Using The QML Profiler

Ulf Hermann

The Qt Company

October 8, 2014 / Qt Developer Days 2014

Outline

Outline

Classical optmization workflow

Minimize total time a program will take to run:

- Instrument binary to count and time function calls
- Or use an emulator that keeps track of function calls

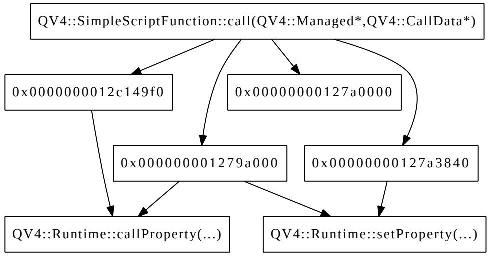
Create call statistics to see:

- which functions took most time
- which functions are called most often

Go back and optimize those.

Problematic with Qt Quick applications ...

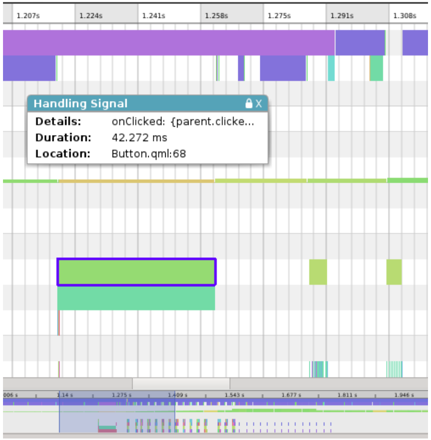
Profiling JIT-compiled code


QV4::SimpleScriptFunction::call(QV4::Managed*,QV4::CallData*)
0x0000000012c149f0
0x00000000127a0000
0x000000001279a000
0x00000000127a3840
QV4::Runtime::callProperty(...)
QV4::Runtime::setProperty(...)

Profiling QML code with Valgrind

- What functions does it call there?
- No useful results on JIT-compiled or interpreted code from general purpose profilers
- No symbolic information available
- No stack unwinding with non-emulating profilers

“Long” run time



Handling Signal

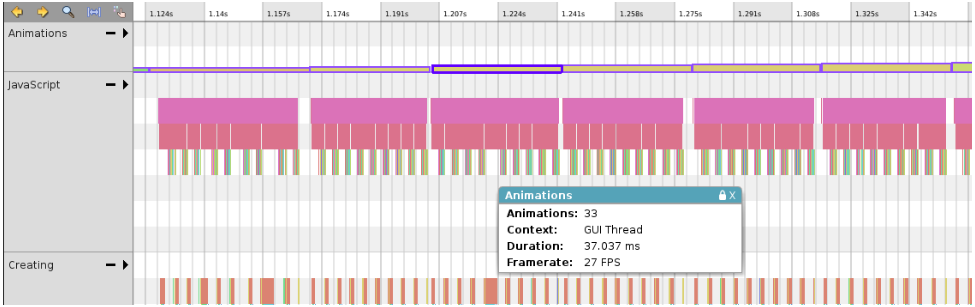
Details: onClicked: {parent.clicke...
Duration: 42.272 ms
Location: Button.qml:68

Single Signal Handler that runs for 40ms

- doesn’t make big dent in statistics
- leads to 2 dropped frames in a row
- might be harmless
- when does it run?

Relate single events on a timescale to pin down problems.

"Many" calls



Animations

Animations: 33
Context: GUI Thread
Duration: 37.037 ms
Framerate: 27 FPS

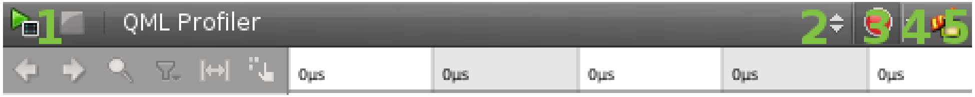
Badly timed object creation

- Time for each object creation isn't significant here.
- Number of calls may be more interesting, but ...
- their distribution over the frames is most important!

