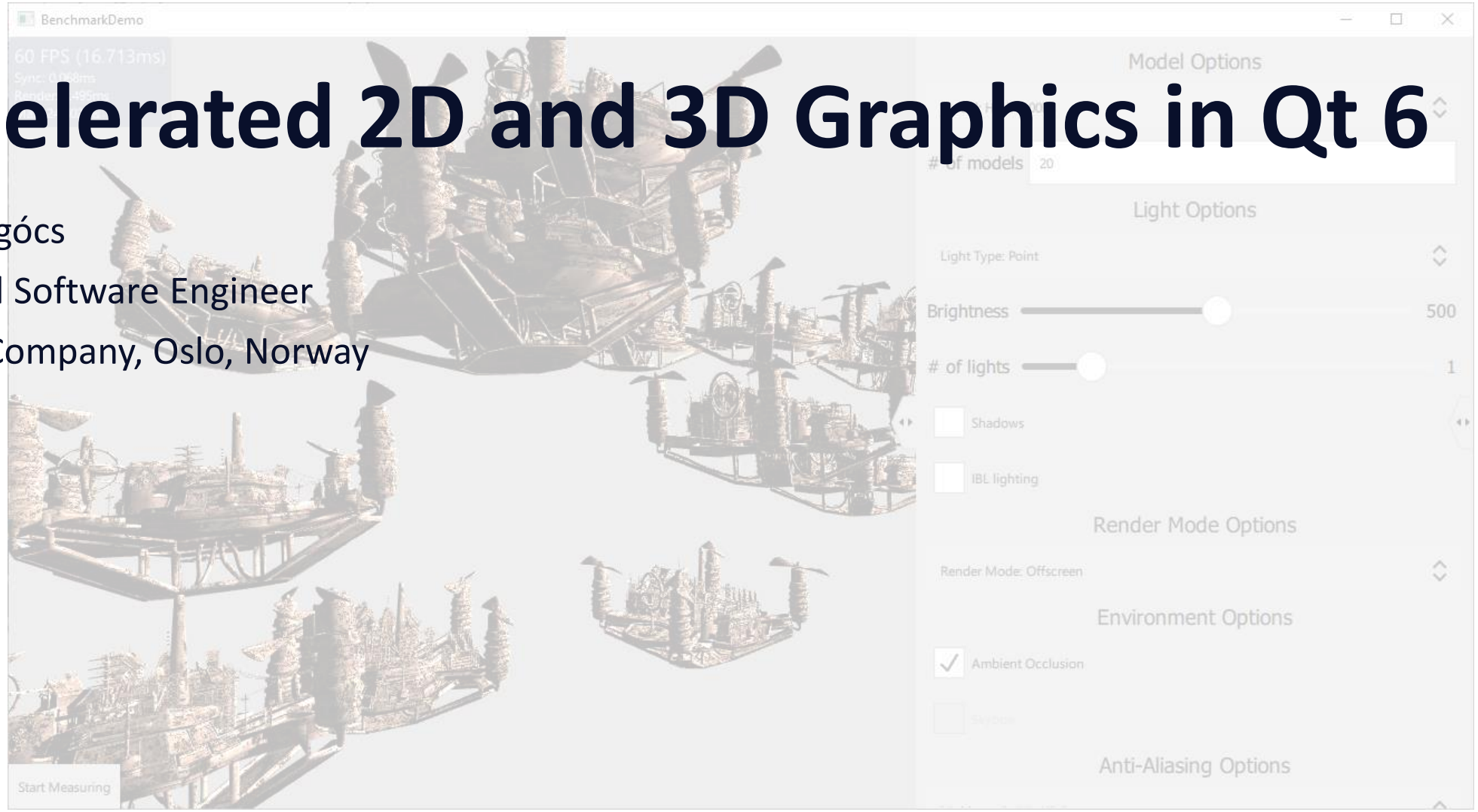
Qt Virtual Tech Con 2020

Accelerated 2D and 3D Graphics in Qt 6



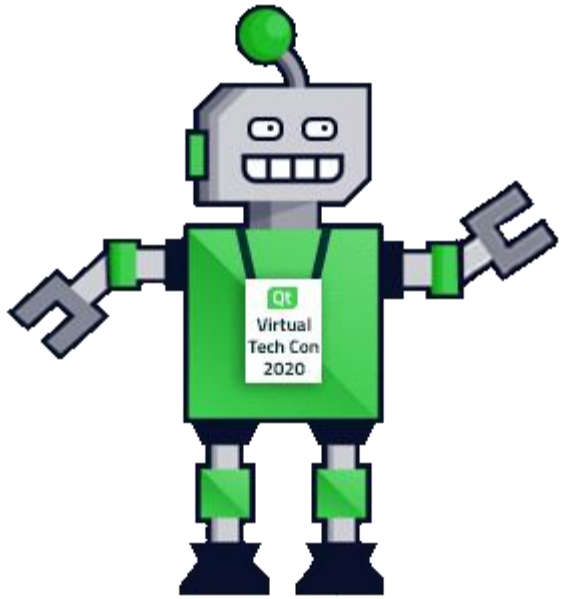
László Agócs

Principal Software Engineer

The Qt Company, Oslo, Norway

Contents

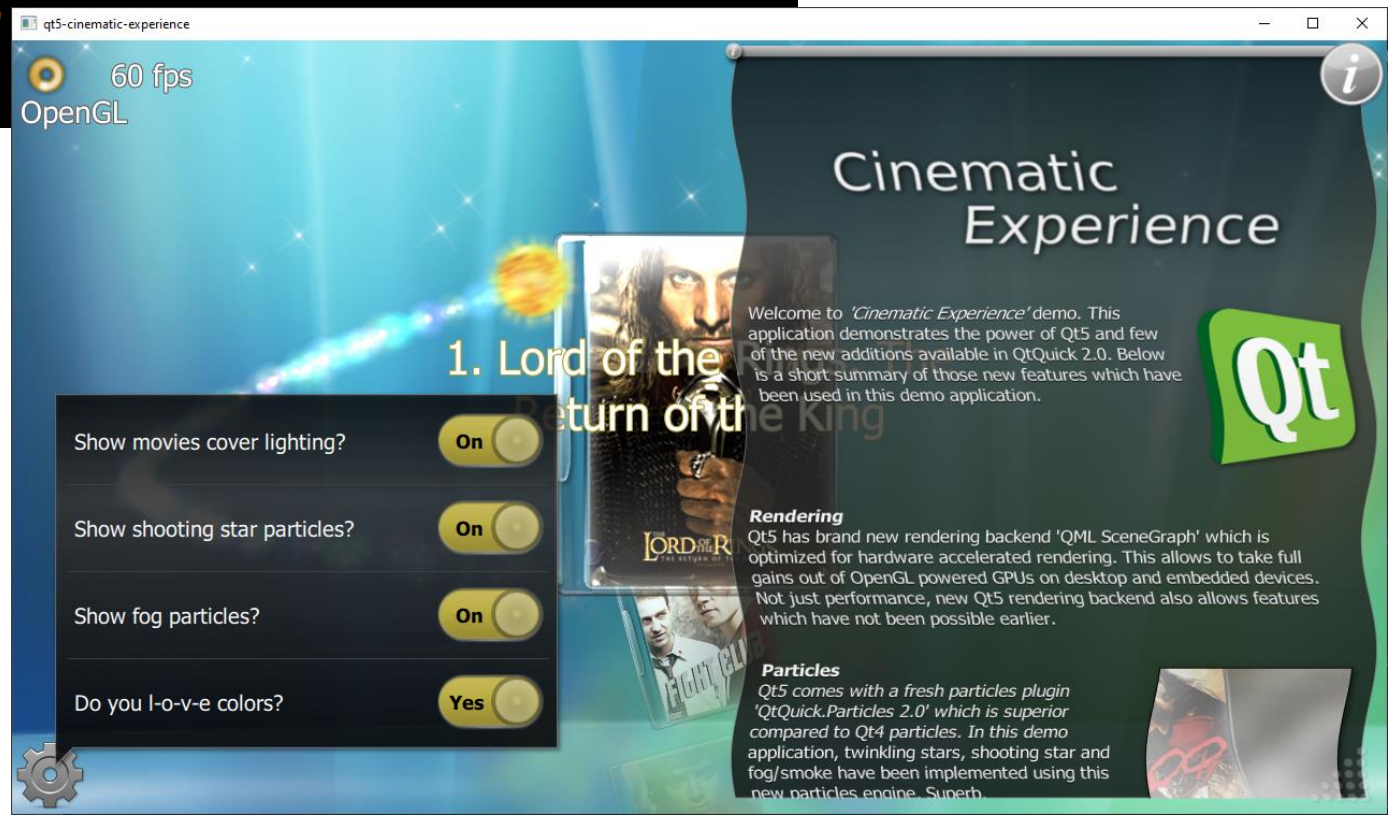
- Qt Quick evolution
- Moving away from OpenGL
- State of things as of May 2020
- The way forward and consequences



Qt Quick

- QML types for creating 2D/2.5D user interfaces
- Backed by a scene graph based rendering engine
  - Uses OpenGL (ES) and GLSL shaders.
- A limited, pure software rendering solution exists too
  - Not in focus for us here – it is expected to stay as-is in Qt 6. (2D only)

