mobius

Профилирование приложений Flutter. Незаслуженно забытые герои.

Андрей
Смирнов

Flutter разработчик, fin hood.team

mobius

В прошлом fullstack c#-developer .net;

В недавнем прошлом frontend разработчик на dart/angular,
а также соавтор курса
flutter mobile developer на платформе otus;

В настоящем flutter developer;

В альтернативном будущем
mobile native developer;

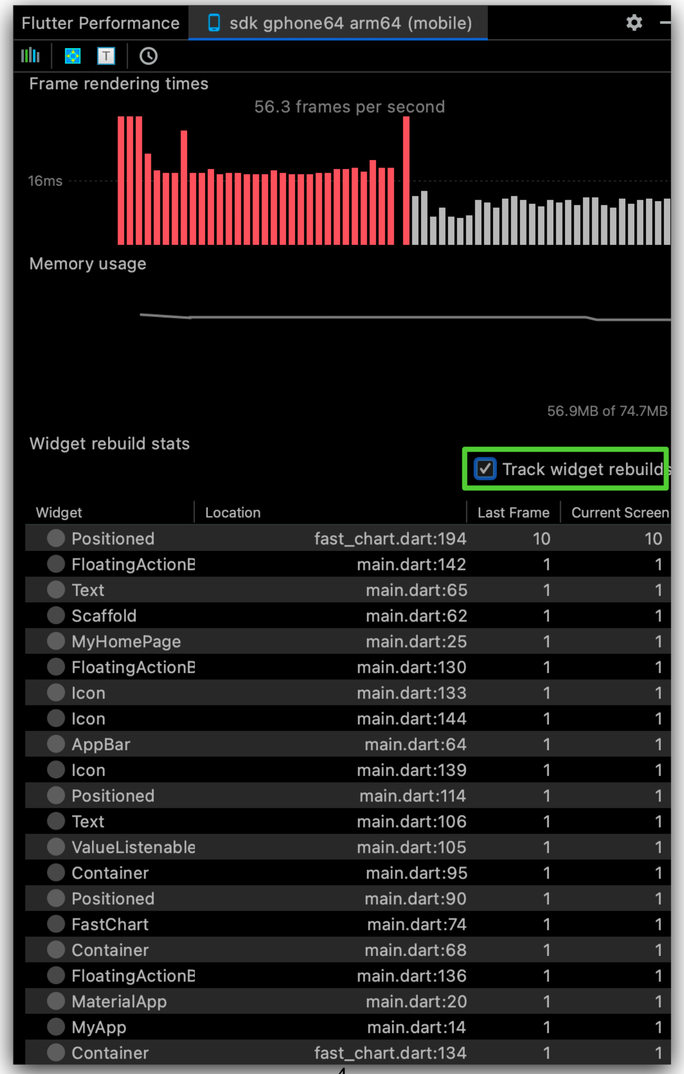
https://twitter.com/dushesatwork

https://t.me/dushes_at_work

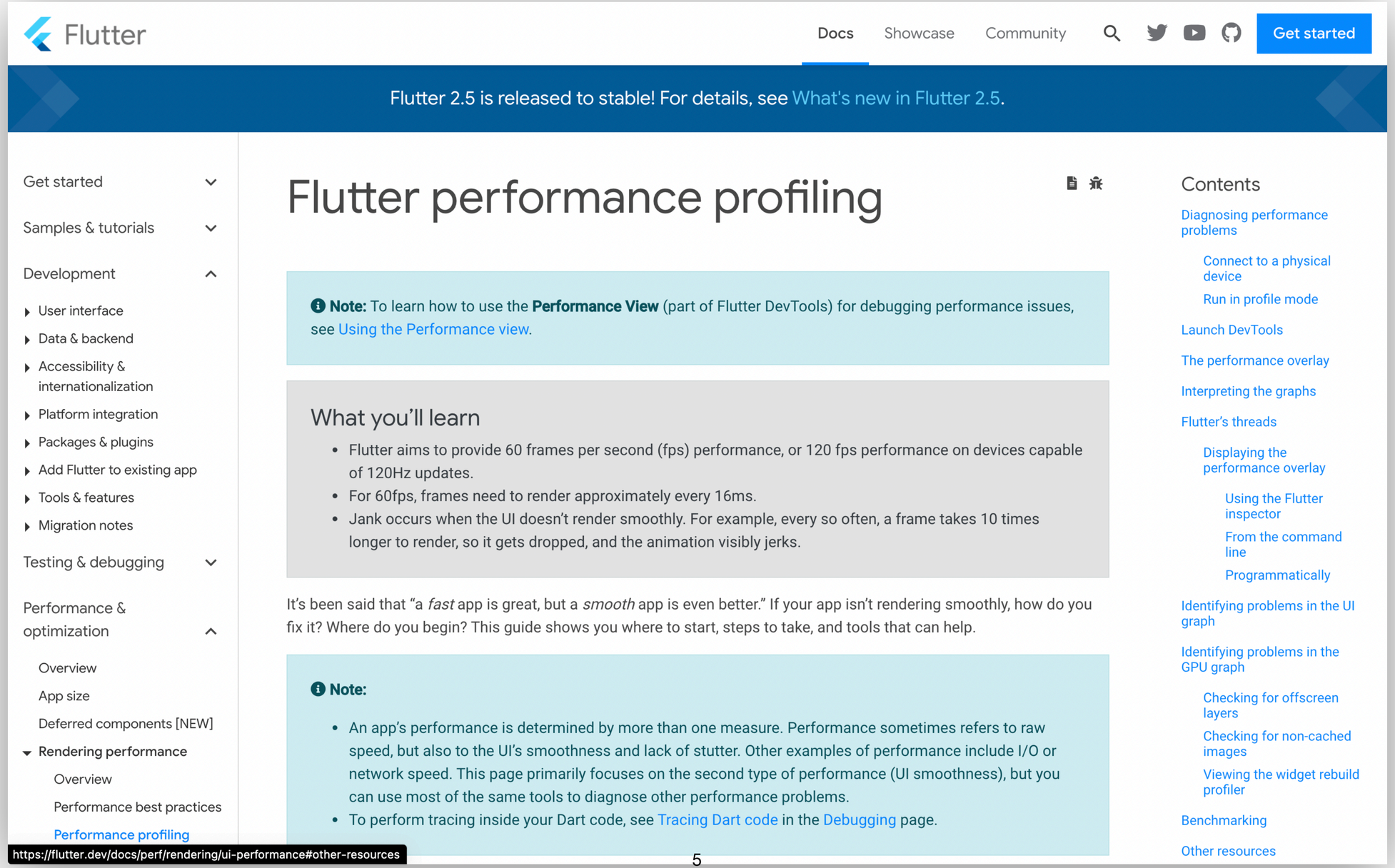
Целью доклада не является пересказ всей документации **flutter.dev**, которая достаточно подробно описывает рекомендации для разработки высокоэффективных приложений

https://flutter.dev/docs/perf/rendering/ui-performance

https://flutter.dev/docs/perf/rendering/ui-performance



https://flutter.dev/docs/perf/rendering/ui-performance



mobius

Начало истории: Fast Chart

mobius

fin_hood

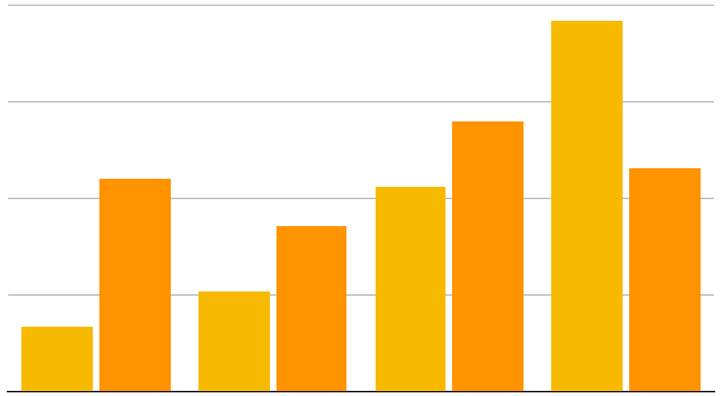
Мы создаем самый
простой и удобный инструмент
для непрофессиональных инвесторов
со всего мира.

finhood.team

Fast Chart -
все имена и события вымышлены,
любые совпадения случайны

Fast Chart -
все имена и события вымышлены,
любые совпадения случайны

Columns Chart



Весь код из демонстраций доступен в репозитории

https://bit.ly/fastchart

Step 1

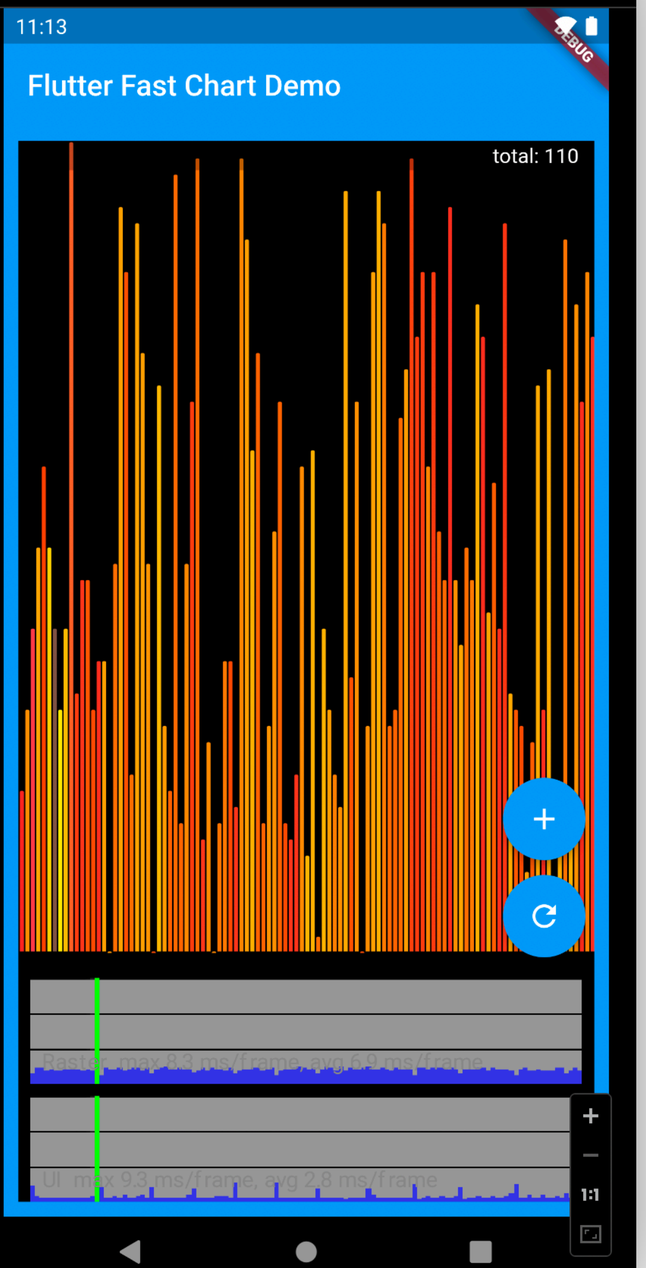
git checkout tags/step1

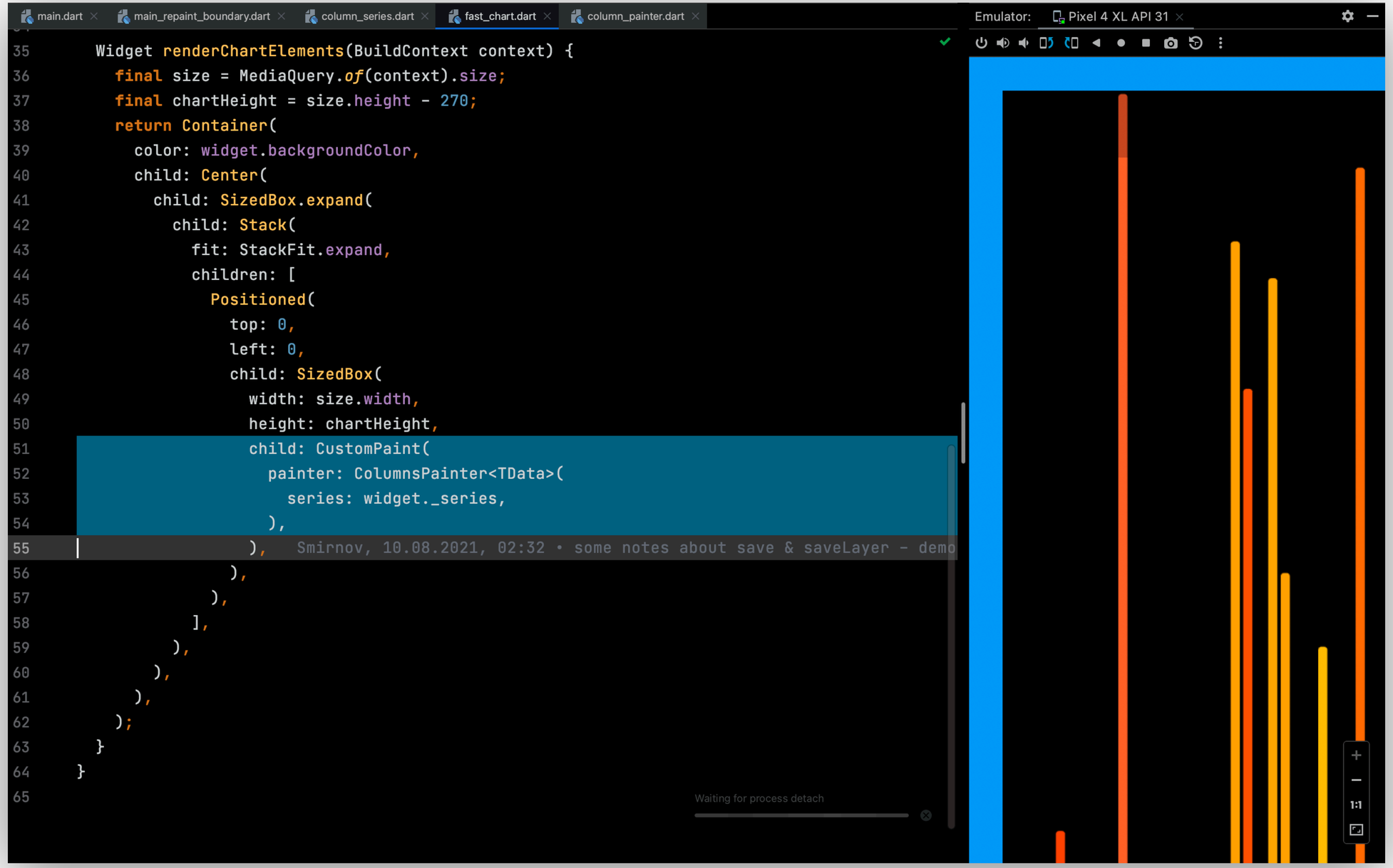
main.dart | main_repaint_boundary.dart | column_series.dart | fast_chart.dart | column_painter.dart

```dart
Widget renderChartElements(BuildContext context) {
  final size = MediaQuery.of(context).size;
  final chartHeight = size.height - 270;
  return Container(
    color: widget.backgroundColor,
    child: Center(
      child: SizedBox.expand(
        child: Stack(
          fit: StackFit.expand,
          children: [
            Positioned(
              top: 0,
              left: 0,
              child: SizedBox(
                width: size.width,
                height: chartHeight,
                child: CustomPaint(
                  painter: ColumnsPainter<TData>(
                    series: widget._series,
                  ),
                ),   Smirnov, 10.08.2021, 02:32 • some notes about save & saveLayer - demo
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
}
```

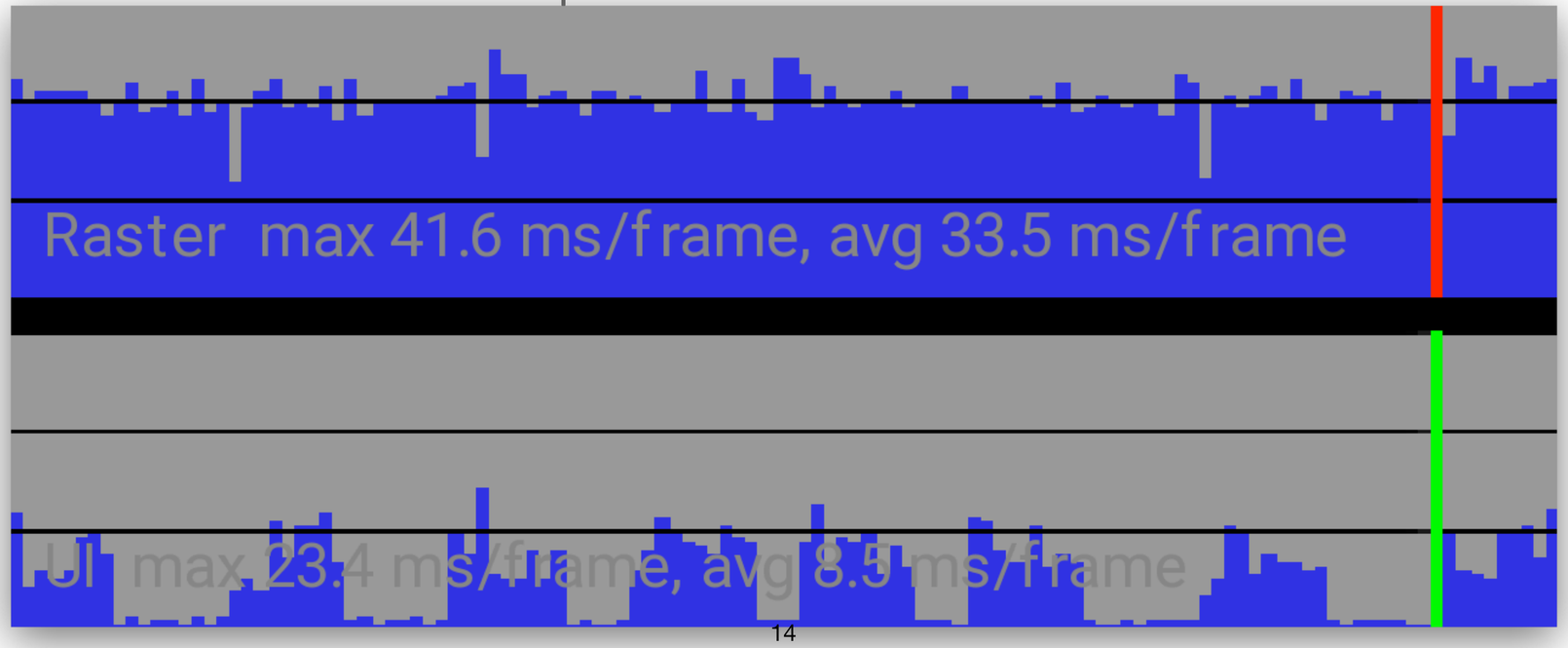
Waiting for process detach

Emulator: Pixel 4 XL API 31




main.dart
main_repaint_boundary.dart
column_series.dart
fast_chart.dart
column_painter.dart
Widget renderChartElements(BuildContext context) {
final size = MediaQuery.of(context).size;
final chartHeight = size.height - 270;
return Container(
color: widget.backgroundColor,
child: Center(
child: SizedBox.expand(
child: Stack(
fit: StackFit.expand,
children: [
Positioned(
top: 0,
left: 0,
child: SizedBox(
width: size.width,
height: chartHeight,
child: CustomPaint(
painter: ColumnsPainter<TData>(
series: widget._series,
),
),
Smirnov, 10.08.2021, 02:32 • some notes about save & saveLayer - demo
),
),
],
),
),
),
);
}
}
Waiting for process detach
Emulator:
Pixel 4 XL API 31
1:1

Step 1 - Flutter Threads



Step 1 - Flutter Threads

Flutter's threads

Flutter uses several threads to do its work, though only two of the threads are shown in the overlay. All of your Dart code runs on the UI thread. Although you have no direct access to any other thread, your actions on the UI thread have performance consequences on other threads.

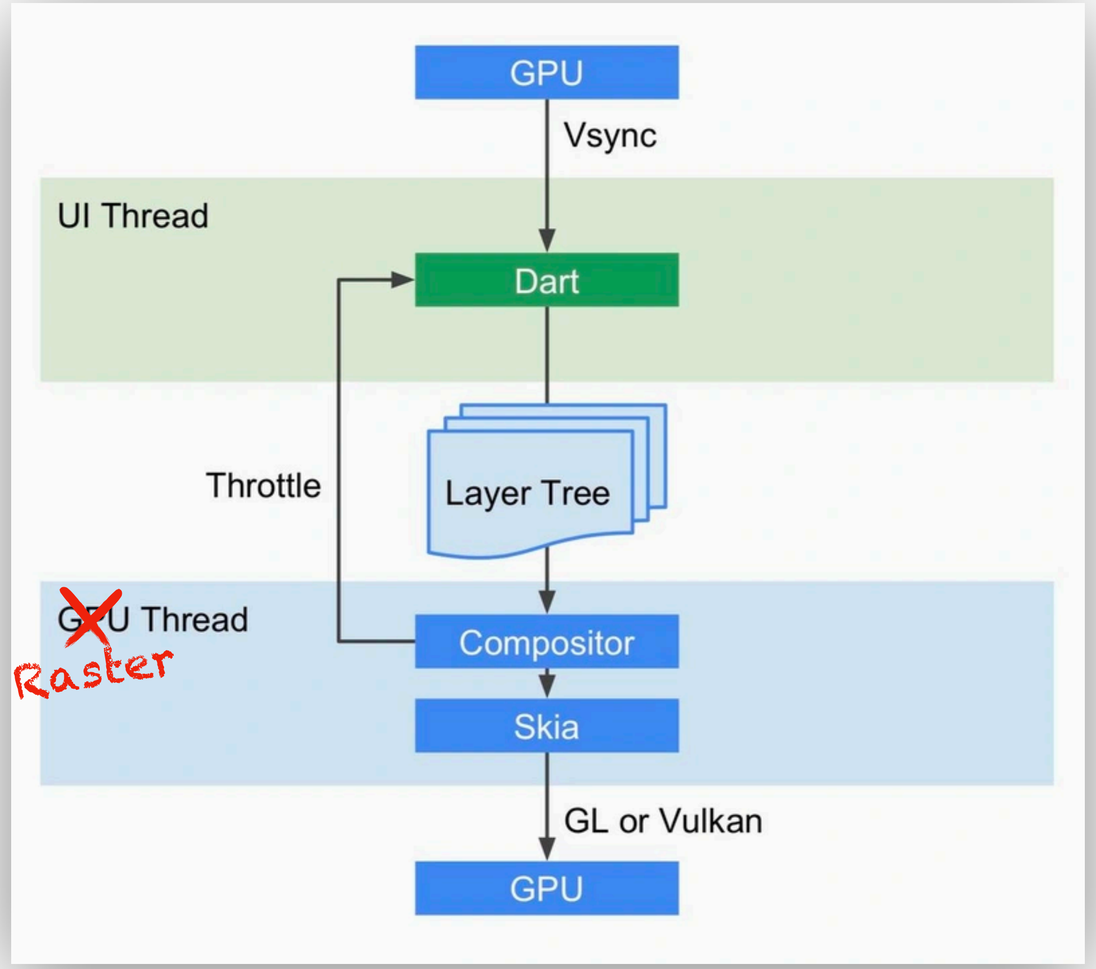
Platform thread
The platform's main thread. Plugin code runs here. For more information, see the UIKit documentation for iOS, or the MainThread documentation for Android. This thread is not shown in the performance overlay.

UI thread
The UI thread executes Dart code in the Dart VM. This thread includes code that you wrote, and code executed by Flutter's framework on your app's behalf. When your app creates and displays a scene, the UI thread creates a *layer tree*, a lightweight object containing device-agnostic painting commands, and sends the layer tree to the raster thread to be rendered on the device. *Don't block this thread!* Shown in the bottom row of the performance overlay.

Raster thread (previously known as the GPU thread)
The raster thread takes the layer tree and displays it by talking to the GPU (graphic processing unit). You cannot directly access the raster thread or its data but, if this thread is slow, it's a result of something you've done in the Dart code. Skia, the graphics library, runs on this thread. Shown in the top row of the performance overlay. This thread was previously known as the "GPU thread" because it rasterizes for the GPU. But it is running on the CPU. We renamed it to "raster thread" because many developers wrongly (but understandably) assumed the thread runs on the GPU unit.

I/O thread
Performs expensive tasks (mostly I/O) that would otherwise block either the UI or raster threads. This thread is not shown in the performance overlay.

For links to more information and videos, see The Framework architecture on the GitHub wiki, and the community article, The Layer Cake.


GPU
Vsync
UI Thread
Dart
Throttle
Layer Tree
GPU Thread
Raster
Compositor
Skia
GL or Vulkan
GPU

Displaying the performance overlay

You can toggle display of the performance overlay as follows:

- Using the Flutter inspector
- From the command line
- Programmatically

Using the Flutter inspector

The easiest way to enable the PerformanceOverlay widget is from the Flutter inspector, which is available in the Inspector view in DevTools. Simply click the **Performance Overlay** button to toggle the overlay on your running app.

From the command line

Toggle the performance overlay using the **P** key from the command line.

Programmatically

To enable the overlay programmatically, see Performance overlay, a section in the Debugging Flutter apps programmatically page.

The performance overlay

The performance overlay displays statistics in two graphs that show where time is being spent in your app. If the UI is janky (skipping frames), these graphs help you figure out why. The graphs display on top of your running app, but they aren't drawn like a normal widget—the Flutter engine itself paints the overlay and only minimally impacts performance. Each graph represents the last 300 frames for that thread.

This section describes how to enable the performance overlay and use it to diagnose the cause of jank in your application. The following screenshot shows the performance overlay running on the Flutter Gallery example:

Performance overlay showing the raster thread (top), and UI thread (bottom).
The vertical green bars represent the current frame.

Interpreting the graphs

The top graph (marked “GPU”) shows the time spent by the raster thread, the bottom one graph shows the time spent by the UI thread. The white lines across the graphs show 16ms increments along the vertical axis; if the graph ever goes over one of these lines then you are running at less than 60Hz. The horizontal axis represents frames. The graph is only updated when your application paints, so if it’s idle the graph stops moving.

The overlay should always be viewed in profile mode, since debug mode performance is intentionally sacrificed in exchange for expensive asserts that are intended to aid development, and thus the results are misleading.

Each frame should be created and displayed within 1/60th of a second (approximately 16ms). A frame exceeding this limit (in either graph) fails to display, resulting in jank, and a vertical red bar appears in one or both of the graphs. If a red bar appears in the UI graph, the Dart code is too expensive. If a red vertical bar appears in the GPU graph, the scene is too complicated to render quickly.

The vertical red bars indicate that the current frame is expensive to both render and paint.
When both graphs display red, start by diagnosing the UI thread.

Step 1 - Flutter Threads

https://flutter.dev/docs/perf/rendering/ui-performance

https://github.com/flutter/flutter/wiki/The-Engine-architecture

Junks

git checkout tags/junks_demo

https://flutter.dev/docs/development/tools/devtools/performance

UI

The UI thread executes Dart code in the Dart VM. This includes code from your application as well as the Flutter framework. When your app creates and displays a scene, the UI thread creates a layer tree, a lightweight object containing device-agnostic painting commands, and sends the layer tree to the raster thread to be rendered on the device. Do **not** block this thread.

Raster

The raster thread (previously known as the GPU thread) executes graphics code from the Flutter Engine. This thread takes the layer tree and displays it by talking to the GPU (graphic processing unit). You cannot directly access the raster thread or its data, but if this thread is slow, it's a result of something you've done in the Dart code. Skia, the graphics library, runs on this thread.

Sometimes a scene results in a layer tree that is easy to construct, but expensive to render on the raster thread. In this case, you need to figure out what your code is doing that is causing rendering code to be slow. Specific kinds of workloads are more difficult for the GPU. They might involve unnecessary calls to `saveLayer()`, intersecting opacities with multiple objects, and clips or shadows in specific situations.

For more information on profiling, see Identifying problems in the GPU graph.

Jank

The frame rendering chart shows jank with a red overlay. A frame is considered to be janky if it takes more than ~16 ms to complete (for 60 FPS devices). To achieve a frame rendering rate of 60 FPS (frames per second), each frame must render in ~16 ms or less. When this target is missed, you may experience UI jank or dropped frames.

For more information on how to analyze your app's performance, see Flutter performance profiling.

