Introduction to GammaRay

QtDay 2018, Firenze

Giuseppe D'Angelo

giuseppe.dangelo@kdab.com

The Qt, OpenGL and C++ experts

About me

- Senior Software Engineer & Trainer at KDAB
- Qt developer since ~2000
- Ask me about QtCore, QtGui, QtQuick, Qt3D
  - And also Modern C++, Modern OpenGL…

Agenda

- Introduction
- GammaRay for QtWidgets
- GammaRay for QtQuick
- GammaRay for Qt3D
- Q&A

KDAB

Why this talk?

KDAB

Introduction

The Qt, OpenGL and C++ experts

What is GammaRay?

- A high-level debugger and profiler for Qt applications
- Developed by KDAB

What can I inspect with GammaRay?

- All the QObjects in your application
  - check their properties, signals, slots, connections, signal emissions
- All the Qt Widgets in your application
  - analyze their painting, layouting, tab ordering
- All the Qt Quick scenes
  - debug sizing and positioning, focus issues, visualize the scene graph contents, have insights into the renderer, inspect QML issues
- All the Qt 3D scenes
  - inspect entities, visualize geometries, materials, framegraph structure
- And much, much more
  - Visualize state machines, inspect QStyle painting, debug model classes, running timers, and so on

Demo

- Traffic lights

Where do I get GammaRay?

- Freely available as standalone tool, or integrated in Qt Creator (part of the Qt Automotive Suite)
- https://github.com/KDAB/GammaRay

How do I use GammaRay?

- No special recompilation of the target application is necessary
- The GammaRay *probe* needs to be compiled with the very same Qt version of the application
- Either launch the application from the GammaRay *client*, or have GammaRay attach to your running application (or launch the probe manually)
- Works both locally and remotely

KDAB

GammaRay for Qt Widgets

The Qt, OpenGL and C++ experts

GammaRay for QtWidgets

- Visualize all the widget trees in the application
- Debug layouting and sizing issues
- Inspect and profile painting
- Visualize tab order chains

Demo

- Contact form

KDAB

GammaRay for Qt Quick

The Qt, OpenGL and C++ experts

GammaRay for Qt Quick scenes

- Visualize the Qt Quick scene's tree structure
- Live preview of the Qt Quick scene, with interaction
- Scene graph information

Example: why is my element not visible?

- Is it covered by another opaque element?
- Has it accidentally got width/height equal to 0?
- Has it got opacity set to 0? What about one of its ancestors?
- Is it outside the scene's visual boundary?

Demo

- Contact list

Profiling options for Qt Quick scenes

- How often is the scene being redrawn, and what are the elements causing a redraw? (check the *changes*)
- How many times is a pixel drawn upon? (check the *overdraw*)
- How many OpenGL draw calls are necessary to draw a scene? (check the *batching*)
- How much GPU memory am I spending for storing images? Are my images ideal for GPU consumption?
- (And more)

KDAB

Demo

- Batching
- Textures

Profiling for the QtQuick software renderer

- Analyze painting and inspect costs

Demo

- Samegame running under the software renderer

KDAB

GammaRay for Qt 3D

The Qt, OpenGL and C++ experts

GammaRay for Qt3D

- Inspect the scenegraph (entity tree) of a Qt3D scene
- Inspect the components attached to entities:
  - Geometries
  - Materials (shaders)
  - …
- Inspect the framegraph

Demo

- Dodge viper

KDAB

Conclusions

Conclusions

- If you're a Qt developer: get GammaRay today!
- Use it as a your debugger companion
- Regularly use it to profile your Qt application
- For more help, join the GammaRay mailing list, or contact me ☺

KDAB

Questions?

KDAB

Thanks!

giuseppe.dangelo@kdab.com

The Qt, OpenGL and C++ experts