# Использование Hierarchy Viewer для просмотра интерфейс профилирования



Эти точки являются показатели:

Левая точка соответствует как долго измерительной работы для этого вид.

Средняя точка представляет сколько отрисовке макета операции для этого представления требуется.

Правая точка соответствует как долго операция для зрения.

Эти показатели могут быть красного, желтого или зеленого цвета и представлять, как каждый вид оказывает по отношению к другим видам дерева. Они не строгое представление о плохой или хороший вид, как таковой.

Красная точка означает, что этот вид отображает самый медленный, по сравнению с видом в иерархии.

Желтая точка означает, что это представление отображается в нижней 50% просмотров в иерархии.

Зеленая точка означает, что это представление отображается в Топ-50% просмотров в иерархии.

При нажатии на определенный вид дерева, Вы также увидите фактическое время исполнения, при которых эти индикаторы основаны. Эти сроки включают время перевода всех дочерних элементов управления. Например, время рендеринга так будет содержать время отрисовки двух элементов графическое представление ребенка.

Шаг 3: используя данные о производительности для повышения производительности пользовательского интерфейса в целом

После того, как Ваш пользовательский интерфейс исчерпал и устранения явных виновных к замедлению рендеринга пользовательского интерфейса, производительности, вы должны начать смотреть на статистику производительности более внимательно. Ты обтекаемый как можно больше базовых ресурсов, так что вы хотите, чтобы переключить свой фокус на времени рендеринга в отличие от индикаторов.

Оптимизация производительности пользовательского интерфейса означает оптимизацию всех его макеты и вид управления так, что они не так быстро, как они могут быть. Ваша задача, как настройка производительности связана с тем сообщается время отображения для каждого окна в иерархии как можно быстрее. Для этого лучше взять базовый. Я хотел, чтобы распечатать дереве иерархии, внося нотку время отрисовки каждого элемента управления, и работы по устранению слабых мест каждого органа управления и привлечения тех времен вниз как можно ниже.

Шаг 4: используя данные о производительности для повышения производительности пользовательского интерфейса в целом

Показатели результативности помочь Вам решить явных виновных к замедлению рендеринга пользовательского интерфейса, производительности, показывая Вам ваши ленивые взгляды, которые имеют длинный для визуализации. Но что вы можете сделать с этой информацией?

Ну, одна легкая вещь, чтобы сделать, это посмотреть на то, что работает, а что нет и соответственно охоту. В нашем примере, управление виджет textview, загружается очень быстро (вся зелень). Нет смысла в контроле, что смотреть слишком пристально прямо сейчас. Однако, так содержащих рыбу и миску контроля imageview является другая история. Красные точки взывают к небольшое расследование.

Шаг 5: Оптимизация Управления Графическое Представление Перевода

Так что мы можем сделать с графическое представление элементов управления, чтобы сделать их быстрее? Ну, вот пара идей:

Оптимизировать Формат изображения для Android. PNG-файлы, как правило, предпочтительным форматом без потерь, с последующим расширением jpg, выбранный в lossy-форматах, которые могут быть переделаны для сжатия. Использование GIF-это уныние, хотя обычно быстро много из того, что мы видели.

Оптимизация размеров изображения для макета. Если размеры вашего графика вверх или вниз, для этого нужно время.

Например, мне удалось улучшить время рисовать аквариума, управления книги, переводя их из файла PNG без потерь (времени рисовать ~40мс), чтобы с потерями, сжатый файл формата jpg (время рисовать ~26ms).Аналогичным образом, я могу сделать время обратить на рыбу-контроль просмотрщик фото хуже, изменения размеров графических и делает ее существенно больше, чем это требуется в приложении.

Когда смотрю на время отрисовки, учитывайте также, что ты рисуешь и что пользователь получает из него. Например, время преобразования 40 мс для одного изображения является неприемлемым для любой гладкой анимации – изображение может быть оказана 25 раз в секунду, достаточно не ровная и не оставляя время для любой другой обработки или визуализации.

Шаг 6: Оптимизация Управления Макетом Перевода

Когда ты смотришь, как оптимизировать ваши макеты (макет, так, уровень мельче, и так далее), у нас есть пару советов для вас:

Не гнездо макеты слишком глубоко.

Смотреть на макеты дублируются в иерархии. Например, если заменить в заменить, какой смысл?

Не используйте шаблоны вам не нужны. Один уровень мельче часто может заменить несколько вложенных linearlayout или TableLayout управления.

Проверьте инструмент layoutopt (см. инструменты SDK для Android: Оптимизация макета), который может помочь вам оптимизировать ваши конструкции макета.