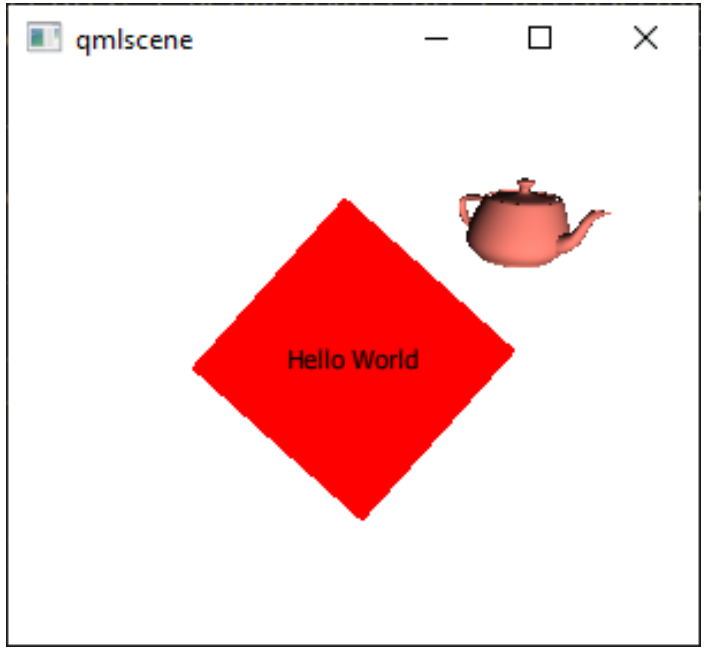
```
import QtQuick 2.0

Rectangle {
    Rectangle {
        width: 100
        height: 100
        anchors.centerIn: parent
        color: "red"
        NumberAnimation on rotation {
            from: 0; to: 360; duration: 2000; loops: Animation.Infinite;
        }
    }

    Text {
        anchors.centerIn: parent
        text: "Hello World"
    }
}
```

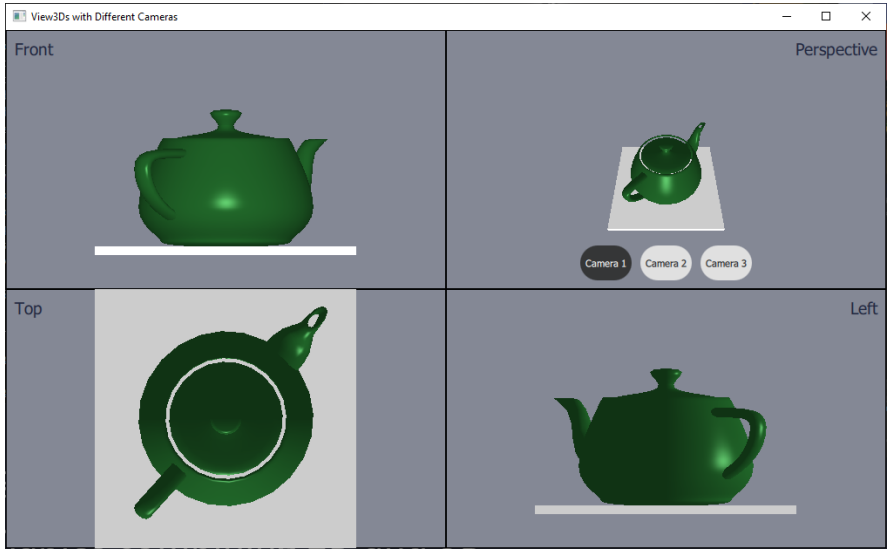


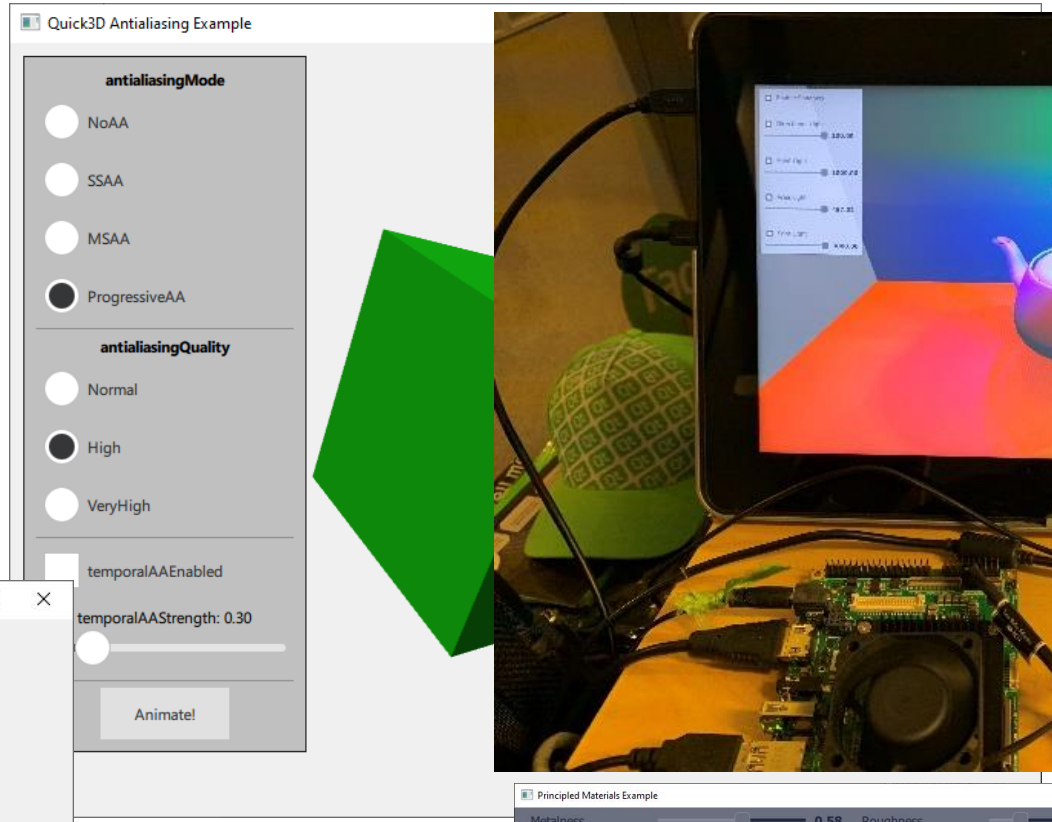
Qt Quick 3D

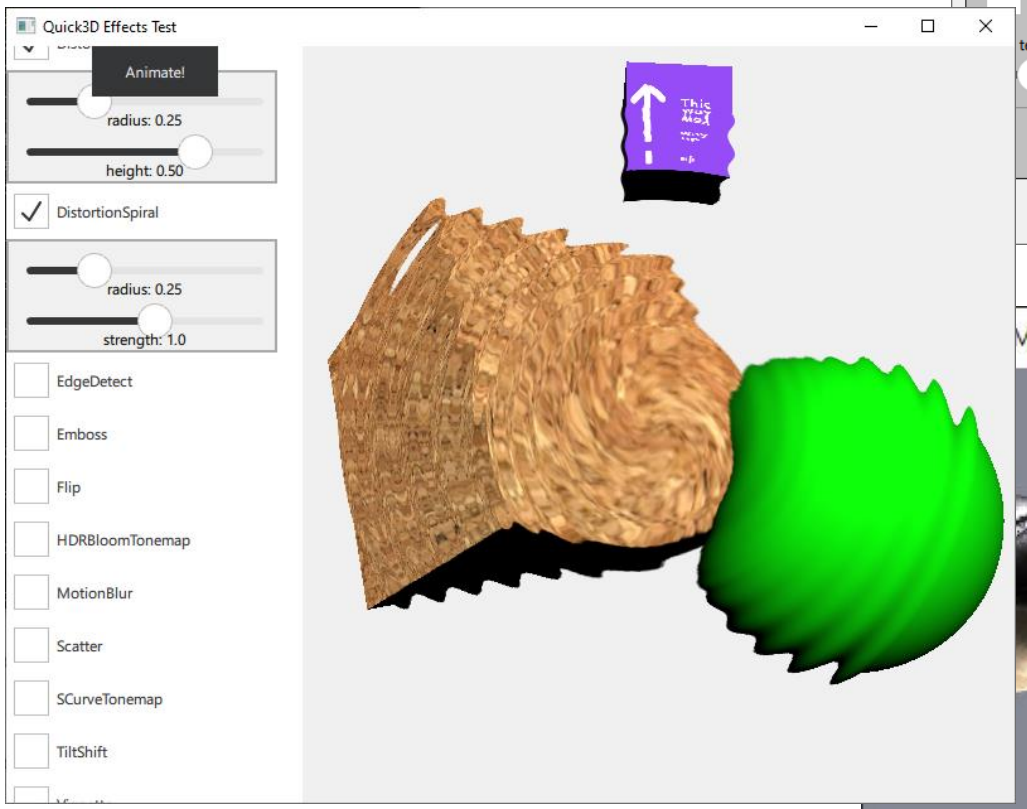
- Not to be confused with Qt 3D.
- QML types for 3D, providing a high-level API for creating 3D content or user interfaces with Qt Quick.
- Tooling
  - Qt Design Studio
  - Serve both designers and developers!
- Combine 2D and 3D more freely.
  - Reduce the confusion and limitations caused by the “separate worlds” of 2D and 3D