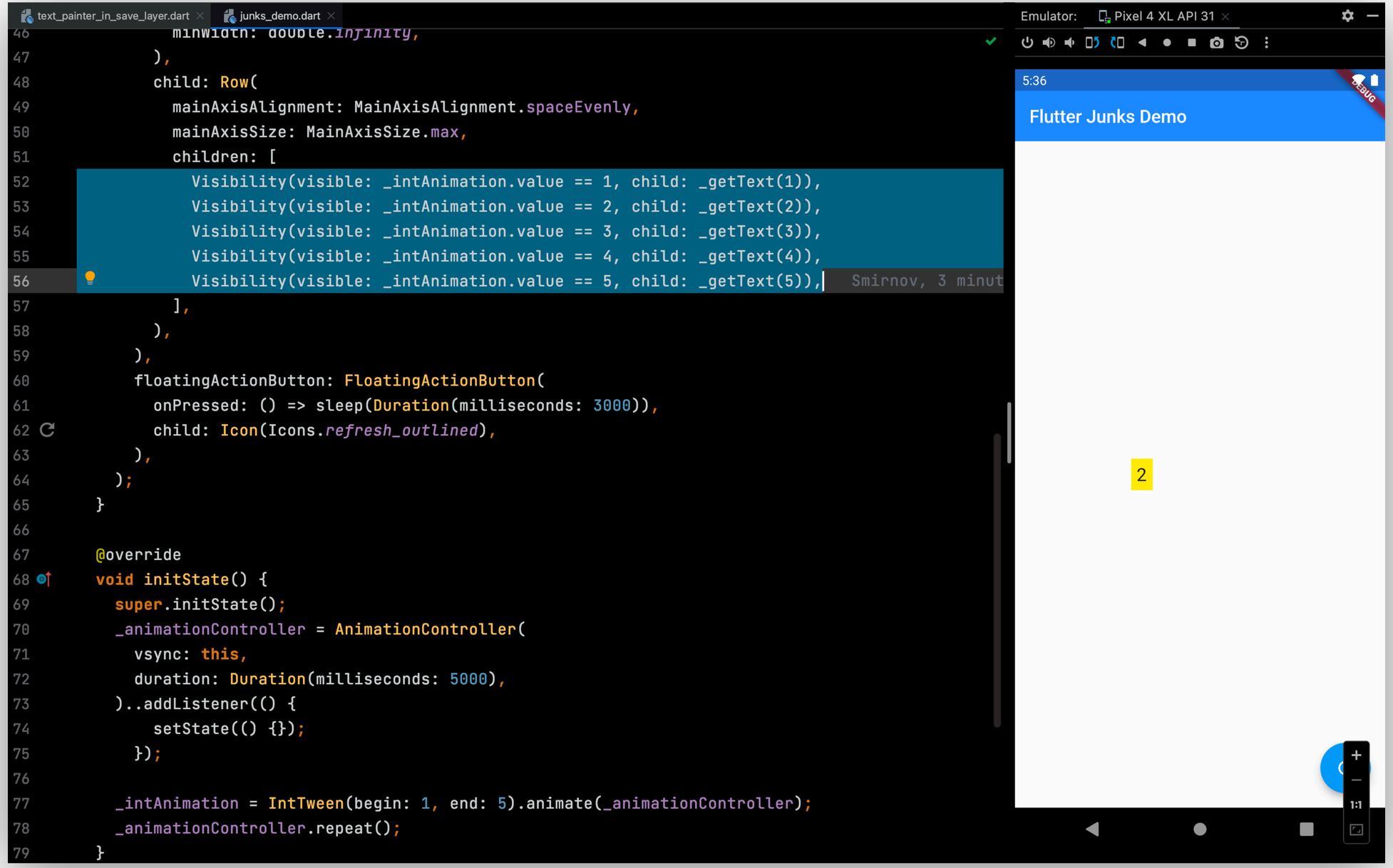
Numeric Animation

Junked Numeric Animation

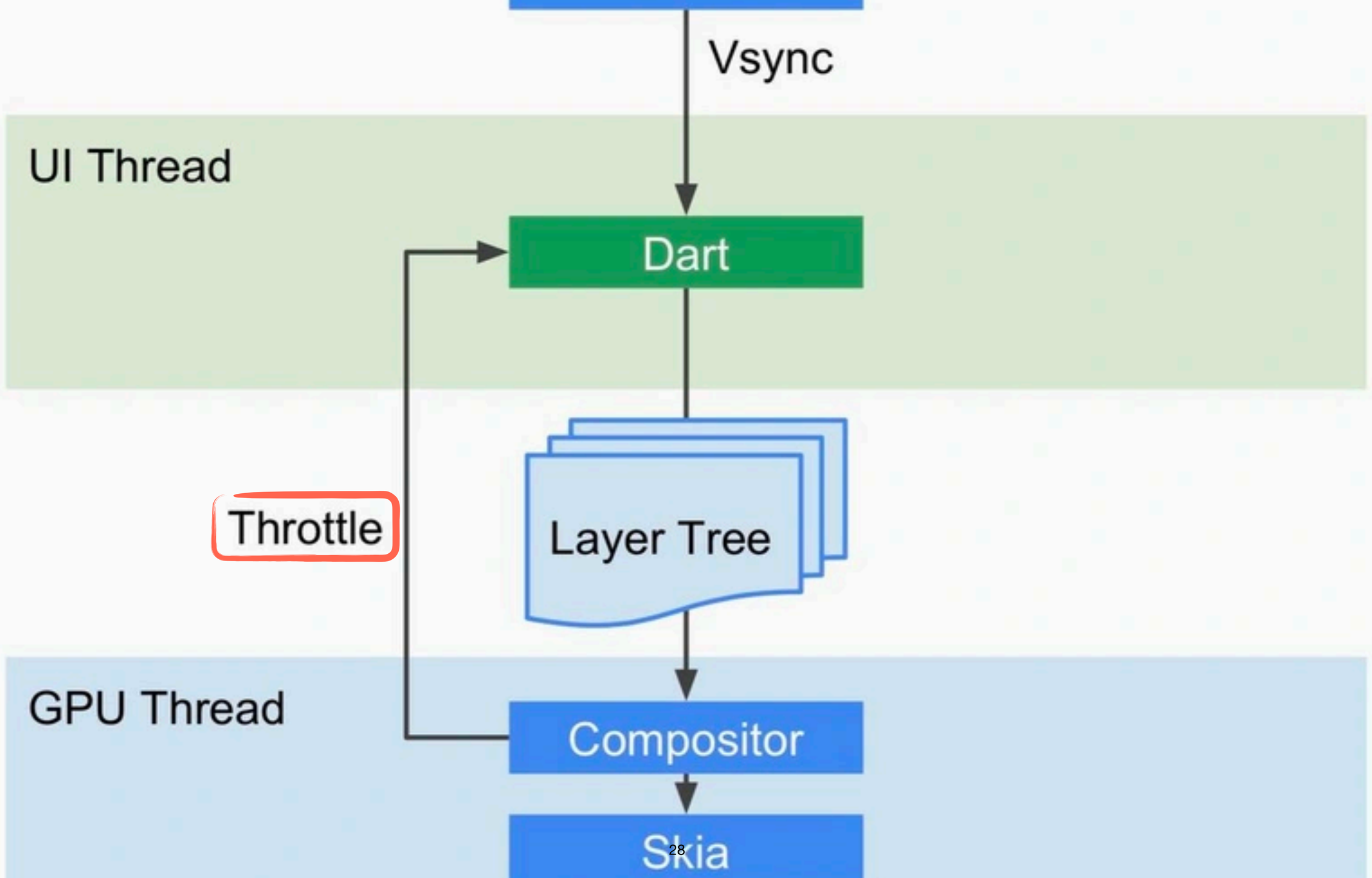
1

5

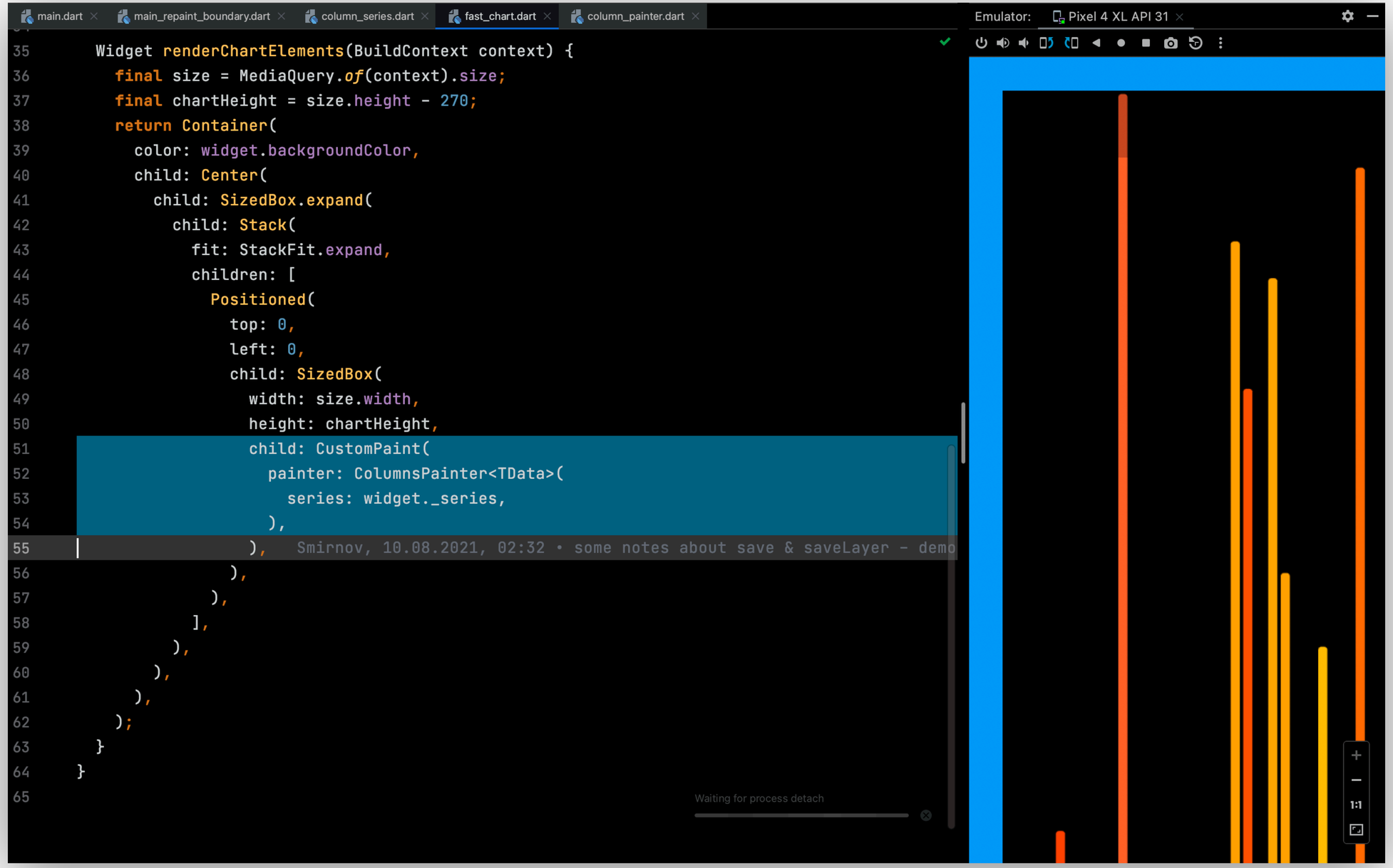
Внезапно....(


text_painter_in_save_layer.dart
junks_demo.dart
minWidth: double.infinity,
),
child: Row(
mainAxisAlignment: MainAxisAlignment.spaceEvenly,
mainAxisSize: MainAxisSize.max,
children: [
Visibility(visible: _intAnimation.value == 1, child: _getText(1)),
Visibility(visible: _intAnimation.value == 2, child: _getText(2)),
Visibility(visible: _intAnimation.value == 3, child: _getText(3)),
Visibility(visible: _intAnimation.value == 4, child: _getText(4)),
Visibility(visible: _intAnimation.value == 5, child: _getText(5)),
Smirnov, 3 minut
],
),
),
floatingActionButton: FloatingActionButton(
onPressed: () => sleep(Duration(milliseconds: 3000)),
child: Icon(Icons.refresh_outlined),
),
);
}
@override
void initState() {
super.initState();
_animationController = AnimationController(
vsync: this,
duration: Duration(milliseconds: 5000),
)..addListener(() {
setState(() {});
});
_intAnimation = IntTween(begin: 1, end: 5).animate(_animationController);
_animationController.repeat();
}
Emulator:
Pixel 4 XL API 31
5:36
DEBUG
Flutter Junks Demo
2
1:1

Почему так происходит?

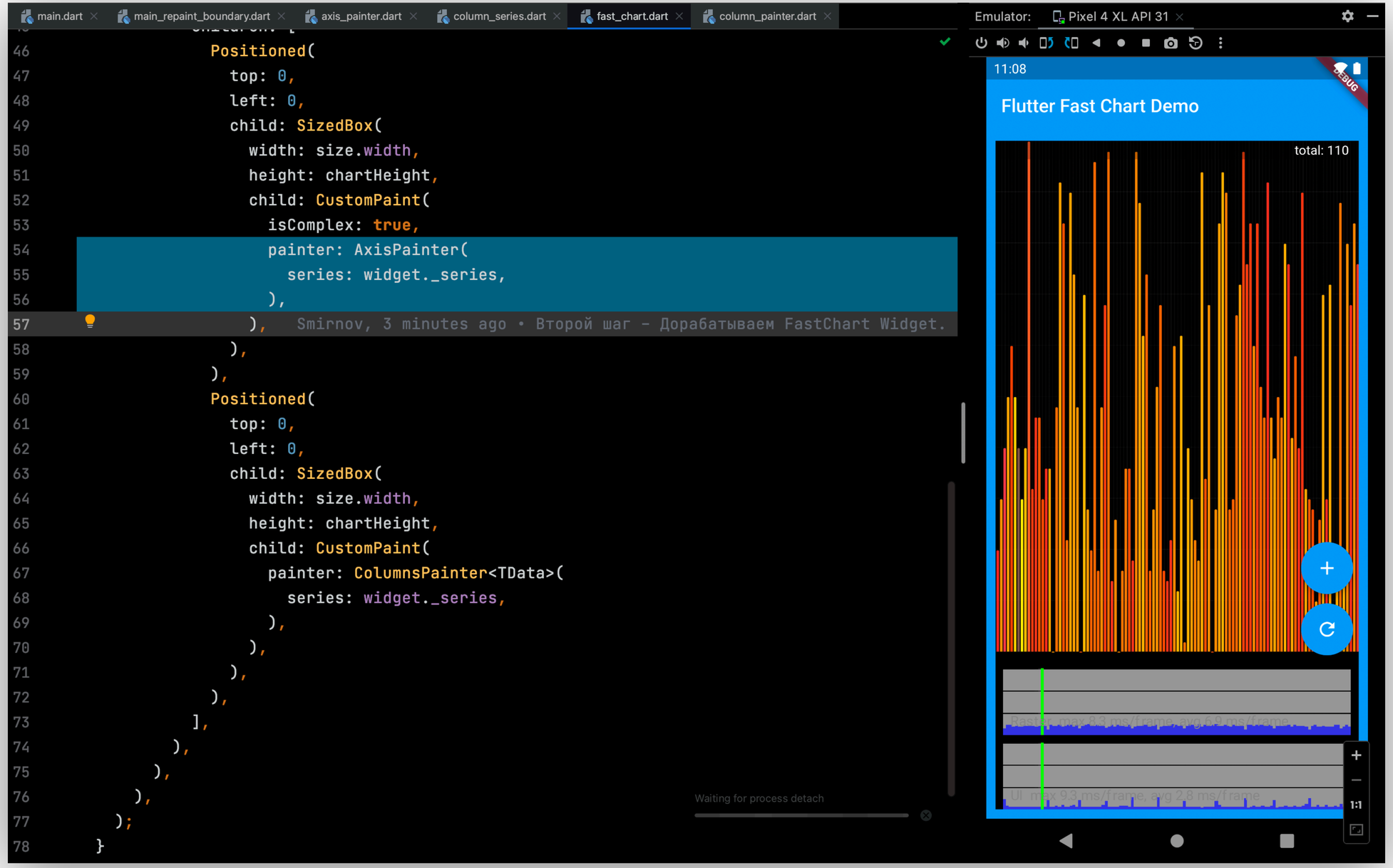

Vsync
UI Thread
Dart
Throttle
Layer Tree
GPU Thread
Compositor
Skia

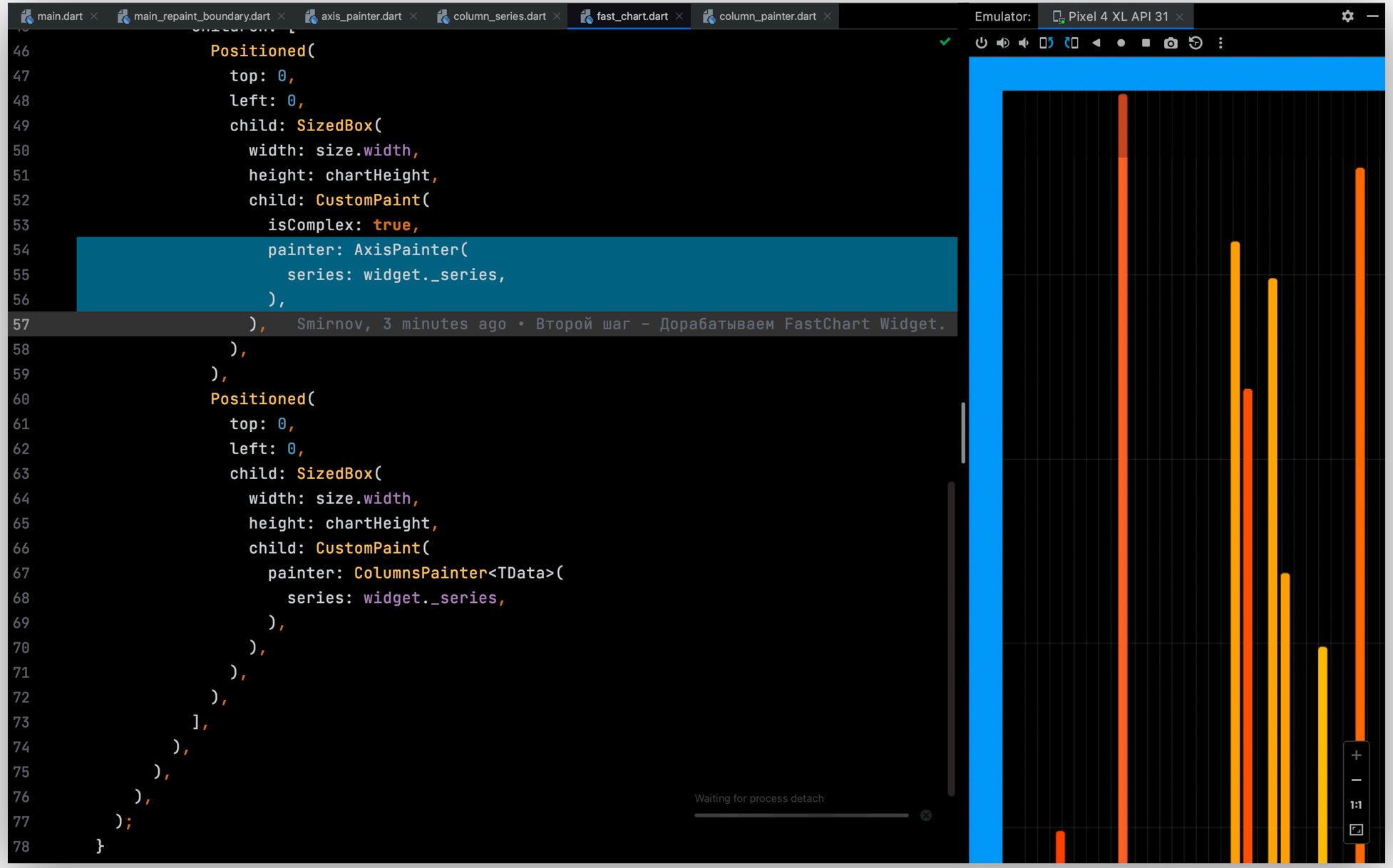
Погрузились немного в теорию,
давайте вернемся к практике


main.dart
main_repaint_boundary.dart
column_series.dart
fast_chart.dart
column_painter.dart
Widget renderChartElements(BuildContext context) {
final size = MediaQuery.of(context).size;
final chartHeight = size.height - 270;
return Container(
color: widget.backgroundColor,
child: Center(
child: SizedBox.expand(
child: Stack(
fit: StackFit.expand,
children: [
Positioned(
top: 0,
left: 0,
child: SizedBox(
width: size.width,
height: chartHeight,
child: CustomPaint(
painter: ColumnsPainter<TData>(
series: widget._series,
),
),
Smirnov, 10.08.2021, 02:32 • some notes about save & saveLayer - demo
),
),
],
),
),
),
);
}
}
Waiting for process detach
Emulator:
Pixel 4 XL API 31
1:1

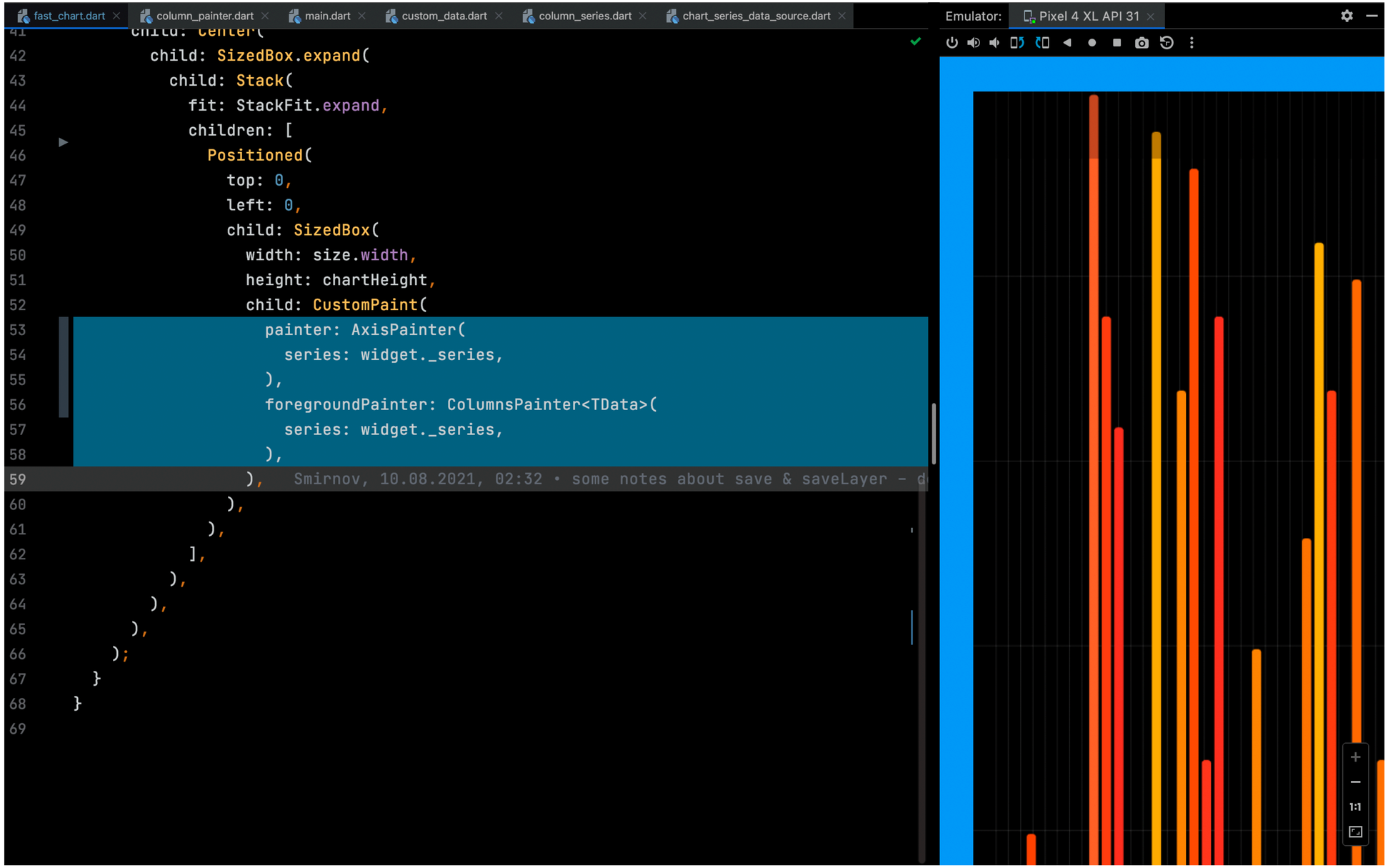
Step 2
axis painter

git checkout tags/step2


main.dart
main_repaint_boundary.dart
axis_painter.dart
column_series.dart
fast_chart.dart
column_painter.dart
Positioned(
top: 0,
left: 0,
child: SizedBox(
width: size.width,
height: chartHeight,
child: CustomPaint(
isComplex: true,
painter: AxisPainter(
series: widget._series,
),
),
Smirnov, 3 minutes ago • Второй шаг - Дорабатываем FastChart Widget.
),
),
Positioned(
top: 0,
left: 0,
child: SizedBox(
width: size.width,
height: chartHeight,
child: CustomPaint(
painter: ColumnsPainter<TData>(
series: widget._series,
),
),
),
),
],
),
),
),
);
}
Waiting for process detach
Emulator:
Pixel 4 XL API 31
11:08
DEBUG
Flutter Fast Chart Demo
total: 110
Raster max 83 ms/frame, avg 6.9 ms/frame.
UI max 93 ms/frame, avg 2.8 ms/frame.
1:1


main.dart
main_repaint_boundary.dart
axis_painter.dart
column_series.dart
fast_chart.dart
column_painter.dart
Emulator:
Pixel 4 XL API 31
Positioned(
top: 0,
left: 0,
child: SizedBox(
width: size.width,
height: chartHeight,
child: CustomPaint(
isComplex: true,
painter: AxisPainter(
series: widget._series,
),
),
Smirnov, 3 minutes ago • Второй шаг - Дорабатываем FastChart Widget.
),
),
Positioned(
top: 0,
left: 0,
child: SizedBox(
width: size.width,
height: chartHeight,
child: CustomPaint(
painter: ColumnsPainter<TData>(
series: widget._series,
),
),
),
),
],
),
),
),
);
}
Waiting for process detach
1:1

git checkout tags/step2_foreground

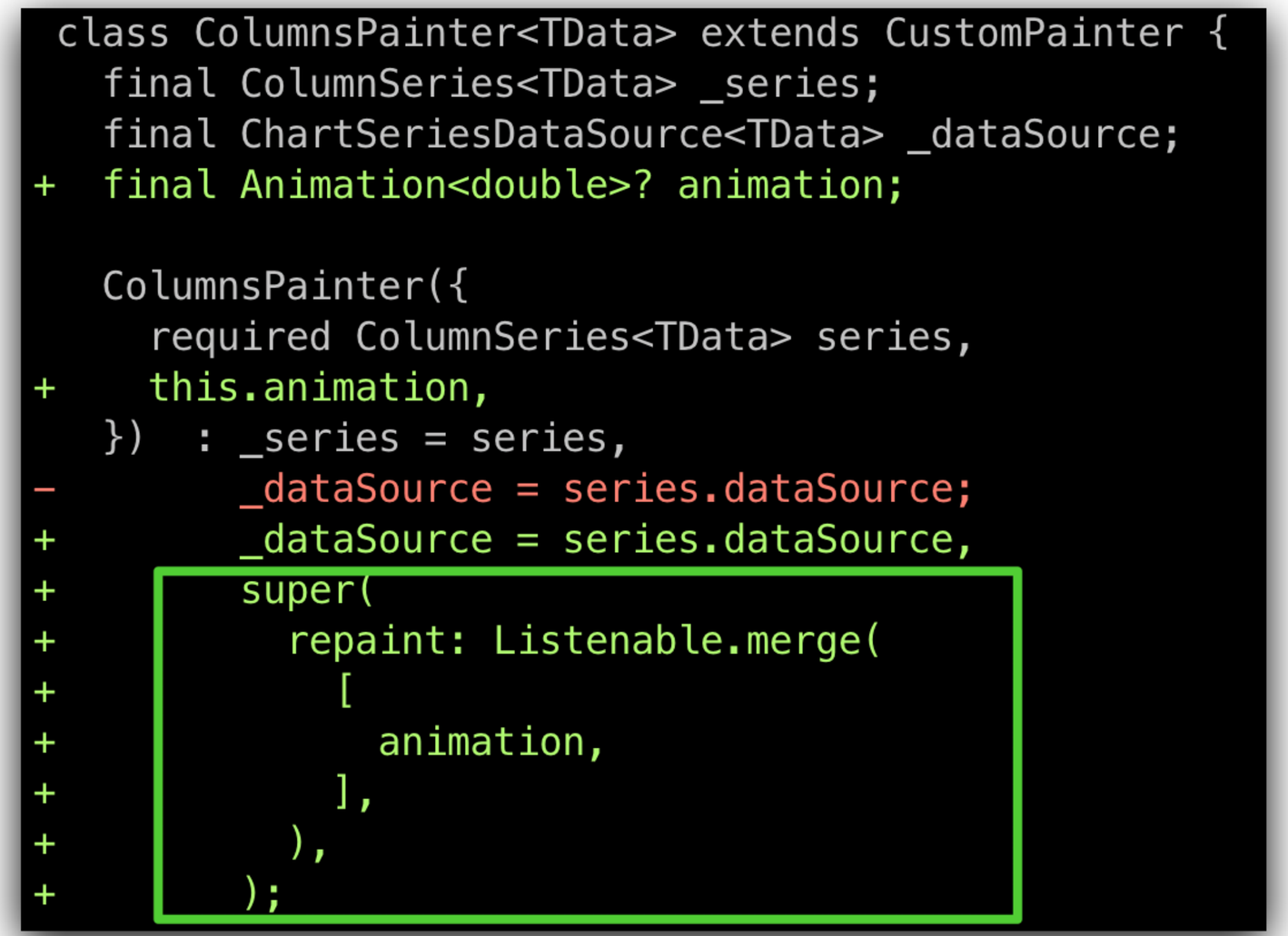

fast_chart.dart
column_painter.dart
main.dart
custom_data.dart
column_series.dart
chart_series_data_source.dart
child: Center(
child: SizedBox.expand(
child: Stack(
fit: StackFit.expand,
children: [
Positioned(
top: 0,
left: 0,
child: SizedBox(
width: size.width,
height: chartHeight,
child: CustomPaint(
painter: AxisPainter(
series: widget._series,
),
foregroundPainter: ColumnsPainter<TData>(
series: widget._series,
),
),
Smirnov, 10.08.2021, 02:32 • some notes about save & saveLayer - d
),
),
),
],
),
),
),
),
);
}
}
Emulator:
Pixel 4 XL API 31
1:1

Step 3
animation

git checkout tags/step3

git diff tags/step2 tags/step3

```dart
abstract class CustomPainter extends Listenable {
  /// Creates a custom painter.
  ///
  /// The painter will repaint whenever `repaint` notifies its listeners.
  const CustomPainter({ Listenable? repaint }) : _repaint = repaint;

```



Здесь я немного извратился и использовал Listenable.merge()

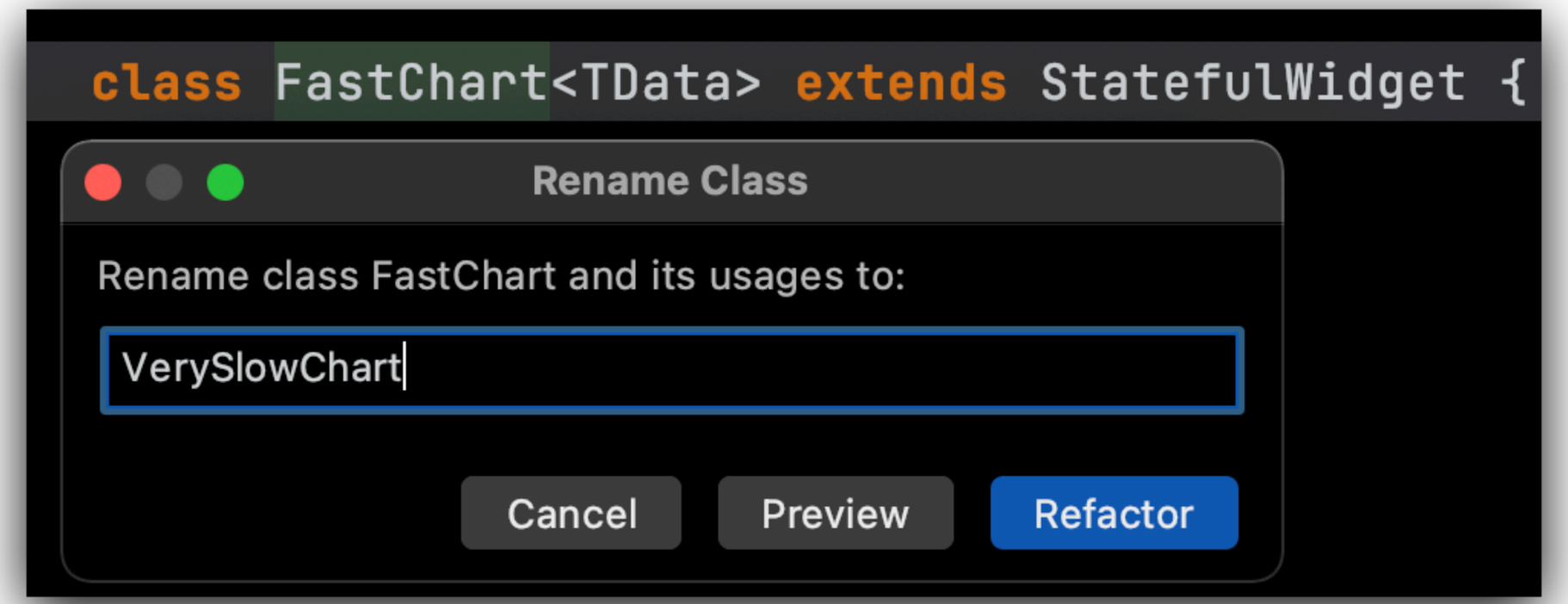
```
   @override
   void paint(Canvas canvas, Size size) {
+    print('!!!! It\'s Columns Painter Logic ');
+
+    final animationFactor = animation != null ? animation!.value : 1;
     late double margin;
     late double radius;

@@ -42,10 +54,12 @@ class ColumnsPainter<TData> extends CustomPainter {
         ..color = _series.pointColorMapper(data, i)
         ..style = PaintingStyle.fill;

-    final yAxisMaxValue = _getMaxYAxisValue();
+      final yAxisMaxValue = _getMaxYAxisValue();        ///  ! O(N^2)
+
       final yAxisValue = _series.yValueMapper(data, i);
       final columnHeight =
-          ((yAxisValue / yAxisMaxValue) * size.height - margin);
+          ((yAxisValue / yAxisMaxValue) * size.height - margin) *
+              animationFactor;
```

aint_boundary.dart | column_series.dart | fast_chart.dart | column_painter.dart | animation.dart | custom_paint.dart | change_notifier.dart

```dart
@override
void paint(Canvas canvas, Size size) {
  print('!!!! It\'s Columns Painter Logic ');

  final animationFactor = animation != null ? animation!.value : 1;   Smirnov, 18 minutes a
  late double margin;
  late double radius;

  final maxAvailableColumnWidth = size.width / _dataSource.length;
  if (maxAvailableColumnWidth < 8 && maxAvailableColumnWidth >= 2) {
    margin = 1;
    radius = 1;
  } else if (maxAvailableColumnWidth < 2) {
    margin = 0;
    radius = 0;
  } else {
    margin = 10;
    radius = 4;
  }

  final columnWidth = (size.width - (margin * _dataSource.length + margin)) /
      _dataSource.length;
  double left = margin;

  for (var i = 0; i < _dataSource.length; i++) {
    final TData data = _dataSource[i];
    Paint columnFillPaint = Paint()
      ..color = _series.pointColorMapper(data, i)
      ..style = PaintingStyle.fill;

    final yAxisMaxValue = _getMaxYAxisValue();        ///  ! O(N^2)

    final yAxisValue = _series.yValueMapper(data, i);
    final columnHeight =
```

Поэтому на данном этапе я решил сделать ребрендинг моего пакета
FastChart —-> SlowChart

Давайте быть честными хотя бы сами с собой и называть свои пакеты так, как они заслуживают



mobius

Profiling

CPU Usage

Дает нам понимание того, каким потоком загружен наш CPU.
Особенно полезно для профилирования многопоточных приложений, но не для нашего случая и для однопоточной среды выполнения кода Dart в Flutter приложениях.

