Domain Specific Debugging Tools

Volker Krause
volker.krause@kdab.com

KDAB

Problem

What's the Problem?

So, where's the bug in your QML?

```
Invalid read of size 1
 at 0x4C2D9B0: bcmp (in /usr/lib64/valgrind/vgpreload_memcheck-amd64-linux.so)
 by 0x6B33AA6: QOpenGLFunctions::glLinkProgram(unsigned int) (qopenglfunctions.h:1098)
 by 0x6B2FD15: QOpenGLShaderProgram::link() (qopenglshaderprogram.cpp:826)
 by 0x4F83AE1: QSGDefaultDistanceFieldGlyphCache::createBlitProgram() (qsgdefaultdistancefieldglyphcache_p.h:118)
 by 0x4F82D50: QSGDefaultDistanceFieldGlyphCache::resizeTexture(QSGDefaultDistanceFieldGlyphCache::TextureInfo*, int, int)
 by 0x4F8262A: QSGDefaultDistanceFieldGlyphCache::storeGlyphs(QHash<unsigned int, QImage> const&)
 by 0x4F77A81: QSGDistanceFieldGlyphCache::update() (qsgadaptationlayer.cpp:169)
 by 0x4F86449: QSGDistanceFieldGlyphNode::preprocess() (qsgdistancefieldglyphnode.cpp:167)
 by 0x4F690E8: QSGRenderer::preprocess() (qsgrenderer.cpp:378)
 by 0x4F68A06: QSGRenderer::renderScene(QSGBindable const&) (qsgrenderer.cpp:248)
 by 0x4F68975: QSGRenderer::renderScene() (qsgrenderer.cpp:229)
 by 0x4F7B48E: QSGContext::renderNextFrame(QSGRenderer*, unsigned int) (qsgcontext.cpp:270)
 by 0x4FBE833: QQuickWindowPrivate::renderSceneGraph(QSize const&) (qquickwindow.cpp:346)
 by 0x50D0217: QQuickTrivialWindowManager::renderWindow(QQuickWindow*) (qquickwindowmanager.cpp:263)
 by 0x50D076F: QQuickTrivialWindowManager::event(QEvent*) (qquickwindowmanager.cpp:351)
 by 0x5B37DB7: QApplicationPrivate::notify_helper(QObject*, QEvent*) (qapplication.cpp:3619)
 by 0x5B354A9: QApplication::notify(QObject*, QEvent*) (qapplication.cpp:3050)
 by 0x79B9479: QCoreApplication::notifyInternal(QObject*, QEvent*) (qcoreapplication.cpp:748)
 by 0x79BCC86: QCoreApplication::sendEvent(QObject*, QEvent*) (in /home/vkrause/dev/qt5/inst/lib/libQtCore.so.5.0.0)
 by 0x79BA53D: QCoreApplicationPrivate::sendPostedEvents(QObject*, int, QThreadData*) (qcoreapplication.cpp:1349)
 by 0x79BA0FE: QCoreApplication::sendPostedEvents(QObject*, int) (qcoreapplication.cpp:1209)
Address 0x7fa847b93a81 is not stack'd, malloc'd or (recently) free'd
```

GDB is not enough

- Increasing abstraction
- Asynchronous API
- Distributed architecture
- Runtime interpreted code
- JIT compilers

General Purpose Tools

- Instruction-level debuggers
- printf
- Profilers measuring:
  - CPU ticks
  - malloc calls

Increasing abstraction

- Examples:
  - Qt Model/View
  - QStateMachine
- Instruction-level view is too far below semantics
- Debug output triggered too often

Asynchronous API

- Examples:
  - QNetworkAccessManager/QNetworkReply
  - Job-based APIs
- Hard to follow control flow

Distributed Architecture

- Examples:
  - D-Bus

- Even harder to follow control flow
- Profilers don't analyze complexity in IPC protocol

Runtime Interpreted Code

- Examples:
  - QtQuick
  - QtWebKit
- Debuggers and profilers analyze interpreter code
- Hard to correlate issues in interpreter to issues in interpreted code

JIT-Compiled Code

- Examples:
  - QtScript
  - QtQuick
- Debuggers and profilers see generated code
- Even harder to correlate issues in generated code to issues in original QML/JavaScript.

General Purpose Tools

- Inefficient/cumbersome
- Require knowledge of framework internals
  - Up to the point of a JIT compiler!
- Can lead to erroneous conclusions
  - “JavaScript is slow!”

Solution

What can we do about it?

Can we have better tools?

- Move knowledge about framework internals to the tool
- Visualize results at the same semantic level

- Downside: Tools become specific to one framework

Where do I get these tools?

- Some tooling exist for Qt
  - cf. Romain Pokrzywka, Volker Krause, “Effective Debugging and Profiling for Qt and Qt Quick”, Qt Dev Days 2011
- Often no tooling exist for your own frameworks

Should I build my own?

- Struggling with complex control flow
- Repeatedly adding the same debug code or printf statements
- Complex internal structures that benefit from dedicated visualization
- Performance metrics lacking correlation to the actual cost cause

Approaches

- Built-in diagnostics
- External observers
- Emulators
- IDEs
- API tracing
- Binary instrumentation
- Qt Introspection

printf isn't dead yet

- qDebug() operator<< overloads

```
QDebug operator <<( QDebug d,
                    const MyType &myObj )
{
        return d << myObj.foo()
                 << myObj.bar();
}
```

- Declare outside of namespaces
- Needs to be exported or inline if provided by a library

Impact

- Minimal increase in code size
- No runtime impact when not used
- Can be disabled completely at compile time
    - QT_NO_DEBUG
    - QT_NO_DEBUG_OUTPUT
    - QT_NO_WARNING_OUTPUT

Built-In Diagnostics

- Enable at compile time or runtime
  - preprocessor define
  - environment variable
  - config file/QSettings
  - triggered via IPC
- Typically perform extra checks or provide verbose diagnostic output

Built-in Diagnostics Examples

- Preprocessor defines
  - QIODEVICE_DEBUG, QSSLSOCKET_DEBUG, ...
  - grep for _DEBUG
- Environment Variables
  - QT_FLUSH_PAINT
  - QDBUS_DEBUG
  - grep for getenv

Built-In Diagnostics DIY

- Compile-time conditional debug output

```
#ifndef FOO_DEBUG
#   define myDebug qDebug
#else
#   define myDebug if (false) qDebug
#endif

…

myDebug(“printf style\n”);
myDebug() << “stream style”;
```

Built-In Diagnostics DIY

- Runtime conditional output

```
static const int debugLevel =
       qgetenv(“MY_DEBUG”).toInt();
...
if (debugLevel > 3)
    dumpInternalState();
```

- Requires application restart to activate

Built-In Diagnostics DIY

- D-Bus triggered diagnostics

```
class MyClass : public QObject {
 Q_OBJECT
 Q_CLASSINFO("D-Bus Interface", "com.kdab.debug")
public:
 MyClass()
 {
  QDBusConnection::sessionBus().
    registerService(“com.kdab.MyApp”);
  QDBusConnection::sessionBus().
    registerObject("/Debug", this,
    QDBusConnection::ExportScriptableSlots);
  …
 }
public slots:
 Q_SCRIPTABLE void dumpInternalState() const
 { … }
}
```