Шаг 7: Оптимизация экраны вообще

В плане оптимизации пользовательского интерфейса, в общем, вот несколько советов:

Не перегружайте экраны с вид контроля. Сохранить его простым.

Адаптировать свои ресурсы на работу. За счет использования альтернативных ресурсов, чтобы поставлять самые лучшие ресурсы в данной ситуации (размеры экрана, ориентацию, разрешение и т. д.).

Заключение

Каждый экран индивидуален и имеет различные требования к производительности и отзывчивости и характеристики. На этом мы завершаем наше краткое обсуждение того, как использовать иерархию приложение для просмотра профиля и оптимизации пользовательских интерфейсов приложений. Иерархия Viewer-это мощный инструмент для визуализации и контроль компонентов, составляющих пользовательского интерфейса Android приложения. Он может быть использован, чтобы помочь упростить дизайн макета и определить, где ваши узкие места в производительности пользовательского интерфейса. С визуализацией информации можно получить из средство просмотра иерархии можно определить исходную точку и затем настроить компоненты пользовательского интерфейса, чтобы сделать их стройнее, злее и быстрее для загрузки, в результате чего повышенное удобство для пользователей.

Как всегда, мы с нетерпением ждем ваших отзывов.

ПЕРВОИСТОЧНИК

Leveraging the Hierarchy Viewer for UI Profiling

This post is part of a series called [Android Essentials](https://code.tutsplus.com/series/android-essentials--mobile-21888).

[Android Essentials: Enhancing Your Applications with App Widgets](https://code.tutsplus.com/tutorials/android-essentials-enhancing-your-applications-with-app-widgets--mobile-6245)

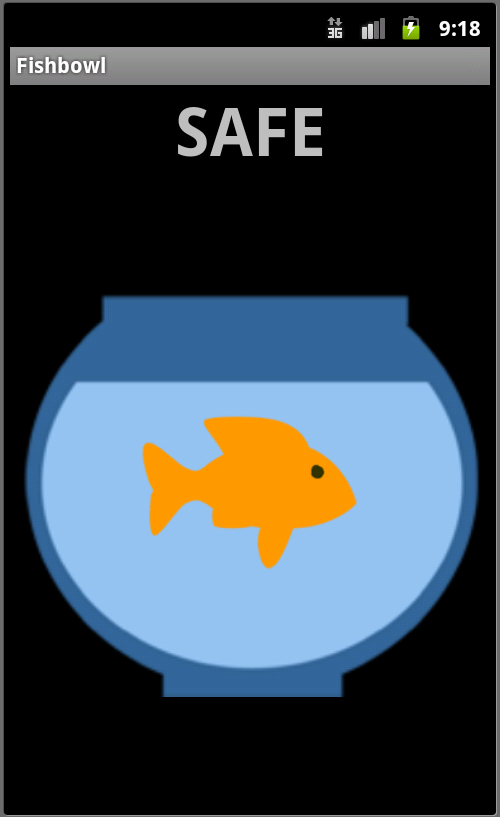
[Android Essentials: Working with Strings and String Arrays](https://code.tutsplus.com/tutorials/android-essentials-working-with-strings-and-string-arrays--mobile-7838)

The Android SDK comes with a number of useful tools to help you design, develop, and test Android applications. One tool that comes in handy for designing, debugging, and profiling your layouts and Android application user interfaces is called the Hierarchy Viewer. The Hierarchy Viewer is available as a standalone executable, and has recently been incorporated into the Eclipse ADT plug-in for Android development. In this tutorial, learn how to use Hierarchy Viewer to optimize your layout resources.

The Hierarchy Viewer is a visual tool that can be used to inspect your application user interfaces in ways that allow you to identify and improve your layout designs.