CPU Usage - это, если совсем упрощенно, аггрегация частоты переключения контекста из одного потока в другой.
И такая аггрегация дает нам понимание того, сколько времени потоки были заняты по отношению друг к другу.

В нашем сценарии мы и так видим, что у нас пострадал UI Thread, потому нам нужна другая техника, а именно

CPU Sampling

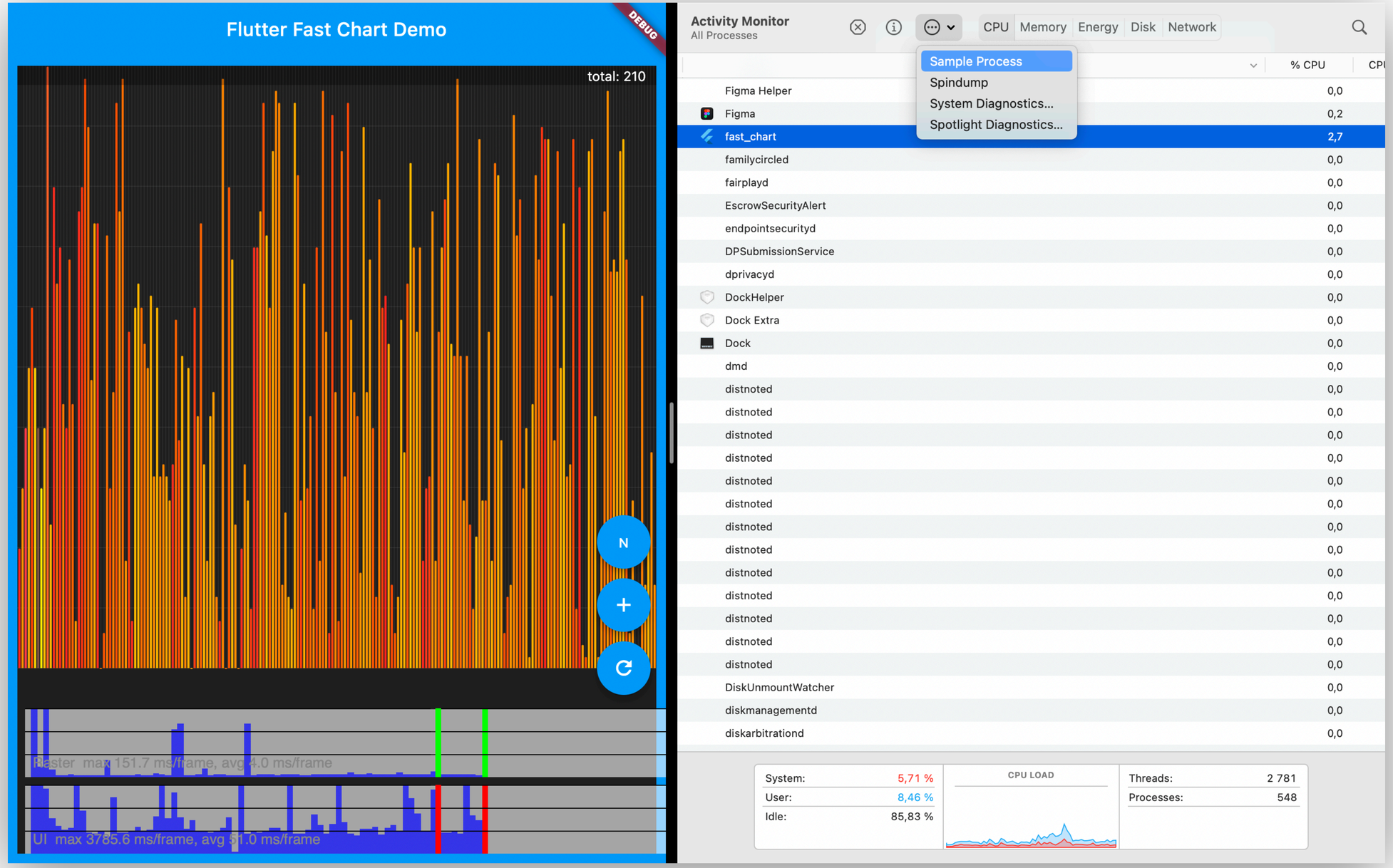
Данная техника делает следующее - с довольной высокой частотой для нашего выполняемого приложения и потока берутся snapshots, а по сути stack traces (или **samples**), где на вершине текущего stack trace выполняется какая-то конкретная функция.

Затем выполняется аггрегация таких данных, и в конечном итоге мы получаем статистику того,
какие функции выполнялись чаще всего за наблюдаемый период времени.

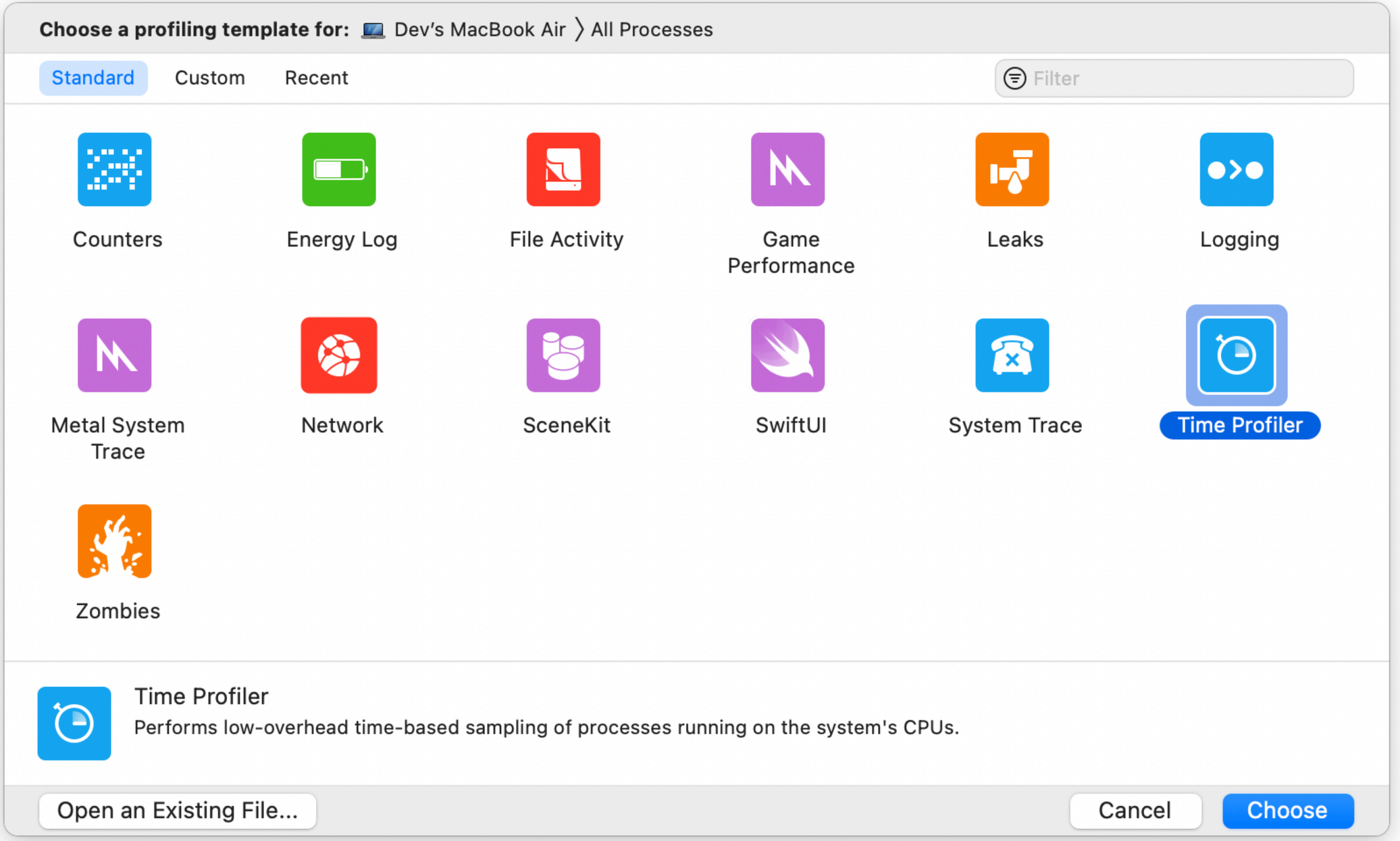
Количество samples, требуемых для наблюдения, зависит от того, насколько точную и детальную статистику вы хотите получить.

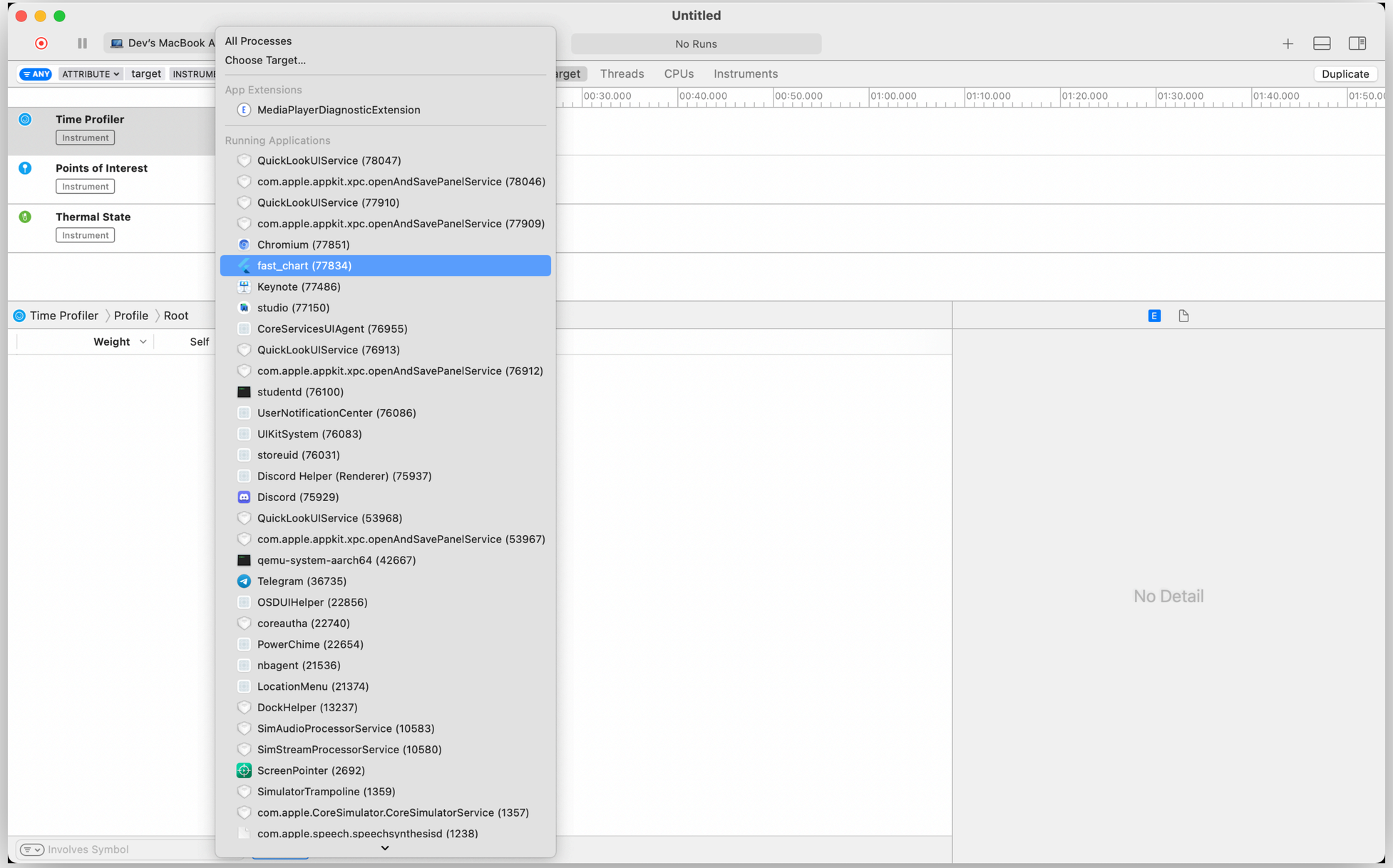
CPU Sampling

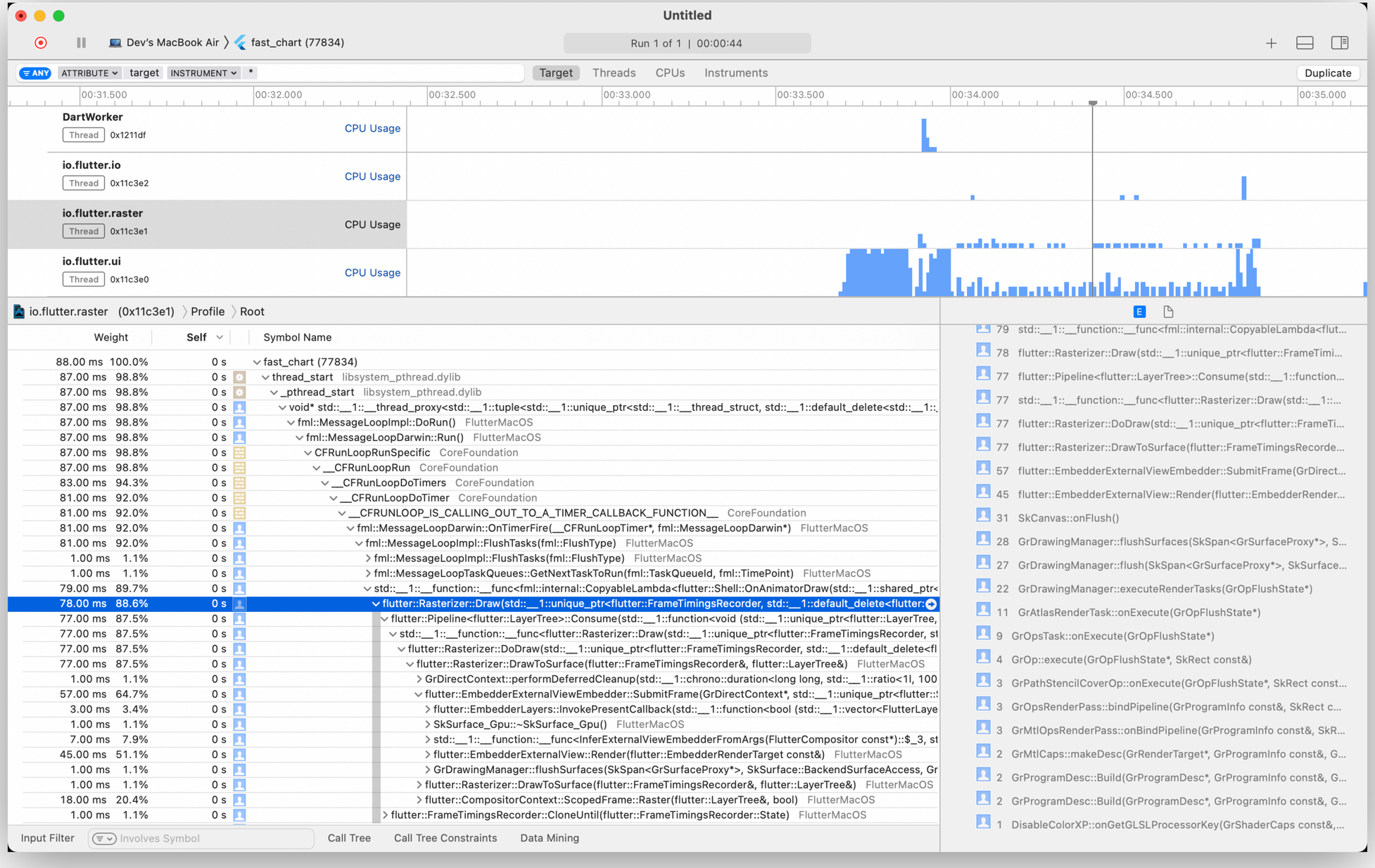
CPU Sampling

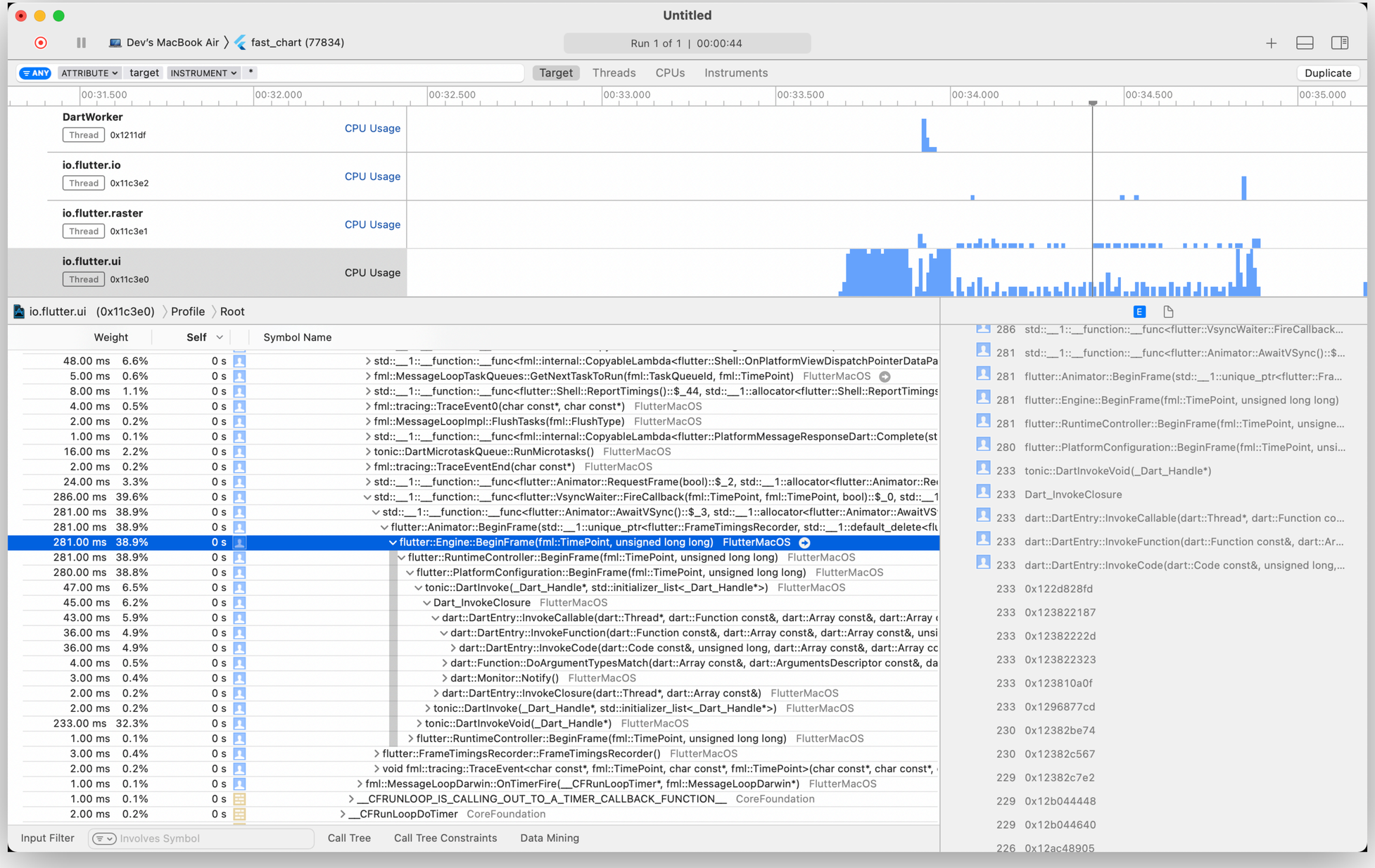

Flutter Fast Chart Demo
DEBUG
total: 210
N
raster max 151.7 ms/frame, avg 4.0 ms/frame
UI max 3785.6 ms/frame, avg 51.0 ms/frame
Activity Monitor
All Processes
CPU
Memory
Energy
Disk
Network
Sample Process
Spindump
System Diagnostics...
Spotlight Diagnostics...
% CPU
Figma Helper 0,0
Figma 0,2
fast_chart 2,7
familycircled 0,0
fairplayd 0,0
EscrowSecurityAlert 0,0
endpointsecurityd 0,0
DPSubmissionService 0,0
dprivacyd 0,0
DockHelper 0,0
Dock Extra 0,0
Dock 0,0
dmd 0,0
distnoted 0,0
DiskUnmountWatcher 0,0
diskmanagementd 0,0
diskarbitrationd 0,0
System: 5,71 %
User: 8,46 %
Idle: 85,83 %
CPU LOAD
Threads: 2 781
Processes: 548

Instruments.app


Choose a profiling template for: Dev's MacBook Air 〉All Processes
Standard
Custom
Recent
Filter
Counters
Energy Log
File Activity
Game Performance
Leaks
Logging
Metal System Trace
Network
SceneKit
SwiftUI
System Trace
Time Profiler
Zombies
Time Profiler
Performs low-overhead time-based sampling of processes running on the system's CPUs.
Open an Existing File...
Cancel
Choose

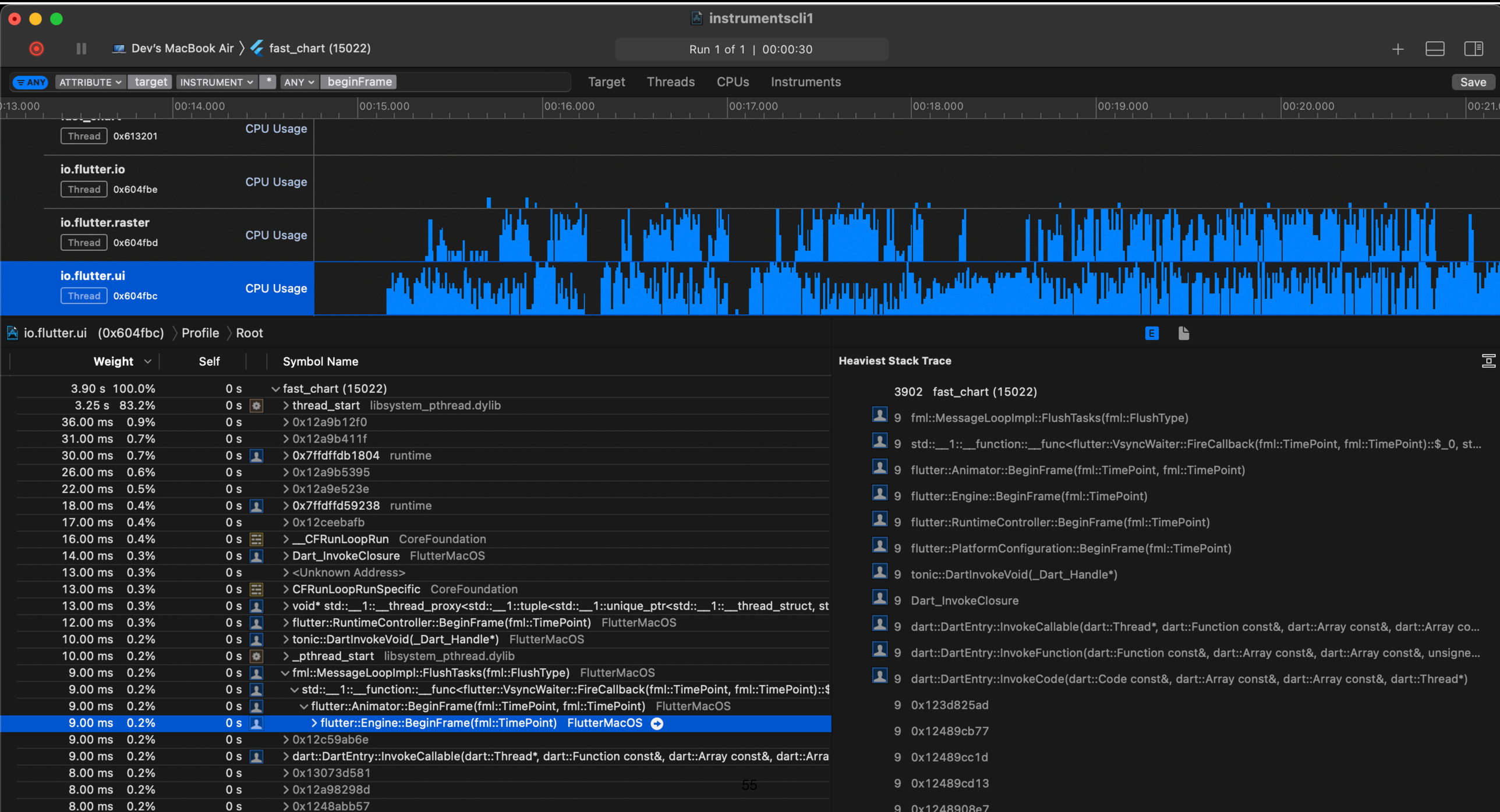

Untitled
Dev's MacBook A
No Runs
All Processes
Choose Target...
App Extensions
MediaPlayerDiagnosticExtension
Running Applications
QuickLookUIService (78047)
com.apple.appkit.xpc.openAndSavePanelService (78046)
QuickLookUIService (77910)
com.apple.appkit.xpc.openAndSavePanelService (77909)
Chromium (77851)
fast_chart (77834)
Keynote (77486)
studio (77150)
CoreServicesUIAgent (76955)
QuickLookUIService (76913)
com.apple.appkit.xpc.openAndSavePanelService (76912)
studentd (76100)
UserNotificationCenter (76086)
UIKitSystem (76083)
storeuid (76031)
Discord Helper (Renderer) (75937)
Discord (75929)
QuickLookUIService (53968)
com.apple.appkit.xpc.openAndSavePanelService (53967)
qemu-system-aarch64 (42667)
Telegram (36735)
OSDUIHelper (22856)
coreautha (22740)
PowerChime (22654)
nbagent (21536)
LocationMenu (21374)
DockHelper (13237)
SimAudioProcessorService (10583)
SimStreamProcessorService (10580)
ScreenPointer (2692)
SimulatorTrampoline (1359)
com.apple.CoreSimulator.CoreSimulatorService (1357)
com.apple.speech.speechsynthesisd (1238)
Time Profiler
Instrument
Points of Interest
Instrument
Thermal State
Instrument
Threads
CPUs
Instruments
Duplicate
00:30.000
00:40.000
00:50.000
01:00.000
01:10.000
01:20.000
01:30.000
01:40.000
Time Profiler
Profile
Root
Weight
Self
No Detail
Involves Symbol


Untitled
Dev's MacBook Air
fast_chart (77834)
Run 1 of 1 | 00:00:44
Target
Threads
CPUs
Instruments
Duplicate
00:31.500
00:32.000
00:32.500
00:33.000
00:33.500
00:34.000
00:34.500
00:35.000
DartWorker
Thread 0x1211df
CPU Usage
io.flutter.io
Thread 0x11c3e2
CPU Usage
io.flutter.raster
Thread 0x11c3e1
CPU Usage
io.flutter.ui
Thread 0x11c3e0
CPU Usage
io.flutter.raster (0x11c3e1) Profile Root
Weight
Self
Symbol Name
88.00 ms 100.0% 0 s fast_chart (77834)
87.00 ms 98.8% 0 s thread_start libsystem_pthread.dylib
87.00 ms 98.8% 0 s _pthread_start libsystem_pthread.dylib
87.00 ms 98.8% 0 s void* std::__1::__thread_proxy<std::__1::tuple<std::__1::unique_ptr<std::__1::__thread_struct, std::__1::default_delete<std::__1::
87.00 ms 98.8% 0 s fml::MessageLoopImpl::DoRun() FlutterMacOS
87.00 ms 98.8% 0 s fml::MessageLoopDarwin::Run() FlutterMacOS
87.00 ms 98.8% 0 s CFRunLoopRunSpecific CoreFoundation
87.00 ms 98.8% 0 s __CFRunLoopRun CoreFoundation
83.00 ms 94.3% 0 s __CFRunLoopDoTimers CoreFoundation
81.00 ms 92.0% 0 s __CFRunLoopDoTimer CoreFoundation
81.00 ms 92.0% 0 s __CFRUNLOOP_IS_CALLING_OUT_TO_A_TIMER_CALLBACK_FUNCTION__ CoreFoundation
81.00 ms 92.0% 0 s fml::MessageLoopDarwin::OnTimerFire(__CFRunLoopTimer*, fml::MessageLoopDarwin*) FlutterMacOS
81.00 ms 92.0% 0 s fml::MessageLoopImpl::FlushTasks(fml::FlushType) FlutterMacOS
1.00 ms 1.1% 0 s fml::MessageLoopImpl::FlushTasks(fml::FlushType) FlutterMacOS
1.00 ms 1.1% 0 s fml::MessageLoopTaskQueues::GetNextTaskToRun(fml::TaskQueueId, fml::TimePoint) FlutterMacOS
79.00 ms 89.7% 0 s std::__1::__function::__func<fml::internal::CopyableLambda<flutter::Shell::OnAnimatorDraw(std::__1::shared_ptr<
78.00 ms 88.6% 0 s flutter::Rasterizer::Draw(std::__1::unique_ptr<flutter::FrameTimingsRecorder, std::__1::default_delete<flutter::
77.00 ms 87.5% 0 s flutter::Pipeline<flutter::LayerTree>::Consume(std::__1::function<void (std::__1::unique_ptr<flutter::LayerTree,
77.00 ms 87.5% 0 s std::__1::__function::__func<flutter::Rasterizer::Draw(std::__1::unique_ptr<flutter::FrameTimingsRecorder, st
77.00 ms 87.5% 0 s flutter::Rasterizer::DoDraw(std::__1::unique_ptr<flutter::FrameTimingsRecorder, std::__1::default_delete<fl
77.00 ms 87.5% 0 s flutter::Rasterizer::DrawToSurface(flutter::FrameTimingsRecorder&, flutter::LayerTree&) FlutterMacOS
1.00 ms 1.1% 0 s GrDirectContext::performDeferredCleanup(std::__1::chrono::duration<long long, std::__1::ratio<1l, 100
57.00 ms 64.7% 0 s flutter::EmbedderExternalViewEmbedder::SubmitFrame(GrDirectContext*, std::__1::unique_ptr<flutter::
3.00 ms 3.4% 0 s flutter::EmbedderLayers::InvokePresentCallback(std::__1::function<bool (std::__1::vector<FlutterLaye
1.00 ms 1.1% 0 s SkSurface_Gpu::~SkSurface_Gpu() FlutterMacOS
7.00 ms 7.9% 0 s std::__1::__function::__func<InferExternalViewEmbedderFromArgs(FlutterCompositor const*)::$_3, st
45.00 ms 51.1% 0 s flutter::EmbedderExternalView::Render(flutter::EmbedderRenderTarget const&) FlutterMacOS
1.00 ms 1.1% 0 s GrDrawingManager::flushSurfaces(SkSpan<GrSurfaceProxy*>, SkSurface::BackendSurfaceAccess, Gr
1.00 ms 1.1% 0 s flutter::Rasterizer::DrawToSurface(flutter::FrameTimingsRecorder&, flutter::LayerTree&) FlutterMacOS
18.00 ms 20.4% 0 s flutter::CompositorContext::ScopedFrame::Raster(flutter::LayerTree&, bool) FlutterMacOS
1.00 ms 1.1% 0 s flutter::FrameTimingsRecorder::CloneUntil(flutter::FrameTimingsRecorder::State) FlutterMacOS
79 std::__1::__function::__func<fml::internal::CopyableLambda<flut...
78 flutter::Rasterizer::Draw(std::__1::unique_ptr<flutter::FrameTimi...
77 flutter::Pipeline<flutter::LayerTree>::Consume(std::__1::function...
77 std::__1::__function::__func<flutter::Rasterizer::Draw(std::__1::...
77 flutter::Rasterizer::DoDraw(std::__1::unique_ptr<flutter::FrameTi...
77 flutter::Rasterizer::DrawToSurface(flutter::FrameTimingsRecorde...
57 flutter::EmbedderExternalViewEmbedder::SubmitFrame(GrDirect...
45 flutter::EmbedderExternalView::Render(flutter::EmbedderRender...
31 SkCanvas::onFlush()
28 GrDrawingManager::flushSurfaces(SkSpan<GrSurfaceProxy*>, S...
27 GrDrawingManager::flush(SkSpan<GrSurfaceProxy*>, SkSurface...
22 GrDrawingManager::executeRenderTasks(GrOpFlushState*)
11 GrAtlasRenderTask::onExecute(GrOpFlushState*)
9 GrOpsTask::onExecute(GrOpFlushState*)
4 GrOp::execute(GrOpFlushState*, SkRect const&)
3 GrPathStencilCoverOp::onExecute(GrOpFlushState*, SkRect const...
3 GrOpsRenderPass::bindPipeline(GrProgramInfo const&, SkRect c...
3 GrMtlOpsRenderPass::onBindPipeline(GrProgramInfo const&, SkR...
2 GrMtlCaps::makeDesc(GrRenderTarget*, GrProgramInfo const&, G...
2 GrProgramDesc::Build(GrProgramDesc*, GrProgramInfo const&, G...
2 GrProgramDesc::Build(GrProgramDesc*, GrProgramInfo const&, G...
1 DisableColorXP::onGetGLSLProcessorKey(GrShaderCaps const&,...
Input Filter
Involves Symbol
Call Tree
Call Tree Constraints
Data Mining