Built-In Diagnostics Impact

- Compile-time diagnostics
  - Can be disabled completely
  - Ideal for very expensive features
- Runtime diagnostics
  - Minimal runtime overhead
  - Diagnostics always available

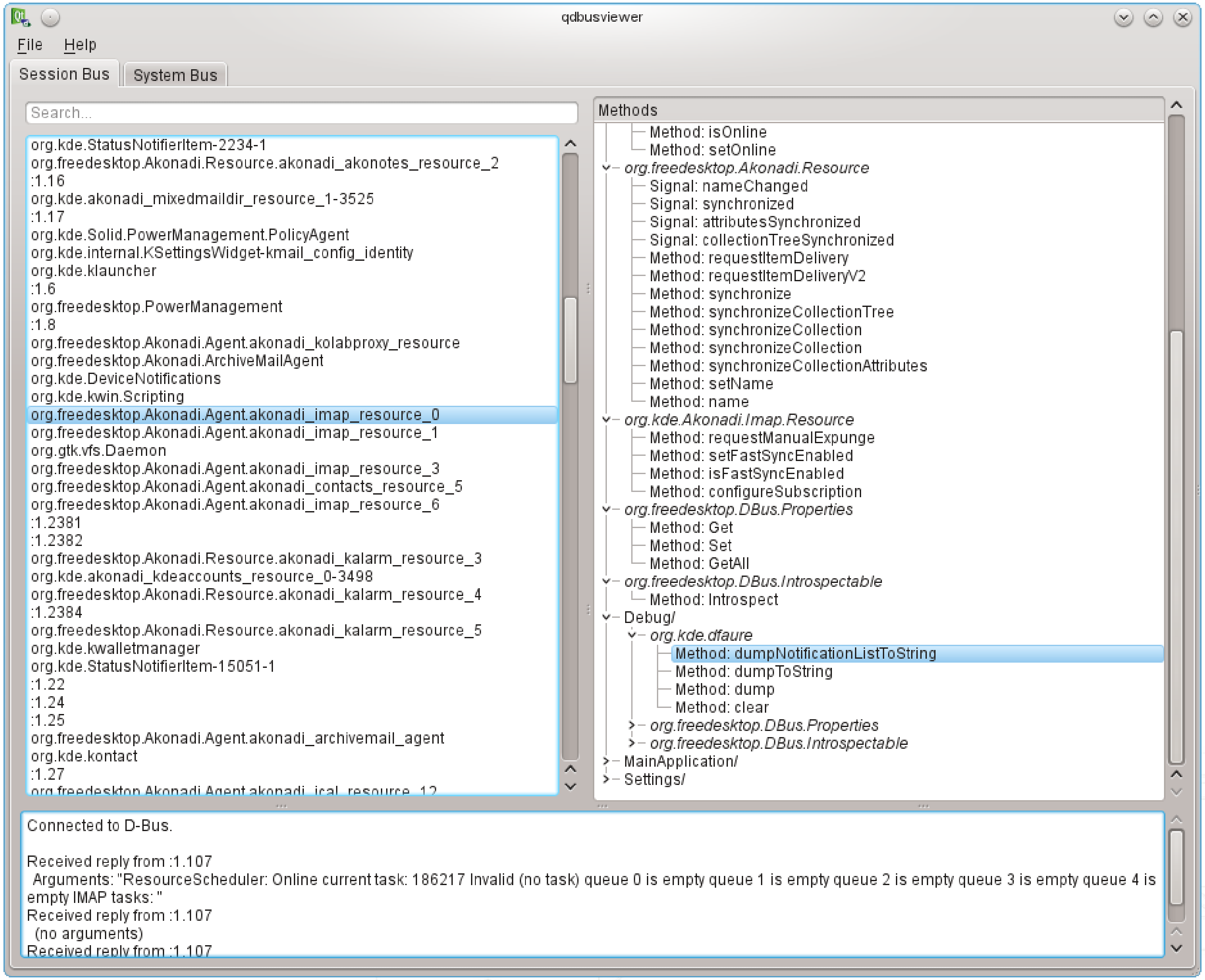
Compiled-In Diagnostics

- Not built into framework, but provided separately
- Has no access to framework internals
- Example: ModelTest
- Useful for non-trivial diagnostics performed using official API

External Observers

- Tools using public interfaces to observe what your application is doing
- Requires communication or other externally visible effects
- Example: qdbusviewer
- Also useful (but not Qt-specific):
  - Network sniffer
  - Database logging/viewers

qdbusviewer



```
org.kde.StatusNotifierItem-2234-1
org.freedesktop.Akonadi.Resource.akonadi_akonotes_resource_2
:1.16
org.kde.akonadi_mixedmaildir_resource_1-3525
:1.17
org.kde.Solid.PowerManagement.PolicyAgent
org.kde.internal.KSettingsWidget-kmail_config_identity
org.kde.klauncher
:1.6
org.freedesktop.PowerManagement
:1.8
org.freedesktop.Akonadi.Agent.akonadi_kolabproxy_resource
org.freedesktop.Akonadi.ArchiveMailAgent
org.kde.DeviceNotifications
org.kde.kwin.Scripting
org.freedesktop.Akonadi.Agent.akonadi_imap_resource_0
org.freedesktop.Akonadi.Agent.akonadi_imap_resource_1
org.gtk.vfs.Daemon
org.freedesktop.Akonadi.Agent.akonadi_imap_resource_3
org.freedesktop.Akonadi.Agent.akonadi_contacts_resource_5
org.freedesktop.Akonadi.Agent.akonadi_imap_resource_6
:1.2381
:1.2382
org.freedesktop.Akonadi.Resource.akonadi_kalarm_resource_3
org.kde.akonadi_kdeaccounts_resource_0-3498
org.freedesktop.Akonadi.Resource.akonadi_kalarm_resource_4
:1.2384
org.freedesktop.Akonadi.Resource.akonadi_kalarm_resource_5
org.kde.kwalletmanager
org.kde.StatusNotifierItem-15051-1
:1.22
:1.24
:1.25
org.freedesktop.Akonadi.Agent.akonadi_archivemail_agent
org.kde.kontact
:1.27
org.freedesktop.Akonadi.Agent.akonadi_ical_resource_12
```

```
    Method: isOnline
    Method: setOnline
v org.freedesktop.Akonadi.Resource
    Signal: nameChanged
    Signal: synchronized
    Signal: attributesSynchronized
    Signal: collectionTreeSynchronized
    Method: requestItemDelivery
    Method: requestItemDeliveryV2
    Method: synchronize
    Method: synchronizeCollectionTree
    Method: synchronizeCollection
    Method: synchronizeCollection
    Method: synchronizeCollectionAttributes
    Method: setName
    Method: name
v org.kde.Akonadi.Imap.Resource
    Method: requestManualExpunge
    Method: setFastSyncEnabled
    Method: isFastSyncEnabled
    Method: configureSubscription
v org.freedesktop.DBus.Properties
    Method: Get
    Method: Set
    Method: GetAll
v org.freedesktop.DBus.Introspectable
    Method: Introspect
v Debug/
  v org.kde.dfaure
      Method: dumpNotificationListToString
      Method: dumpToString
      Method: dump
      Method: clear
  > org.freedesktop.DBus.Properties
  > org.freedesktop.DBus.Introspectable
> MainApplication/
> Settings/
```

```
Connected to D-Bus.

Received reply from :1.107
 Arguments: "ResourceScheduler: Online current task: 186217 Invalid (no task) queue 0 is empty queue 1 is empty queue 2 is empty queue 3 is empty queue 4 is
empty IMAP tasks: "
Received reply from :1.107
  (no arguments)
Received reply from :1.107
```