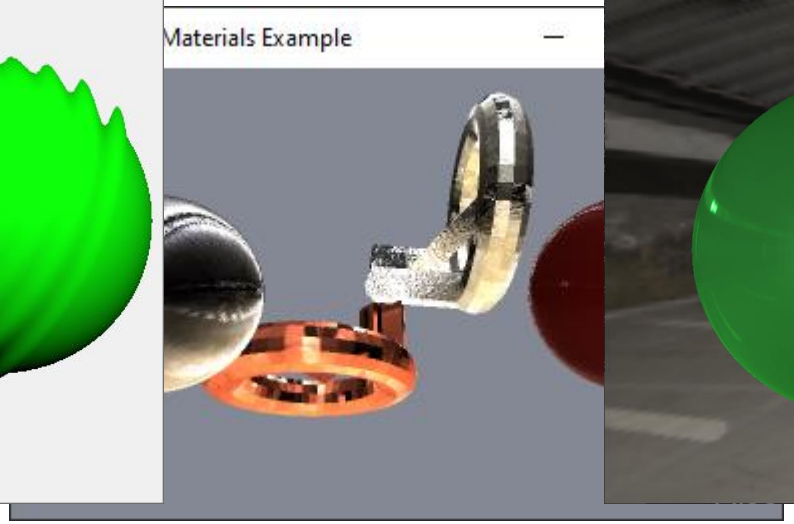
```
import QtQuick 2.15
import QtQuick3D 1.15

Rectangle {
    View3D {
        anchors.fill: parent
        camera: camera
        Model {
            source: "teapot.mesh"
            x: 100
            y: 100
            scale: Qt.vector3d(15, 15, 15)
            materials: [ DefaultMaterial { diffuseColor: "salmon" } ]
            PropertyAnimation on eulerRotation.y { from: 0; to: 360; duration: 2000; loops: -1 }
        }
        DirectionalLight {
            id: dirLight
            ambientColor: Qt.rgba(0.1, 0.1, 0.1, 1.0);
        }
        PerspectiveCamera {
            id: camera
            position: Qt.vector3d(0, 200, 300)
            eulerRotation: Qt.vector3d(-30, 0, 0)
        }
    }

    Rectangle {
        width: 100
        height: 100
        anchors.centerIn: parent
        color: "red"
        NumberAnimation on rotation {
            from: 0; to: 360; duration: 2000; loops: Animation.Infinite;
        }
    }

    Text {
        anchors.centerIn: parent
        text: "Hello World"
    }
}
```

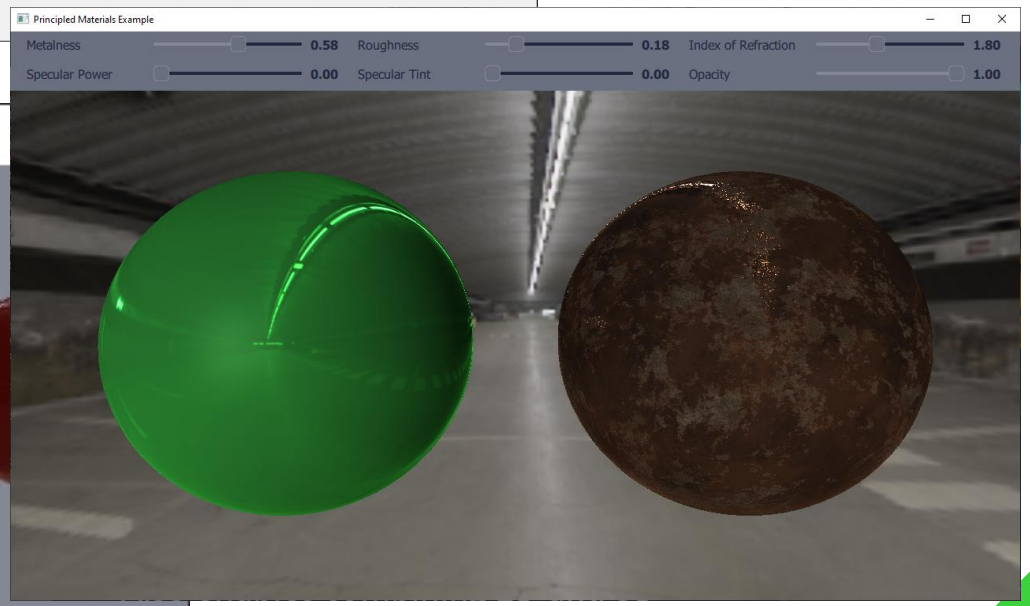



View3Ds with Different Cameras
Front
Perspective
Camera 1
Camera 2
Camera 3
Top
Left


Quick3D Antialiasing Example
antialiasingMode
NoAA
SSAA
MSAA
ProgressiveAA
antialiasingQuality
Normal
High
VeryHigh
temporalAAEnabled
temporalAAStrength: 0.30
Animate!

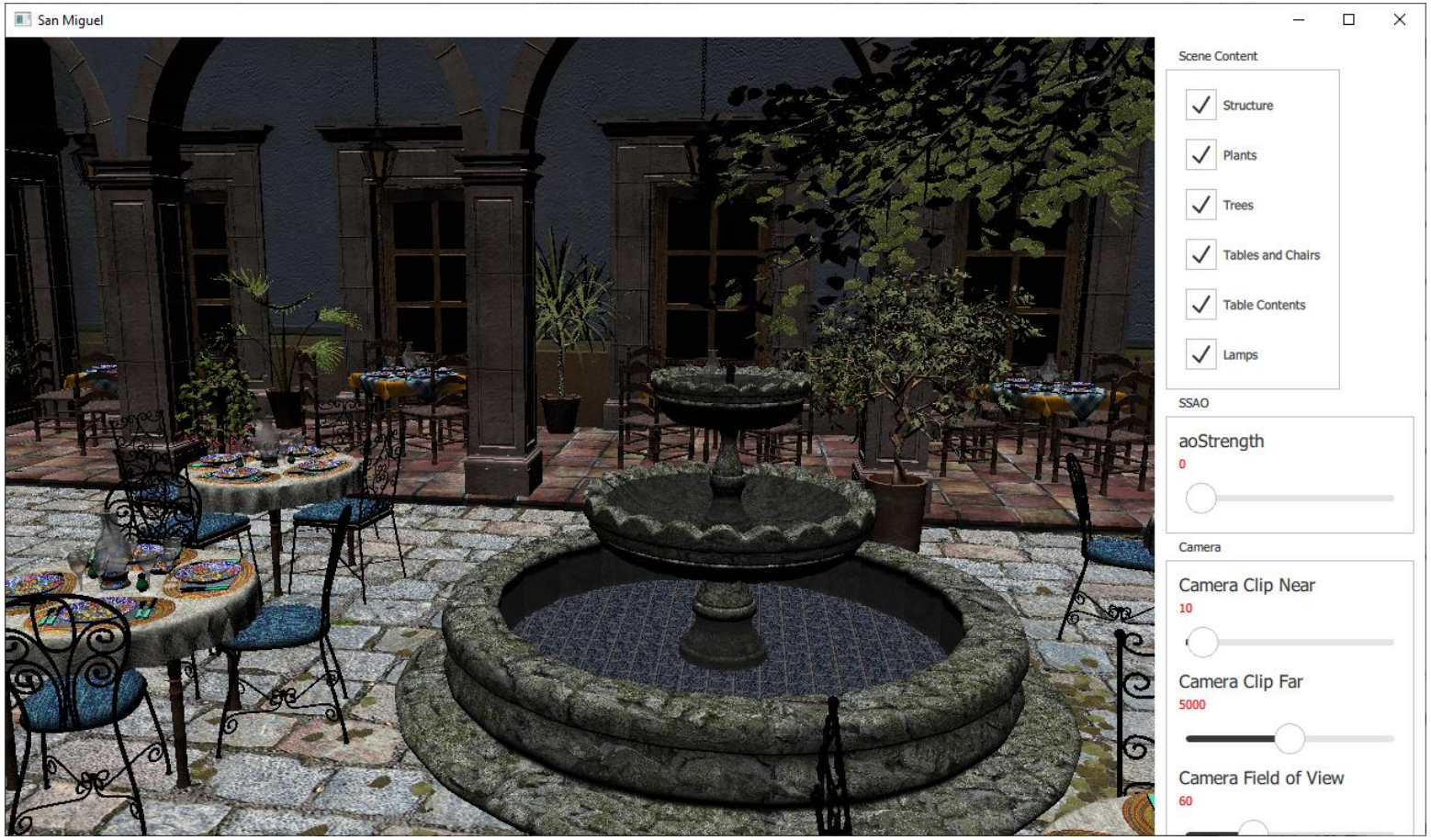

Quick3D Effects Test
Animate!
radius: 0.25
height: 0.50
DistortionSpiral
radius: 0.25
strength: 1.0
EdgeDetect
Emboss
Flip
HDRBloomTonemap
MotionBlur
Scatter
SCurveTonemap
TiltShift

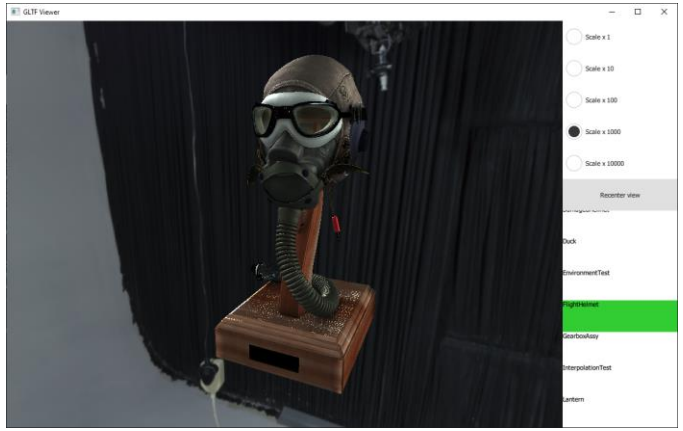

Materials Example


Principled Materials Example
Metalness
0.58
Roughness
0.18
Index of Refraction
1.80
Specular Power
0.00
Specular Tint
0.00
Opacity
1.00