Untitled
Dev's MacBook Air
fast_chart (77834)
Run 1 of 1 | 00:00:44
Target
Threads
CPUs
Instruments
Duplicate
00:31.500
00:32.000
00:32.500
00:33.000
00:33.500
00:34.000
00:34.500
00:35.000
DartWorker
Thread 0x1211df
CPU Usage
io.flutter.io
Thread 0x11c3e2
CPU Usage
io.flutter.raster
Thread 0x11c3e1
CPU Usage
io.flutter.ui
Thread 0x11c3e0
CPU Usage
io.flutter.ui (0x11c3e0) Profile Root
Weight
Self
Symbol Name
48.00 ms 6.6% 0 s std::__1::__function::__func<fml::internal::CopyableLambda<flutter::Shell::OnPlatformViewDispatchPointerDataPa
5.00 ms 0.6% 0 s fml::MessageLoopTaskQueues::GetNextTaskToRun(fml::TaskQueueId, fml::TimePoint) FlutterMacOS
8.00 ms 1.1% 0 s std::__1::__function::__func<flutter::Shell::ReportTimings()::$_44, std::__1::allocator<flutter::Shell::ReportTimings
4.00 ms 0.5% 0 s fml::tracing::TraceEvent0(char const*, char const*) FlutterMacOS
2.00 ms 0.2% 0 s fml::MessageLoopImpl::FlushTasks(fml::FlushType) FlutterMacOS
1.00 ms 0.1% 0 s std::__1::__function::__func<fml::internal::CopyableLambda<flutter::PlatformMessageResponseDart::Complete(st
16.00 ms 2.2% 0 s tonic::DartMicrotaskQueue::RunMicrotasks() FlutterMacOS
2.00 ms 0.2% 0 s fml::tracing::TraceEventEnd(char const*) FlutterMacOS
24.00 ms 3.3% 0 s std::__1::__function::__func<flutter::Animator::RequestFrame(bool)::$_2, std::__1::allocator<flutter::Animator::Re
286.00 ms 39.6% 0 s std::__1::__function::__func<flutter::VsyncWaiter::FireCallback(fml::TimePoint, fml::TimePoint, bool)::$_0, std::__1
281.00 ms 38.9% 0 s std::__1::__function::__func<flutter::Animator::AwaitVSync()::$_3, std::__1::allocator<flutter::Animator::AwaitVS
281.00 ms 38.9% 0 s flutter::Animator::BeginFrame(std::__1::unique_ptr<flutter::FrameTimingsRecorder, std::__1::default_delete<flu
281.00 ms 38.9% 0 s flutter::Engine::BeginFrame(fml::TimePoint, unsigned long long) FlutterMacOS
281.00 ms 38.9% 0 s flutter::RuntimeController::BeginFrame(fml::TimePoint, unsigned long long) FlutterMacOS
280.00 ms 38.8% 0 s flutter::PlatformConfiguration::BeginFrame(fml::TimePoint, unsigned long long) FlutterMacOS
47.00 ms 6.5% 0 s tonic::DartInvoke(_Dart_Handle*, std::initializer_list<_Dart_Handle*>) FlutterMacOS
45.00 ms 6.2% 0 s Dart_InvokeClosure FlutterMacOS
43.00 ms 5.9% 0 s dart::DartEntry::InvokeCallable(dart::Thread*, dart::Function const&, dart::Array const&, dart::Array
36.00 ms 4.9% 0 s dart::DartEntry::InvokeFunction(dart::Function const&, dart::Array const&, dart::Array const&, unsi
36.00 ms 4.9% 0 s dart::DartEntry::InvokeCode(dart::Code const&, unsigned long, dart::Array const&, dart::Array cc
4.00 ms 0.5% 0 s dart::Function::DoArgumentTypesMatch(dart::Array const&, dart::ArgumentsDescriptor const&, da
3.00 ms 0.4% 0 s dart::Monitor::Notify() FlutterMacOS
2.00 ms 0.2% 0 s dart::DartEntry::InvokeClosure(dart::Thread*, dart::Array const&) FlutterMacOS
2.00 ms 0.2% 0 s tonic::DartInvoke(_Dart_Handle*, std::initializer_list<_Dart_Handle*>) FlutterMacOS
233.00 ms 32.3% 0 s tonic::DartInvokeVoid(_Dart_Handle*) FlutterMacOS
1.00 ms 0.1% 0 s flutter::RuntimeController::BeginFrame(fml::TimePoint, unsigned long long) FlutterMacOS
3.00 ms 0.4% 0 s flutter::FrameTimingsRecorder::FrameTimingsRecorder() FlutterMacOS
2.00 ms 0.2% 0 s void fml::tracing::TraceEvent<char const*, fml::TimePoint, char const*, fml::TimePoint>(char const*, char const*,
1.00 ms 0.1% 0 s fml::MessageLoopDarwin::OnTimerFire(__CFRunLoopTimer*, fml::MessageLoopDarwin*) FlutterMacOS
1.00 ms 0.1% 0 s __CFRUNLOOP_IS_CALLING_OUT_TO_A_TIMER_CALLBACK_FUNCTION__ CoreFoundation
2.00 ms 0.2% 0 s __CFRunLoopDoTimer CoreFoundation
286 std::__1::__function::__func<flutter::VsyncWaiter::FireCallback...
281 std::__1::__function::__func<flutter::Animator::AwaitVSync()::$...
281 flutter::Animator::BeginFrame(std::__1::unique_ptr<flutter::Fra...
281 flutter::Engine::BeginFrame(fml::TimePoint, unsigned long long)
281 flutter::RuntimeController::BeginFrame(fml::TimePoint, unsigne...
280 flutter::PlatformConfiguration::BeginFrame(fml::TimePoint, unsi...
233 tonic::DartInvokeVoid(_Dart_Handle*)
233 Dart_InvokeClosure
233 dart::DartEntry::InvokeCallable(dart::Thread*, dart::Function co...
233 dart::DartEntry::InvokeFunction(dart::Function const&, dart::Ar...
233 dart::DartEntry::InvokeCode(dart::Code const&, unsigned long,...
233 0x122d828fd
233 0x123822187
233 0x12382222d
233 0x123822323
233 0x123810a0f
233 0x1296877cd
230 0x12382be74
230 0x12382c567
229 0x12382c7e2
229 0x12b044448
229 0x12b044640
226 0x12ac48905
Input Filter
Involves Symbol
Call Tree
Call Tree Constraints
Data Mining

instrumentscli

```
~ instruments -l 30000 -t Time\ Profiler -p 15022
```



```
~ sample 15022 30 -file /Users/andreysmirnov/Desktop/sample_trace.txt
Analysis of sampling fast_chart (pid 15022) every 1 millisecond
Process:         fast_chart [15022]
Path:            /Users/USER/*/fast_chart.app/Contents/MacOS/fast_chart
Load Address:    0x102120000
Identifier:      com.example.fastChart
Version:         1.0.0 (1)
Code Type:       X86-64 (translated)
Platform:        macOS
Parent Process:  dart [14838]

Date/Time:       2021-08-05 09:48:30.564 +0300
Launch Time:     2021-08-05 09:36:51.081 +0300
OS Version:      macOS 11.2.3 (20D91)
Report Version:  7
Analysis Tool:   /usr/bin/sample

Physical footprint:         413.8M
Physical footprint (peak):  423.2M
----

Call graph:
    4536 Thread_6311746   DispatchQueue_1: com.apple.main-thread  (serial)
    + 4536 start  (in libdyld.dylib) + 1  [0x7fff20379621]
    +   4536 main  (in fast_chart) + 13  [0x10212273d]  AppDelegate.swift:5
    +     4536 NSApplicationMain  (in AppKit) + 816  [0x7fff22b6e96f]
    +       4463 -[NSApplication run]  (in AppKit) + 586  [0x7fff22b9a68a]
    +       ! 4436 -[NSApplication(NSEvent) _nextEventMatchingEventMask:untilDate:inMode:dequeue:]  (in AppKit) + 1366  [0x7fff22ba8177]
    +       ! : 4434 _DPSNextEvent  (in AppKit) + 883  [0x7fff22ba99b1]
    +       ! : | 4433 _BlockUntilNextEventMatchingListInModeWithFilter  (in HIToolbox) + 64  [0x7fff2892214f]
    +       ! : | + 4337 ReceiveNextEventCommon  (in HIToolbox) + 709  [0x7fff2892242c]
    +       ! : | + ! 4336 RunCurrentEventLoopInMode  (in HIToolbox) + 292  [0x7fff28922630]
    +       ! : | + ! : 4331 CFRunLoopRunSpecific  (in CoreFoundation) + 567  [0x7fff20453fe2]
    +       ! : | + ! : | 4306 __CFRunLoopRun  (in CoreFoundation) + 1319  [0x7fff20454bf6]
    +       ! : | + ! : | + 4306 __CFRunLoopServiceMachPort  (in CoreFoundation) + 316  [0x7fff2045650d]
    +       ! : | + ! : | +   4306 mach_msg_trap  (in libsystem_kernel.dylib) + 10  [0x7fff20329e7e]
    +       ! : | + ! : | +     4306 ???  (in <unknown binary>)  [0x7ffe96372ab8]
    +       ! : | + ! : | 10 __CFRunLoopRun  (in CoreFoundation) + 2745  [0x7fff20455188]

    4536 Thread_6311868: io.flutter.ui
    + 4407 thread_start  (in libsystem_pthread.dylib) + 15  [0x7fff2035a47b]
    + ! 4407  pthread_start  (in libsystem_pthread.dylib) + 224  [0x7fff2035e950]
```

mobius

CPU Sampling For Dart Applications

Dart VM

Dart 1.0 was released on November 14, 2013

Dart VM

Flutter Application == Dart Application

```
~/projects/fast_chart 〉 master *4 dart lib/main.dart
```

```
❯ dart lib/main.dart
lib/main.dart: Warning: Interpreting this as package URI, 'package:fast_chart/main.dart'.
lib/save_layer/save_layer_demo.dart:3:8: Error: Not found: 'dart:ui'
import 'dart:ui' as ui;
       ^
lib/chart/axis_painter.dart:1:8: Error: Not found: 'dart:ui'
import 'dart:ui';
       ^
lib/chart/column_painter.dart:1:8: Error: Not found: 'dart:ui'
import 'dart:ui';
       ^
```

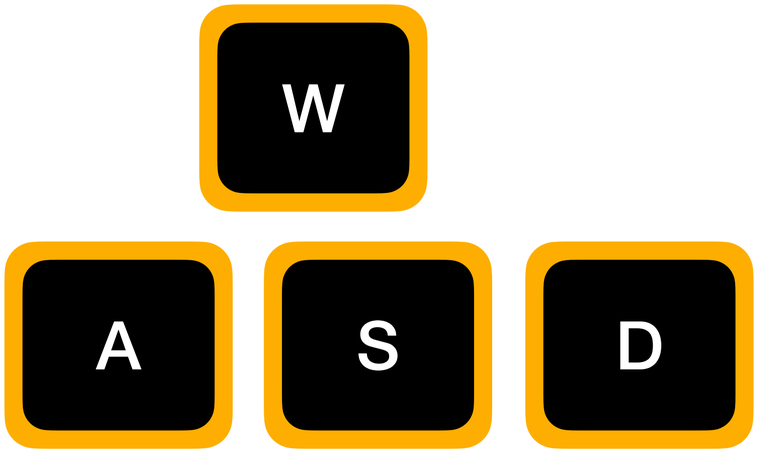
```
❯ flutter run --debug
Using hardware rendering with device sdk gphone64 arm64. If you notice graphics artifacts, consider enabling
"--enable-software-rendering".
Launching lib/main.dart on sdk gphone64 arm64 in debug mode...
Running Gradle task 'assembleDebug'...                             15.3s
✓  Built build/app/outputs/flutter-apk/app-debug.apk.
W/FlutterActivityAndFragmentDelegate(10804): A splash screen was provided to Flutter, but this is deprecated.
lash-migration for migration steps.
W/FlutterActivityAndFragmentDelegate(10841): A splash screen was provided to Flutter, but this is deprecated.
lash-migration for migration steps.
Syncing files to device sdk gphone64 arm64...                      131ms

Flutter run key commands.
r Hot reload. 🔥🔥🔥
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

💪 Running with sound null safety 💪

An Observatory debugger and profiler on sdk gphone64 arm64 is available at: http://127.0.0.1:53252/pKFvGs6p96o=/
```

mobius

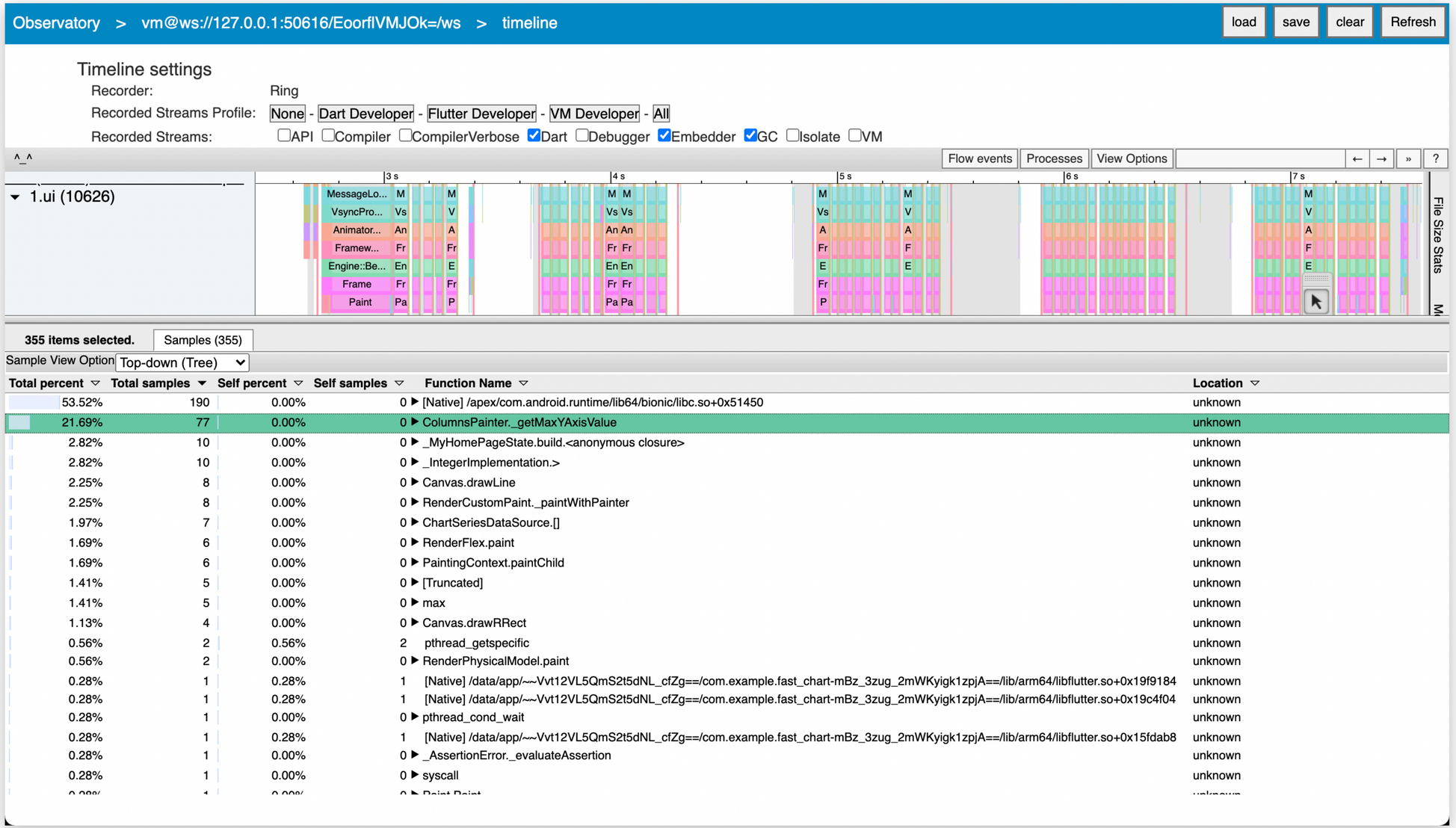
Observatory

Observatory Timeline Demo
CPU Sampling

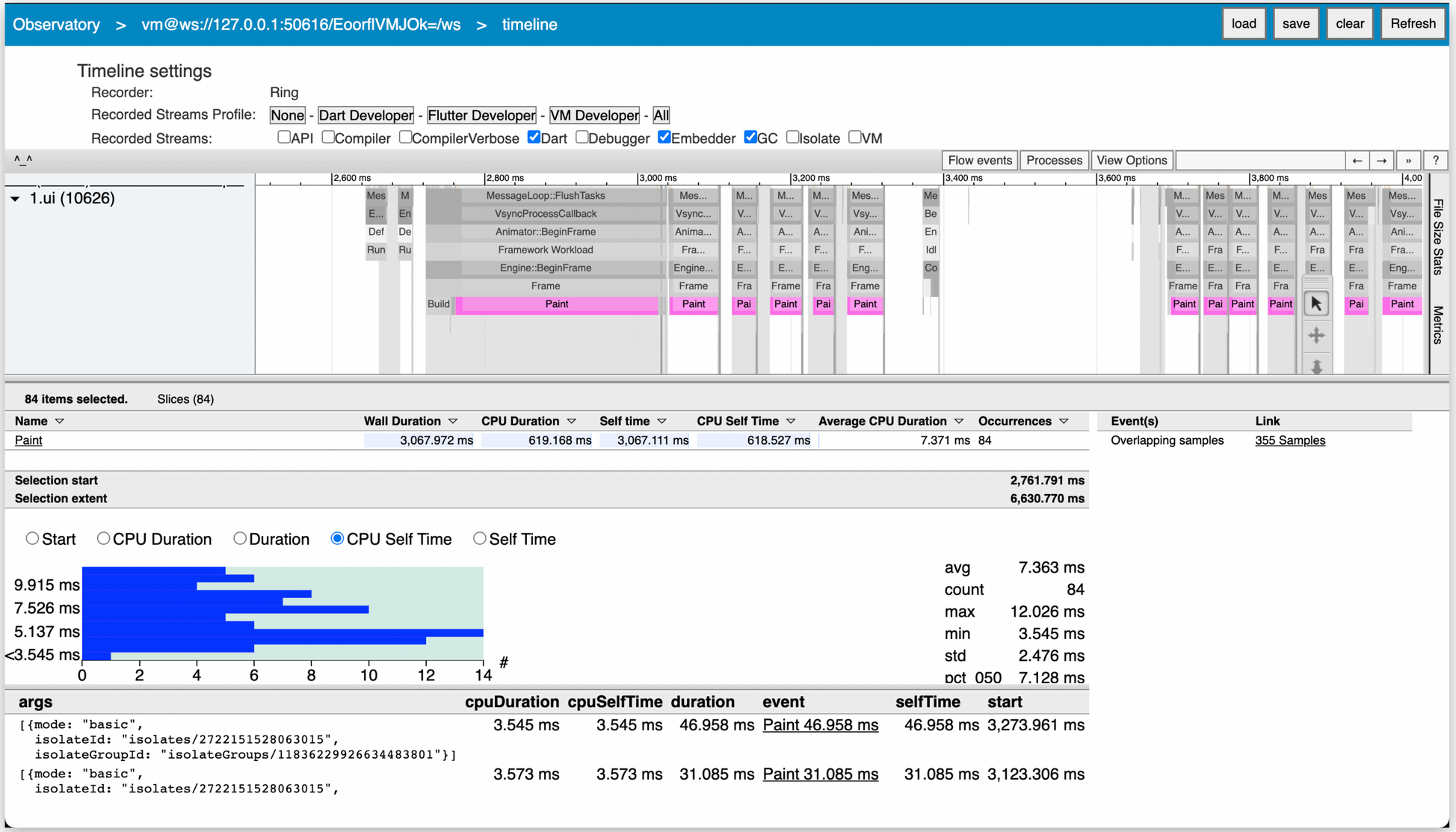
Observatory Timeline Hotkeys



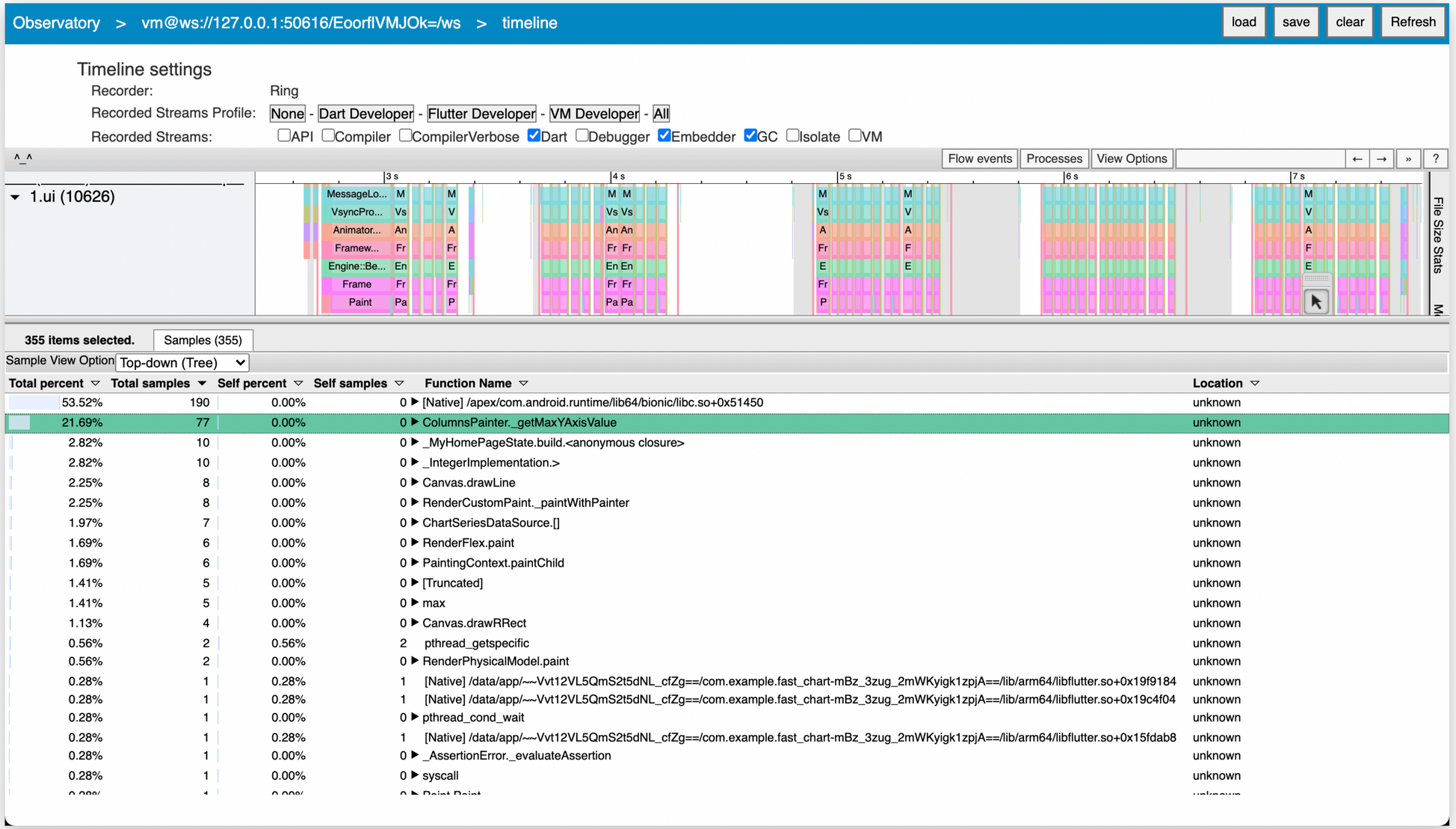
Selection Mode and Slices



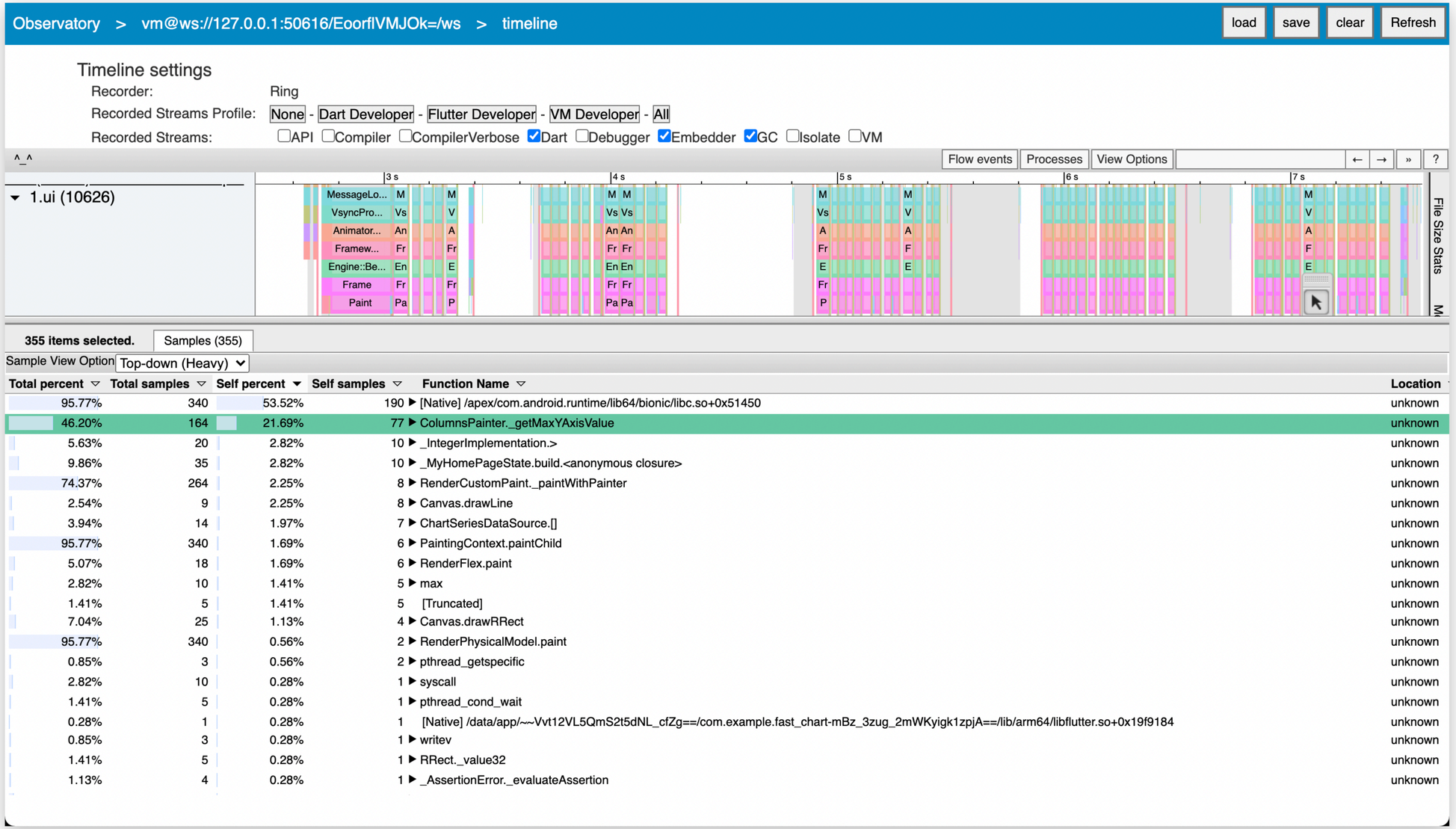
Анализируем выбранный Slice



TopDown Tree

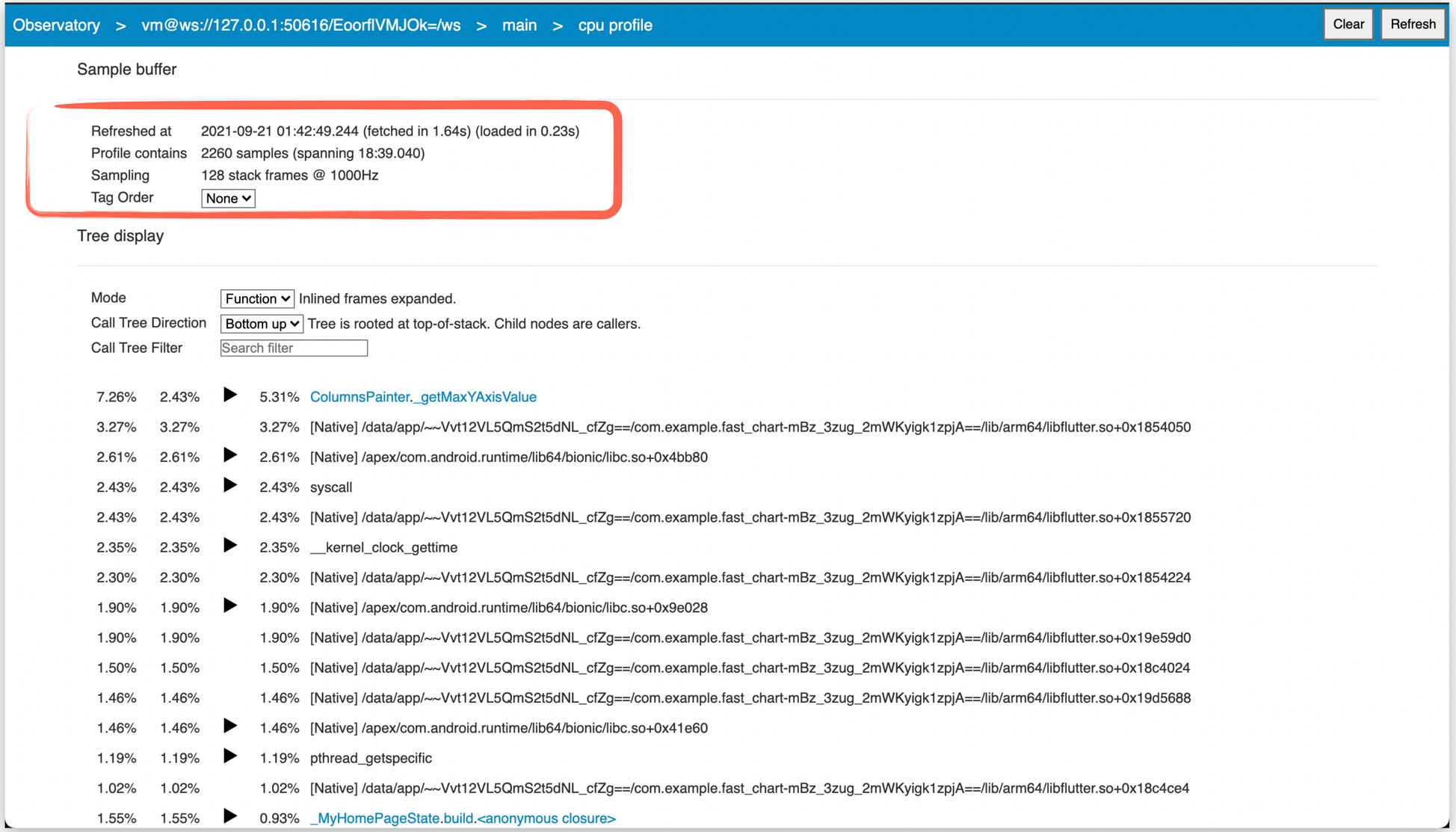


TopDown



Total percent	Total samples	Self percent	Self samples	Function Name	Location
95.77%	340	53.52%	190	▶ [Native] /apex/com.android.runtime/lib64/bionic/libc.so+0x51450	unknown
46.20%	164	21.69%	77	▶ ColumnsPainter._getMaxYAxisValue	unknown
5.63%	20	2.82%	10	▶ _IntegerImplementation.>	unknown
9.86%	35	2.82%	10	▶ _MyHomePageState.build.<anonymous closure>	unknown
74.37%	264	2.25%	8	▶ RenderCustomPaint._paintWithPainter	unknown
2.54%	9	2.25%	8	▶ Canvas.drawLine	unknown
3.94%	14	1.97%	7	▶ ChartSeriesDataSource.[]	unknown
95.77%	340	1.69%	6	▶ PaintingContext.paintChild	unknown
5.07%	18	1.69%	6	▶ RenderFlex.paint	unknown
2.82%	10	1.41%	5	▶ max	unknown
1.41%	5	1.41%	5	[Truncated]	unknown
7.04%	25	1.13%	4	▶ Canvas.drawRRect	unknown
95.77%	340	0.56%	2	▶ RenderPhysicalModel.paint	unknown
0.85%	3	0.56%	2	▶ pthread_getspecific	unknown
2.82%	10	0.28%	1	▶ syscall	unknown
1.41%	5	0.28%	1	▶ pthread_cond_wait	unknown
0.28%	1	0.28%	1	[Native] /data/app/~~Vvt12VL5QmS2t5dNL_cfZg==/com.example.fast_chart-mBz_3zug_2mWKyigk1zpjA==/lib/arm64/libflutter.so+0x19f9184	unknown
0.85%	3	0.28%	1	▶ writev	unknown
1.41%	5	0.28%	1	▶ RRect._value32	unknown
1.13%	4	0.28%	1	▶ _AssertionError._evaluateAssertion	unknown