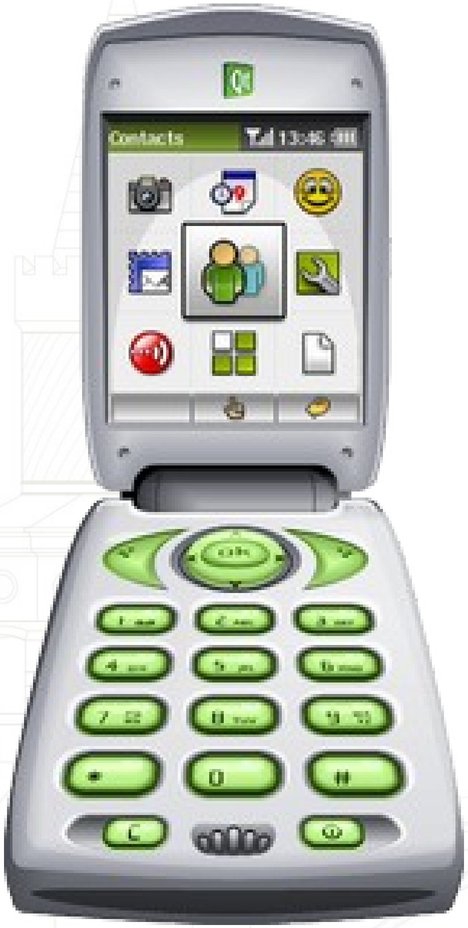
External Observers

- No changes required in your application
- Don't require application restart but can be used on-demand
- Requires interceptable communication channels
  - Problematic with e.g. TLS/SSL
- Example for DIY project: QDataStream viewer

Emulators

- Simulate the real environment your application runs in
- Makes you independent of hardware or physical constraints
- Example: qvfb

qvfb



Emulators

- Allows replay of recorded input
- Allows easy testing of corner cases and “that should never happen” conditions
- Very useful for CI systems

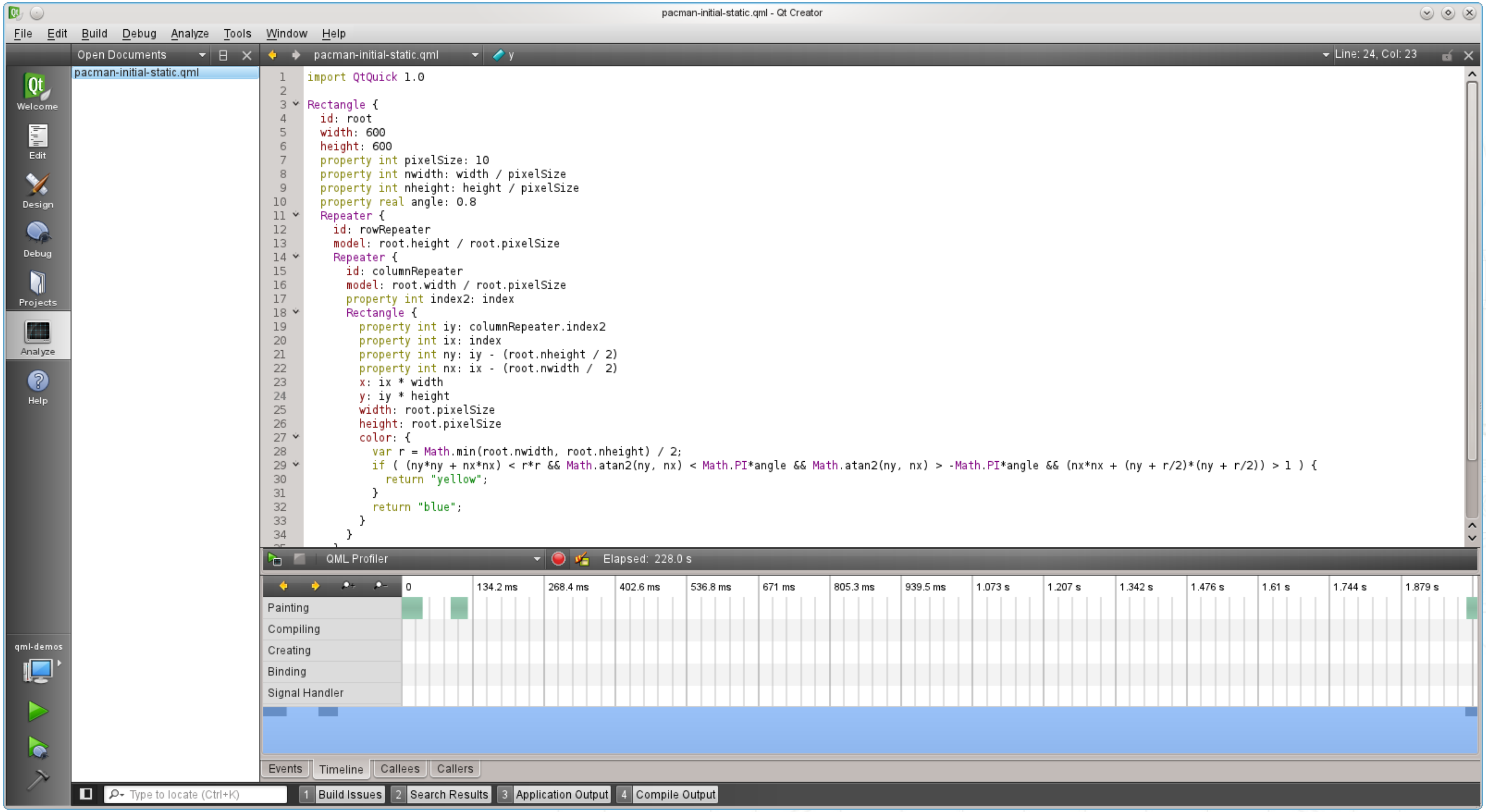
Emulators DIY

- Find the right interface
  - API-compatible drop-in replacement DLL
  - Using existing backend abstractions (e.g. QtSensors)
  - IPC or network protocols
- Feed data
  - manually, with custom UI
  - manually, from code
  - from previously recorded file

Full IDE

- Fully integrated suite for the entire development workflow, including debugging and profiling
- Example: QtCreator for QML
- Usually overkill, but worth considering when providing a complex domain specific language
  - Existing IDEs (QtCreator, KDevelop, …) can be extended by plug-ins