**Getting Started**

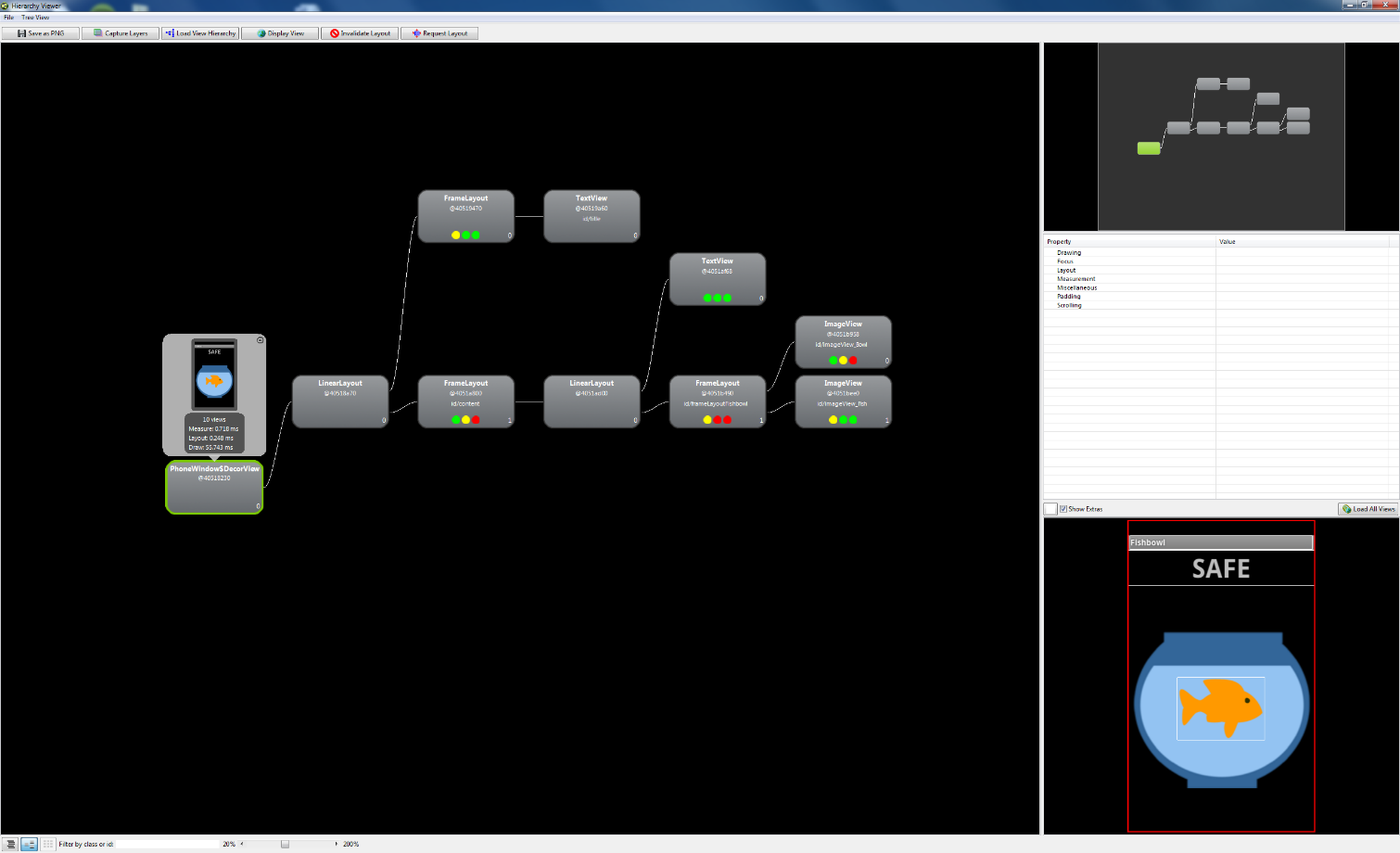
We’ve whipped up a very simple application with one screen. The screen uses a vertical LinearLayout control with two “cells”: A TextView control that says “Safe” and a FrameLayout which contains two ImageView controls. Two underlying drawable graphics are used: a bowl and a fish. Figure 1 shows our simple Fishbowl app screen:



We will use the Hierarchy Viewer tool to optimize our layout. We introduced this tool in the [Android Tools: Using the Hierarchy Viewer tutorial](http://j.mp/qbGFn9). If you are unfamiliar with the Hierarchy Viewer, you will want to read that tutorial before continuing.

**Step 1: Inspect the View Hierarchy of the Screen**

Let’s begin by looking at the general picture of what’s going on with the screen by launching the Hierarchy Viewer and loading the View Hierarchy of the screen. Figure 2 shows what the screen might look like:



**Step 2: Using the View Performance Indicators**

Notice how each View within your application has a set of three colored dots (Figure 3).



These dots are performance indicators:

* The left dot represents how long the measuring operation for this view takes.
* The middle dot represents how long the layout rendering operation for this view takes.
* The right dot represents how long the drawing operation for this view takes.

These indicators can be red, yellow, or green and represent how each view renders in relation to other views in the tree. They are not a strict representation of a bad or good view, per se.

* A red dot means that this view renders the slowest, compared to all views in your hierarchy.
* A yellow dot means that this view renders in the bottom 50% of all views in your hierarchy.
* A green dot means that this view renders in the top 50% of all views in your hierarchy.

When you click on a specific view within the Tree, you will also see the actual performance times upon which these indicators are based. These times include the time of rendering all child controls. For example, the rendering times for the FrameLayout will contain the rendering times of its two child ImageView controls.