CPU Profile - Информация по sampling



Allocation Profile

Allocation Profile

Observatory > vm@ws://127.0.0.1:50616/EoorfIVMJOk=/ws > main > allocation profile

Auto-refresh on GC | Refresh | GC | Download

Allocation Profile

last forced GC at ---

New Generation

used	13.0MB of 16.0MB
external	252.9KB
collections	7
average collection time	2.00 ms

Old Generation

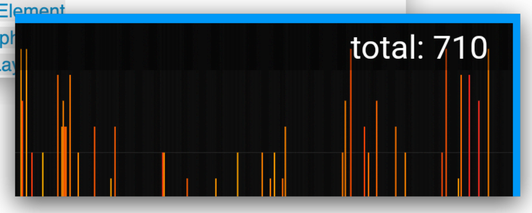
used	57.0MB of 58.7MB
external	81.2KB
collections	7
average collection time	compact ▲

Total

used	70.0MB of 74.7MB
external	334.1KB
collections	14
average collection time	7.58 ms

customdata ▲ 1 / 1 ▼ ✕

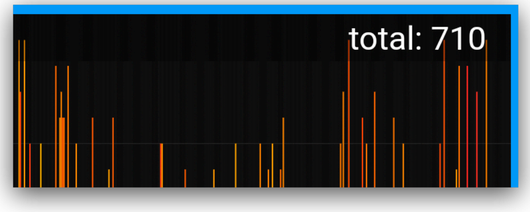
New Generation				Old Generation				Total				
Internal	External	Size	Instances	Internal	External	Size	Instances	Internal	External	Size▲	Instances	Class
45.6KB	0B	45.6KB	1458	32B	0B	32B	1	45.6KB	0B	45.6KB		
0B	0B	0B	0	45.0KB	0B	45.0KB	961	45.0KB	0B	45.0KB		
0B	0B	0B	0	40.8KB	0B	40.8KB	870	40.8KB	0B	40.8KB	870	ClosureData
[illegible]								22.5KB	0B	22.5KB	479	Rect
12.5KB	0B	12.5KB	400	9.69KB	0B	9.69KB	310	22.2KB	0B	22.2KB	710	CustomData
[illegible]								21.3KB	0B	21.3KB	609	_Int64List
1.47KB	19.1KB	20.6KB	94	32B	416B	448B	2	1.50KB	19.5KB	21.0KB	96	Path
19.8KB	0B	19.8KB	143	960B	0B	960B	6	20.8KB	0B	20.8KB	149	_Float64List
19.9KB	0B	19.9KB	318	0B	0B	0B	0	19.9KB	0B	19.9KB	318	StringBuffer
13.6KB	0B	13.6KB	29	6.09KB	0B	6.09KB	13	19.7KB	0B	19.7KB	42	SemanticsProperties
960B	0B	960B	20	18.7KB	0B	18.7KB	398	19.6KB	0B	19.6KB	418	_HashSet
0B	0B	0B	0	18.9KB	0B	18.9KB	404	18.9KB	0B	18.9KB	404	PatchClass
17.9KB	0B	17.9KB	229	0B	0B	0B	0	17.9KB	0B	17.9KB	229	_CompactIterator
0B	0B	0B	0	16.8KB	0B	16.8KB	98	16.8KB	0B	16.8KB	98	SingleChildRenderObjectElement
16.2KB	0B	16.2KB	259	0B	0B	0B	0	16.2KB	0B	16.2KB	259	_HashSetIterator
0B	0B	0B	0	14.8KB	0B	14.8KB	22	14.8KB	0B	14.8KB	22	RenderSemanticsAnnotations
6.61KB	0B	6.61KB	423	6.92KB	0B	6.92KB	443	13.5KB	0B	13.5KB	866	Color
0B	0B	0B	0	13.3KB	0B	13.3KB	284	13.3KB	0B	13.3KB	284	_Location
13.2KB	0B	13.2KB	211	0B	0B	0B	0	13.2KB	0B	13.2KB	211	_CompactIterable
0B	0B	0B	0	12.4KB	0B	12.4KB	72	12.4KB	0B	12.4KB	72	StatefulElement
32B	3.91KB	3.94KB	2	64B	7.81KB	7.88KB	4	96B	11.7KB	11.8KB	6	Paragraph
10.9KB	0B	10.9KB	87	640B	0B	640B	5	11.5KB	0B	11.5KB	92	PictureLa



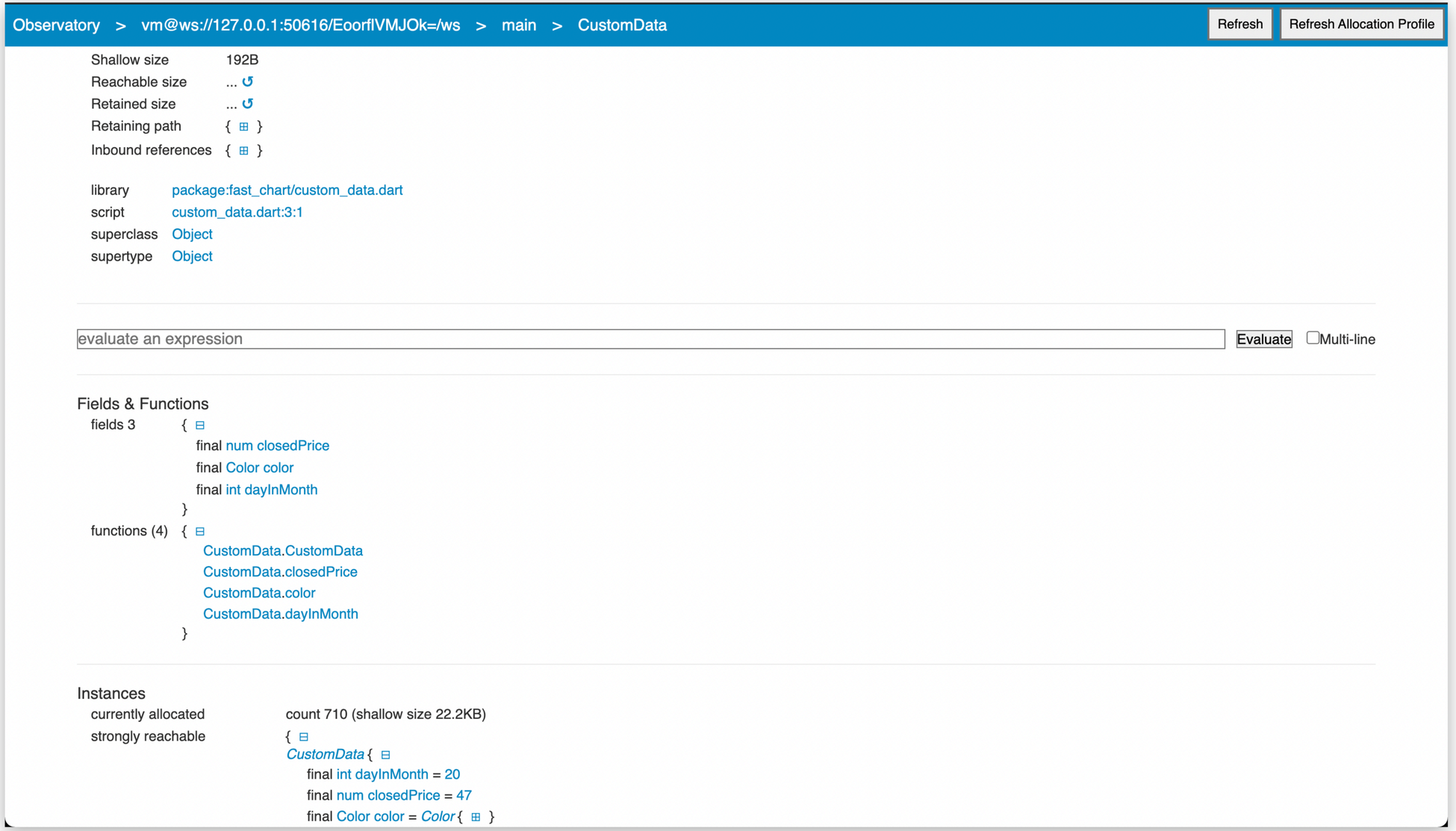
Allocation Profile - New Generation - Old Generation

New Generation		Old Generation		Total	
used	1.00MB of 16.0MB	used	57.1MB of 59.7MB	used	58.1MB of 75.7MB
external	233.9KB	external	102.3KB	external	336.1KB
collections	9	collections	8	collections	17
average collection time	2.92 ms	average collection time	compact ▲	average collection time	8.36 ms

New Generation				Old Generation				Total				
Internal	External	Size	Instances	Internal	External	Size	Instances	Internal	External	Size▲	Instances	Class
0B	0B	0B	0	45.0KB	0B	45.0KB	961	45.0KB	0B	45.0KB		customdata ▲ 1 / 1 ▼ ✕
0B	0B	0B	0	42.7KB	0B	42.7KB	911	42.7KB	0B	42.7KB		
6.94KB	0B	6.94KB	111	[illegible]				36.3KB	0B	36.3KB	580	_Closure
0B	0B	0B	0	22.2KB	0B	22.2KB	710	22.2KB	0B	22.2KB	710	CustomData
[illegible]				[illegible]				18.7KB	0B	18.7KB	399	_HashSet
0B	0B	0B	0	18.5KB	0B	18.5KB	394	18.5KB	0B	18.5KB	394	PatchClass
0B	0B	0B	0	16.8KB	0B	16.8KB	98	16.8KB	0B	16.8KB	98	SingleChildRenderObjectElement
0B	0B	0B	0	15.0KB	0B	15.0KB	120	15.0KB	0B	15.0KB	120	KernelProgramInfo
0B	0B	0B	0	14.8KB	0B	14.8KB	22	14.8KB	0B	14.8KB	22	RenderSemanticsAnnotations
2.53KB	0B	2.53KB	71	12.2KB	0B	12.2KB	344	14.8KB	0B	14.8KB	415	Context



Allocation Profile - профилирование структур данных в памяти

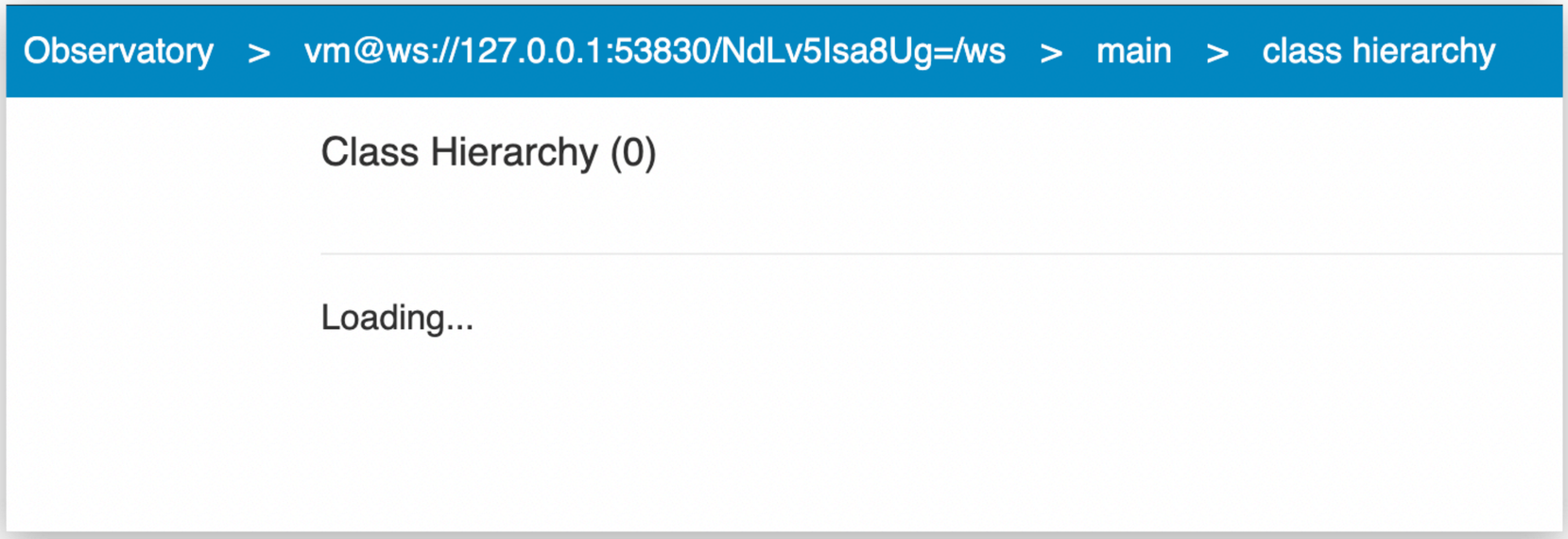


OutOfMemoryError

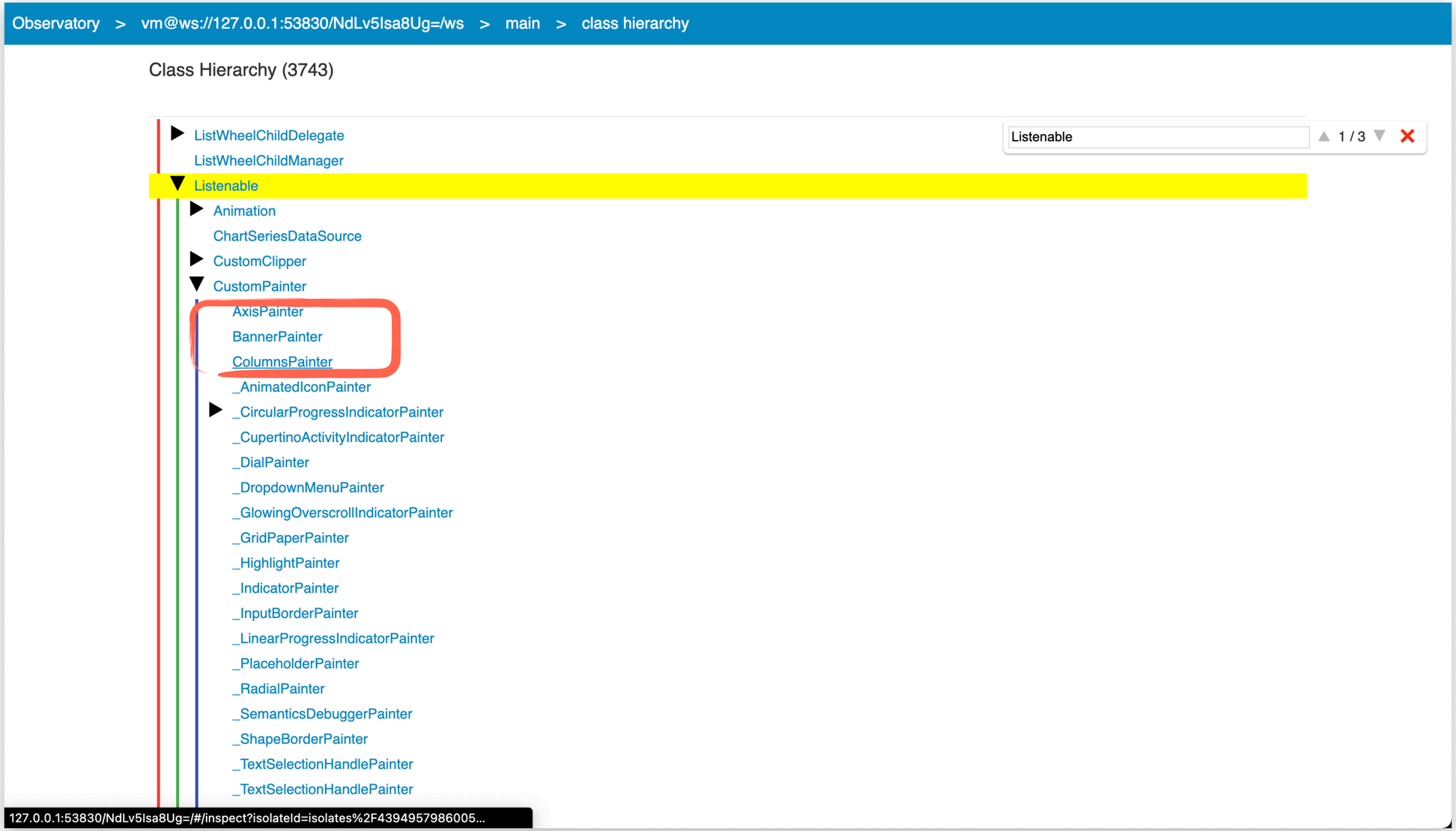
Allocation Profile может быть очень полезен для отслеживания memory leaks вплоть до отслеживания тех объектов, которые начинают тратить ее не так, как планировалось

Class Hierarchy

Class Hierarchy - имейте терпение)



Class Hierarchy - поиск только по базовому классу



Профилирование кода Dart через Class Hierarchy

Observatory > vm@ws://127.0.0.1:53830/NdLv5Isa8Ug=/ws > main > ColumnsPainter

Refresh | Refresh Allocation Profile

```
51                    for (var i = 0; i < _dataSource.length; i++) {
52                      final TData data = _dataSource[i];
53   0.05%  0.01%       Paint columnFillPaint = Paint()
54   0.06%  0.02%         ..color = _series.pointColorMapper(data, i)
55   0.01%  0.00%         ..style = PaintingStyle.fill;
56
57  31.78%  0.00%       final yAxisMaxValue = _getMaxYAxisValue();        ///  ! O(N^2)
58
59   0.01%  0.01%       final yAxisValue = _series.yValueMapper(data, i);
60                       final columnHeight =
61                           ((yAxisValue / yAxisMaxValue) * size.height - margin) *
62                               animationFactor;
63
64                       final top = size.height - columnHeight;
65   0.02%  0.00%       final columnRRect = RRect.fromRectAndCorners(
66   0.02%  0.01%         Rect.fromLTWH(left, top, columnWidth, columnHeight),
67                         bottomLeft: Radius.zero,
68                         bottomRight: Radius.zero,
69                         topLeft: Radius.circular(radius),
70   0.01%  0.00%         topRight: Radius.circular(radius),
71                       );
72   1.24%  0.00%       canvas.drawRRect(columnRRect, columnFillPaint);
73
74                       left += columnWidth + margin;
75                     }
76                   }
77
78                   @override
79                   bool shouldRepaint(ColumnsPainter oldDelegate) => false;
80
81                   num _getMaxYAxisValue() {
82                     num max = 0;
83   1.50%  1.50%      for (var i = 0; i < _dataSource.length; i++) {
84                       final data = _dataSource[i];
85  24.93%  7.16%       final yAxisValue = _series.yValueMapper(data, i);
86   2.36%  2.35%       max = math.max(max, yAxisValue);
87                     }
88
89                     return max;
90                   }
91                 }
```

Профилирование кода Dart через переход CPU Profile

Observatory > vm@ws://127.0.0.1:50616/EoorfIVMJOk=/ws > main > package:fast_chart/chart/column_painter.dart > object

Refresh

```
B  49                      double left = margin;
   50
B  51                      for (var i = 0; i < _dataSource.length; i++) {
B  52                        final TData data = _dataSource[i];
B  53   0.04%   0.01%        Paint columnFillPaint = Paint()
B  54   0.01%   0.01%          ..color = _series.pointColorMapper(data, i)
B  55                          ..style = PaintingStyle.fill;
   56
B  57  12.48%   0.01%        final yAxisMaxValue = _getMaxYAxisValue();        ///  ! O(N^2)
   58
B  59                        final yAxisValue = _series.yValueMapper(data, i);
   60                        final columnHeight =
B  61                            ((yAxisValue / yAxisMaxValue) * size.height - margin) *
   62                                animationFactor;
   63
B  64                        final top = size.height - columnHeight;
B  65                        final columnRRect = RRect.fromRectAndCorners(
B  66   0.01%   0.01%          Rect.fromLTWH(left, top, columnWidth, columnHeight),
   67                          bottomLeft: Radius.zero,
   68                          bottomRight: Radius.zero,
B  69                          topLeft: Radius.circular(radius),
B  70                          topRight: Radius.circular(radius),
   71                        );
B  72   0.72%   0.01%        canvas.drawRRect(columnRRect, columnFillPaint);
   73
B  74                        left += columnWidth + margin;
   75                      }
B  76                    }
   77
   78                    @override
B  79                    bool shouldRepaint(ColumnsPainter oldDelegate) => false;
   80
B  81                    num _getMaxYAxisValue() {
B  82                      num max = 0;
B  83   0.54%   0.54%      for (var i = 0; i < _dataSource.length; i++) {
B  84                        final data = _dataSource[i];
B  85   9.55%   2.73%        final yAxisValue = _series.yValueMapper(data, i);
B  86   0.88%   0.88%        max = math.max(max, yAxisValue);
   87                      }
   88
B  89                      return max;
   90                    }
   91                  }
```

Total %

Observatory - дополнительные возможности
Custom Timelines & Metrics

Timeline class Null safety

Add to the timeline.

Timeline's methods add synchronous events to the timeline. When generating a timeline in Chrome's tracing format, using Timeline generates "Complete" events. Timeline's startSync and finishSync can be used explicitly, or implicitly by wrapping a closure in timeSync. For example:

```
Timeline.startSync("Doing Something");
doSomething();
Timeline.finishSync();
```

Or:

```
Timeline.timeSync("Doing Something", () {
  doSomething();
});
```

mobius

Замечания насчет профилирования асинхронных операций на timeline

git checkout tags/timeline_task

TimelineTask class Null safety

An asynchronous task on the timeline. An asynchronous task can have many (nested) synchronous operations. Synchronous operations can live longer than the current isolate event. To pass a TimelineTask to another isolate, you must first call pass to get the task id and then construct a new TimelineTask in the other isolate.

Constructors

TimelineTask({TimelineTask? parent, String? filterKey})
Create a task. The task ID will be set by the system. [...]

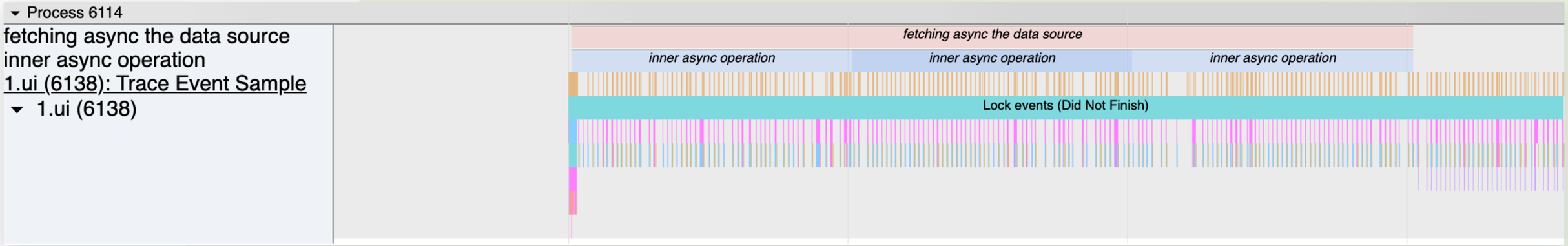
TimelineTask.withTaskId(int taskId, {String? filterKey})
Create a task with an explicit `taskId`. This is useful if you are passing a task from one isolate to another. [...]

Properties

hashCode → int
The hash code for this object. [...]
read-only, inherited

runtimeType → Type
A representation of the runtime type of the object.
read-only, inherited

Вывод на time line асинхронных операций



Вывод на time line операции, выполняемой в isolate

Пример вывода на timeline синхронной операции

Observatory Documentation

Самое печальное то, что по данному
инструменту уже и документации нет (

thenextweb.com/news/google-releases-dart-sdk-1-4-new-observatory-tool-looking-inside-running-virtual-machines-demand

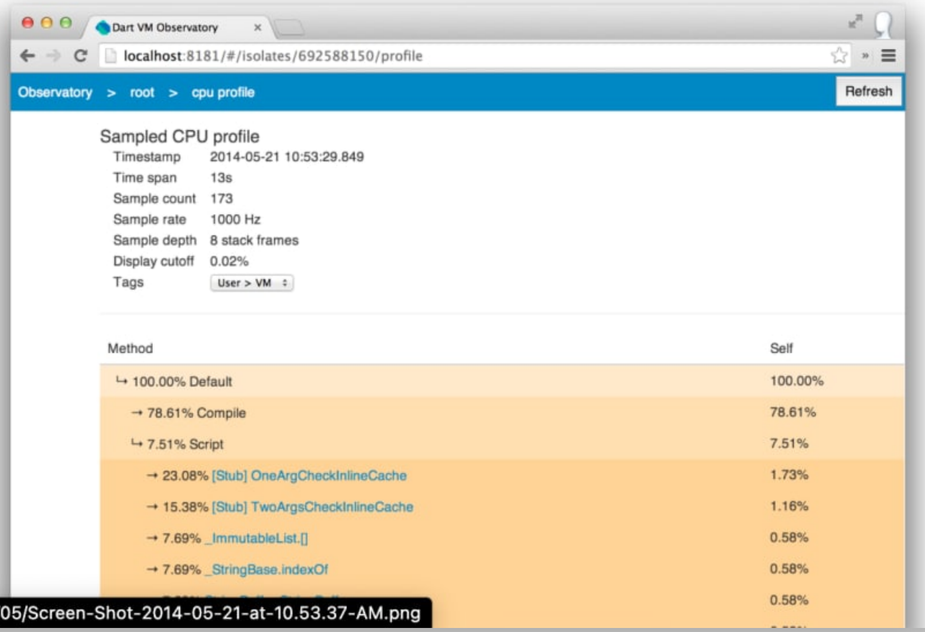
TNW | NEWS

HOME NEWS EVENTS PROGRAMS SPACES MORE

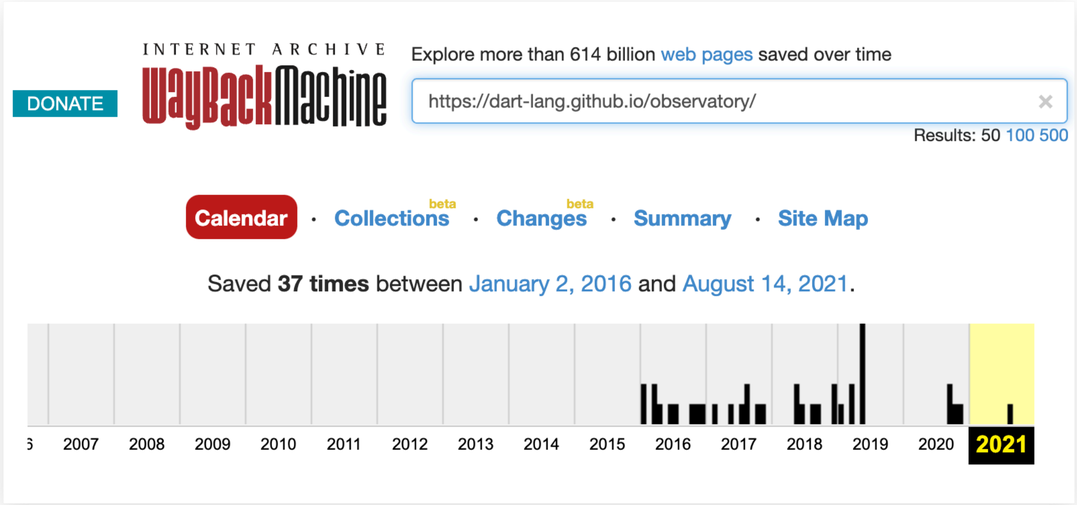
STORY BY

Emil Protalinski

Google today released Dart SDK version 1.4 with a new Observatory tool that allows you to look inside a running Dart virtual machine on demand and see live reporting of data. It shows information about garbage collection, the instance count for all in-memory objects by their type, line-by-line code coverage, and evaluation of arbitrary expressions. You can download the latest stable release of Dart, the company's JavaScript competitor, now from dartlang.org.



https://thenextweb.com/wp-content/blogs.dir/1/files/2014/05/Screen-Shot-2014-05-21-at-10.53.37-AM.png


DONATE
INTERNET ARCHIVE
WayBackMachine
Explore more than 614 billion web pages saved over time
https://dart-lang.github.io/observatory/
Results: 50 100 500
Calendar · Collections beta · Changes beta · Summary · Site Map
Saved 37 times between January 2, 2016 and August 14, 2021.
2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021

Observatory: A Profiler for Dart Apps

Observatory is a tool for profiling and debugging your Dart applications.

> You get Observatory, for free, when you download the Dart SDK.
>
> You can file issues and requests on dartbug.com.

Observatory allows you to peek inside a running Dart virtual machine (VM) on demand and provides live, immediate reporting of data. You can use it to browse most aspects of an application. Some of Observatory's features allow you to:

- Determine where an **app is spending its time**.
- Examine **allocated memory**.
- See which lines of **code have executed**.
- Debug **memory leaks**.
- Debug **memory fragmentation**.