QtCreator



pacman-initial-static.qml - Qt Creator

```
import QtQuick 1.0

Rectangle {
  id: root
  width: 600
  height: 600
  property int pixelSize: 10
  property int nwidth: width / pixelSize
  property int nheight: height / pixelSize
  property real angle: 0.8
  Repeater {
    id: rowRepeater
    model: root.height / root.pixelSize
    Repeater {
      id: columnRepeater
      model: root.width / root.pixelSize
      property int index2: index
      Rectangle {
        property int iy: columnRepeater.index2
        property int ix: index
        property int ny: iy - (root.nheight / 2)
        property int nx: ix - (root.nwidth /  2)
        x: ix * width
        y: iy * height
        width: root.pixelSize
        height: root.pixelSize
        color: {
          var r = Math.min(root.nwidth, root.nheight) / 2;
          if ( (ny*ny + nx*nx) < r*r && Math.atan2(ny, nx) < Math.PI*angle && Math.atan2(ny, nx) > -Math.PI*angle && (nx*nx + (ny + r/2)*(ny + r/2)) > 1 ) {
            return "yellow";
          }
          return "blue";
        }
      }
```

Line: 24, Col: 23

Elapsed: 228.0 s

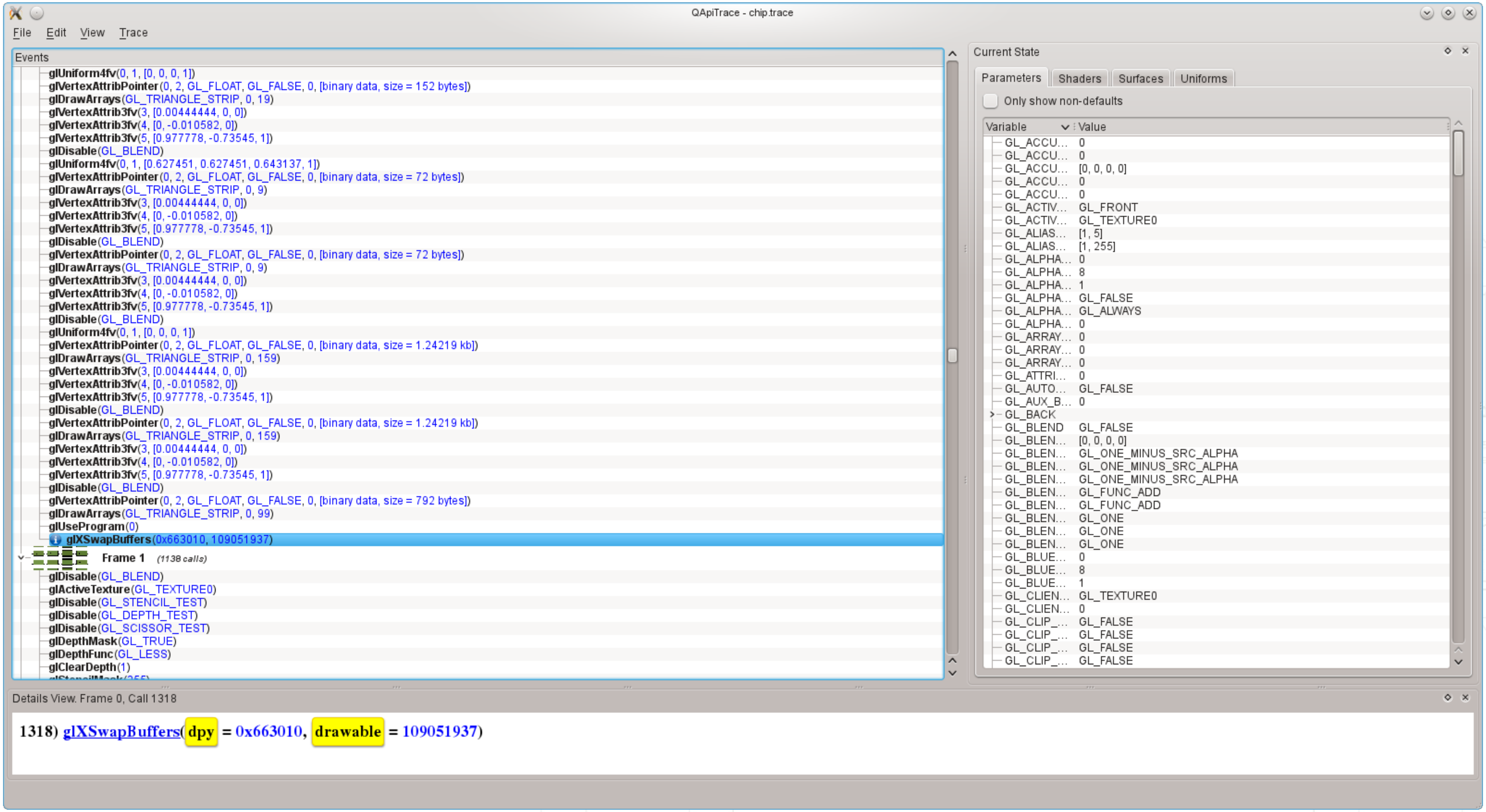
API Tracing

- Trace all calls (and arguments) to a specific API
- Visualization for the massive amount of data gathered
- Approach:
  - Intercept API call
  - Record call and its arguments
  - Call the original method

API Tracing Examples

- strace
  - Traces all system calls
- apitrace
  - Traces OpenGL/Direct3d calls
  - http://github.com/apitrace/apitrace
  - Qt visualization UI for OpenGL state at an arbitrary point in time

API Tracing Examples



```
glUniform4fv(0, 1, [0, 0, 0, 1])
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 0, [binary data, size = 152 bytes])
glDrawArrays(GL_TRIANGLE_STRIP, 0, 19)
glVertexAttrib3fv(3, [0.00444444, 0, 0])
glVertexAttrib3fv(4, [0, -0.010582, 0])
glVertexAttrib3fv(5, [0.977778, -0.73545, 1])
glDisable(GL_BLEND)
glUniform4fv(0, 1, [0.627451, 0.627451, 0.643137, 1])
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 0, [binary data, size = 72 bytes])
glDrawArrays(GL_TRIANGLE_STRIP, 0, 9)
glVertexAttrib3fv(3, [0.00444444, 0, 0])
glVertexAttrib3fv(4, [0, -0.010582, 0])
glVertexAttrib3fv(5, [0.977778, -0.73545, 1])
glDisable(GL_BLEND)
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 0, [binary data, size = 72 bytes])
glDrawArrays(GL_TRIANGLE_STRIP, 0, 9)
glVertexAttrib3fv(3, [0.00444444, 0, 0])
glVertexAttrib3fv(4, [0, -0.010582, 0])
glVertexAttrib3fv(5, [0.977778, -0.73545, 1])
glDisable(GL_BLEND)
glUniform4fv(0, 1, [0, 0, 0, 1])
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 0, [binary data, size = 1.24219 kb])
glDrawArrays(GL_TRIANGLE_STRIP, 0, 159)
glVertexAttrib3fv(3, [0.00444444, 0, 0])
glVertexAttrib3fv(4, [0, -0.010582, 0])
glVertexAttrib3fv(5, [0.977778, -0.73545, 1])
glDisable(GL_BLEND)
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 0, [binary data, size = 1.24219 kb])
glDrawArrays(GL_TRIANGLE_STRIP, 0, 159)
glVertexAttrib3fv(3, [0.00444444, 0, 0])
glVertexAttrib3fv(4, [0, -0.010582, 0])
glVertexAttrib3fv(5, [0.977778, -0.73545, 1])
glDisable(GL_BLEND)
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 0, [binary data, size = 792 bytes])
glDrawArrays(GL_TRIANGLE_STRIP, 0, 99)
glUseProgram(0)
glXSwapBuffers(0x663010, 109051937)
Frame 1   (1138 calls)
glDisable(GL_BLEND)
glActiveTexture(GL_TEXTURE0)
glDisable(GL_STENCIL_TEST)
glDisable(GL_DEPTH_TEST)
glDisable(GL_SCISSOR_TEST)
glDepthMask(GL_TRUE)
glDepthFunc(GL_LESS)
glClearDepth(1)
glStencilMask(255)
```