**Step 3: Using Performance Data to Improve UI Performance as a Whole**

Once your user interface has settled and you’ve addressed the obvious culprits to slowing down your UI rendering performance, you have to start looking at the performance statistics more closely. You’ve streamlined the underlying resources as much as possible, so you’ll want to switch your focus on the rendering times as opposed to the indicators.

Optimizing user interface performance means optimizing all its layouts and view controls so that they are as fast as they can be. Your job as a performance tuner is to make those reported rendering times for each view in your hierarchy as fast as possible. To do this, it’s best to take a baseline. I like to print out the hierarchy tree, making a note of each control’s rendering times, and then work on addressing each control’s weak spots and bringing those times down as low as possible.

**Step 4: Using Performance Data to Improve UI Performance as a Whole**

The performance indicators help you address the obvious culprits to slowing down your UI rendering performance by showing you your lazy views that take the longest to render. But what can you do with this information?

Well, one easy thing to do is look for what’s working and what’s not and prioritize your hunt accordingly. In our example, the TextView control is loading very quickly (all greens). No point in scrutinizing that view too closely right away. However, the FrameLayout containing the fish and bowl ImageView controls is another story. Those red dots cry out for a little investigation.

**Step 5: Optimizing ImageView Control Rendering**

So what can we do with ImageView controls to make them render faster? Well, here are a couple of ideas:

* Optimize the image format for Android. PNG files are generally the preferred lossless format, followed by JPG, the preferred lossy format, which can be tweaked for compression. GIF usage is discouraged, although it’s generally plenty fast from what we’ve seen.
* Optimize the image dimensions for the layout. If you are sizing your graphics up or down a lot, this takes time.

For example, I was able to improve the draw time of the fishbowl ImageView control by switching it from a lossless PNG file (draw time ~40ms) to a lossy, compressed JPG format file (draw time ~26ms).Similarly, I can make the draw time on the fish ImageView control worse by resizing the graphic and making it substantially larger than what is needed in the application.

When looking at rendering times, consider also what you're drawing and what the user gets out of it. For example, a rendering time of 40ms for a single image is unacceptable for any sort of smooth animation – the image could only be rendered 25 times a second, not smooth enough and not leaving any time for any other processing or rendering.

**Step 6: Optimizing Layout Control Rendering**

When you’re looking at how to optimize your layouts (LinearLayout, FrameLayout, RelativeLayout, and so on), we’ve got a couple pieces of advice for you:

* Don’t nest your layouts too deeply.
* Watch for layouts duplicated in the hierarchy. For example, if you have a FrameLayout within a FrameLayout, what’s the point?
* Don’t use layouts you don’t need. One single RelativeLayout can often take the place of several nested LinearLayout or TableLayout controls.
* Check out the layoutopt tool (see [Android SDK Tools: Layout Optimization](http://j.mp/nX5HN9)), which can help you optimize your layout designs.

**Step 7: Optimizing Screens in General**

In terms of UI optimization in general, here are a few more tips:

* Don’t overload your screens with view controls. Keep it simple.
* Tailor your resources to the job. By using alternative resources to supply the best resources for a given situation (screen dimensions, orientation, resolution, etc).

**Conclusion**

Each screen is different and has different performance and responsiveness requirements and characteristics. This concludes our brief discussion of how to use the Hierarchy Viewer application to profile and optimize your application user interfaces. The Hierarchy Viewer is a powerful tool for visualizing and inspecting the components that make up your Android app user interfaces. It can be used to help streamline layout design and determine where your UI performance bottlenecks are. With the rendering information you can derive from the Hierarchy Viewer tool, you can establish a baseline and then tweak your user interface components to make them leaner, meaner, and faster to load, resulting in an improved experience for users.

As always, we look forward to your feedback.

Advertisement

**About the Authors**

Mobile developers Lauren Darcey and Shane Conder have coauthored several books on Android development: an in-depth programming book entitled [*Android Wireless Application Development, Second Edition*](http://www.amazon.com/dp/0321743016/?tag=mamlambo-20) and [*Sams Teach Yourself Android Application Development in 24 Hours, Second Edition*](http://www.amazon.com/dp/0672335697/?tag=mamlambo-20). When not writing, they spend their time developing mobile software at their company and providing consulting services. They can be reached at via email to [androidwirelessdev+mt@gmail.com](mailto:androidwirelessdev+mt@gmail.com), via their blog at[androidbook.blogspot.com](http://androidbook.blogspot.com/), and on Twitter [@androidwireless](http://twitter.com/androidwireless).

**Need More Help Writing Android Apps? Check out our Latest Books and Resources!**