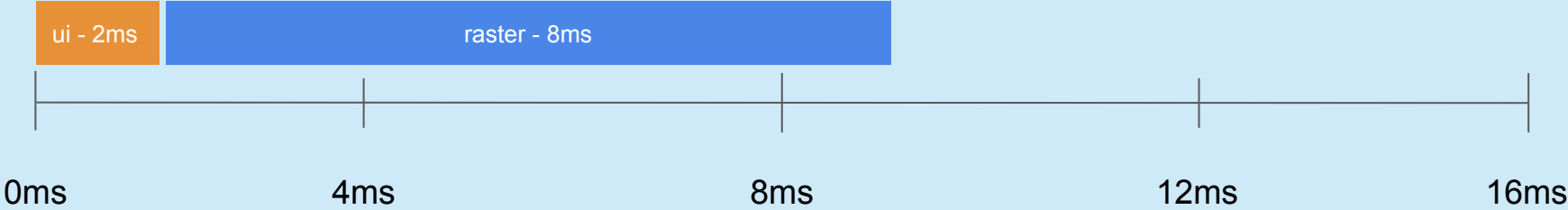


### **3. What does this all mean for performance profiling?**

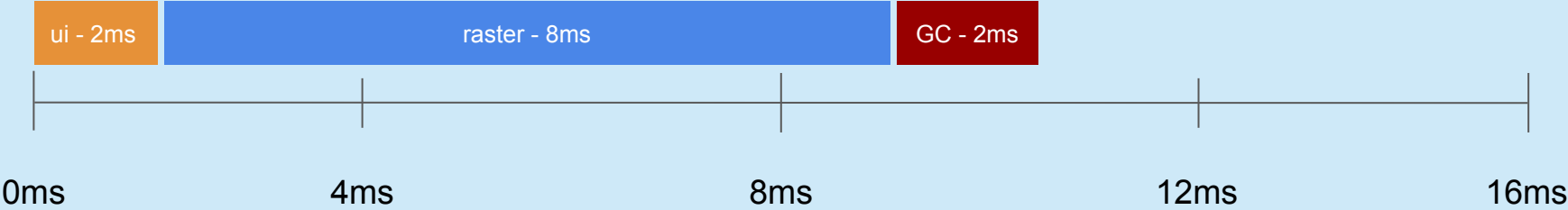




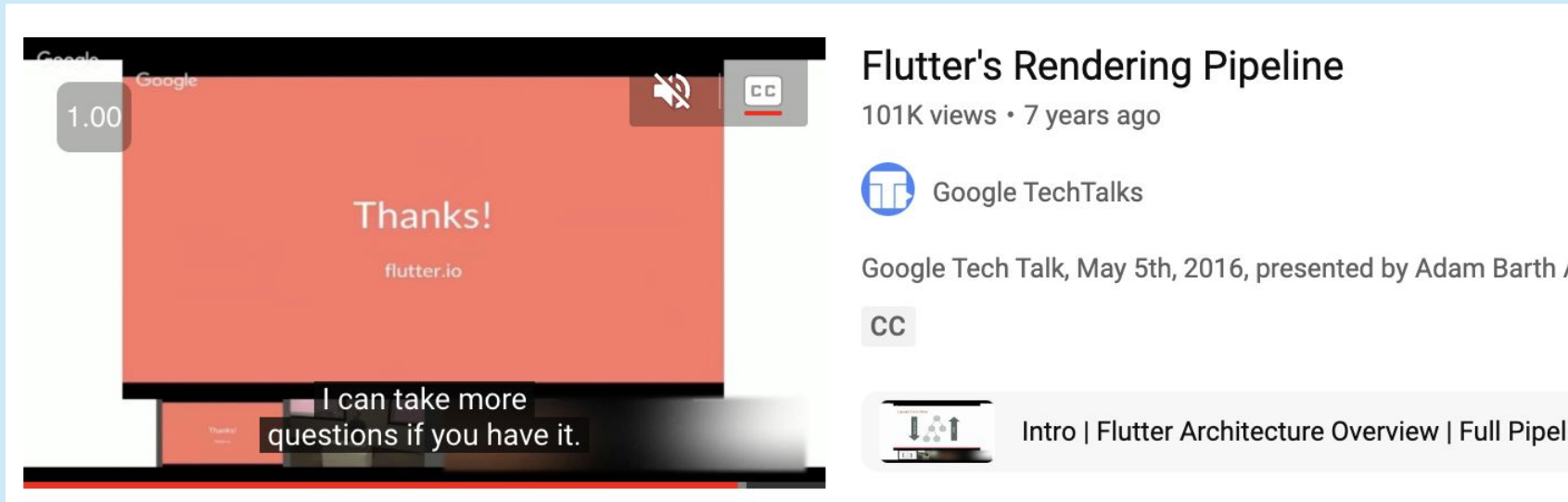
Life cycle of a RenderObject



Life cycle of a RenderObject



<https://www.youtube.com/watch?v=UUfXWzp0-DU>



The image shows a YouTube video player interface. The video player itself is a large red rectangle with the text "Thanks!" in white, and "flutter.io" in a smaller font below it. Above the video, there's a "Google" logo. To the left of the video, there's a "1.00" timestamp. To the right, there are icons for mute and closed captions. Below the video, there's a subtitle that reads "I can take more questions if you have it." To the right of the video player, the video title "Flutter's Rendering Pipeline" is displayed, followed by "101K views • 7 years ago". Below that is the Google TechTalks logo and name. Further down, it says "Google Tech Talk, May 5th, 2016, presented by Adam Barth". There is a "CC" icon for closed captions. At the bottom right, there's a navigation bar with a thumbnail icon and the text "Intro | Flutter Architecture Overview | Full Pipeline".

Thanks!

flutter.io

I can take more questions if you have it.

### Flutter's Rendering Pipeline

101K views • 7 years ago

Google TechTalks

Google Tech Talk, May 5th, 2016, presented by Adam Barth

CC

Intro | Flutter Architecture Overview | Full Pipeline



## Special thanks!



**Dan Field**



**Jonah  
Williams**



**Michael  
Goderbauer**



**Thank you!**