□ Only show non-defaults

Variable	Value
GL_ACCU…	0
GL_ACCU…	0
GL_ACCU…	[0, 0, 0, 0]
GL_ACCU…	0
GL_ACCU…	0
GL_ACTIV…	GL_FRONT
GL_ACTIV…	GL_TEXTURE0
GL_ALIAS…	[1, 5]
GL_ALIAS…	[1, 255]
GL_ALPHA…	0
GL_ALPHA…	8
GL_ALPHA…	1
GL_ALPHA…	GL_FALSE
GL_ALPHA…	GL_ALWAYS
GL_ALPHA…	0
GL_ARRAY…	0
GL_ARRAY…	0
GL_ARRAY…	0
GL_ATTRI…	0
GL_AUTO…	GL_FALSE
GL_AUX_B…	0
GL_BACK	
GL_BLEND	GL_FALSE
GL_BLEN…	[0, 0, 0, 0]
GL_BLEN…	GL_ONE_MINUS_SRC_ALPHA
GL_BLEN…	GL_ONE_MINUS_SRC_ALPHA
GL_BLEN…	GL_ONE_MINUS_SRC_ALPHA
GL_BLEN…	GL_FUNC_ADD
GL_BLEN…	GL_FUNC_ADD
GL_BLEN…	GL_ONE
GL_BLEN…	GL_ONE
GL_BLEN…	GL_ONE
GL_BLUE…	0
GL_BLUE…	8
GL_BLUE…	1
GL_CLIEN…	GL_TEXTURE0
GL_CLIEN…	0
GL_CLIP_…	GL_FALSE
GL_CLIP_…	GL_FALSE
GL_CLIP_…	GL_FALSE
GL_CLIP_…	GL_FALSE

Details View. Frame 0, Call 1318

1318) glXSwapBuffers(dpy = 0x663010, drawable = 109051937)

API Tracing DIY

- OS-level system-wide tracing tools:
  - DTrace
  - SystemTap, perf, uprobes
- POSIX ptrace
- Library pre-loading and forwarding
  - LD_PRELOAD, dlsym(RTLD_NEXT, …)
  - even more ugly on Windows

API Tracing Impact

- Overhead usually comparable to one extra function call
- Be prepared to handle large amounts of data
- Requires no modifications on traced code
- Also works if no source code is available

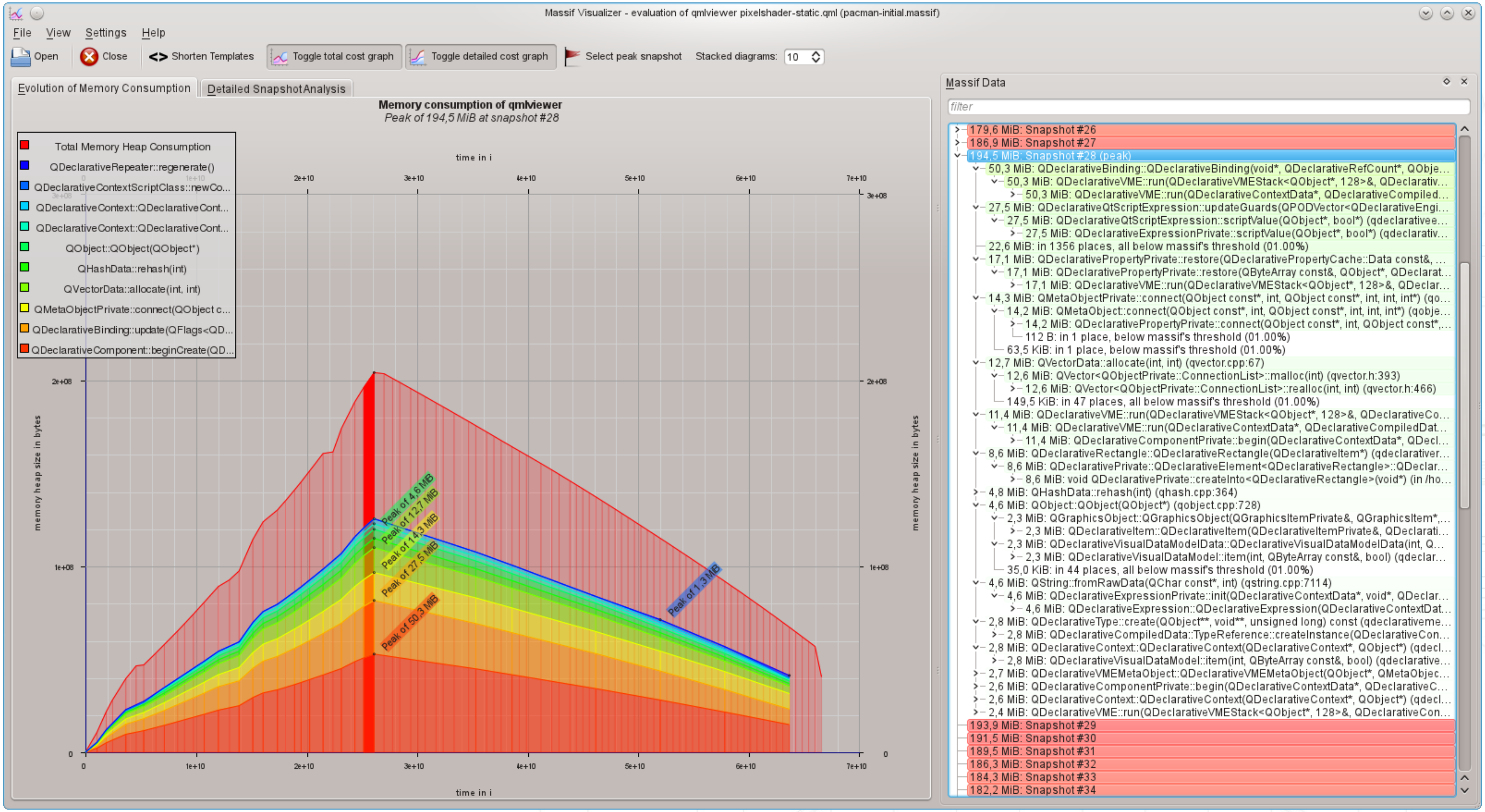
Binary Instrumentation

- Interpret or JIT rewrite binary code
- Example: Valgrind suite
- Requires in-depth knowledge of binary code execution
- Allows analysis of very low-level details, e.g. for memory profiling

