Usage of QML Tools

Coding, Debugging & Performance

Aurindam Jana
Digia
Aurindam Jana
IRC: #qt #qt-creator: auri__
Objective

Overview of existing tools

Get feedback and feature requests
Contents

Coding
  QML/JS Editor
  Qt Quick Designer

Debugging
  C++/QML Debugging
  Inspector
  Console

Profiling
  QML Profiler

Q&A
How many of you use Qt Creator for Qt Quick application development?
Quick Overview

Available in Qt 5.0.0 and onwards.
1) Qt Creator 2.6.0

2) Qt Quick 1 – Qt 4.8.x and Qt 5.0.0 and Qt Quick 2 – Qt 5.0.0
   (Deviations are indicated with ✹ )
do {
    var pill = getPill();
} while (pill.color === Qt.color("blue"))
It understands the QML code model

Code faster
- Code navigation
- Auto-completion
- Qt Quick toolbars

Reduce errors
- Syntax check

Maintain code
- Code refactor

Easy to read
- Semantic Highlight

Syntax Checks

```
function loadPuzzle() {
    if (gameCanvas.mode != "")
        Logic.cleanUp();
    Logic.startNewGame(gameCanvas, "puzzle", "levels/level"+acc+".qml")
}
function nextPuzzle() {
    acc = (acc + 1) % 10;
    loadPuzzle();
    if (a == 1)
        acc = 10
    this is Surely an Error
}
Timer {
    id: gameOverTimer
    interval: 1500
    running : gameCanvas.gameOver && gameCanvas.mode == "puzzle" //mode
    repeat : false
    onTriggered: {
        Logic.cleanUp();
        nextPuzzle();
    }
}
Image {
    source: "coin.png"
    anchors.fill: widget
}
GameArea {
    id: gameCanvas
    z: 1
    width: parent.width
    ...}
Coding: Qt Quick Designer

Code faster
  Minimize manual coding
  UI component library
  Visual feedback

Quick Prototype

Easy to use
  Lets you create apps even if you are not a coder.

⚠️ Currently supports Qt Quick 1. Support for Qt Quick 2 is ongoing.
Client Server architecture

1) A TCP server is started that listens to connections on a specified port.

3) Server advertises available services.

2) A TCP client connects to specified port.

4) Service clients connect to respective services.
   (Only one client per service is accepted. All clients share the same port.)

⚠️ An open port presents a security risk. Ensure that the port is properly protected by a firewall.
Debugging: Steps

1) Enable TCP Server
   Compile with qmake argument
   CONFIG+=declarative_debug for Qt Quick 1 apps or
   CONFIG+=qml_debug for Qt Quick 2 apps.

2) Specify Port
   • Pass -qmljsdebugger=port:xxxx as a command line argument.
   • [,host:<ip address>] optional arg specifies the IP address
   • [,block] optional arg blocks the GUI thread until a profiling
     client is connected to the TCP server.

3) Attach a Profiling Client
   Connect a profiling client to the TCP server at known address
   and port.
Press the Debug Button
To debug a running application that has QML debugging enabled, specify the port and the corresponding kit.
Inspect the QML object tree when debugger is not on a debug break.

Modify properties of QML elements.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>object</td>
<td>QQuickRectangle_QML_9</td>
</tr>
<tr>
<td>Keys</td>
<td>object</td>
<td>Keys</td>
</tr>
<tr>
<td>stateChangeAnim</td>
<td>object</td>
<td>SequentialAnimation</td>
</tr>
<tr>
<td>Connections</td>
<td>object</td>
<td>Connections</td>
</tr>
<tr>
<td>bottomBar</td>
<td>object</td>
<td>QQuickImage_QML_18</td>
</tr>
<tr>
<td>scoreBar</td>
<td>object</td>
<td>QQuickImage_QML_10</td>
</tr>
<tr>
<td>puzzleScore</td>
<td>object</td>
<td>QQuickText_QML_17</td>
</tr>
<tr>
<td>puzzleTime</td>
<td>object</td>
<td>QQuickText_QML_16</td>
</tr>
<tr>
<td>puzzleMoves</td>
<td>object</td>
<td>QQuickText_QML_15</td>
</tr>
<tr>
<td>p2Score</td>
<td>object</td>
<td>QQuickText_QML_14</td>
</tr>
<tr>
<td>p1Score</td>
<td>object</td>
<td>QQuickText_QML_13</td>
</tr>
<tr>
<td>arcadeHighScore</td>
<td>object</td>
<td>QQuickText_QML_12</td>
</tr>
<tr>
<td>arcadeScore</td>
<td>object</td>
<td>QQuickText_QML_11</td>
</tr>
<tr>
<td>Behavior</td>
<td>object</td>
<td>Behavior</td>
</tr>
<tr>
<td>Properties</td>
<td>list</td>
<td></td>
</tr>
<tr>
<td>menu</td>
<td>object</td>
<td>Item</td>
</tr>
<tr>
<td>gameCanvas</td>
<td>object</td>
<td>GameArea</td>
</tr>
<tr>
<td>Image</td>
<td>object</td>
<td>Image</td>
</tr>
<tr>
<td>gameOverTimer</td>
<td>object</td>
<td>Timer</td>
</tr>
<tr>
<td>State</td>
<td>object</td>
<td>State</td>
</tr>
<tr>
<td>Transition</td>
<td>object</td>
<td>Transition</td>
</tr>
<tr>
<td>Properties</td>
<td>list</td>
<td></td>
</tr>
<tr>
<td>acc</td>
<td>0</td>
<td>int</td>
</tr>
<tr>
<td>activeFocus</td>
<td>false</td>
<td>bool</td>
</tr>