Outline

The QML Profiler

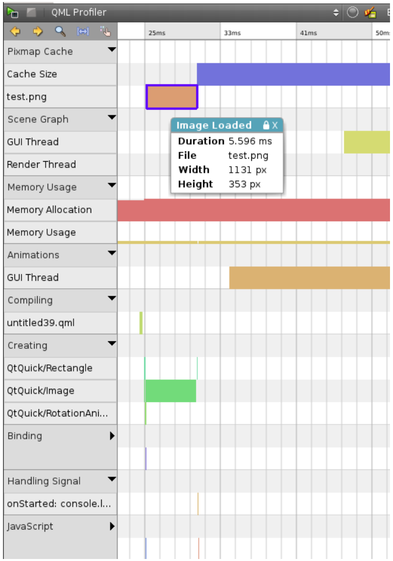
In Analyze mode of Qt Creator



1. start/stop profiling
2. control execution directly or profile external process
3. switch recording on and off while application is running to receive traces for specific time ranges.
4. select event types to be recorded (Qt Creator 3.3+)
5. clear current trace

Save and load trace files from context menu.

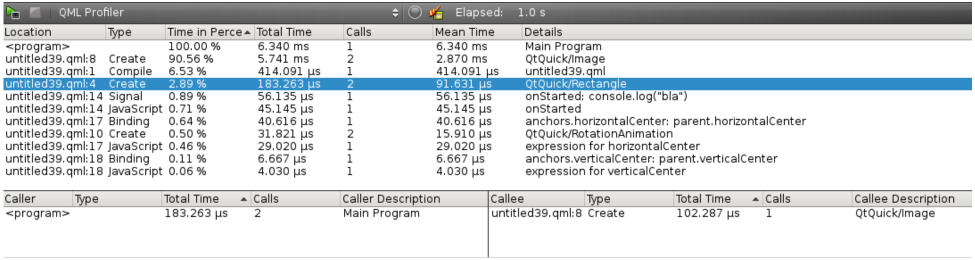
Timeline View



Duration	5.596 ms
File	test.png
Width	1131 px
Height	353 px

- Pixmap Cache: slow loading or large pictures
- Animations, Scene Graph: composition of scene graph
- Memory Usage: JavaScript heap and garbage collector
- Binding, Signal Handling, JavaScript: QML/JavaScript execution times

Events View



Location	Type	Time in Percent	Total Time	Calls	Mean Time	Details
<program>		100.00 %	6.340 ms	1	6.340 ms	Main Program
untitled39.qml:8	Create	90.56 %	5.741 ms	2	2.870 ms	QtQuick/Image
untitled39.qml:1	Compile	6.53 %	414.091 µs	1	414.091 µs	untitled39.qml
untitled39.qml:4	Create	2.89 %	183.263 µs	2	91.631 µs	QtQuick/Rectangle
untitled39.qml:14	Signal	0.89 %	56.135 µs	1	56.135 µs	onStarted: console.log("bla")
untitled39.qml:14	JavaScript	0.71 %	45.145 µs	1	45.145 µs	onStarted
untitled39.qml:17	Binding	0.64 %	40.616 µs	1	40.616 µs	anchors.horizontalCenter: parent.horizontalCenter
untitled39.qml:10	Create	0.50 %	31.821 µs	2	15.910 µs	QtQuick/RotationAnimation
untitled39.qml:17	JavaScript	0.46 %	29.020 µs	1	29.020 µs	expression for horizontalCenter
untitled39.qml:18	Binding	0.11 %	6.667 µs	1	6.667 µs	anchors.verticalCenter: parent.verticalCenter
untitled39.qml:18	JavaScript	0.06 %	4.030 µs	1	4.030 µs	expression for verticalCenter

Caller	Type	Total Time	Calls	Caller Description
<program>		183.263 µs	2	Main Program

Callee	Type	Total Time	Calls	Callee Description
untitled39.qml:8	Create	102.287 µs	1	QtQuick/Image

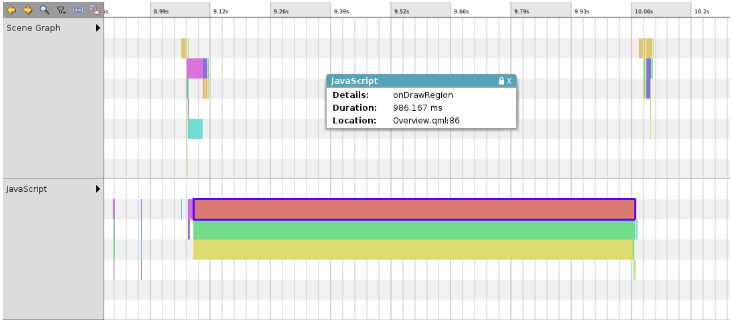
- Statistical profile of QML/JavaScript
- For problems that lend themselves to the classical workflow
- Optimize the overall most expensive bits to get a general speedup

Outline

It’s slow. What is wrong?