Binary Instrumentation DIY

- Existing frameworks for binary instrumentation
  - Valgrind (http://www.valgrind.org/)
  - Pin (http://www.pintool.org/)
- Example use-case: runtime attachable Massif

Massif


Massif Visualizer - evaluation of qmlviewer pixelshader-static.qml (pacman-initial.massif)
File
View
Settings
Help
Open
Close
Shorten Templates
Toggle total cost graph
Toggle detailed cost graph
Select peak snapshot
Stacked diagrams:
10
Evolution of Memory Consumption
Detailed Snapshot Analysis
Memory consumption of qmlviewer
Peak of 194,5 MiB at snapshot #28
Total Memory Heap Consumption
QDeclarativeRepeater::regenerate()
QDeclarativeContextScriptClass::newCo...
QDeclarativeContext::QDeclarativeCont...
QDeclarativeContext::QDeclarativeCont...
QObject::QObject(QObject*)
QHashData::rehash(int)
QVectorData::allocate(int, int)
QMetaObjectPrivate::connect(QObject c...
QDeclarativeBinding::update(QFlags<QD...
QDeclarativeComponent::beginCreate(QD...
time in i
memory heap size in bytes
0
1e+10
2e+10
3e+10
4e+10
5e+10
6e+10
7e+10
1e+08
2e+08
3e+08
Peak of 4,6 MiB
Peak of 12,7 MiB
Peak of 14,3 MiB
Peak of 27,5 MiB
Peak of 50,3 MiB
Peak of 1,3 MiB
Massif Data
filter

```
> 179,6 MiB: Snapshot #26
> 186,9 MiB: Snapshot #27
v 194,5 MiB: Snapshot #28 (peak)
   v 50,3 MiB: QDeclarativeBinding::QDeclarativeBinding(void*, QDeclarativeRefCount*, QObje...
      v 50,3 MiB: QDeclarativeVME::run(QDeclarativeVMEStack<QObject*, 128>&, QDeclarativ...
         > 50,3 MiB: QDeclarativeVME::run(QDeclarativeContextData*, QDeclarativeCompiled...
   v 27,5 MiB: QDeclarativeQtScriptExpression::updateGuards(QPODVector<QDeclarativeEngi...
      v 27,5 MiB: QDeclarativeQtScriptExpression::scriptValue(QObject*, bool*) (qdeclarativee...
         > 27,5 MiB: QDeclarativeExpressionPrivate::scriptValue(QObject*, bool*) (qdeclarativ...
   22,6 MiB: in 1356 places, all below massif's threshold (01.00%)
   v 17,1 MiB: QDeclarativePropertyPrivate::restore(QDeclarativePropertyCache::Data const&, ...
      v 17,1 MiB: QDeclarativePropertyPrivate::restore(QByteArray const&, QObject*, QDeclarat...
         > 17,1 MiB: QDeclarativeVME::run(QDeclarativeVMEStack<QObject*, 128>&, QDeclar...
   v 14,3 MiB: QMetaObjectPrivate::connect(QObject const*, int, QObject const*, int, int, int*) (qo...
      v 14,2 MiB: QMetaObject::connect(QObject const*, int, QObject const*, int, int, int*) (qobje...
         > 14,2 MiB: QDeclarativePropertyPrivate::connect(QObject const*, int, QObject const*,...
           112 B: in 1 place, below massif's threshold (01.00%)
        63,5 KiB: in 1 place, below massif's threshold (01.00%)
   v 12,7 MiB: QVectorData::allocate(int, int) (qvector.cpp:67)
      v 12,6 MiB: QVector<QObjectPrivate::ConnectionList>::malloc(int) (qvector.h:393)
         > 12,6 MiB: QVector<QObjectPrivate::ConnectionList>::realloc(int, int) (qvector.h:466)
        149,5 KiB: in 47 places, all below massif's threshold (01.00%)
   v 11,4 MiB: QDeclarativeVME::run(QDeclarativeVMEStack<QObject*, 128>&, QDeclarativeCo...
      v 11,4 MiB: QDeclarativeVME::run(QDeclarativeContextData*, QDeclarativeCompiledDat...
         > 11,4 MiB: QDeclarativeComponentPrivate::begin(QDeclarativeContextData*, QDecl...
   v 8,6 MiB: QDeclarativeRectangle::QDeclarativeRectangle(QDeclarativeItem*) (qdeclarativer...
      v 8,6 MiB: QDeclarativePrivate::QDeclarativeElement<QDeclarativeRectangle>::QDeclar...
         > 8,6 MiB: void QDeclarativePrivate::createInto<QDeclarativeRectangle>(void*) (in /ho...
   > 4,8 MiB: QHashData::rehash(int) (qhash.cpp:364)
   v 4,6 MiB: QObject::QObject(QObject*) (qobject.cpp:728)
      v 2,3 MiB: QGraphicsObject::QGraphicsObject(QGraphicsItemPrivate&, QGraphicsItem*,...
         > 2,3 MiB: QDeclarativeItem::QDeclarativeItem(QDeclarativeItemPrivate&, QDeclarati...
      v 2,3 MiB: QDeclarativeVisualDataModelData::QDeclarativeVisualDataModelData(int, Q...
         > 2,3 MiB: QDeclarativeVisualDataModel::item(int, QByteArray const&, bool) (qdeclar...
        35,0 KiB: in 44 places, all below massif's threshold (01.00%)
   v 4,6 MiB: QString::fromRawData(QChar const*, int) (qstring.cpp:7114)
      v 4,6 MiB: QDeclarativeExpressionPrivate::init(QDeclarativeContextData*, void*, QDeclar...
         > 4,6 MiB: QDeclarativeExpression::QDeclarativeExpression(QDeclarativeContextDat...
   v 2,8 MiB: QDeclarativeType::create(QObject**, void**, unsigned long) const (qdeclarativeme...
      > 2,8 MiB: QDeclarativeCompiledData::TypeReference::createInstance(QDeclarativeCon...
   v 2,8 MiB: QDeclarativeContext::QDeclarativeContext(QDeclarativeContext*, QObject*) (qdecl...
      > 2,8 MiB: QDeclarativeVisualDataModel::item(int, QByteArray const&, bool) (qdeclarative...
   > 2,7 MiB: QDeclarativeVMEMetaObject::QDeclarativeVMEMetaObject(QObject*, QMetaObjec...
   > 2,6 MiB: QDeclarativeComponentPrivate::begin(QDeclarativeContextData*, QDeclarativeC...
   > 2,6 MiB: QDeclarativeContext::QDeclarativeContext(QDeclarativeContext*, QObject*) (qdecl...
   > 2,4 MiB: QDeclarativeVME::run(QDeclarativeVMEStack<QObject*, 128>&, QDeclarativeCon...
  193,9 MiB: Snapshot #29
  191,5 MiB: Snapshot #30
  189,5 MiB: Snapshot #31
  186,3 MiB: Snapshot #32
  184,3 MiB: Snapshot #33
  182,2 MiB: Snapshot #34
```

Qt Introspection

- QObject Introspection
  - QMetaObject
  - signals, slots, properties, enums, object types
- Global hooks
  - object creation/destruction
  - application start
- Examples: Squish, GammaRay