You can use Observatory to examine the internals of a running Dart VM in real time, at any time.

In the following video, recorded at the recent Dart Developer Summit, John McCutchan and Todd Turnidge show how to profile and debug a Dart application.

Build Modes

https://flutter.dev/docs/testing/build-modes

- **Debug**

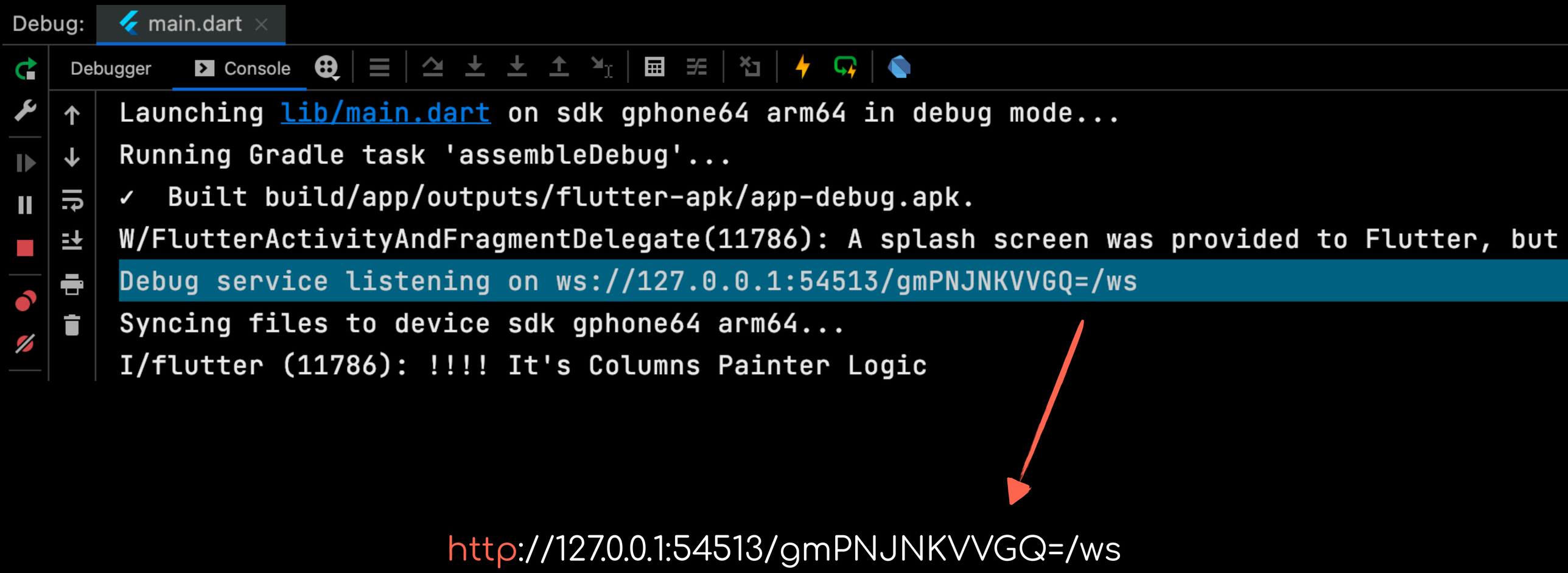
- **Profile** - Этот режим недоступен в эмуляторе (симуляторе) по причине, что поведение в эмуляторе не дает реальной картины производительности приложения.

  Этот режим очень близок к режиму Release, за исключением того, что есть логгинг, трассировка событий и VM Service, который позволяет вам подключиться при помощи DevTools или Observatory. Flutter умышленно не разрешает запуск приложения в Profile режиме на эмуляторе, так как это не имеет смысла.

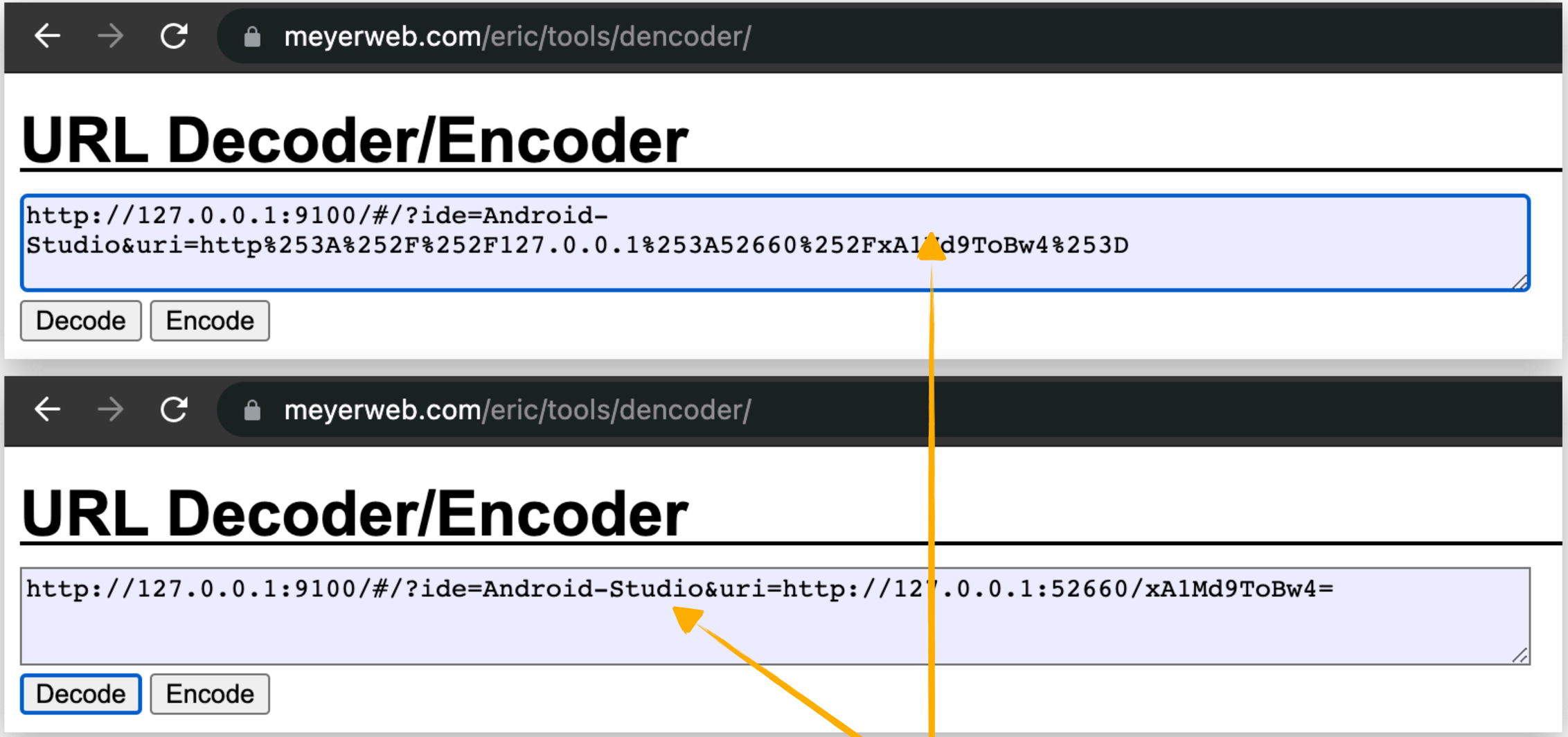
  Этот режим рекомендуется для профилирования приложений Flutter - https://flutter.dev/docs/perf/rendering/ui-performance#connect-to-a-physical-device

- **Release** - В этом режиме мы получаем AoT сборку, нет никакой отладочной информации, нет трассировки событий. Tree Shaking выбросил весь неиспользуемый код.

Другие способы открыть Observatory



http://127.0.0.1:54513/gmPNJNKVVGQ=/ws


meyerweb.com/eric/tools/dencoder/
URL Decoder/Encoder
http://127.0.0.1:9100/#/?ide=Android-Studio&uri=http%253A%252F%252F127.0.0.1%253A52660%252FxA1Md9ToBw4%253D
Decode
Encode
meyerweb.com/eric/tools/dencoder/
URL Decoder/Encoder
http://127.0.0.1:9100/#/?ide=Android-Studio&uri=http://127.0.0.1:52660/xA1Md9ToBw4=
Decode
Encode
Flutter DevTools Uri

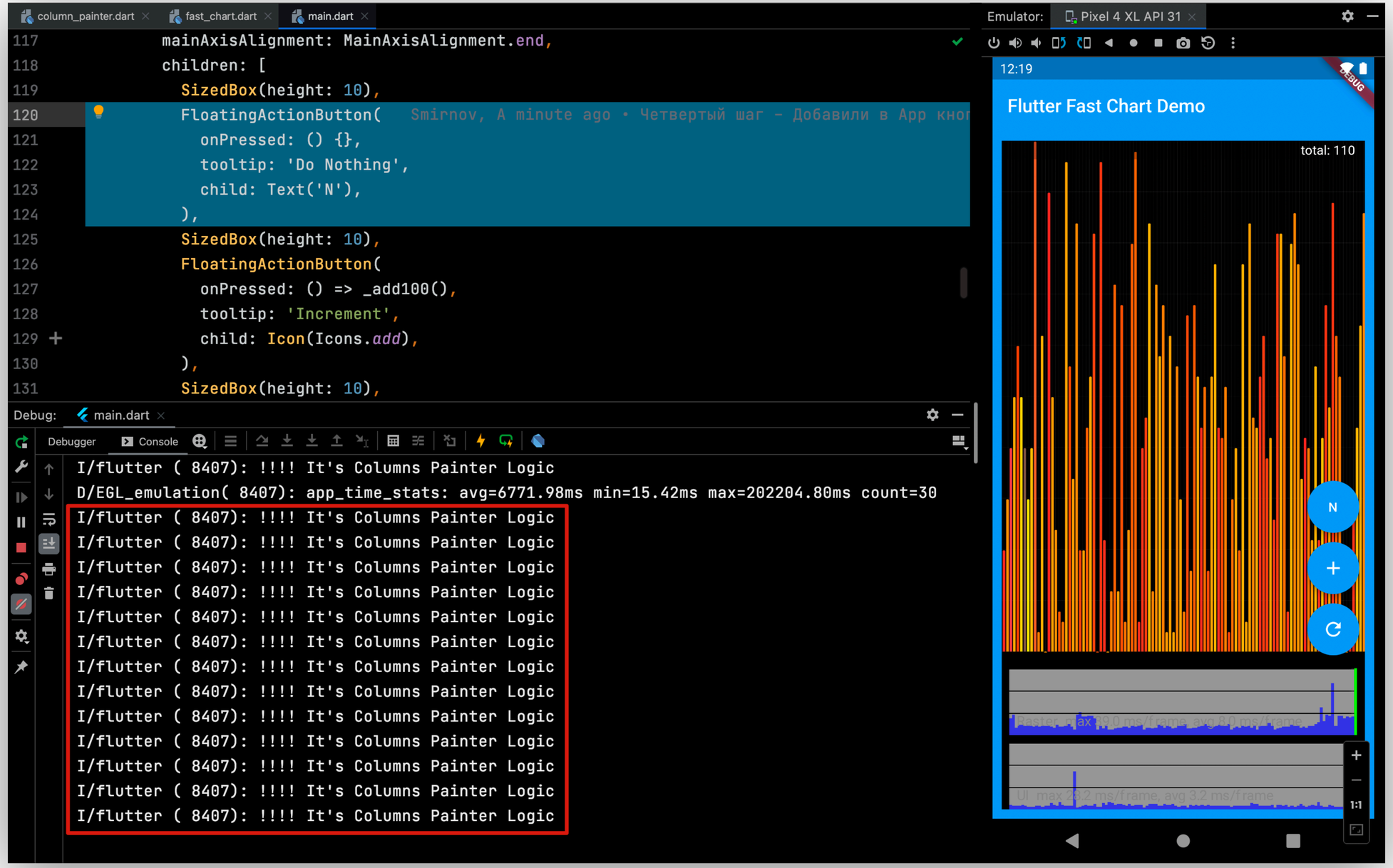
Observatory - кажется, полезный инструмент?

Фикс проблемы

git checkout tags/step4

Step 4
button doing the nothing

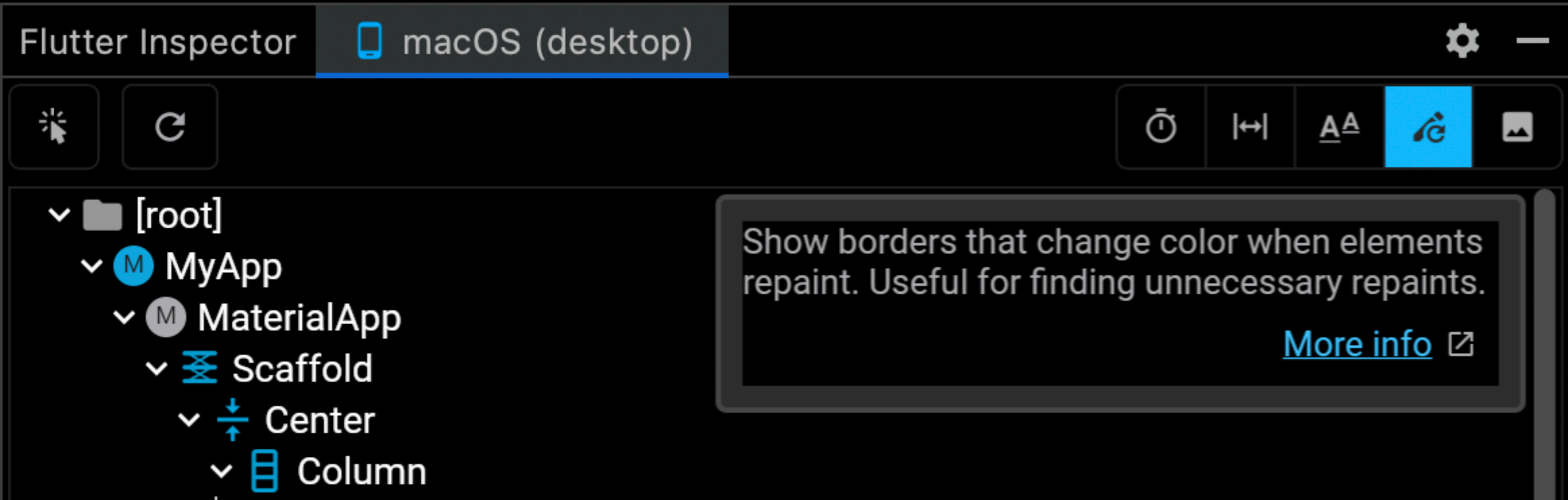
git checkout tags/step4_CircularProgressIndicator


column_painter.dart
fast_chart.dart
main.dart
117 mainAxisAlignment: MainAxisAlignment.end,
118 children: [
119 SizedBox(height: 10),
120 FloatingActionButton(Smirnov, A minute ago • Четвертый шаг - Добавили в App кно
121 onPressed: () {},
122 tooltip: 'Do Nothing',
123 child: Text('N'),
124),
125 SizedBox(height: 10),
126 FloatingActionButton(
127 onPressed: () => _add100(),
128 tooltip: 'Increment',
129 child: Icon(Icons.add),
130),
131 SizedBox(height: 10),
Debug:
main.dart
Debugger
Console
I/flutter (8407): !!!! It's Columns Painter Logic
D/EGL_emulation(8407): app_time_stats: avg=6771.98ms min=15.42ms max=202204.80ms count=30
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
I/flutter (8407): !!!! It's Columns Painter Logic
Emulator:
Pixel 4 XL API 31
12:19
DEBUG
Flutter Fast Chart Demo
total: 110
N
1:1

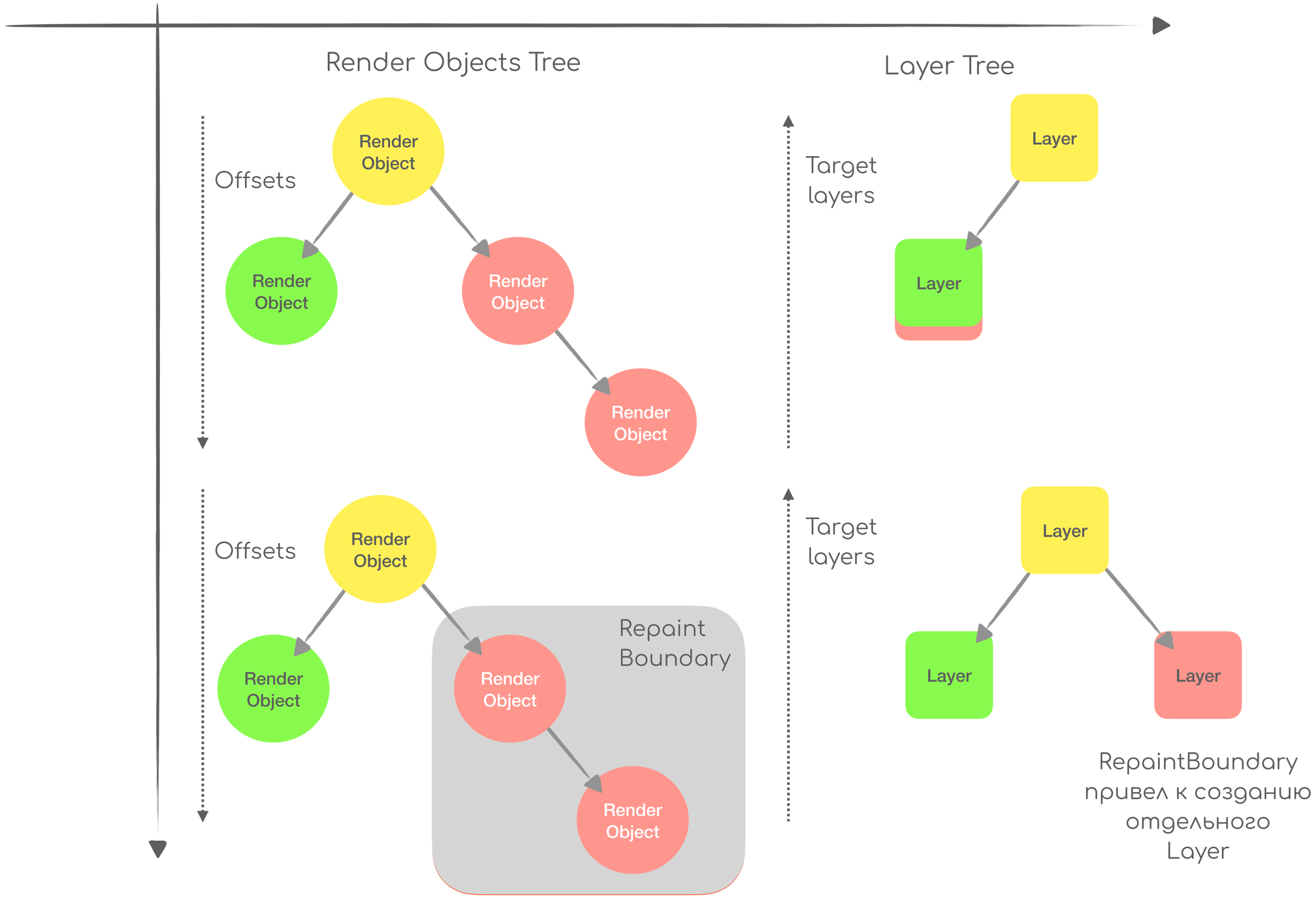
mobius

RepaintBoundary

```dart
/// Overlay a rotating set of colors when repainting layers in checked mode.
///
/// See also:
///
///  * [RepaintBoundary], which can be used to contain repaints when unchanged
///    areas are being excessively repainted.
bool debugRepaintRainbowEnabled = false;
```

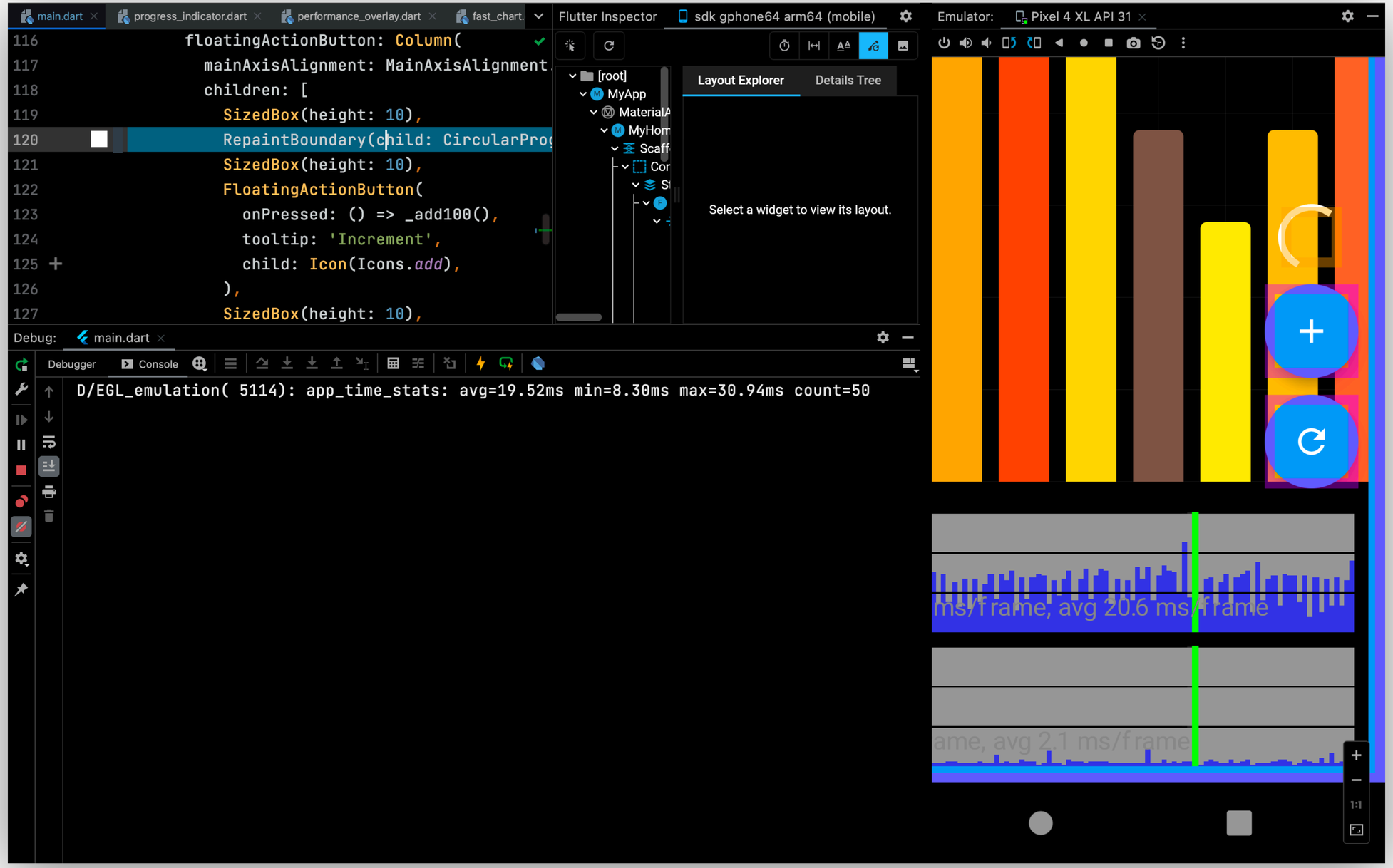


```dart
  /// It's possible that the [paint] method will get called even if
  /// [shouldRepaint] returns false (e.g. if an ancestor or descendant needed to
  /// be repainted). It's also possible that the [paint] method will get called
  /// without [shouldRepaint] being called at all (e.g. if the box changes
  /// size).
  ///
  /// If a custom delegate has a particularly expensive paint function such that
  /// repaints should be avoided as much as possible, a [RepaintBoundary] or
  /// [RenderRepaintBoundary] (or other render object with
  /// [RenderObject.isRepaintBoundary] set to true) might be helpful.
  ///
  /// The `oldDelegate` argument will never be null.
  bool shouldRepaint(covariant CustomPainter oldDelegate);
```


Render Objects Tree
Layer Tree
Offsets
Render Object
Render Object
Render Object
Render Object
Target layers
Layer
Layer
Offsets
Render Object
Render Object
Render Object
Render Object
Repaint Boundary
Target layers
Layer
Layer
Layer
RepaintBoundary привел к созданию отдельного Layer

Step 4
Фикс проблемы с использованием RepaintBondary

git checkout tags/step4_RepaintBoundaryInFastChart


main.dart
progress_indicator.dart
performance_overlay.dart
fast_chart.
floatingActionButton: Column(
mainAxisAlignment: MainAxisAlignment.
children: [
SizedBox(height: 10),
RepaintBoundary(child: CircularProg
SizedBox(height: 10),
FloatingActionButton(
onPressed: () => _add100(),
tooltip: 'Increment',
child: Icon(Icons.add),
),
SizedBox(height: 10),
Flutter Inspector
sdk gphone64 arm64 (mobile)
[root]
MyApp
Layout Explorer
Details Tree
Select a widget to view its layout.
Debug:
main.dart
Debugger
Console
D/EGL_emulation(5114): app_time_stats: avg=19.52ms min=8.30ms max=30.94ms count=50
Emulator:
Pixel 4 XL API 31
ms/frame, avg 20.6 ms/frame
ame, avg 2.1 ms/frame

addRepaintBoundary property Null safety

bool addRepaintBoundary
final

Indicate to automatically add a `RepaintBoundary` widget around the animation. This allows to optimize the app performance by isolating the animation in its own `Layer`.

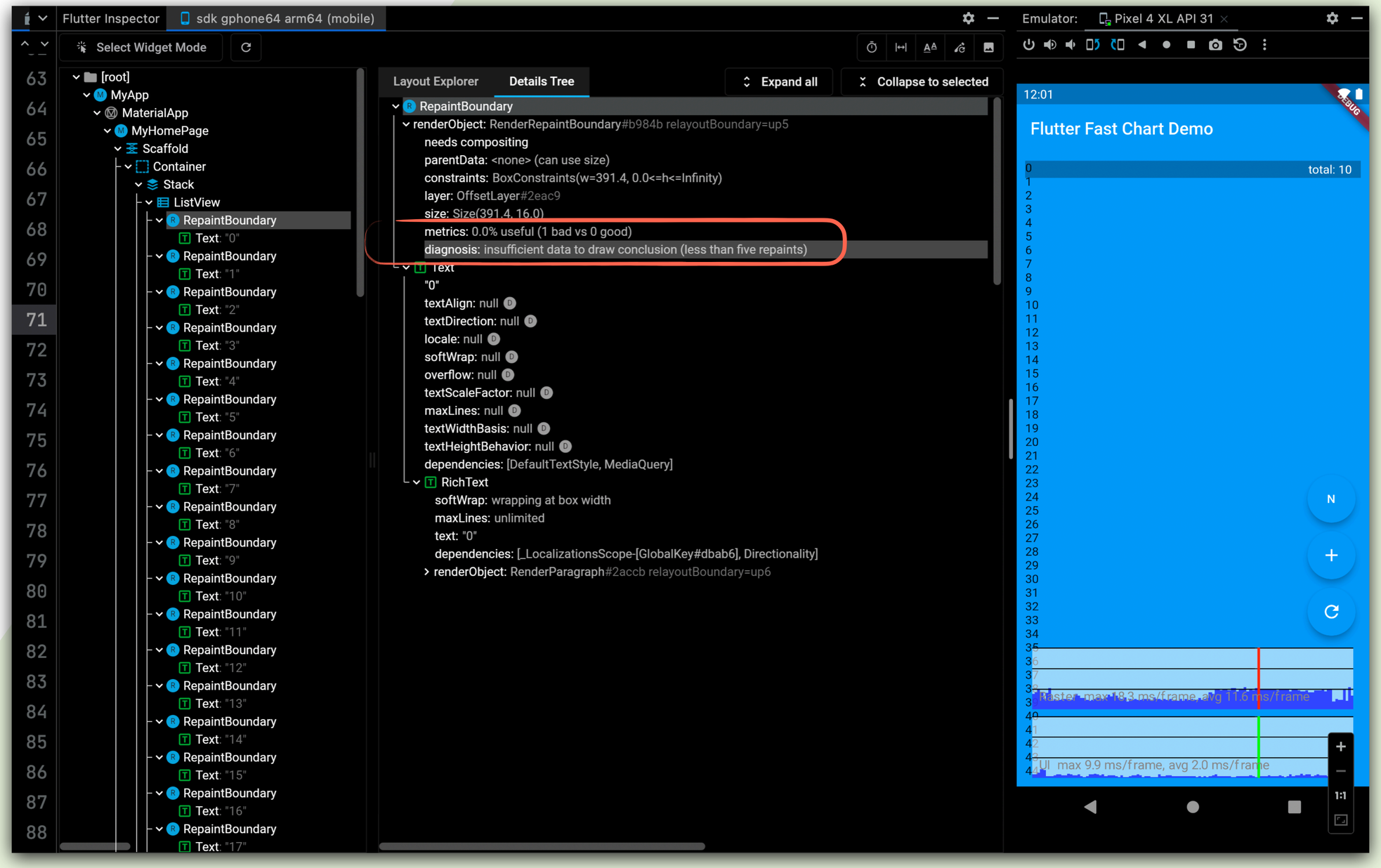
This property is `true` by default.