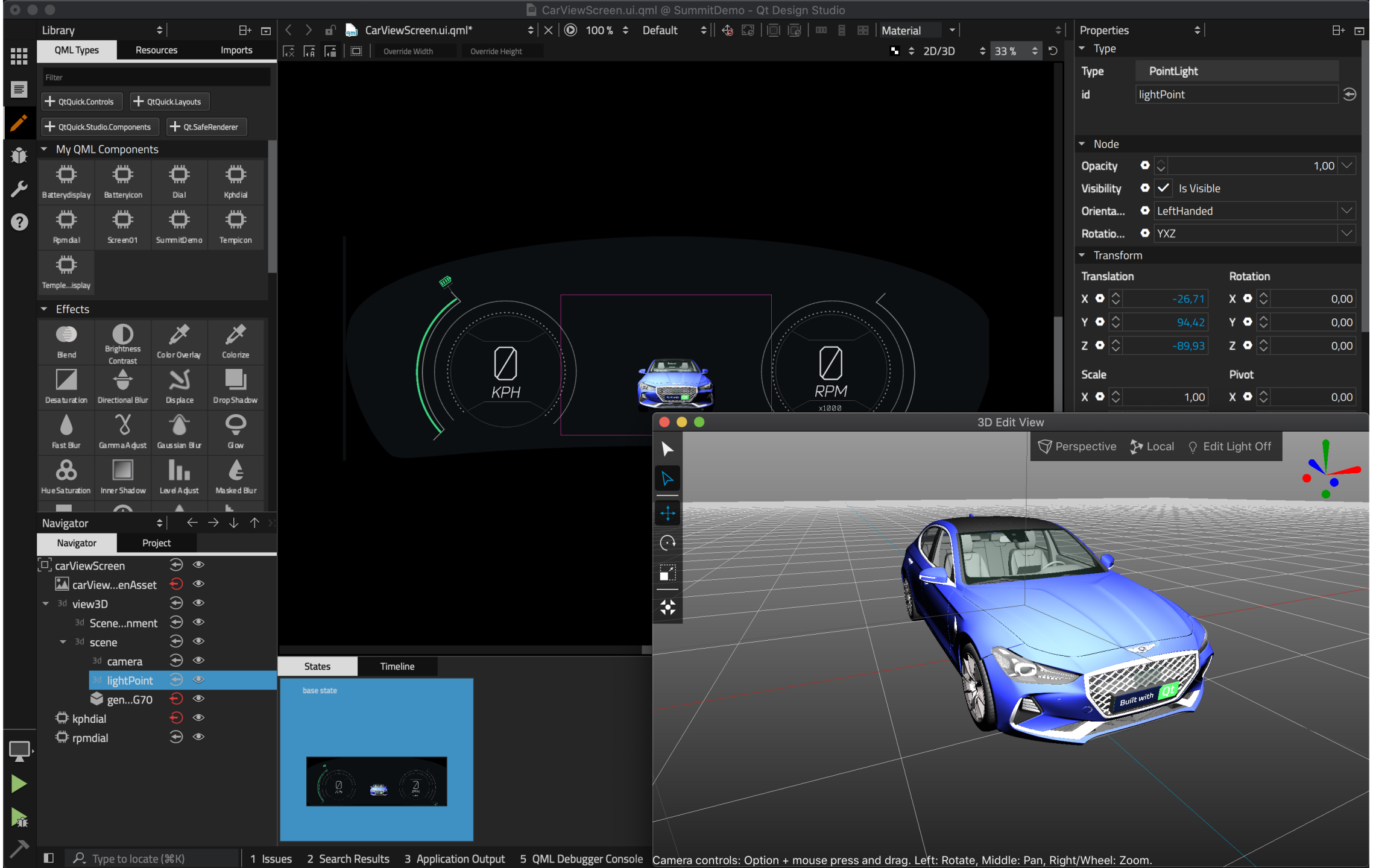
Qt Quick 3D

- Forward renderer
- Default/Principled material (PBR)
- Lights (directional, point, area, spot)
- Light probes
- Shadow mapping, SSAO, skybox
- Progressive, temporal AA
- Post-processing effects
- Custom materials
- Assets: import FBX, glTF2, etc.

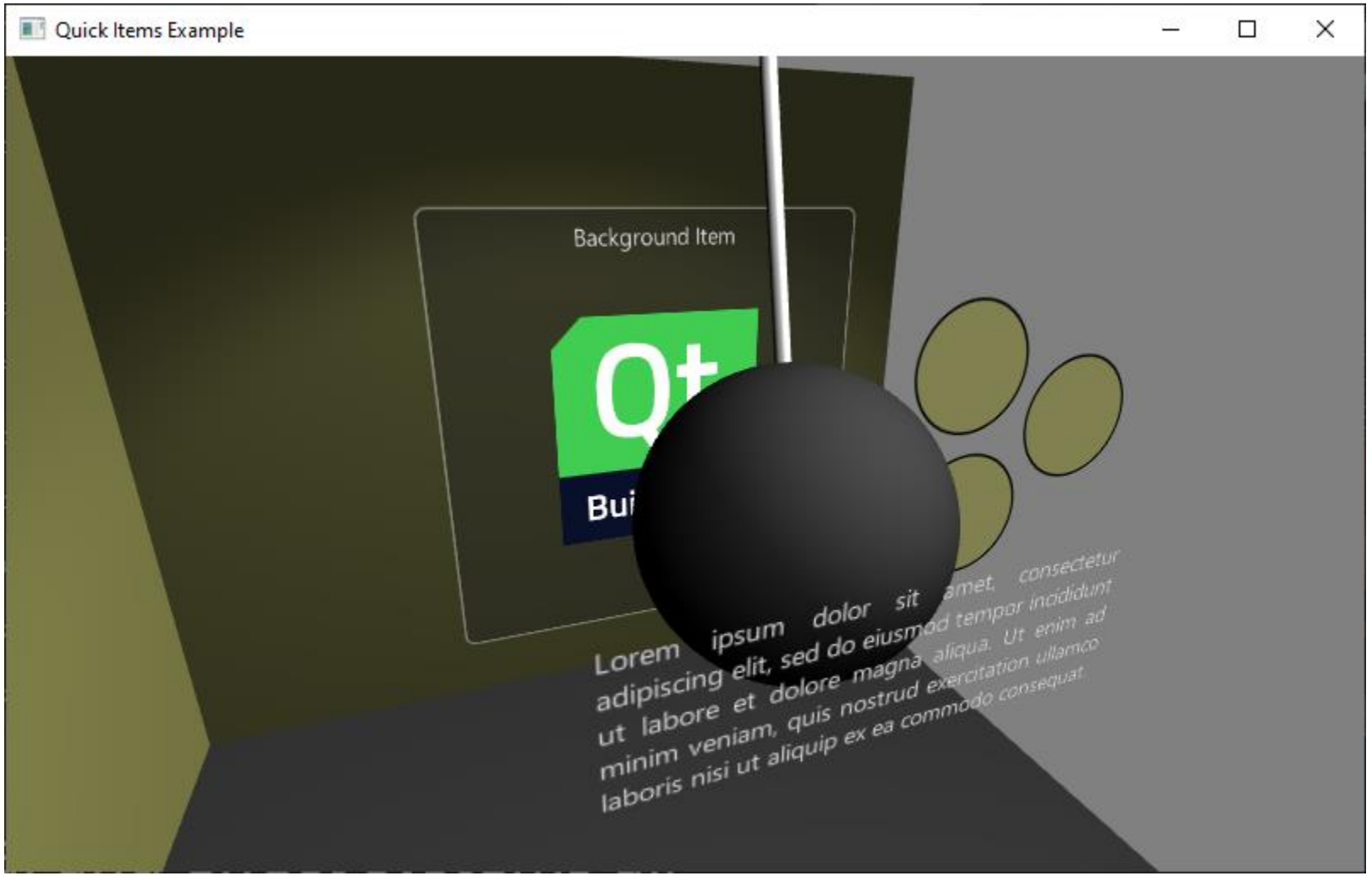





BenchmarkDemo


CarViewScreen.ui.qml @ SummitDemo - Qt Design Studio
Library
QML Types
Resources
Imports
Filter
QtQuick.Controls
QtQuick.Layouts
QtQuick.Studio.Components
Qt.SafeRenderer
My QML Components
Batterydisplay
Batteryicon
Dial
Kphdial
Rpmdial
Screen01
SummitDemo
Tempicon
Temple...isplay
Effects
Blend
Brightness Contrast
Color Overlay
Colorize
Desaturation
Directional Blur
Displace
Drop Shadow
Fast Blur
Gamma Adjust
Gaussian Blur
Glow
Hue Saturation
Inner Shadow
Level Adjust
Masked Blur
Navigator
Navigator
Project
carViewScreen
carView...enAsset
view3D
Scene...nment
scene
camera
lightPoint
gen...G70
kphdial
rpmdial
CarViewScreen.ui.qml*
100 %
Default
Material
Override Width
Override Height
2D/3D
33 %
KPH
RPM
x1000
States
Timeline
base state
Properties
Type
Type
PointLight
id
lightPoint
Node
Opacity
1,00
Visibility
Is Visible
Orienta...
LeftHanded
Rotatio...
YXZ
Transform
Translation
Rotation
X
-26,71
0,00
Y
94,42
0,00
Z
-89,93
0,00
Scale
Pivot
1,00
0,00
3D Edit View
Perspective
Local
Edit Light Off
Built with Qt
Type to locate (⌘K)
1 Issues
2 Search Results
3 Application Output
5 QML Debugger Console
Camera controls: Option + mouse press and drag. Left: Rotate, Middle: Pan, Right/Wheel: Zoom.