Qt Introspection DIY

- qt_startup_hook()
- Triggered from QCoreApplication constructor
- Allows you to run your diagnostics code early inside any Qt application
- Use event filter or object creation hooks to wait for interesting events
- Overwriting the hook is platform-specific

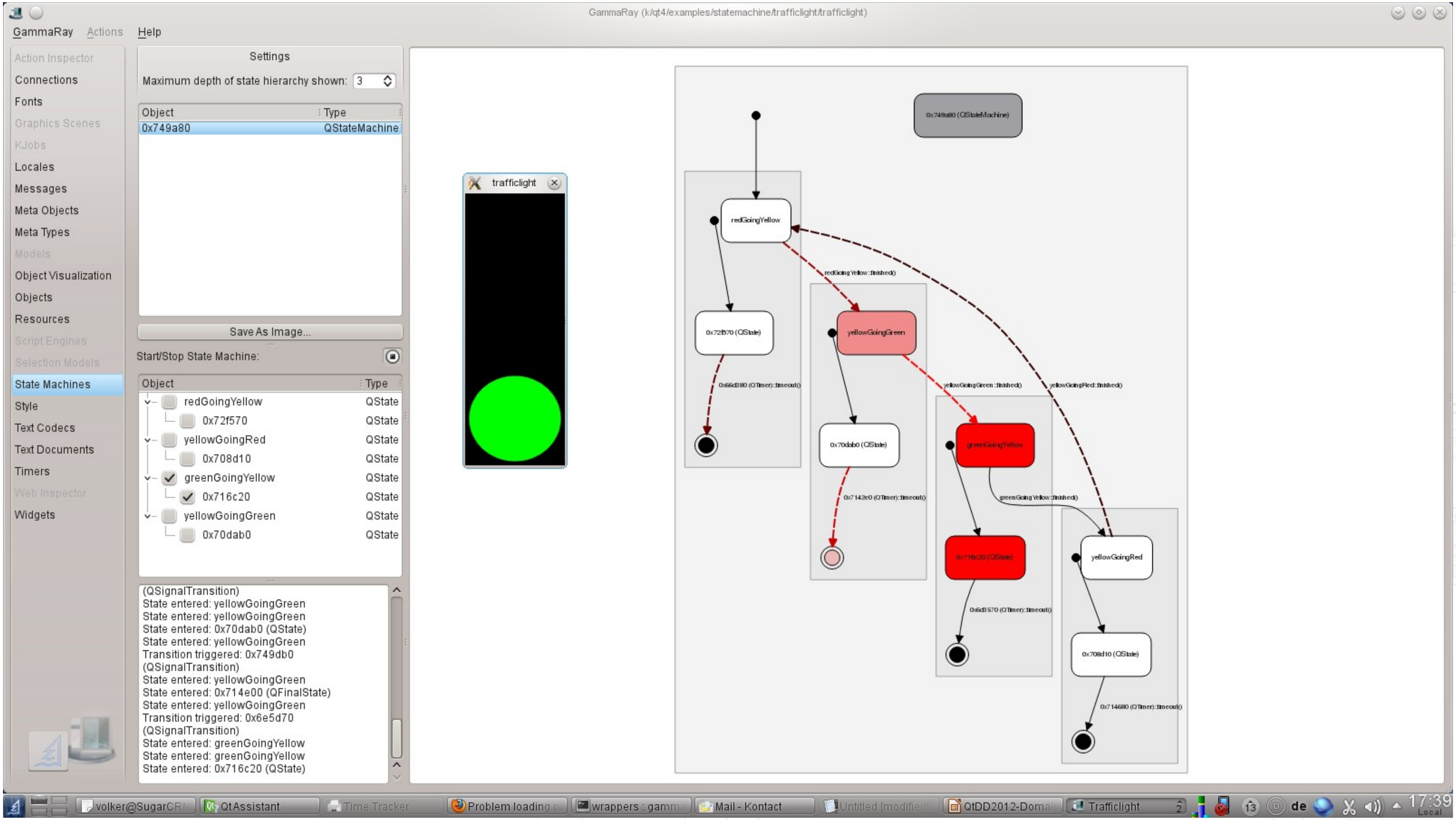
Qt Introspection DIY

- qt_[add|remove]Object(QObject*)
- Triggered from QObject constructor/destructor
  - Too early/late for the virtual table to be complete
  - Consider multi-threading
  - Only covers QObjects
- Powerful, but slightly dangerous.

GammaRay

- GammaRay provides comprehensive visualization for various Qt frameworks
- http://www.kdab.com/gammaray
- Free Software (GPL)
- Introspection from start or runtime attaching
- Framework for building Qt introspection tools