- Too much JavaScript executed in few frames?
  - All JavaScript must return before GUI thread advances
  - Frames delayed/dropped if GUI thread not ready
  - Result: Unresponsive, stuttering UI
- Creating/Painting/Updating invisible items?
  - Takes time in GUI thread
  - Same effect as “Too much JavaScript”
- Triggering long running C++ functions?
  - Paint methods, signal handlers, etc. triggered from QML
  - Also takes time in GUI thread
  - Harder to see in the QML profiler as C++ isn’t profiled

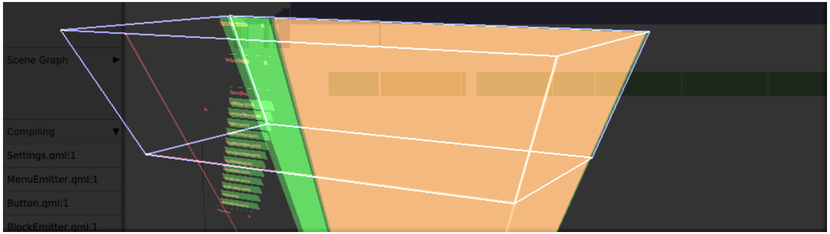
Too much Javascript



JavaScript
Details: onDrawRegion
Duration: 986.167 ms
Location: Overview.qml:86

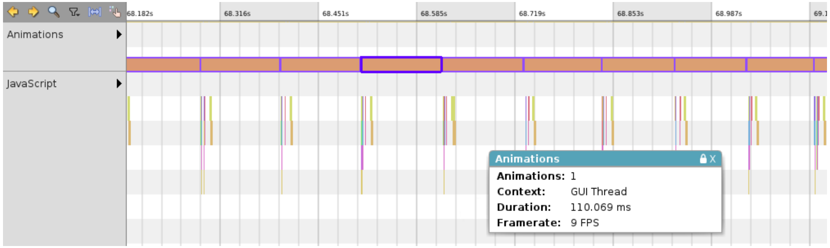
- Watch frame rate in Animations and Scene Graph
- Gaps and orange animation events are bad
- JavaScript category shows functions and run time
- Stay under $1000/60 \approx 16\text{ms}$ per frame

Invisible Items



- Check again for dropped frames
- Check for many short bindings or signal handlers => Too many items updated per frame
- QSG_VISUALIZE=overdraw shows scene layout
- Watch for items outside the screen or underneath visible elements

Long running C++ functions



Animations

Animations: 1
Context: GUI Thread
Duration: 110.069 ms
Framerate: 9 FPS

- Dropped frames, but no JavaScript running?
- Large unexplained gaps in the timeline?
- Check your custom QQuickItem implementations
- Use general purpose profiler to explore the details

Outline

Example 1: Too much JavaScript

Glitch in SameGame example when starting new game

Example 1: Too much JavaScript

Glitch in SameGame example when starting new game

- All items created from one JavaScript function call
- Takes about 100ms
- About 7 dropped frames in a row
- Enough unused CPU time during menu removal animation

Example 1: Too much JavaScript

Glitch in SameGame example when starting new game

- All items created from one JavaScript function call
- Takes about 100ms
- About 7 dropped frames in a row
- Enough unused CPU time during menu removal animation

Solution:

- Create invisible items during menu animation
- Later only set them visible
- Setting visibility is cheaper than creating items

Conventions for profiling Qt Creator

- gray color scheme: profiling one of the others
- red color scheme: buggy pre-3.0 as “bad” example
- green color scheme: v3.3 preview
- blue color scheme: patched v3.3 preview
- Trace files are just loaded into “colored” Qt Creators to trigger activity. Don’t interpret the data.

Example 2: Even more JavaScript

QML Profiler stutters on horizontal resizing.

Example 2: Even more JavaScript

QML Profiler stutters on horizontal resizing.

- Overview always iterates all events to paint itself
- is implemented in JavaScript
- but: only updated on loading and resizing

Example 2: Even more JavaScript

QML Profiler stutters on horizontal resizing.

- Overview always iterates all events to paint itself
- is implemented in JavaScript
- but: only updated on loading and resizing

Solution[1]:

- Stretch the code over multiple frames
- Use Timer to trigger deferred JavaScript execution
- onTriggered should not take longer than a frame (around 16ms)
- Downside: Overview painting is “animated” now

[1]with potential for further optimization

Example 3: Painting outside viewport

Slow scrolling if timeline categories expanded

Example 3: Painting outside viewport

Slow scrolling if timeline categories expanded

- Coordinate system marks cover a large space in vertical direction
- can take a long time to paint (up to 10ms)
- are mostly invisible most of the time.