Combine 2D and 3D items in the scene - The Next Level



```
    title: qsTr("Quick Items Example")

    View3D {
        anchors.fill: parent
        environment: SceneEnvironment {
            clearColor: "#808080"
            backgroundMode: SceneEnvironment.Color
            antialiasingMode: SceneEnvironment.MSAA
            antialiasingQuality: SceneEnvironment.High
        }

        PerspectiveCamera {
            id: camera
            property real cameraAnimation: 1
            SequentialAnimation on cameraAnimation {
                loops: Animation.Infinite
                NumberAnimation {
                    to: -1
                    duration: 5000
                    easing.type: Easing.InOutQuad
                }
                NumberAnimation {
                    to: 1
                    duration: 5000
                    easing.type: Easing.InOutQuad
                }
            }
            position: Qt.vector3d(200 * cameraAnimation, 300, 500)
            eulerRotation.x: -20
            eulerRotation.y: 20 * cameraAnimation
        }

        DirectionalLight {
            eulerRotation: Qt.vector3d(-135, -110, 0)
            brightness: 100
        }
-\--- main.qml      27% (94,0)    Git-dev  (JavaScript +2)
```

```
                    color: "#80808020"
                    border.color: "black"
                    border.width: 2
                }
                Rectangle {
                    x: 210
                    y: 200
                    width: 100
                    height: 100
                    radius: 50
                    color: "#80808020"
                    border.color: "black"
                    border.width: 2
                }
            }
        }
        //! [circles item]

        //! [text item]
        Node {
            position: Qt.vector3d(0, 80, 250)
            Text {
                width: 300
                wrapMode: Text.WordWrap
                horizontalAlignment: Text.AlignJustify
                font.pixelSize: 14
                color: "#e0e0e0"
                style: Text.Raised
                text: "Lorem ipsum dolor sit amet, consectetur adipisci
ng elit, sed do eiusmod " +
                      "tempor incididunt ut labore et dolore magna aliq
ua. Ut enim ad minim " +
                      "veniam, quis nostrud exercitation ullamco labori
s nisi ut aliquip ex ea " +
                      "commodo consequat."
                SequentialAnimation on opacity {
-\--- main.qml      72% (223,0)    Git-dev  (JavaScript +2)
```

2D item added as child of a 3D one.
(compare to previous sample code on slide 6)

2D vs. 3D scene

Qt Quick

- QQuickItem tree
  - described in QML
  - main thread
- QSGNode tree + QSGMaterial
  - the 2D scene graph
  - render thread (if there is one)
- OpenGL (ES 2.0)
  - plus a few ES 3.0 level things, optionally

Qt Quick 3D

- QQuick3DObject tree
  - described in QML
  - main thread
- QSSGRenderGraphObject tree + material system
  - the 3D scene graph
  - hooks into the 2D scene graph renderer
    - so same concept of render loops, render thread, etc.
- OpenGL 3.3+ or ES 3.0+
  - feature-limited ES 2.0 renderer path exist

2D vs. 3D scene

Qt Quick

- QQuickItem tree
  - described in QML
  - main thread
- QSGNode tree + QSGMaterial
  - the 2D scene graph
  - render thread (if there is one)
- OpenGL (ES 2.0)
  - plus a few ES 3.0 level things, optionally

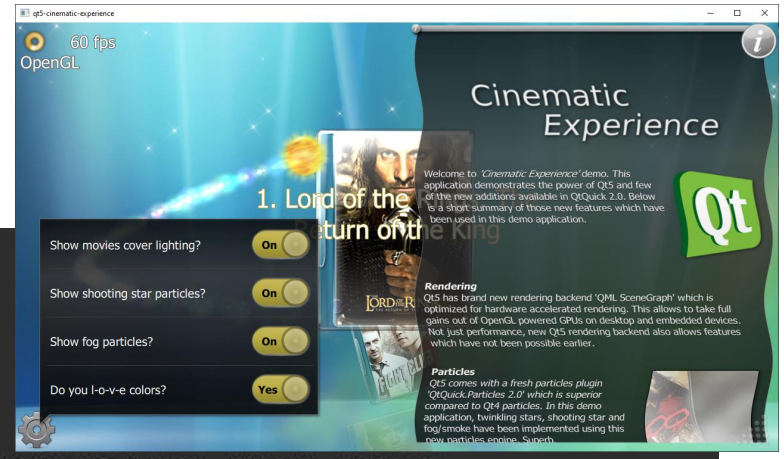
Qt Quick 3D

- QQuick3DObject tree
  - described in QML
  - main thread
- QSSGRenderGraphObject tree + material system
  - the 3D scene graph
  - hooks into the 2D scene graph renderer
    - so same concept of render loops, render thread, etc.
- OpenGL 3.3+ or ES 3.0+
  - feature-limited ES 2.0 renderer path exist

As of Qt 5.15. A bit simplified (many differences in how meshes and materials work), but the 3D part has come a long way since NVIDIA Drive Design.