GammaRay



Maximum depth of state hierarchy shown: 3

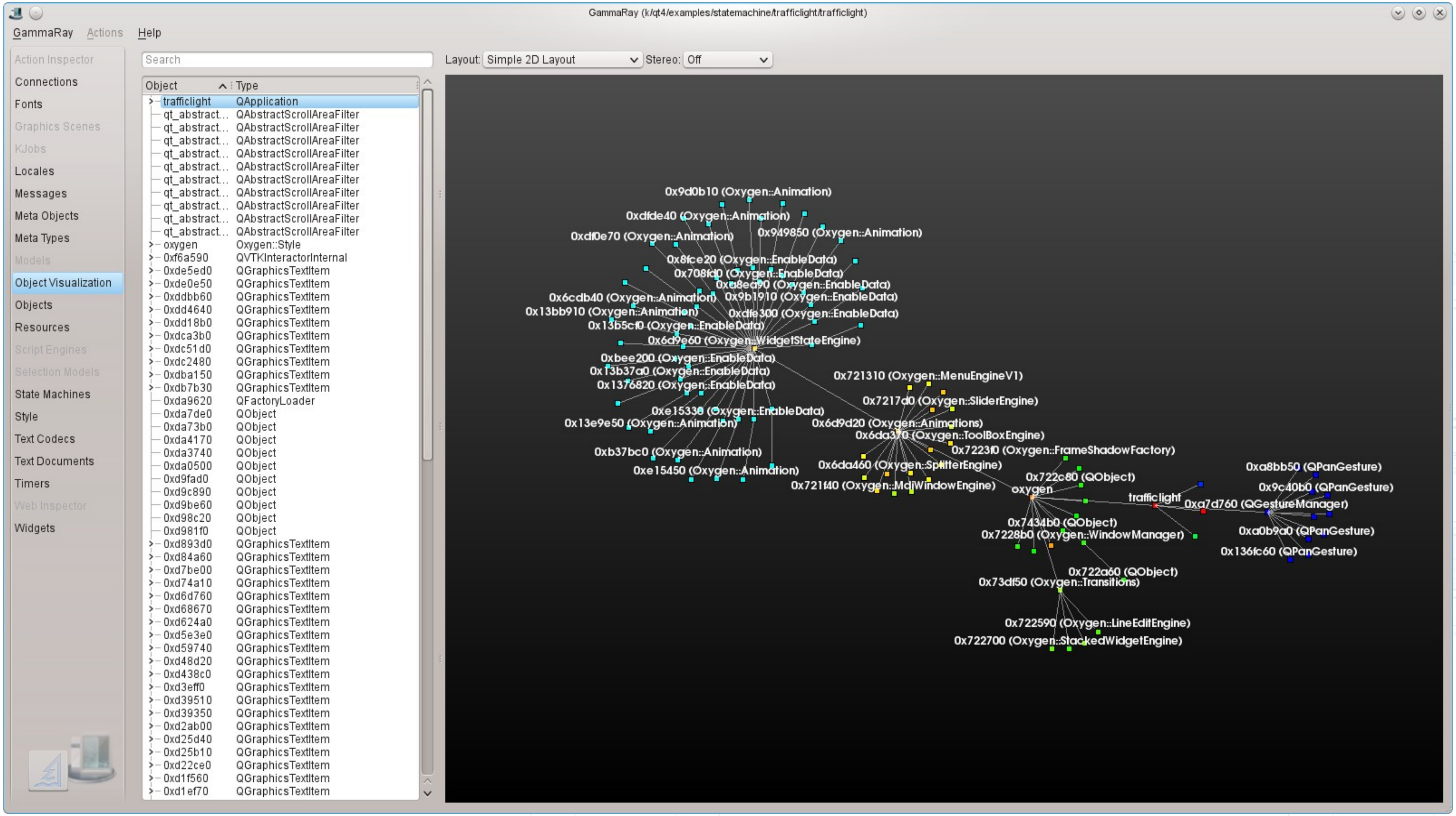
Object	Type
0x749a80	QStateMachine

Start/Stop State Machine: ●

Object	Type
□ redGoingYellow	QState
□ 0x72f570	QState
□ yellowGoingRed	QState
□ 0x708d10	QState
☑ greenGoingYellow	QState
☑ 0x716c20	QState
□ yellowGoingGreen	QState
□ 0x70dab0	QState

```
(QSignalTransition)
State entered: yellowGoingGreen
State entered: yellowGoingGreen
State entered: 0x70dab0 (QState)
State entered: yellowGoingGreen
Transition triggered: 0x749db0
(QSignalTransition)
State entered: yellowGoingGreen
State entered: 0x714e00 (QFinalState)
State entered: yellowGoingGreen
Transition triggered: 0x6e5d70
(QSignalTransition)
State entered: greenGoingYellow
State entered: greenGoingYellow
State entered: 0x716c20 (QState)
```

GammaRay



GammaRay (k/qt4/examples/statemachine/trafficlight/trafficlight)

Layout: Simple 2D Layout

Stereo: Off

Object	Type
trafficlight	QApplication
qt_abstract...	QAbstractScrollAreaFilter
qt_abstract...	QAbstractScrollAreaFilter
qt_abstract...	QAbstractScrollAreaFilter
qt_abstract...	QAbstractScrollAreaFilter
qt_abstract...	QAbstractScrollAreaFilter
qt_abstract...	QAbstractScrollAreaFilter
qt_abstract...	QAbstractScrollAreaFilter
qt_abstract...	QAbstractScrollAreaFilter
qt_abstract...	QAbstractScrollAreaFilter
qt_abstract...	QAbstractScrollAreaFilter
oxygen	Oxygen::Style
0xf6a590	QVTKInteractorInternal
0xde5ed0	QGraphicsTextItem
0xde0e50	QGraphicsTextItem
0xddbb60	QGraphicsTextItem
0xdd4640	QGraphicsTextItem
0xdd18b0	QGraphicsTextItem
0xdca3b0	QGraphicsTextItem
0xdc51d0	QGraphicsTextItem
0xdc2480	QGraphicsTextItem
0xdba150	QGraphicsTextItem
0xdb7b30	QGraphicsTextItem
0xda9620	QFactoryLoader
0xda7de0	QObject
0xda73b0	QObject
0xda4170	QObject
0xda3740	QObject
0xda0500	QObject
0xd9fad0	QObject
0xd9c890	QObject
0xd9be60	QObject
0xd98c20	QObject
0xd981f0	QObject
0xd893d0	QGraphicsTextItem
0xd84a60	QGraphicsTextItem
0xd7be00	QGraphicsTextItem
0xd74a10	QGraphicsTextItem
0xd6d760	QGraphicsTextItem
0xd68670	QGraphicsTextItem
0xd624a0	QGraphicsTextItem
0xd5e3e0	QGraphicsTextItem
0xd59740	QGraphicsTextItem
0xd48d20	QGraphicsTextItem
0xd438c0	QGraphicsTextItem
0xd3eff0	QGraphicsTextItem
0xd39510	QGraphicsTextItem
0xd39350	QGraphicsTextItem
0xd2ab00	QGraphicsTextItem
0xd25d40	QGraphicsTextItem
0xd25b10	QGraphicsTextItem
0xd22ce0	QGraphicsTextItem
0xd1f560	QGraphicsTextItem
0xd1ef70	QGraphicsTextItem

0x9d0b10 (Oxygen::Animation)
0xdfde40 (Oxygen::Animation)
0xdf0e70 (Oxygen::Animation)
0x949850 (Oxygen::Animation)
0x8fce20 (Oxygen::EnableData)
0x708fd0 (Oxygen::EnableData)
0xa8ea90 (Oxygen::EnableData)
0x6cdb40 (Oxygen::Animation)
0x9b1910 (Oxygen::EnableData)
0x13bb910 (Oxygen::Animation)
0xdfe300 (Oxygen::EnableData)
0x13b5cf0 (Oxygen::EnableData)
0x6d9e60 (Oxygen::WidgetStateEngine)
0xbee200 (Oxygen::EnableData)
0x13b37a0 (Oxygen::EnableData)
0x1376820 (Oxygen::EnableData)
0x721310 (Oxygen::MenuEngineV1)
0x7217d0 (Oxygen::SliderEngine)
0xe15330 (Oxygen::EnableData)
0x13e9e50 (Oxygen::Animation)
0x6d9d20 (Oxygen::Animations)
0x6da370 (Oxygen::ToolBoxEngine)
0x7223f0 (Oxygen::FrameShadowFactory)
0xb37bc0 (Oxygen::Animation)
0xe15450 (Oxygen::Animation)
0x6da460 (Oxygen::SplitterEngine)
0xa8bb50 (QPanGesture)
0x722c80 (QObject)
0x9c40b0 (QPanGesture)
0x721f40 (Oxygen::MdiWindowEngine)
oxygen
trafficlight
0xa7d760 (QGestureManager)
0x7434b0 (QObject)
0x7228b0 (Oxygen::WindowManager)
0xa0b9a0 (QPanGesture)
0x136fc60 (QPanGesture)
0x722a60 (QObject)
0x73df50 (Oxygen::Transitions)
0x722590 (Oxygen::LineEditEngine)
0x722700 (Oxygen::StackedWidgetEngine)

Extending GammaRay

- Plug-in based
- Hides the nasty details of the Qt hooks
- Simple API
  - thread-safe object creation/destruction notifications, delayed until the virtual table exists
  - flat or hierarchical object models
  - built-in filtering by object types

Conclusion

- Increased complexity requires better tooling
- Time invested in tooling easily pays off
- Don't be scared about overhead
- Consider turning your repeatedly added debug output into something more reusable :-)

Thanks for listening!

Questions?

Volker Krause
volker.krause@kdab.com
KDAB