Example 3: Painting outside viewport

Slow scrolling if timeline categories expanded

- Coordinate system marks cover a large space in vertical direction
- can take a long time to paint (up to 10ms)
- are mostly invisible most of the time.

Solution[2]:

- Only paint visible part of coordinate system
- Directly set virtual contentHeight on Flickable
- Painted area “sliding” in virtual contentHeight
- Reduces painting time to about 1 - 2ms

[2]with potential for further optimization

Example 4: Expensive C++

Timeline scrolling still slow for some traces

Example 4: Expensive C++

Timeline scrolling still slow for some traces

- Timeline data painted for all categories, no matter how many are visible
- Takes a lot of time, especially in “dense” places.
- Hard to see in QML Profiler, as painting is implemented in C++.
- QSG_VISUALIZE=overdraw can help.

Example 4: Expensive C++

Timeline scrolling still slow for some traces

- Timeline data painted for all categories, no matter how many are visible
- Takes a lot of time, especially in “dense” places.
- Hard to see in QML Profiler, as painting is implemented in C++.
- QSG_VISUALIZE=overdraw can help.

Solution[3]:

- Again, only paint visible part of timeline.
- Same technique as with coordinate system.

[3]with potential for further optimization

Example 5: What about the labels?

Hiccup when expanding large categories

Example 5: What about the labels?

Hiccup when expanding large categories

- Repeater creates all elements at the same time.
- Use ListView to create and delete on demand?
- Potentially save some memory?

Example 5: What about the labels?

Hiccup when expanding large categories

- Repeater creates all elements at the same time.
- Use ListView to create and delete on demand?
- Potentially save some memory?

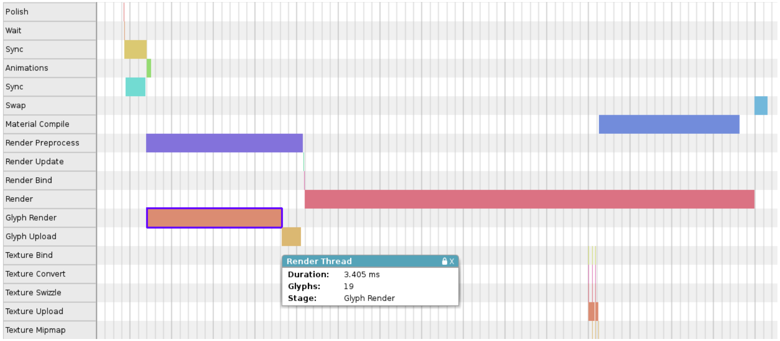
But:

- Labels are rarely updated.
- On-demand creation and removal during scrolling, when a lot of other code has to run?
- Creation and removal triggers garbage collector.

Solution: Probably not worth it in this case

Outline

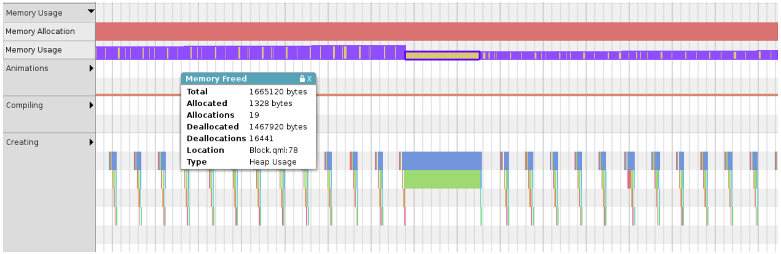
Better Scene Graph Profiling



Duration: 3.405 ms
Glyphs: 19
Stage: Glyph Render

- Will be included in Professional and Enterprise packages of Qt Creator 3.3
- Visualizes all the timing information available from the scene graph

JavaScript Heap profiler



Memory Freed

Total	1665120 bytes
Allocated	1328 bytes
Allocations	19
Deallocated	1467920 bytes
Deallocations	16441
Location	Block.qml:78
Type	Heap Usage

- UI in Qt Creator 3.2+ (Professional and Enterprise)
- Will be usable with Qt 5.4+
- Tracks page allocations of the memory manager
- Tracks memory allocations on JavaScript heap
- Shows when the garbage collector runs

Selective recording

- Switch off recording of events you’re not interested in
- Reduces amount of data created
- Record longer traces without running into memory bottlenecks
- Smaller trace files, faster loading

Various UI improvements

- Drag&Drop reordering of categories
- Completely hide categories to reduce height of timeline
- Resize rows in timeline