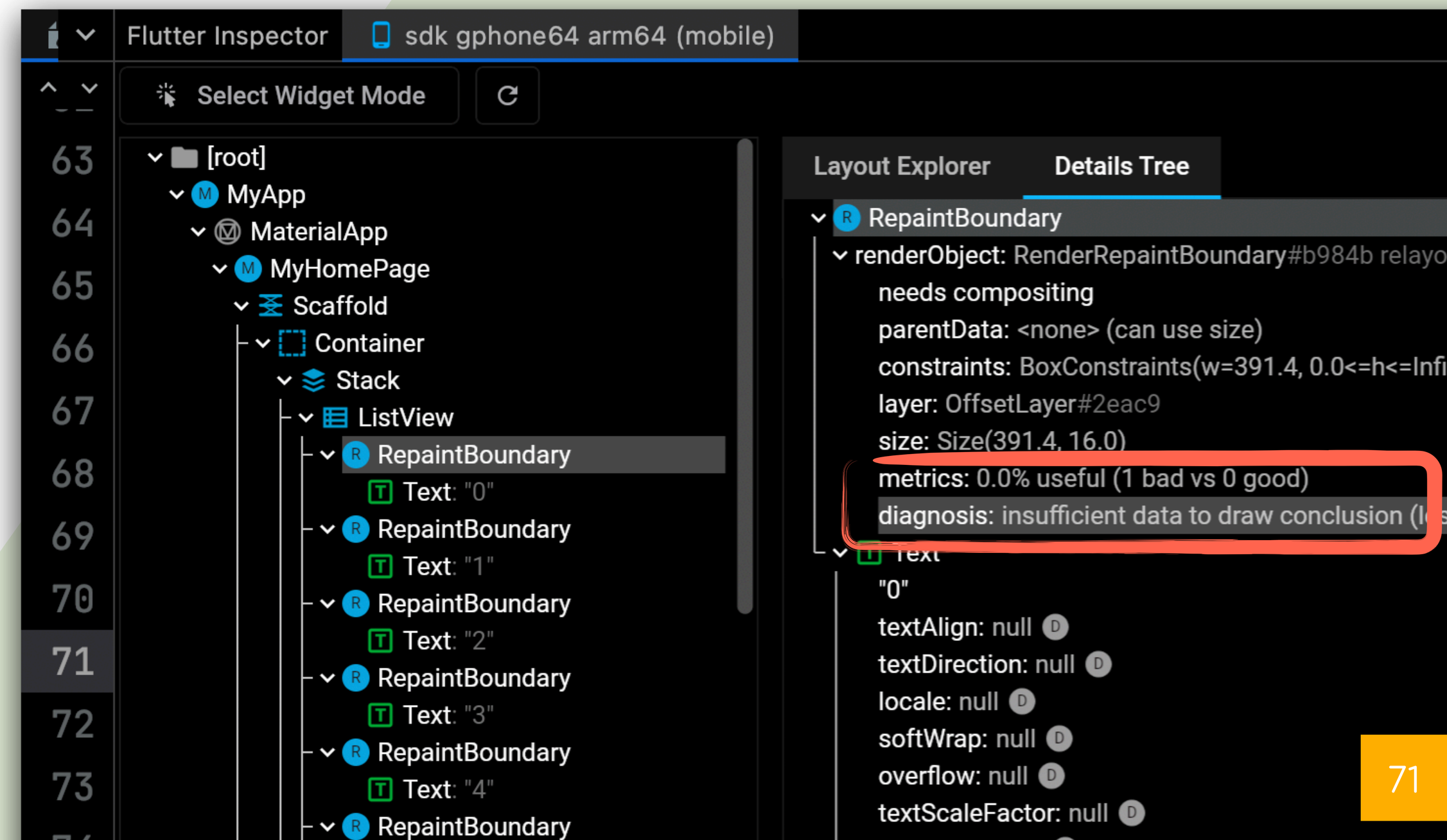
Implementation

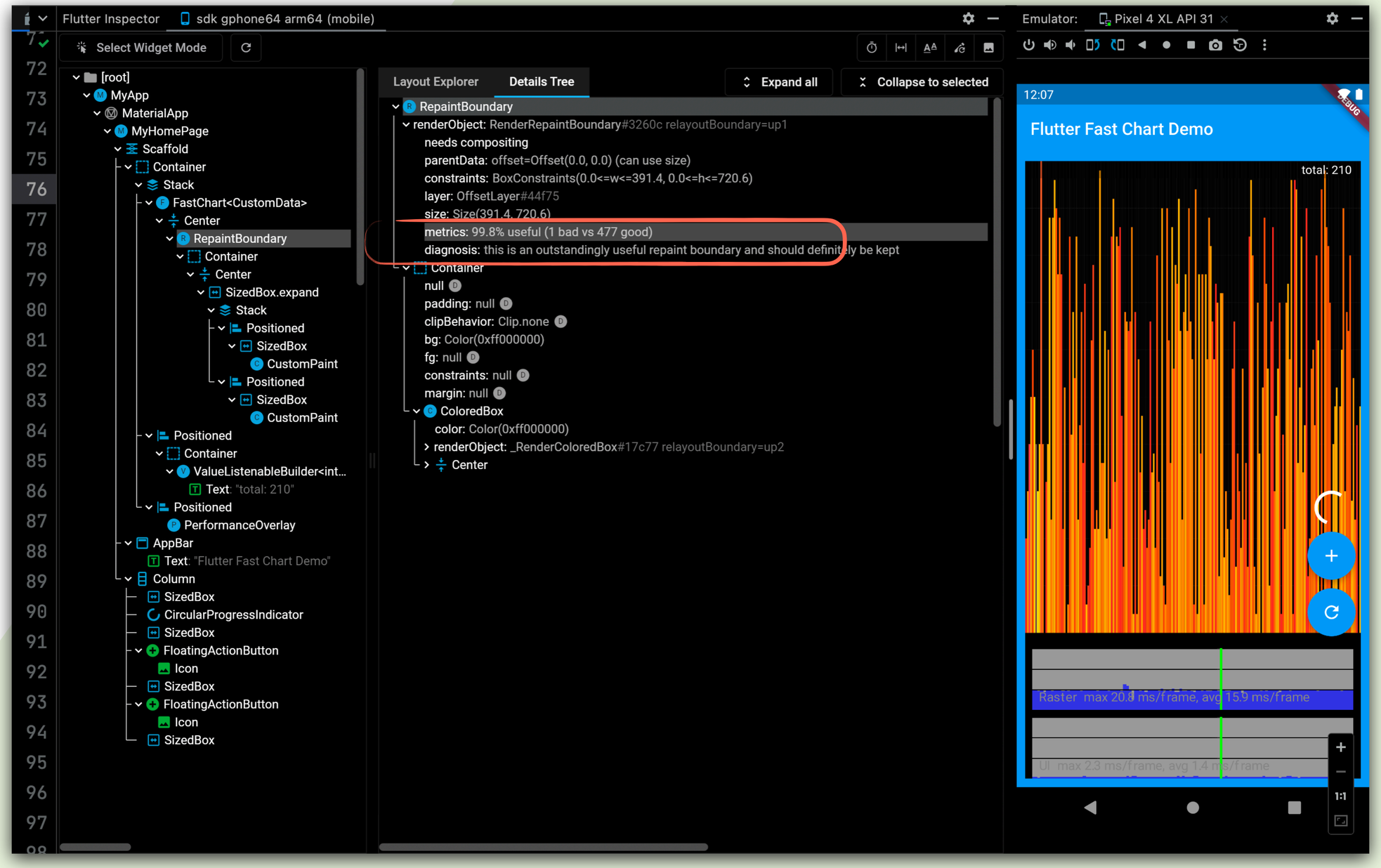
```
final bool addRepaintBoundary;
```

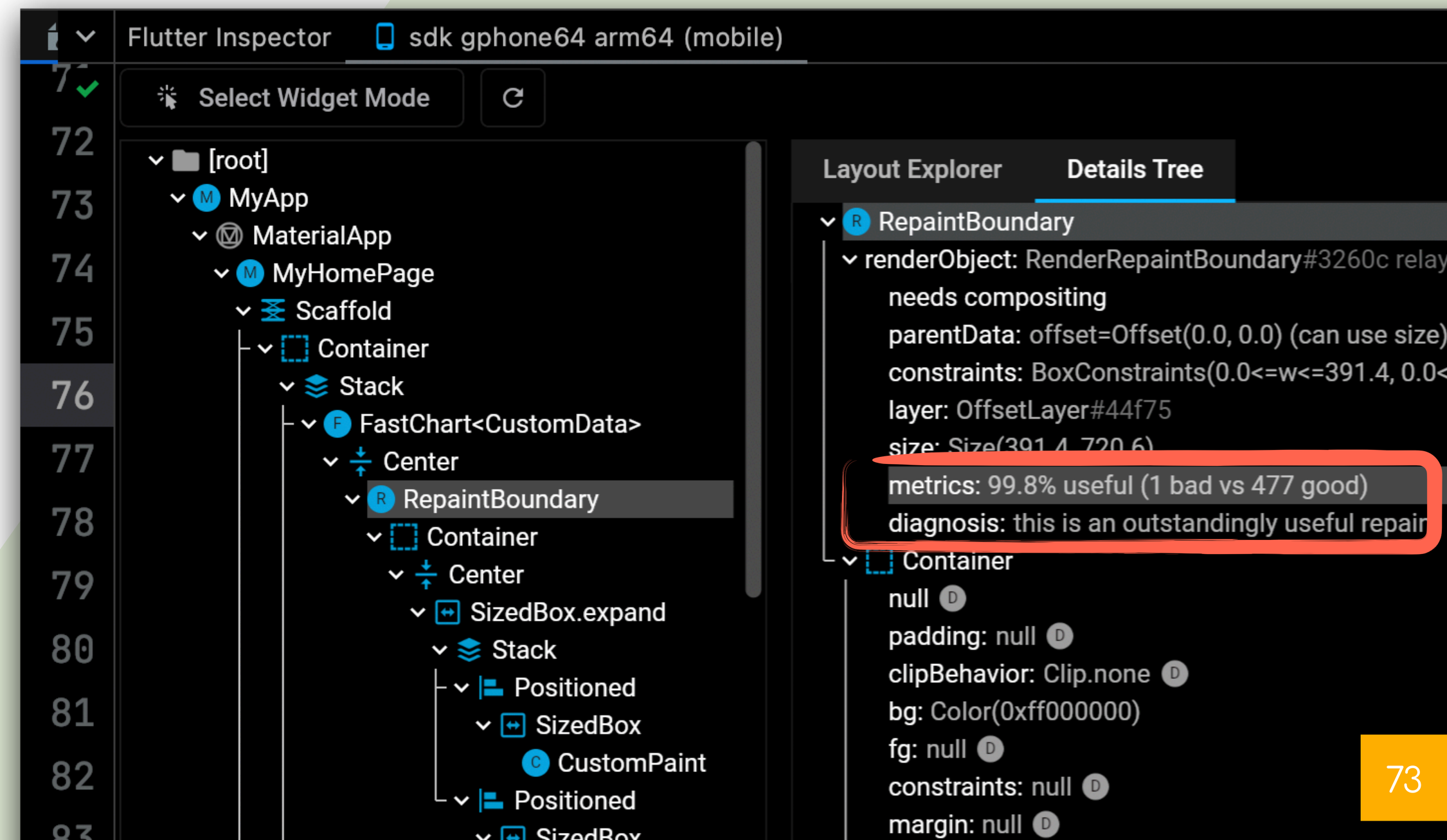
mobius

Замечания насчет использования RepaintBoundary


Flutter Inspector
sdk gphone64 arm64 (mobile)
Select Widget Mode
[root]
MyApp
MaterialApp
MyHomePage
Scaffold
Container
Stack
ListView
RepaintBoundary
Text: "0"
Text: "1"
Text: "2"
Text: "3"
Text: "4"
Text: "5"
Text: "6"
Text: "7"
Text: "8"
Text: "9"
Text: "10"
Text: "11"
Text: "12"
Text: "13"
Text: "14"
Text: "15"
Text: "16"
Text: "17"
Layout Explorer
Details Tree
Expand all
Collapse to selected
RepaintBoundary
renderObject: RenderRepaintBoundary#b984b relayoutBoundary=up5
needs compositing
parentData: <none> (can use size)
constraints: BoxConstraints(w=391.4, 0.0<=h<=Infinity)
layer: OffsetLayer#2eac9
size: Size(391.4, 16.0)
metrics: 0.0% useful (1 bad vs 0 good)
diagnosis: insufficient data to draw conclusion (less than five repaints)
Text
"0"
textAlign: null
textDirection: null
locale: null
softWrap: null
overflow: null
textScaleFactor: null
maxLines: null
textWidthBasis: null
textHeightBehavior: null
dependencies: [DefaultTextStyle, MediaQuery]
RichText
softWrap: wrapping at box width
maxLines: unlimited
text: "0"
dependencies: [_LocalizationsScope-[GlobalKey#dbab6], Directionality]
renderObject: RenderParagraph#2accb relayoutBoundary=up6
Emulator:
Pixel 4 XL API 31
12:01
DEBUG
Flutter Fast Chart Demo
total: 10
Raster max 18.3 ms/frame, avg 11.6 ms/frame
UI max 9.9 ms/frame, avg 2.0 ms/frame
1:1

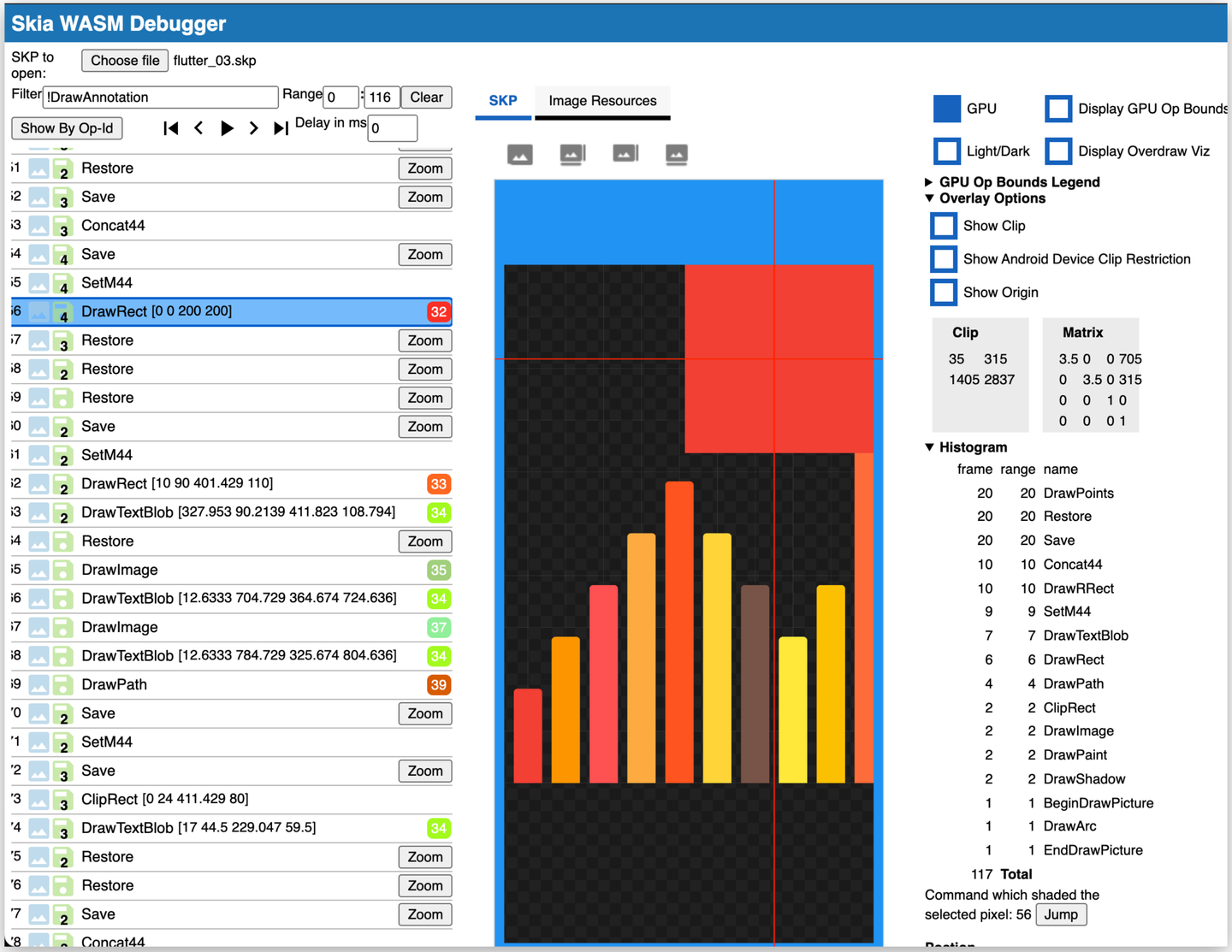

Flutter Inspector
sdk gphone64 arm64 (mobile)
Select Widget Mode
[root]
MyApp
MaterialApp
MyHomePage
Scaffold
Container
Stack
ListView
RepaintBoundary
Text: "0"
RepaintBoundary
Text: "1"
RepaintBoundary
Text: "2"
RepaintBoundary
Text: "3"
RepaintBoundary
Text: "4"
RepaintBoundary
Layout Explorer
Details Tree
RepaintBoundary
renderObject: RenderRepaintBoundary#b984b relayo
needs compositing
parentData: <none> (can use size)
constraints: BoxConstraints(w=391.4, 0.0<=h<=Infi
layer: OffsetLayer#2eac9
size: Size(391.4, 16.0)
metrics: 0.0% useful (1 bad vs 0 good)
diagnosis: insufficient data to draw conclusion (l
"0"
textAlign: null
textDirection: null
locale: null
softWrap: null
overflow: null
textScaleFactor: null
63
64
65
66
67
68
69
70
71
72
73

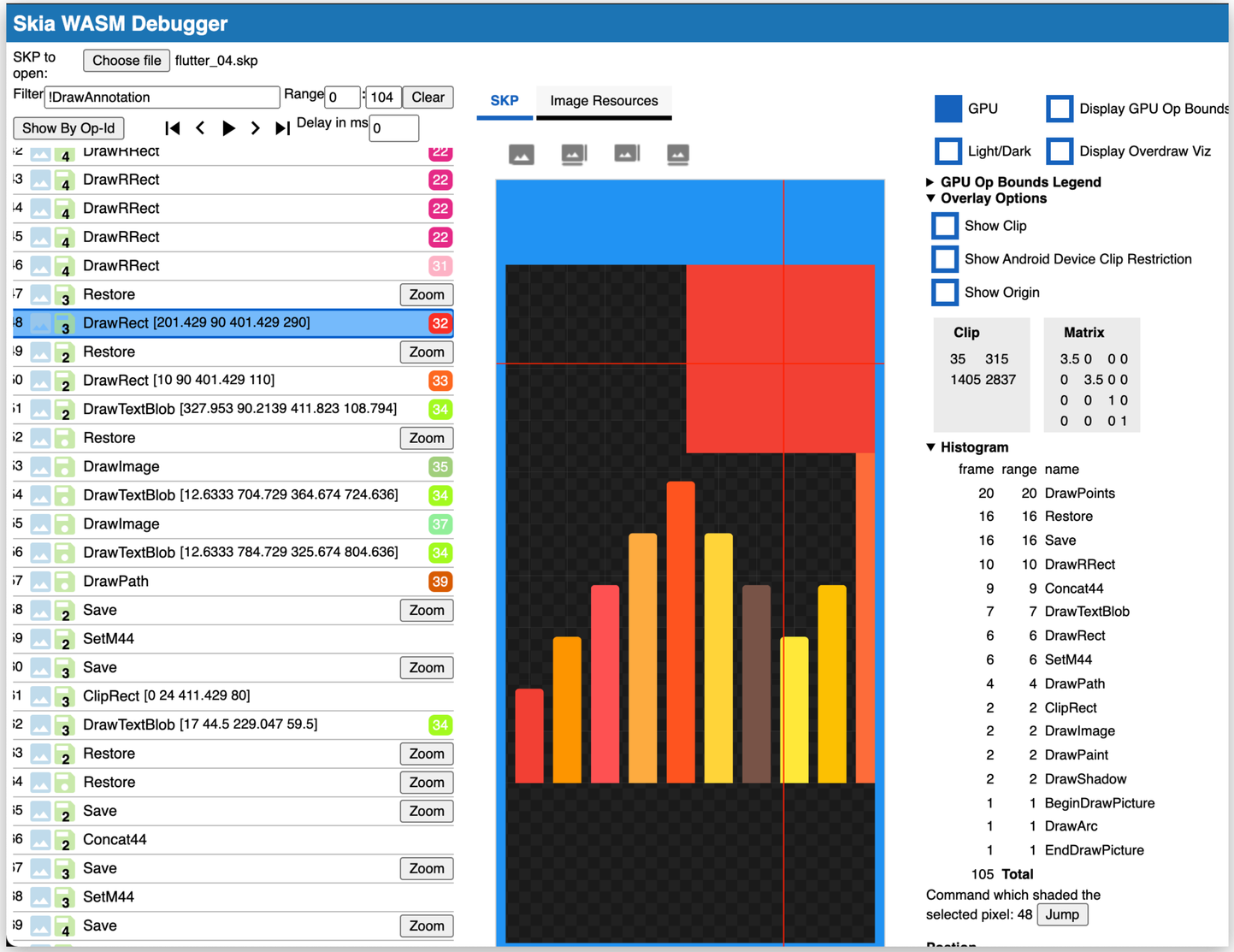

Flutter Inspector
sdk gphone64 arm64 (mobile)
Select Widget Mode
Layout Explorer
Details Tree
Expand all
Collapse to selected
[root]
MyApp
MaterialApp
MyHomePage
Scaffold
Container
Stack
FastChart<CustomData>
Center
RepaintBoundary
Container
Center
SizedBox.expand
Stack
Positioned
SizedBox
CustomPaint
Positioned
SizedBox
CustomPaint
Positioned
Container
ValueListenableBuilder<int...
Text: "total: 210"
Positioned
PerformanceOverlay
AppBar
Text: "Flutter Fast Chart Demo"
Column
SizedBox
CircularProgressIndicator
SizedBox
FloatingActionButton
Icon
SizedBox
FloatingActionButton
Icon
SizedBox
RepaintBoundary
renderObject: RenderRepaintBoundary#3260c relayoutBoundary=up1
needs compositing
parentData: offset=Offset(0.0, 0.0) (can use size)
constraints: BoxConstraints(0.0<=w<=391.4, 0.0<=h<=720.6)
layer: OffsetLayer#44f75
size: Size(391.4, 720.6)
metrics: 99.8% useful (1 bad vs 477 good)
diagnosis: this is an outstandingly useful repaint boundary and should definitely be kept
Container
null
padding: null
clipBehavior: Clip.none
bg: Color(0xff000000)
fg: null
constraints: null
margin: null
ColoredBox
color: Color(0xff000000)
renderObject: _RenderColoredBox#17c77 relayoutBoundary=up2
Center
Emulator:
Pixel 4 XL API 31
12:07
DEBUG
Flutter Fast Chart Demo
total: 210
Raster max 20.8 ms/frame, avg 15.9 ms/frame
UI max 2.3 ms/frame, avg 1.4 ms/frame
1:1


Flutter Inspector
sdk gphone64 arm64 (mobile)
Select Widget Mode
[root]
MyApp
MaterialApp
MyHomePage
Scaffold
Container
Stack
FastChart<CustomData>
Center
RepaintBoundary
Container
Center
SizedBox.expand
Stack
Positioned
SizedBox
CustomPaint
Positioned
Layout Explorer
Details Tree
RepaintBoundary
renderObject: RenderRepaintBoundary#3260c relay
needs compositing
parentData: offset=Offset(0.0, 0.0) (can use size)
constraints: BoxConstraints(0.0<=w<=391.4, 0.0<
layer: OffsetLayer#44f75
metrics: 99.8% useful (1 bad vs 477 good)
diagnosis: this is an outstandingly useful repair
Container
null
padding: null
clipBehavior: Clip.none
bg: Color(0xff000000)
fg: null
constraints: null
margin: null

mobius

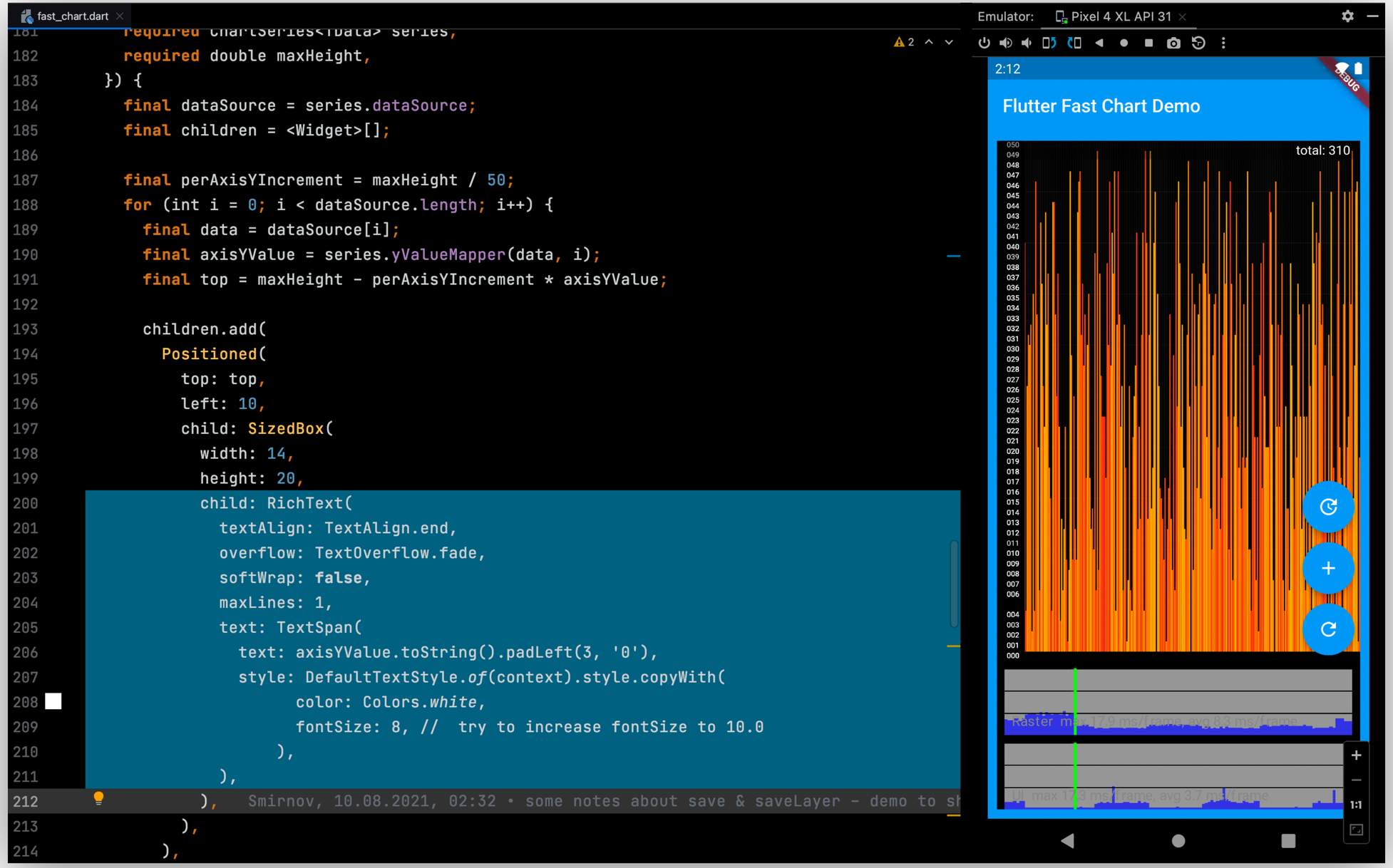
Skia Rendering With and Without RepaintBoundary


Skia WASM Debugger
SKP to open:
Choose file
flutter_03.skp
Filter
!DrawAnnotation
Range
0
116
Clear
Show By Op-Id
Delay in ms
0
SKP
Image Resources
GPU
Display GPU Op Bounds
Light/Dark
Display Overdraw Viz
GPU Op Bounds Legend
Overlay Options
Show Clip
Show Android Device Clip Restriction
Show Origin
Clip
35 315
1405 2837
Matrix
3.5 0 0 705
0 3.5 0 315
0 0 1 0
0 0 0 1
Histogram
frame range name
20 20 DrawPoints
20 20 Restore
20 20 Save
10 10 Concat44
10 10 DrawRRect
9 9 SetM44
7 7 DrawTextBlob
6 6 DrawRect
4 4 DrawPath
2 2 ClipRect
2 2 DrawImage
2 2 DrawPaint
2 2 DrawShadow
1 1 BeginDrawPicture
1 1 DrawArc
1 1 EndDrawPicture
117 Total
Command which shaded the selected pixel: 56
Jump
Restore
Save
Concat44
SetM44
DrawRect [0 0 200 200]
DrawRect [10 90 401.429 110]
DrawTextBlob [327.953 90.2139 411.823 108.794]
DrawImage
DrawTextBlob [12.6333 704.729 364.674 724.636]
DrawTextBlob [12.6333 784.729 325.674 804.636]
DrawPath
ClipRect [0 24 411.429 80]
DrawTextBlob [17 44.5 229.047 59.5]
Zoom


Skia WASM Debugger
SKP to open:
Choose file
flutter_04.skp
Filter
!DrawAnnotation
Range
0
104
Clear
Show By Op-Id
Delay in ms
0
DrawRRect 22
DrawRRect 22
DrawRRect 22
DrawRRect 22
DrawRRect 31
Restore Zoom
DrawRect [201.429 90 401.429 290] 32
Restore Zoom
DrawRect [10 90 401.429 110] 33
DrawTextBlob [327.953 90.2139 411.823 108.794] 34
Restore Zoom
DrawImage 35
DrawTextBlob [12.6333 704.729 364.674 724.636] 34
DrawImage 37
DrawTextBlob [12.6333 784.729 325.674 804.636] 34
DrawPath 39
Save Zoom
SetM44
Save Zoom
ClipRect [0 24 411.429 80]
DrawTextBlob [17 44.5 229.047 59.5] 34
Restore Zoom
Restore Zoom
Save Zoom
Concat44
Save Zoom
SetM44
Save Zoom
SKP
Image Resources
GPU
Display GPU Op Bounds
Light/Dark
Display Overdraw Viz
GPU Op Bounds Legend
Overlay Options
Show Clip
Show Android Device Clip Restriction
Show Origin
Clip
35 315
1405 2837
Matrix
3.5 0 0 0
0 3.5 0 0
0 0 1 0
0 0 0 1
Histogram
frame range name
20 20 DrawPoints
16 16 Restore
16 16 Save
10 10 DrawRRect
9 9 Concat44
7 7 DrawTextBlob
6 6 DrawRect
6 6 SetM44
4 4 DrawPath
2 2 ClipRect
2 2 DrawImage
2 2 DrawPaint
2 2 DrawShadow
1 1 BeginDrawPicture
1 1 DrawArc
1 1 EndDrawPicture
105 Total
Command which shaded the selected pixel: 48
Jump

Деградация Raster Thread

git checkout tags/rasterThreadDegradation

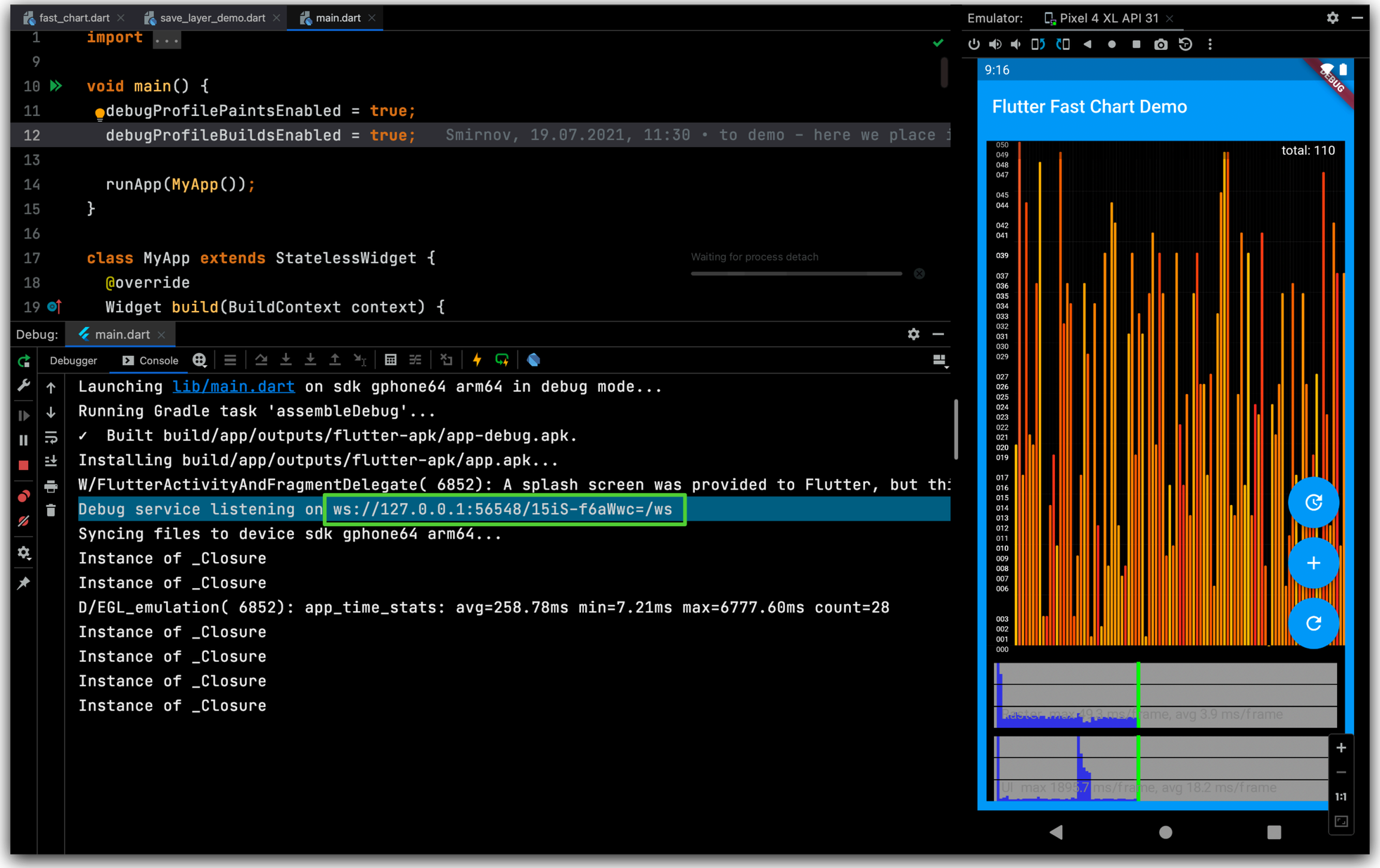

fast_chart.dart
required ChartSeries<TData> series,
required double maxHeight,
}) {
final dataSource = series.dataSource;
final children = <Widget>[];
final perAxisYIncrement = maxHeight / 50;
for (int i = 0; i < dataSource.length; i++) {
final data = dataSource[i];
final axisYValue = series.yValueMapper(data, i);
final top = maxHeight - perAxisYIncrement * axisYValue;
children.add(
Positioned(
top: top,
left: 10,
child: SizedBox(
width: 14,
height: 20,
child: RichText(
textAlign: TextAlign.end,
overflow: TextOverflow.fade,
softWrap: false,
maxLines: 1,
text: TextSpan(
text: axisYValue.toString().padLeft(3, '0'),
style: DefaultTextStyle.of(context).style.copyWith(
color: Colors.white,
fontSize: 8, // try to increase fontSize to 10.0
),
),
),
),
Smirnov, 10.08.2021, 02:32 • some notes about save & saveLayer - demo to sh
),
),
Emulator: Pixel 4 XL API 31
2:12
DEBUG
Flutter Fast Chart Demo
total: 310
Raster max 17.9 ms/frame, avg 8.3 ms/frame
1:1