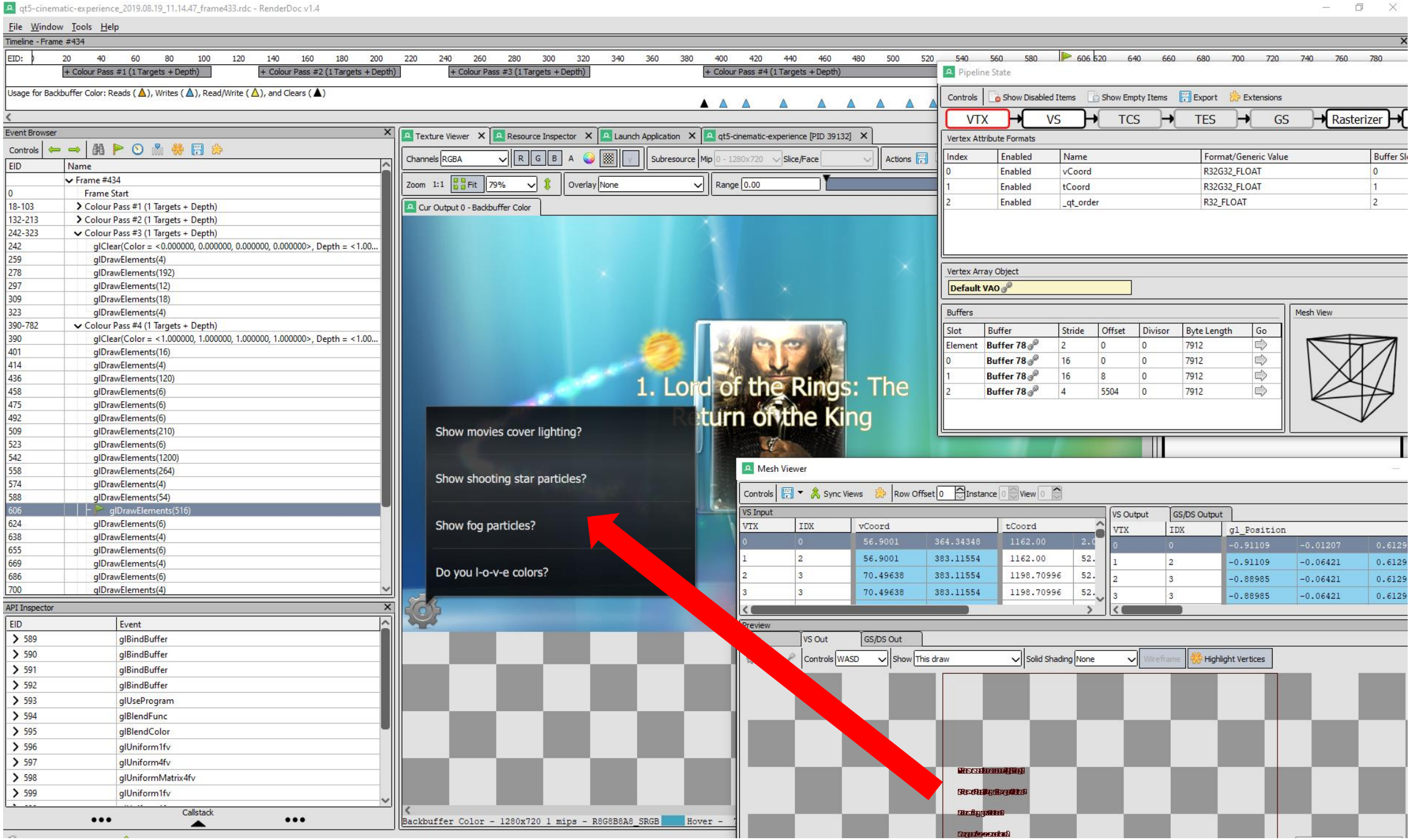
Qt Quick

Batching, atlasing, …




```
[ ] RootNode 0x7fe48800f310  "" ) 1
 [ ] TransformNode( 0x7fe48800f390 identity "QQuickItem(QQuickRootItem:)"  ) 0
  [ ] TransformNode( 0x7fe488015fc0 identity "QQuickItem(QQuickItem:)"  ) 0
   [ ] TransformNode( 0x7fe488016220 identity "QQuickItem(MainView_QMLTYPE_8:)"  ) 0
    [ ] TransformNode( 0x7fe48808f450 identity "QQuickItem(QQuickItem:)"  ) 0
     [ ] TransformNode( 0x7fe48808f620 identity "QQuickItem(QQuickItem:)"  ) 0
      [ ] TransformNode( 0x7fe4881b70e0 identity "QQuickItem(QQuickImage_QML_2:)"  ) 0
       [ ] GeometryNode( 0x7fe4881b7c20 strip #V: 4 #I: 0 x1= 0 y1= 0 x2= 1280 y2= 720 materialtype= 0x7fe4c166ad32 ) "internalimage" 1000001 order 0
      [ ] TransformNode( 0x7fe4881b7290 identity "QQuickItem(QQuickParticleSystem:)"  ) 0
       [ ] TransformNode( 0x7fe4881b7480 identity "QQuickItem(QQuickImageParticle:)"  ) 0
       [ ] TransformNode( 0x7fe4881b7620 identity "QQuickItem(QQuickParticleEmitter:)"  ) 0
      [ ] TransformNode( 0x7fe48808f730 identity "QQuickItem(QQuickListView_QML_9:)"  ) 0
       [ ] TransformNode( 0x7fe4881ad830 translate 0 220 0 "QQuickItem(QQuickItem:)"  ) 0
        [ ] TransformNode( 0x7fe4881ad910 det= 0.444444 "QQuickItem(DelegateItem_QMLTYPE_0:)"  ) 0
         [ ] OpacityNode( 0x7fe4881ada10 opacity= 0.5 combined= 1  "" ) 1
          [ ] TransformNode( 0x7fe4881adf20 identity "QQuickItem(QQuickMouseArea:)"  ) 0
          [ ] TransformNode( 0x7fe4881ae0b0 translate 512 0 0 "QQuickItem(QQuickImage:)"  ) 0
           [ ] RootNode 0x7fe4881b0b90  "" ) 1
            [ ] GeometryNode( 0x7fe4881b1de0 strip #V: 4 #I: 0 x1= 0 y1= 0 x2= 256 y2= 256 materialtype= 0x7fe4c166ad32 ) "internalimage" 1000001 order 0
[ ] TransformNode( 0x7fe4881adaf0 det= 0.444444 "QQuickItem(DelegateItem_QMLTYPE_0:)"  ) 0
acity= 0.7  combined= 1  "" ) 1
 identity  QQuickItem(QQuickMouseArea:)"  ) 0
 translat  512 0 0 "QQuickItem(QQuickImage:)"  ) 0
) 1
0 strip #V: 4 #I: 0 x1= 0 y1= 0 x2= 256 y2= 256 materialtype= 0x7fe4c166ad32 ) "internalimage" 1000001 order 0
entity "QQuickItem(QQuickItem:)"  ) 0
t= 0.0948417 "QQuickItem(DelegateItem_QMLTYPE_0:)"  ) 0
acity= 0.825694 combined= 1  "" ) 1
 identity "QQuickItem(QQuickMouseArea:)"  ) 0
 translate 512 0 0 "QQuickItem(QQuickImage:)"  ) 0
) 1
0 strip #V: 4 #I: 0 x1= 0 y1= 0 x2= 256 y2= 256 materialtype= 0x7fe4c166ad32 ) "internalimage" 1000001 order 0
nslate 390 267 0 "QQuickItem(QQuickText_QML_10:)"  ) 0
entity "text"  ) 1
iangles #V: 140 #I: 210 x1= 20.0244 y1= 8.79277 x2= 479.55 y2= 107.761 materialtype= 0x7fe4c166b051 ) "glyphs" 1000003 order 0
nslate 480 528 0 "QQuickItem(QQuickImage_QML_11:)"  ) 0
```

```
Item {
    id: mainViewArea
    anchors.fill: parent

    Background {
        id: background
    }

    ListView {
        id: listView

        property real globalLightPosX: lightImage.x / root.width
        property real globalLightPosY: lightImage.y / root.height

        // Normal-mapped cover shared among delegates
        ShaderEffectSource {
            id: coverNmapSource
            sourceItem: Image { source: "images/cover_nmap.png" }
            hideSource: true
            visible: false
        }

        anchors.fill: parent
        spacing: -60
        model: moviesModel
        delegate: DelegateItem {
            name: model.name
```


qt5-cinematic-experience_2019.08.19_11.14.47_frame433.rdc - RenderDoc v1.4
File Window Tools Help
Timeline - Frame #434
+ Colour Pass #1 (1 Targets + Depth)
+ Colour Pass #2 (1 Targets + Depth)
+ Colour Pass #3 (1 Targets + Depth)
+ Colour Pass #4 (1 Targets + Depth)
Usage for Backbuffer Color: Reads ( ), Writes ( ), Read/Write ( ), and Clears ( )
Event Browser
Controls
EID Name
Frame #434
Frame Start
Colour Pass #1 (1 Targets + Depth)
Colour Pass #2 (1 Targets + Depth)
Colour Pass #3 (1 Targets + Depth)
glClear(Color = <0.000000, 0.000000, 0.000000, 0.000000>, Depth = <1.00...
glDrawElements(4)
glDrawElements(192)
glDrawElements(12)
glDrawElements(18)
glDrawElements(4)
Colour Pass #4 (1 Targets + Depth)
glClear(Color = <1.000000, 1.000000, 1.000000, 1.000000>, Depth = <1.00...
glDrawElements(16)
glDrawElements(4)
glDrawElements(120)
glDrawElements(6)
glDrawElements(6)
glDrawElements(6)
glDrawElements(210)
glDrawElements(6)
glDrawElements(1200)
glDrawElements(264)
glDrawElements(4)
glDrawElements(54)
glDrawElements(516)
glDrawElements(6)
glDrawElements(4)
glDrawElements(6)
glDrawElements(4)
glDrawElements(6)
glDrawElements(4)
API Inspector
EID Event
glBindBuffer
glBindBuffer
glBindBuffer
glBindBuffer
glUseProgram
glBlendFunc
glBlendColor
glUniform1fv
glUniform4fv
glUniformMatrix4fv
glUniform1fv
Callstack
Texture Viewer
Resource Inspector
Launch Application
qt5-cinematic-experience [PID 39132]
Channels RGBA
Subresource
Actions
Zoom 1:1 Fit 79%
Overlay None
Range 0.00
Cur Output 0 - Backbuffer Color
1. Lord of the Rings: The Return of the King
Show movies cover lighting?
Show shooting star particles?
Show fog particles?
Do you l-o-v-e colors?
Backbuffer Color - 1280x720 1 mips - R8G8B8A8_SRGB
Pipeline State
Controls Show Disabled Items Show Empty Items Export Extensions
VTX VS TCS TES GS Rasterizer
Vertex Attribute Formats
Index Enabled Name Format/Generic Value
0 Enabled vCoord R32G32_FLOAT
1 Enabled tCoord R32G32_FLOAT
2 Enabled _qt_order R32_FLOAT
Vertex Array Object
Default VAO
Buffers
Slot Buffer Stride Offset Divisor Byte Length
Element Buffer 78 2 0 0 7912
0 Buffer 78 16 0 0 7912
1 Buffer 78 16 8 0 7912
2 Buffer 78 4 5504 0 7912
Mesh View
Mesh Viewer
Controls Sync Views Row Offset 0 Instance 0 View 0
VS Input
VTX IDX vCoord tCoord
0 0 56.9001 364.34348 1162.00
1 2 56.9001 383.11554 1162.00
2 3 70.49638 383.11554 1198.70996
3 3 70.49638 383.11554 1198.70996
VS Output GS/DS Output
VTX IDX gl_Position
0 0 -0.91109 -0.01207 0.6129
1 2 -0.91109 -0.06421 0.6129
2 3 -0.88985 -0.06421 0.6129
3 3 -0.88985 -0.06421 0.6129
Preview
VS Out GS/DS Out
Controls WASD Show This draw Solid Shading None Wireframe Highlight Vertices

Qt Quick 3D

› View3D is a viewport, a “2D surface” for the 3D content.