<tr>
<td>anchors</td>
<td>false</td>
<td>QQuickAnchors*</td>
</tr>
<tr>
<td>antialiasing</td>
<td>false</td>
<td>bool</td>
</tr>
<tr>
<td>baseline</td>
<td>QQuickAnchorLine</td>
<td></td>
</tr>
<tr>
<td>baselineOffset</td>
<td>0</td>
<td>double</td>
</tr>
<tr>
<td>border</td>
<td>QQuickPen</td>
<td></td>
</tr>
<tr>
<td>bottom</td>
<td>QQuickAnchorLine</td>
<td></td>
</tr>
<tr>
<td>children</td>
<td>&lt;unknown v...</td>
<td>QQmlListProperty&lt;QQuickitem&gt;</td>
</tr>
<tr>
<td>childrenRect</td>
<td>false</td>
<td>QRectF</td>
</tr>
<tr>
<td>clip</td>
<td>false</td>
<td>bool</td>
</tr>
<tr>
<td>color</td>
<td>#ffffff</td>
<td>QColor</td>
</tr>
</tbody>
</table>
'Application on top' tool tries to ensure the debugee is always the top level window.

'Select' tool selects UI elements in the view. It can be used to identify a particular element and view its properties.

'Zoom' tool provides zoom in and zoom out functionality.

For Qt Quick 2 applications, Select tool also provides zoom functionality. The Zoom tool is hence disabled.
Console APIs

- Firebug console APIs
- Logging (console.log(), console.warn(), etc.)
- Profiling (console.time(), console.timeEnd(), etc.)
- console.assert(), console.trace(), etc.

Interactive console in Qt Creator

- Evaluate expressions
- Filter messages
- Find functionality

* For Qt Quick 1 applications, a subset of console APIs is available.
Debugging: Salient features

- Modify register values of locals.
- Modify property values of QML objects.
- Watch expressions.
- Evaluate JavaScript expressions.
- Break on JavaScript exceptions.
- Select, zoom UI elements in the application view.

*Available in Qt 5.0.0 and onwards.*
Profiling
Re-uses Debugging Client Server architecture

1) A TCP server is started that listens to connections on a specified port.

3) Server advertises available profiling services.

An open port presents a security risk. Ensure that the port is properly protected by a firewall.

2) A TCP client connects to specified port.

4) Profiling service clients connect to respective profiling services. (Only one client per service is accepted. All clients share the same port.)
1) Enable TCP Server
   Compile with qmake argument
   CONFIG+=declarative_debug for Qt Quick 1 apps or
   CONFIG+=qml_debug for Qt Quick 2 apps.

2) Specify Port
   • Pass -qmljsdebugger=port:xxxxx as a command line argument.
   • [,host:<ip address>] optional arg specifies the IP address
   • [,block] optional arg blocks the GUI thread until a profiling
     client is connected to the TCP server.

3) Attach a Profiling Client
   Connect a profiling client to the TCP server at known address
   and port.
Press the QML Profiler Start Button
To profile a running application that has QML debugging enabled, specify the host and port.
To start an application with the profiler,
qmlprofiler [options] [program] [program args]

To profile a running application that has QML debugging enabled,
qmlprofiler [options] -attach [hostname]

Options
- `-fromStart` to record as soon as the QML engine is started.
- `-p [-port] <number>` specifies the TCP/IP port to use.

Commands
- `r [record]` to toggle recording.
- `q [quit]` to quit.

Profile data is saved in XML format.

Available in Qt 5.0.0 and onwards.
Overview of events on a timeline.

Zoom in or out in Timeline view.

Step through events in either chronological or reverse chronological order.

Detailed view of events in tabular form.

Filter events within a time period.

View callers and callees of functions.

Profile JavaScript code.

* Available in Qt 5.0.0 and onwards.
Profiling: Some use cases

Debug Code!
Find binding loops in your code.

Optimize Code.
Find binding evaluations during animations and state changes.
Available in Qt 5.0.0 and onwards.
Documentation
http://doc.qt.digia.com/qtcreator/index.html

THANK YOU

Contact

Qt mailing lists
Aurindam Jana – aurindam.jana@digia.com
Ensure 'Enable QML Debugging' is checked in Build Settings. The default is checked.
Ensure 'Enable QML' is checked in Run Settings.
To enable Inspector view, ensure 'Show QML object tree' is checked in Debugger Options. The default is checked.
http://qt-projects.org – Qt and Qt Creator icons
http://www.apple.com – iOS icon
http://www.clker.com – Berlin skyline
http://svengraph.deviantart.com – Tools icon
http://www.damieng.com - Tablet icon
http://www.fasticon.com - Display off icon
http://www.cosmicwise.com – Swat fly image
http://turbomilk.com - Black asterisk icon
http://www.saveyourinnertortoise.com - Tortoise with rocket image