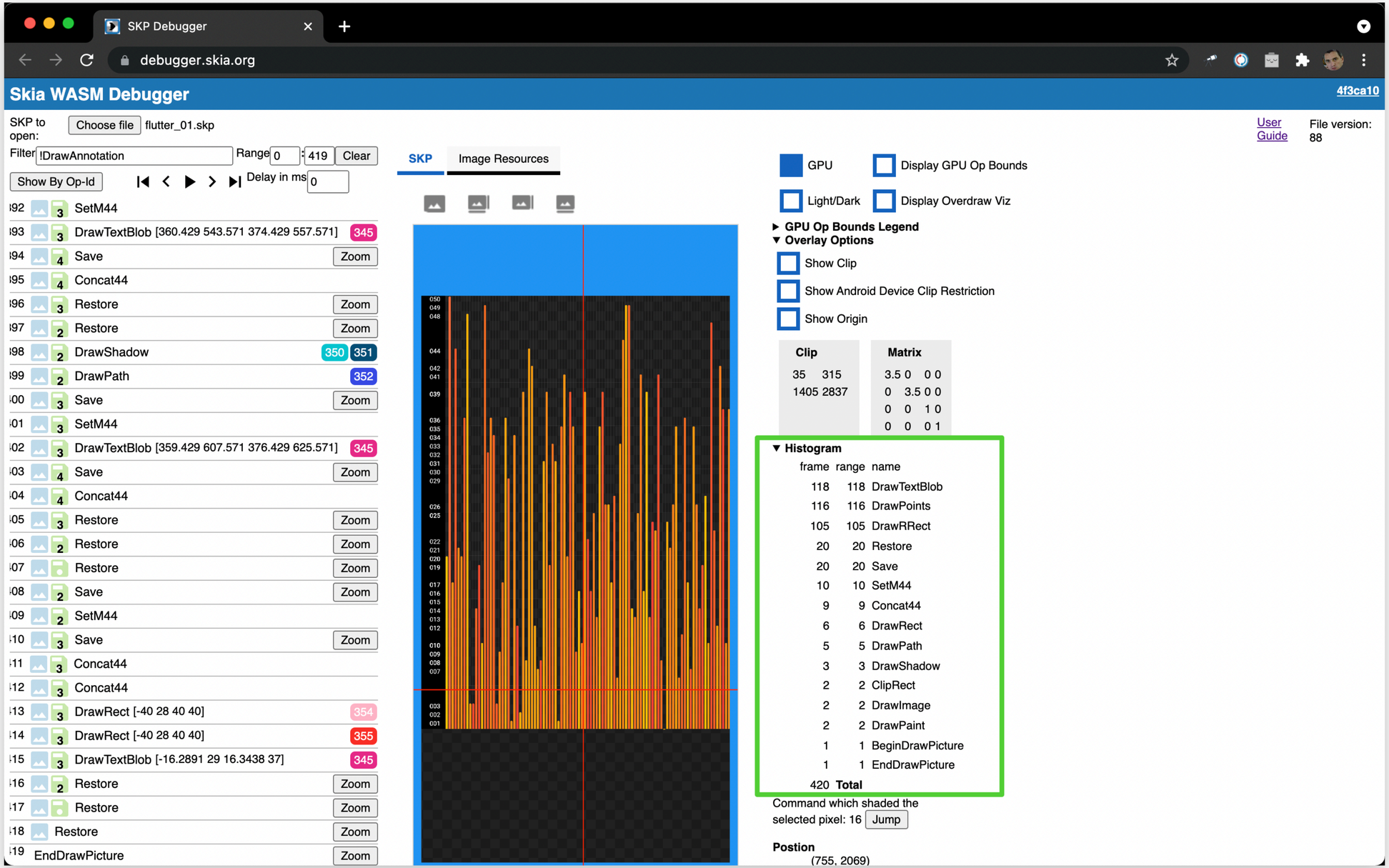
Как же понять, где проблема в Raster Thread?
Как проанализировать сложность нашего рендеринга?
SkiaDebugger!

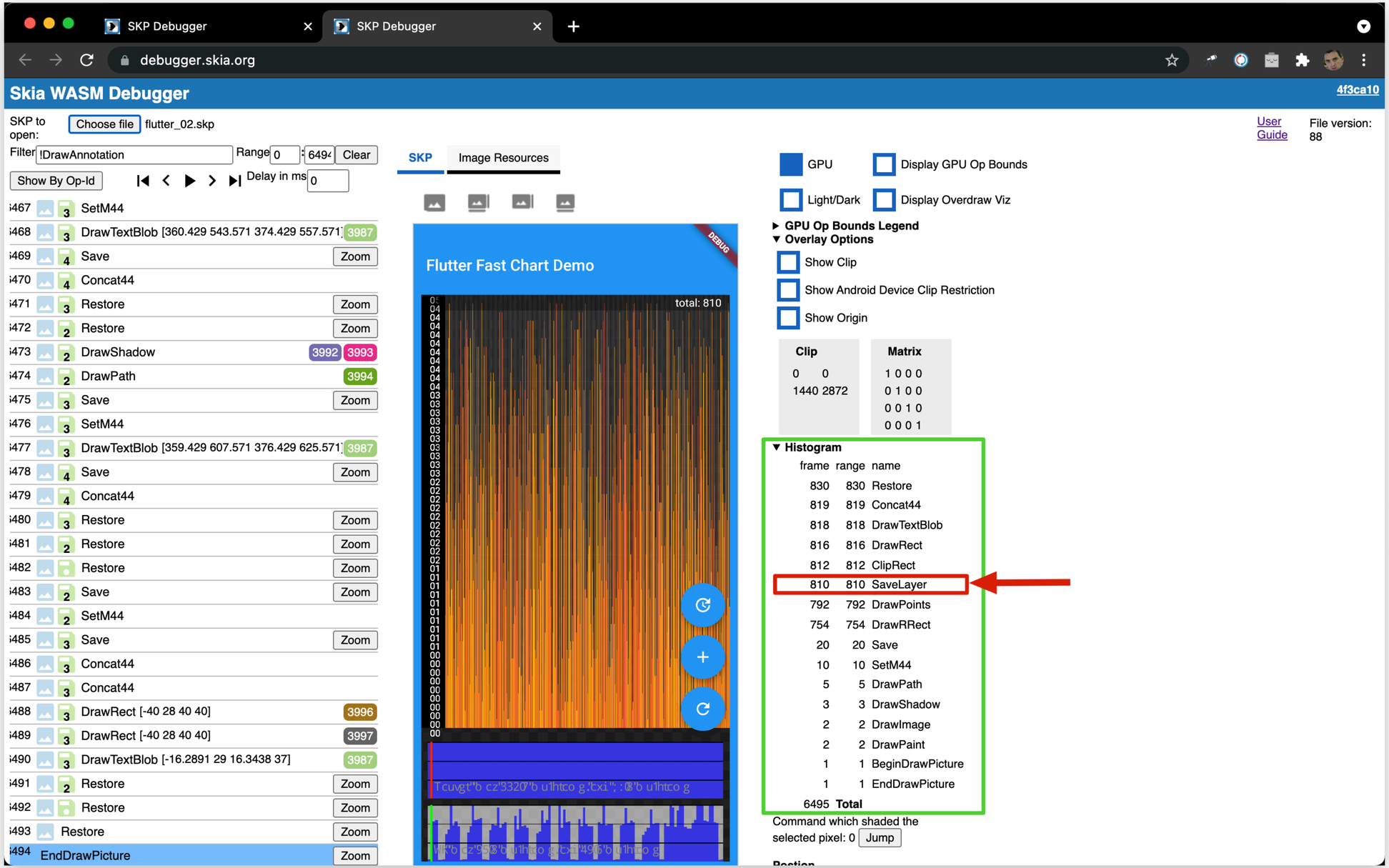
```
❯ flutter screenshot --type=skia --observatory-url=http://127.0.0.1:56548/15iS-f6aWwc=/ws
```

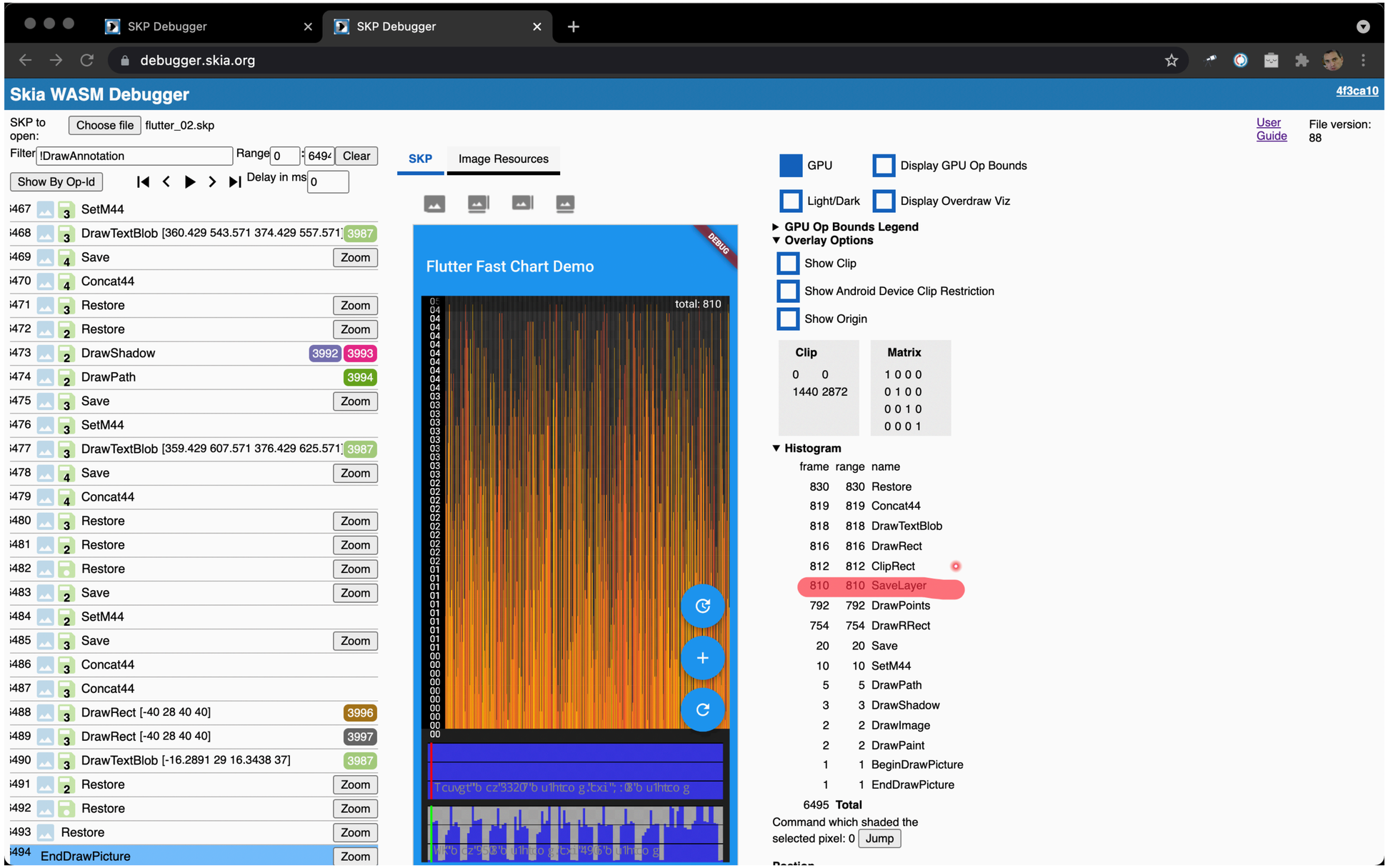
mobius

SkiaDebugger


fast_chart.dart
save_layer_demo.dart
main.dart
import ...
void main() {
debugProfilePaintsEnabled = true;
debugProfileBuildsEnabled = true;
runApp(MyApp());
}
class MyApp extends StatelessWidget {
@override
Widget build(BuildContext context) {
Waiting for process detach
Debug:
main.dart
Debugger
Console
Launching lib/main.dart on sdk gphone64 arm64 in debug mode...
Running Gradle task 'assembleDebug'...
✓ Built build/app/outputs/flutter-apk/app-debug.apk.
Installing build/app/outputs/flutter-apk/app.apk...
W/FlutterActivityAndFragmentDelegate(6852): A splash screen was provided to Flutter, but th
Debug service listening on ws://127.0.0.1:56548/15iS-f6aWwc=/ws
Syncing files to device sdk gphone64 arm64...
Instance of _Closure
Instance of _Closure
D/EGL_emulation(6852): app_time_stats: avg=258.78ms min=7.21ms max=6777.60ms count=28
Instance of _Closure
Instance of _Closure
Instance of _Closure
Instance of _Closure
Emulator:
Pixel 4 XL API 31
9:16
DEBUG
Flutter Fast Chart Demo
total: 110


SKP Debugger
debugger.skia.org
Skia WASM Debugger
4f3ca10
SKP to open:
Choose file
flutter_01.skp
User Guide
File version: 88
Filter
!DrawAnnotation
Range
0
419
Clear
Show By Op-Id
Delay in ms
0
SKP
Image Resources
GPU
Display GPU Op Bounds
Light/Dark
Display Overdraw Viz
GPU Op Bounds Legend
Overlay Options
Show Clip
Show Android Device Clip Restriction
Show Origin
Clip
35 315
1405 2837
Matrix
3.5 0 0 0
0 3.5 0 0
0 0 1 0
0 0 0 1
Histogram
frame range name
118 118 DrawTextBlob
116 116 DrawPoints
105 105 DrawRRect
20 20 Restore
20 20 Save
10 10 SetM44
9 9 Concat44
6 6 DrawRect
5 5 DrawPath
3 3 DrawShadow
2 2 ClipRect
2 2 DrawImage
2 2 DrawPaint
1 1 BeginDrawPicture
1 1 EndDrawPicture
420 Total
Command which shaded the selected pixel: 16
Jump
Postion
(755, 2069)
192 SetM44
193 DrawTextBlob [360.429 543.571 374.429 557.571] 345
194 Save Zoom
195 Concat44
196 Restore Zoom
197 Restore Zoom
198 DrawShadow 350 351
199 DrawPath 352
200 Save Zoom
201 SetM44
202 DrawTextBlob [359.429 607.571 376.429 625.571] 345
203 Save Zoom
204 Concat44
205 Restore Zoom
206 Restore Zoom
207 Restore Zoom
208 Save Zoom
209 SetM44
210 Save Zoom
211 Concat44
212 Concat44
213 DrawRect [-40 28 40 40] 354
214 DrawRect [-40 28 40 40] 355
215 DrawTextBlob [-16.2891 29 16.3438 37] 345
216 Restore Zoom
217 Restore Zoom
218 Restore Zoom
219 EndDrawPicture Zoom


SKP Debugger
SKP Debugger
debugger.skia.org
Skia WASM Debugger
4f3ca10
SKP to open:
Choose file
flutter_02.skp
User Guide
File version: 88
Filter
!DrawAnnotation
Range
0
649
Clear
Show By Op-Id
Delay in ms
0
SKP
Image Resources
GPU
Display GPU Op Bounds
Light/Dark
Display Overdraw Viz
GPU Op Bounds Legend
Overlay Options
Show Clip
Show Android Device Clip Restriction
Show Origin
Clip
0 0
1440 2872
Matrix
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
Histogram
frame range name
830 830 Restore
819 819 Concat44
818 818 DrawTextBlob
816 816 DrawRect
812 812 ClipRect
810 810 SaveLayer
792 792 DrawPoints
754 754 DrawRRect
20 20 Save
10 10 SetM44
5 5 DrawPath
3 3 DrawShadow
2 2 DrawImage
2 2 DrawPaint
1 1 BeginDrawPicture
1 1 EndDrawPicture
6495 Total
Command which shaded the selected pixel: 0
Jump
Flutter Fast Chart Demo
DEBUG
total: 810
#467 SetM44
#468 DrawTextBlob [360.429 543.571 374.429 557.571] 3987
#469 Save Zoom
#470 Concat44
#471 Restore Zoom
#472 Restore Zoom
#473 DrawShadow 3992 3993
#474 DrawPath 3994
#475 Save Zoom
#476 SetM44
#477 DrawTextBlob [359.429 607.571 376.429 625.571] 3987
#478 Save Zoom
#479 Concat44
#480 Restore Zoom
#481 Restore Zoom
#482 Restore Zoom
#483 Save Zoom
#484 SetM44
#485 Save Zoom
#486 Concat44
#487 Concat44
#488 DrawRect [-40 28 40 40] 3996
#489 DrawRect [-40 28 40 40] 3997
#490 DrawTextBlob [-16.2891 29 16.3438 37] 3987
#491 Restore Zoom
#492 Restore Zoom
#493 Restore Zoom
#494 EndDrawPicture Zoom


SKP Debugger
SKP Debugger
debugger.skia.org
Skia WASM Debugger
4f3ca10
SKP to open:
Choose file
flutter_02.skp
User Guide
File version: 88
Filter
!DrawAnnotation
Range
0
6494
Clear
Show By Op-Id
Delay in ms
0
SKP
Image Resources
GPU
Display GPU Op Bounds
Light/Dark
Display Overdraw Viz
GPU Op Bounds Legend
Overlay Options
Show Clip
Show Android Device Clip Restriction
Show Origin
Clip
0 0
1440 2872
Matrix
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
Histogram
frame range name
830 830 Restore
819 819 Concat44
818 818 DrawTextBlob
816 816 DrawRect
812 812 ClipRect
810 810 SaveLayer
792 792 DrawPoints
754 754 DrawRRect
20 20 Save
10 10 SetM44
5 5 DrawPath
3 3 DrawShadow
2 2 DrawImage
2 2 DrawPaint
1 1 BeginDrawPicture
1 1 EndDrawPicture
6495 Total
Command which shaded the selected pixel: 0
Jump
Flutter Fast Chart Demo
DEBUG
total: 810
3467 SetM44
3468 DrawTextBlob [360.429 543.571 374.429 557.571] 3987
3469 Save Zoom
3470 Concat44
3471 Restore Zoom
3472 Restore Zoom
3473 DrawShadow 3992 3993
3474 DrawPath 3994
3475 Save Zoom
3476 SetM44
3477 DrawTextBlob [359.429 607.571 376.429 625.571] 3987
3478 Save Zoom
3479 Concat44
3480 Restore Zoom
3481 Restore Zoom
3482 Restore Zoom
3483 Save Zoom
3484 SetM44
3485 Save Zoom
3486 Concat44
3487 Concat44
3488 DrawRect [-40 28 40 40] 3996
3489 DrawRect [-40 28 40 40] 3997
3490 DrawTextBlob [-16.2891 29 16.3438 37] 3987
3491 Restore Zoom
3492 Restore Zoom
3493 Restore Zoom
3494 EndDrawPicture Zoom

Из документации про saveLayer

https://api.flutter.dev/flutter/dart-ui/Canvas/saveLayer.html

Performance considerations

Generally speaking, saveLayer is relatively expensive.

There are a several different hardware architectures for GPUs (graphics processing units, the hardware that handles graphics), but most of them involve batching commands and reordering them for performance. When layers are used, they cause the rendering pipeline to have to switch render target (from one layer to another).

Render target switches can flush the GPU's command buffer, which typically means that optimizations that one could get with larger batching are lost.

Render target switches also generate a lot of memory churn because the GPU needs to copy out the current frame buffer contents from the part of memory that's optimized for writing, and then needs to copy it back in once the previous render target (layer) is restored.

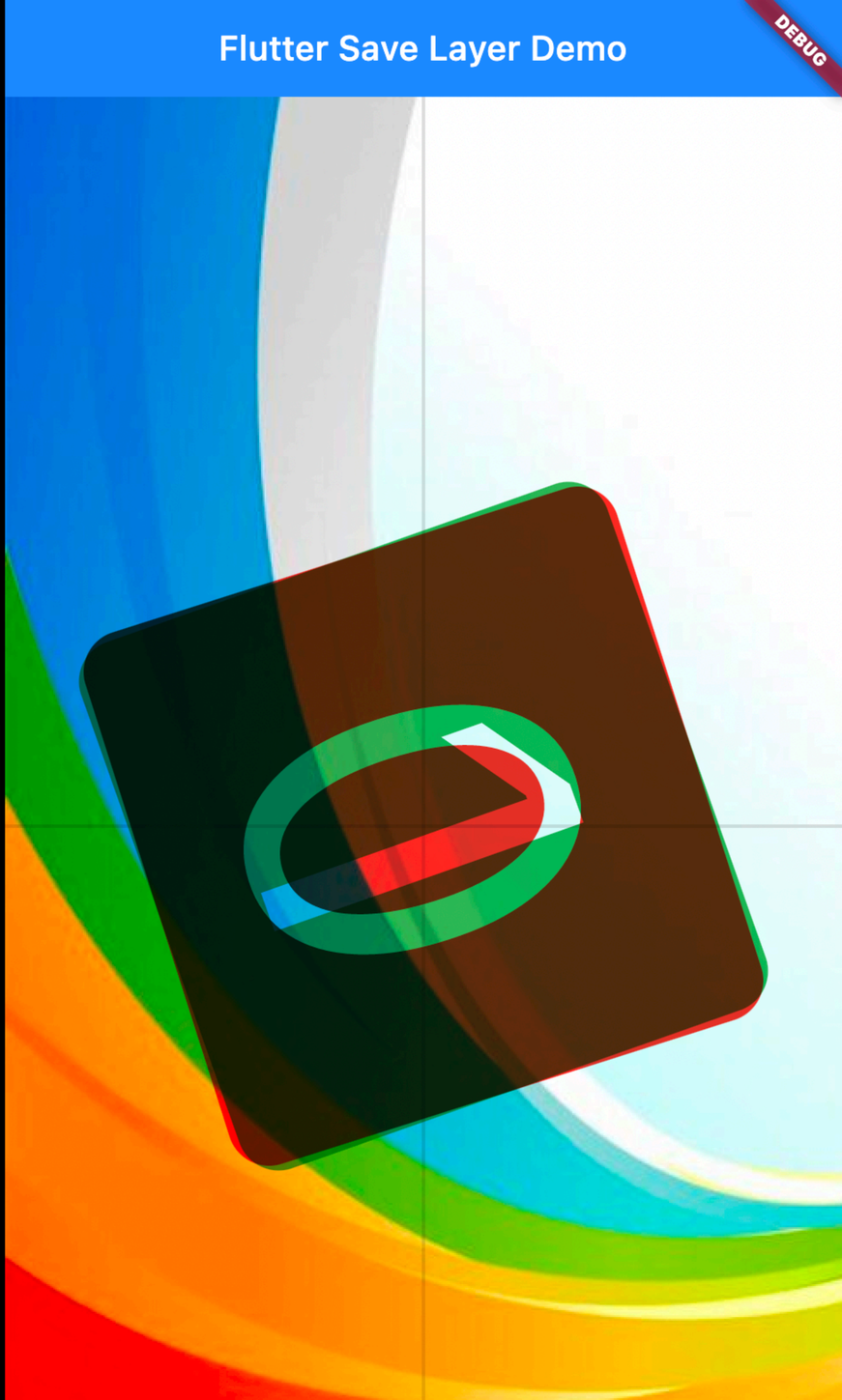
mobius

Обсуждаем разницу между применением Save & SaveLayer

git checkout tags/saveLayerDemoToShowTheDifferences

transparent_items_painter.dart | text_painter_in_save_layer.dart | geometry.dart | save_layer_demo.dart | license.txt | asset_bundle

```dart
///
/// Мы можем в принципе обойтись и без save layer, указав blend mode
/// при формировании кисти
///
if (useSaveLayer) {
  canvas.saveLayer(
      Offset.zero & size, Paint()..blendMode = BlendMode.multiply);

  rectPaint = Paint()
    ..color = index.isEven
        ? Colors.red.withOpacity(1)
        : Colors.green.withOpacity(.9)
    ..style = PaintingStyle.fill;
} else {
  rectPaint = Paint()
    ..color = index.isEven ? Colors.red : Colors.green
    ..style = PaintingStyle.fill
    ..blendMode = BlendMode.multiply; // try to remove it
}

var positive = math.Random().nextBool();
canvas.translate(size.width / 2, size.height / 2);
canvas.rotate((positive ? 2 : -2) * math.pi * math.Random().nextDouble());
canvas.translate(-size.width / 2, -size.height / 2);

canvas.drawRRect(
    RRect.fromRectAndRadius(rect, Radius.circular(24)), rectPaint);

final span = TextSpan(
```



```
Colors.green

ry to remove it

ht / 2);
i * math.Random().nextDouble());
ight / 2);
```

mobius

Рекомендации для saveLayer

https://flutter.dev/docs/perf/rendering/best-practices#apply-effects-only-when-needed

Apply effects only when needed

Use effects carefully, as they can be expensive. Some of them invoke `saveLayer()` behind the scenes, which can be an expensive operation.

Why is savelayer expensive?

Calling `saveLayer()` allocates an offscreen buffer. Drawing content into the offscreen buffer might trigger render target switches that are particularly slow in older GPUs.

Some general rules when applying specific effects:

- Use the `Opacity` widget only when necessary. See the Transparent image section in the Opacity API page for an example of applying opacity directly to an image, which is faster than using the Opacity widget.
- **Clipping** doesn't call `saveLayer()` (unless explicitly requested with `Clip.antiAliasWithSaveLayer`) so these operations aren't as expensive as Opacity, but clipping is still costly, so use with caution. By default, clipping is disabled (`Clip.none`), so you must explicitly enable it when needed.

Other widgets that might trigger `saveLayer()` and are potentially costly:

- `ShaderMask`
- `ColorFilter`
- `Chip`—might cause call to `saveLayer()` if `disabledColorAlpha != 0xff`
- `Text`—might cause call to `saveLayer()` if there's an `overflowShader`

Ways to avoid calls to `saveLayer()`:

- To implement fading in an image, consider using the `FadeInImage` widget, which applies a gradual opacity using the GPU's fragment shader. For more information, see the `Opacity` docs.
- To create a rectangle with rounded corners, instead of applying a clipping rectangle, consider using the `borderRadius` property offered by many of the widget classes.

mobius

Выводы

Рекомендации

https://flutter.dev/docs/perf/rendering/best-practices

https://flutter.dev/docs/perf/rendering/ui-performance

- Старайтесь обновлять состояния виджетов небольшими порциями
- Обновляйте состояние только тогда, когда это действительно необходимо
- Не держите большие вычисления в build методе. Если есть тяжелые операции, выносите их в отдельный изолят
- Минимизируйте использование дорогих виджетов, использующих тяжелые операции с битмапами, эффектами типа Blur, Opacity, и прочие, которые являются причиной появления saveLayer (https://flutter.dev/docs/perf/rendering/best-practices#apply-effects-only-when-needed).
- Используете RepaintBoundary для изоляции своих анимаций.

Инструменты

- Flutter Dev Tools
- Flutter Performance Tool Window
- Performance Overlay

- Observatory
- SkiaDebugger (https://debugger.skia.org/)

Преждевременная оптимизация?

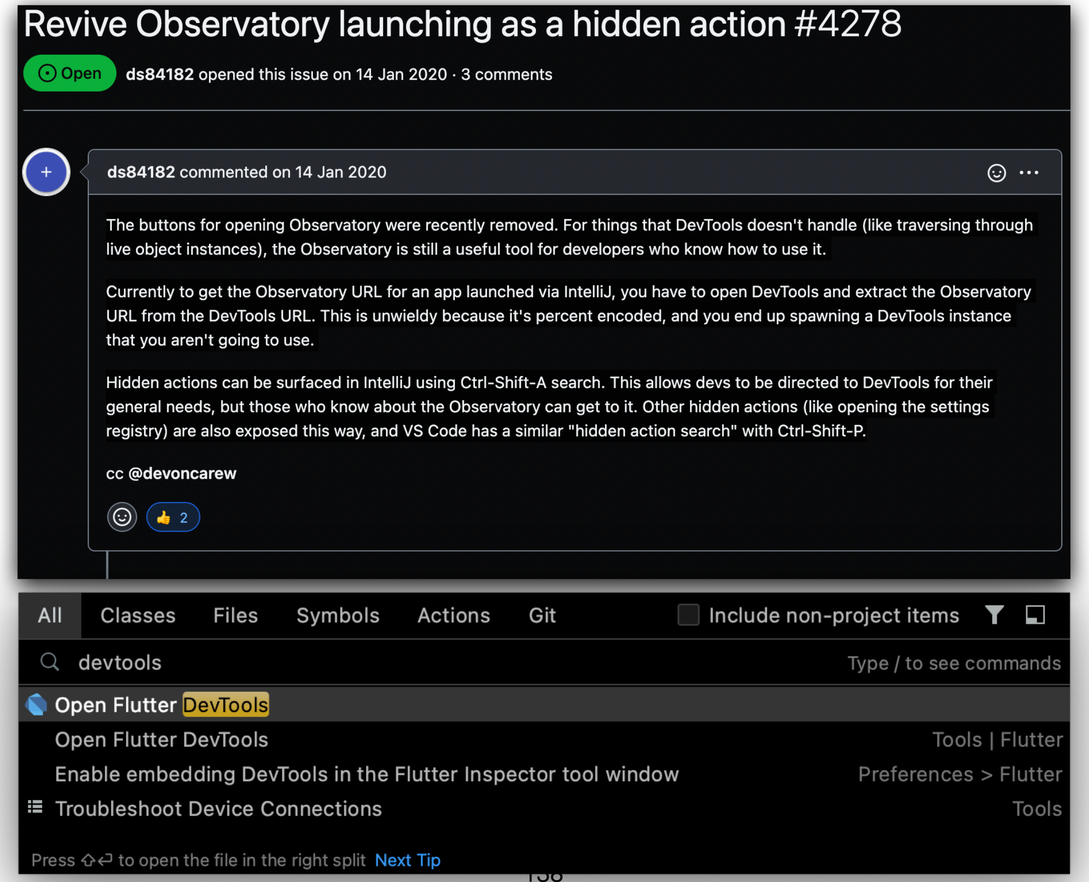
- Действительность такова, что на многие вопросы об оптимизации производительности есть такой ответ - «зависит от обстоятельств». Стоит ли эта конкретная оптимизация для этого конкретного виджета усилий и затрат на обслуживание? Имеет ли смысл такой подход в данной конкретной ситуации?

- Чтобы ответить на эти вопросы - необходимо тестирование и измерения. Определите количественно, какое влияние может оказать такая оптимизация на производительность, и примите решение на основе этих данных.

Если вам понравилась Observatory, поддержите Петицию
#ReviveObservatory

Revive Observatory launching as a hidden action

https://github.com/flutter/flutter-intellij/issues/4278



mobius

Спасибо)

Андрей Смирнов

https://twitter.com/dushesatwork

https://t.me/dushes_at_work