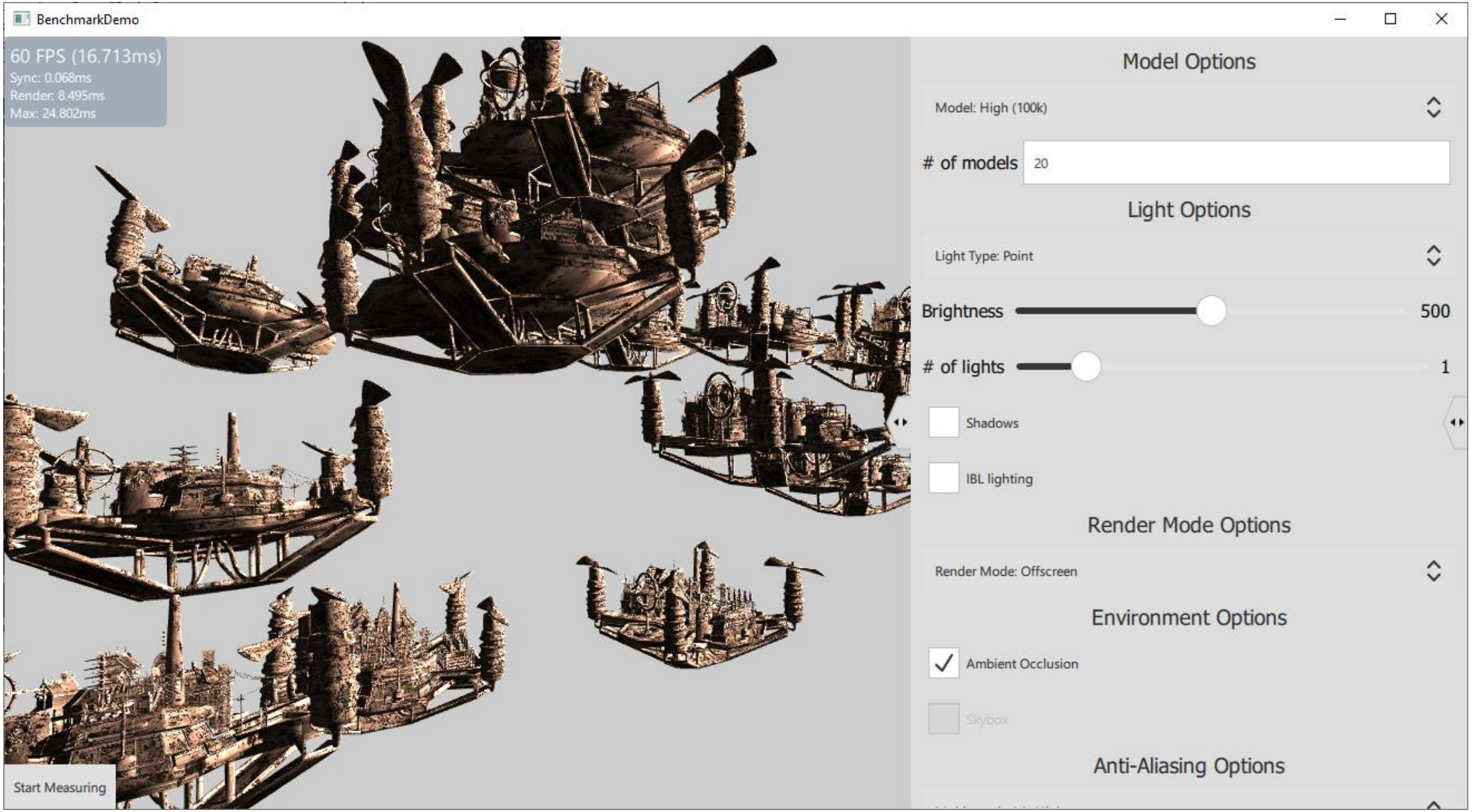
› Those who worked on integrating custom OpenGL rendering into Qt Quick will likely find View3D’s operating modes familiar:

renderMode : enumeration

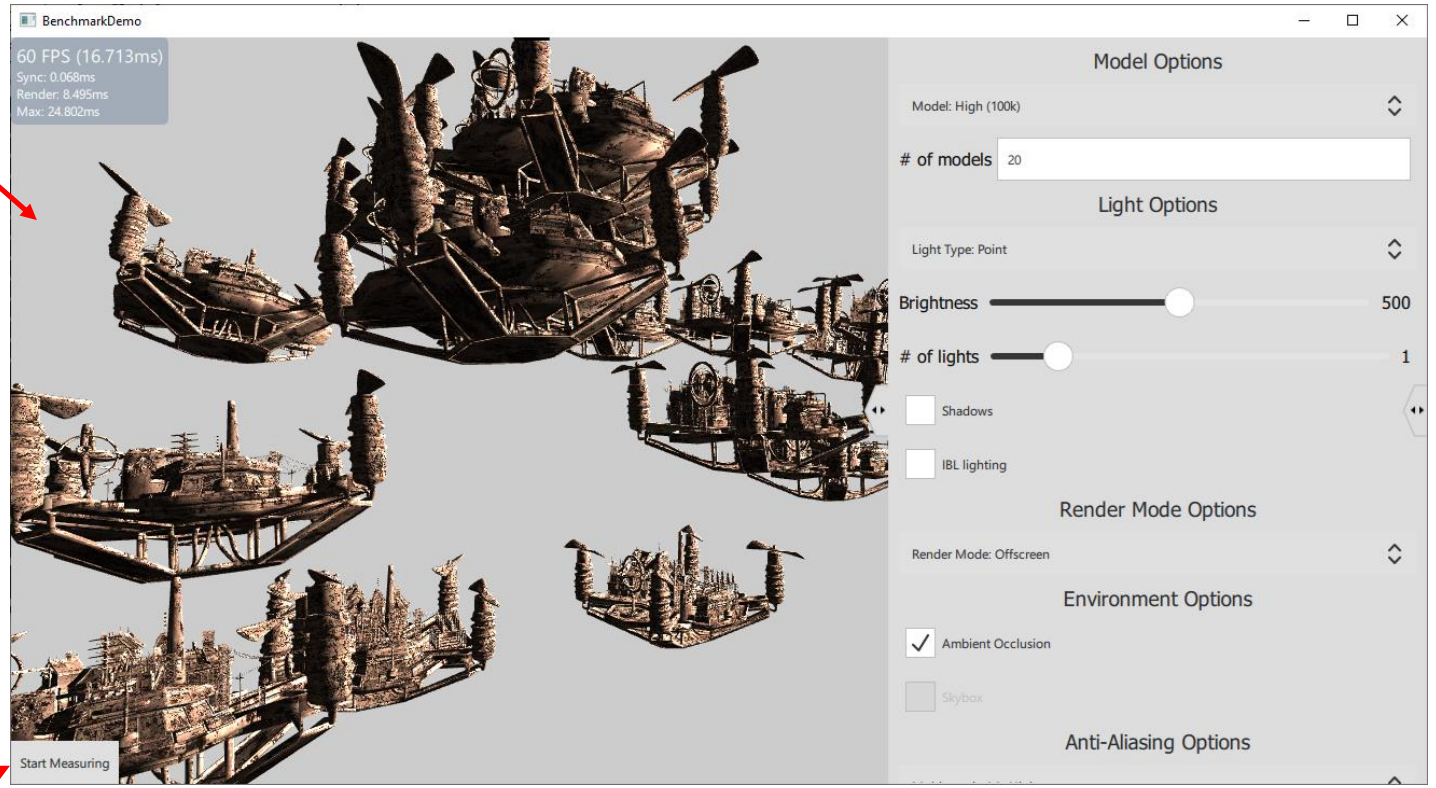
This property determines how the scene is rendered to the viewport.

Constant	Description
`View3D.Offscreen`	Scene is rendered to a texture. Comes with no limitations.
`View3D.Underlay`	Scene is rendered directly to the window before Qt Quick is rendered.
`View3D.Overlay`	Scene is rendered directly to the window after Qt Quick is rendered.
`View3D.Inline`	Scene is rendered to the current render target using QSGRenderNode.

The default mode is `View3D.Offscreen` as this is the offers the best compatibility.


BenchmarkDemo
60 FPS (16.713ms)
Sync: 0.068ms
Render: 8.495ms
Max: 24.802ms
Start Measuring
Model Options
Model: High (100k)
# of models
20
Light Options
Light Type: Point
Brightness
500
# of lights
1
Shadows
IBL lighting
Render Mode Options
Render Mode: Offscreen
Environment Options
Ambient Occlusion
Skybox
Anti-Aliasing Options

```
View3D {
    id: view3D
    anchors.fill: parent
    visible: benchmarkRoot.visible
    renderMode: View3D.Offscreen

    Node {
        LightSpawner {
            id: lightSpawner
            visible: benchmarkRoot.visible
        }

        PerspectiveCamera {
            id: camera
            z: 600
            clipFar: 1200
            clipNear: 1
        }

        MeshSpawner {
            id: modelSpawner
            visible: benchmarkRoot.visible
        }
    }
}

Button {
    id: measureButton
    anchors.bottom: parent.bottom
    anchors.left: parent.left
    text: "Start Measuring"
}
```



BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc - RenderDoc v1.7

File Window Tools Help

Event Browser

Controls

EID	Name
	Frame #1251
0	Capture Start
12-358	Colour Pass #1 (1 Targets)
12	glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
70	glDrawElements(4800)
109	glDrawElements(18186)
148	glDrawElements(4800)
187	glDrawElements(152730)
194	glDrawElements(50799)
201	glDrawElements(4926)
208	glDrawElements(11952)
215	glDrawElements(42480)
222	glDrawElements(5208)
229	glDrawElements(95460)
236	glDrawElements(9498)
275	glDrawElements(4800)
314	glDrawElements(71952)
353	glDrawElements(4800)
358	glBlitFramebuffer(**Framebuffer 3309** , **Framebuffer 3310**)
522-1381	Colour Pass #2 (1 Targets + Depth)
522	glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
544	glDrawElements(768)
555	glDrawElements(4)
567	glDrawElements(26)
575	glDrawElements(4)
590	glDrawElements(4)
599	glDrawElements(16)
627	glDrawElements(90)
651	glDrawElements(228)
675	glDrawElements(84)
689	glDrawElements(16)
717	glDrawElements(72)

API Inspector

EID	Event
13	glDisable
14	glEnable
15	glCullFace
16	glFrontFace
17	glColorMask
18	glDisable
19	glEnable
20	glDepthMask
21	glDepthFunc
22	glDisable
23	glDisable

Callstack

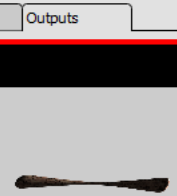
Launch Application | Diagnostic Log | Resource Inspector | Texture Viewer | Pipeline State | BenchmarkDemo [PID 11196] | BenchmarkDemo [PID 10552] | BenchmarkDemo [PID 7076]

Channels RGBA R G B A Subresource Sample Sample 0 Slice/Face Actions

Zoom 1:1 Fit 80% Overlay None Range 0.00 1.00

Cur Output 0 - Renderbuffer 2088

Inputs Outputs



FB0 Renderbuffer 2088

Mesh Viewer

Controls Sync Views Row Offset 0 Instance 0 View 0

VS Input

VTX	IDX	attr_textan			attr_uv0
0	0	0.83312	0.2153	-0.50947	0.33655

VS Output GS/DS Output

VTX	IDX	gl_Position			
0	0	-162.948	61.15884	290.88037	29

Preview

VS In VS Out GS/DS Out

Controls Arcball Show This draw Solid Shading None Wireframe Highlight Vertices

Renderbuffer 2088 - 1302x677 MS{4x 0Q} 1 mips - R8G

Replay Context: Local BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc loaded. No problems detected.

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc - RenderDoc v1.7

File Window Tools Help

Event Browser

Controls

EID	Name
	Frame #1251
0	Capture Start
12-358	Colour Pass #1 (1 Targets)
12	glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
70	glDrawElements(4800)
109	glDrawElements(18186)
148	glDrawElements(4800)
187	glDrawElements(152730)
194	glDrawElements(50799)
201	glDrawElements(4926)
208	glDrawElements(11952)
215	glDrawElements(42480)
222	glDrawElements(5208)
229	glDrawElements(95460)
236	glDrawElements(9498)
275	glDrawElements(4800)
314	glDrawElements(71952)
353	glDrawElements(4800)
358	glBlitFramebuffer(**Framebuffer 3309** , **Framebuffer 3310**)
522-1381	Colour Pass #2 (1 Targets + Depth)
522	glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
544	glDrawElements(768)
555	glDrawElements(4)
567	glDrawElements(26)
575	glDrawElements(4)
590	glDrawElements(4)
599	glDrawElements(16)
627	glDrawElements(90)
651	glDrawElements(228)
675	glDrawElements(84)
689	glDrawElements(16)
717	glDrawElements(72)

API Inspector

EID	Event
71	glUniform3fv
72	glUniform3fv
73	glUniform4fv
74	glUniformMatrix4fv
75	glUniformMatrix4fv
76	glUniformMatrix3fv
77	glUniform4fv
78	glUniform1f
79	glUniform2fv
80	glUniform3fv
81	glUniform3fv

Callstack

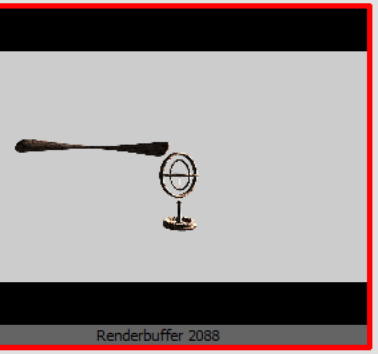
Launch Application | Diagnostic Log | Resource Inspector | Texture Viewer | Pipeline State | BenchmarkDemo [PID 11196] | BenchmarkDemo [PID 10552] | BenchmarkDemo [PID 7076]

Channels RGBA R G B A Subresource Sample Sample 0 Slice/Face Actions

Zoom 1:1 Fit 80% Overlay None Range 0.00 1.00

Cur Output 0 - Renderbuffer 2088

Inputs Outputs



FB0 Renderbuffer 2088

Mesh Viewer

Controls Sync Views Row Offset 0 Instance 0 View 0

VS Input

VTX	IDX	attr_textan			attr_uv0
0	150983	-0.99949	-0.01137	0.0298	0.51108

VS Output GS/DS Output

VTX	IDX	gl_Position			
0	150983	65.36328	-219.96742	460.50226	46

Preview

VS In VS Out GS/DS Out

Controls Arcball Show This draw Solid Shading None Wireframe Highlight Vertices

Renderbuffer 2088 - 1302x677 MS{4x 0Q} 1 mips - R8G

Replay Context: Local

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc loaded. No problems detected.

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc - RenderDoc v1.7

File Window Tools Help

Event Browser

Controls

EID	Name
	Frame #1251
0	Capture Start
12-358	Colour Pass #1 (1 Targets)
12	glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
70	glDrawElements(4800)
109	glDrawElements(18186)
148	glDrawElements(4800)
187	glDrawElements(152730)
194	glDrawElements(50799)
201	glDrawElements(4926)
208	glDrawElements(11952)
215	glDrawElements(42480)
222	glDrawElements(5208)
229	glDrawElements(95460)
236	glDrawElements(9498)
275	glDrawElements(4800)
314	glDrawElements(71952)
353	glDrawElements(4800)
358	glBlitFramebuffer(**Framebuffer 3309** , **Framebuffer 3310**)
522-1381	Colour Pass #2 (1 Targets + Depth)
522	glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
544	glDrawElements(768)
555	glDrawElements(4)
567	glDrawElements(26)
575	glDrawElements(4)
590	glDrawElements(4)
599	glDrawElements(16)
627	glDrawElements(90)
651	glDrawElements(228)
675	glDrawElements(84)
689	glDrawElements(16)
717	glDrawElements(72)

API Inspector

EID	Event
110	glUniform3fv
111	glUniform3fv
112	glUniform4fv
113	glUniformMatrix4fv
114	glUniformMatrix4fv
115	glUniformMatrix3fv
116	glUniform4fv
117	glUniform1f
118	glUniform2fv
119	glUniform3fv
120	glUniform3fv

Callstack

Launch Application | Diagnostic Log | Resource Inspector | Texture Viewer | Pipeline State | BenchmarkDemo [PID 11196] | BenchmarkDemo [PID 10552] | BenchmarkDemo [PID 7076]

Channels RGBA R G B A Subresource Sample Sample 0 Slice/Face Actions

Zoom 1:1 Fit 80% Overlay None Range 0.00 1.00

Cur Output 0 - Renderbuffer 2088

Inputs Outputs

FB0 Renderbuffer 2088

Renderbuffer 2088 - 1302x677 MS{4x 0Q} 1 mips - R8G

Mesh Viewer

Controls Sync Views Row Offset 0 Instance 0 View 0

VS Input

VTX	IDX	attr_textan			attr_uv0
0	0	0.83312	0.2153	-0.50947	0.33655

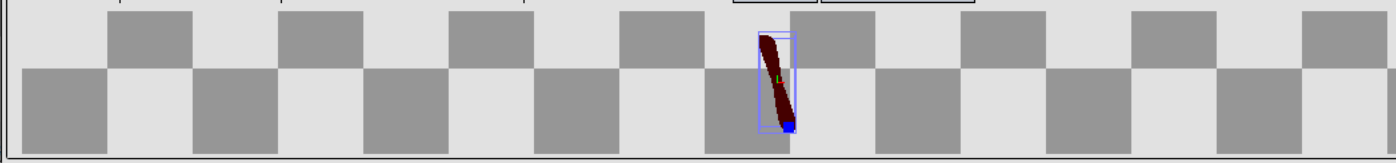
VS Output GS/DS Output

VTX	IDX	gl_Position			
0	0	155.4817	61.15884	493.79886	49

Preview

VS In VS Out GS/DS Out

Controls Arcball Show This draw Solid Shading None Wireframe Highlight Vertices



Replay Context: Local

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc loaded. No problems detected.

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc - RenderDoc v1.7

File Window Tools Help

Event Browser

Controls

EID	Name
	Frame #1251
0	Capture Start
12-358	Colour Pass #1 (1 Targets)
12	glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
70	glDrawElements(4800)
109	glDrawElements(18186)
148	glDrawElements(4800)
187	glDrawElements(152730)
194	glDrawElements(50799)
201	glDrawElements(4926)
208	glDrawElements(11952)
215	glDrawElements(42480)
222	glDrawElements(5208)
229	glDrawElements(95460)
236	glDrawElements(9498)
275	glDrawElements(4800)
314	glDrawElements(71952)
353	glDrawElements(4800)
358	glBlitFramebuffer(**Framebuffer 3309** , **Framebuffer 3310**)
522-1381	Colour Pass #2 (1 Targets + Depth)
522	glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
544	glDrawElements(768)
555	glDrawElements(4)
567	glDrawElements(26)
575	glDrawElements(4)
590	glDrawElements(4)
599	glDrawElements(16)
627	glDrawElements(90)
651	glDrawElements(228)
675	glDrawElements(84)
689	glDrawElements(16)
717	glDrawElements(72)

API Inspector

EID	Event
315	glUniform3fv
316	glUniform3fv
317	glUniform4fv
318	glUniformMatrix4fv
319	glUniformMatrix4fv
320	glUniformMatrix3fv
321	glUniform4fv
322	glUniform1f
323	glUniform2fv
324	glUniform3fv
325	glUniform3fv

Callstack

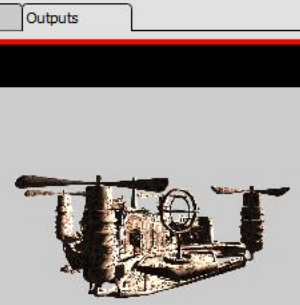
Launch Application | Diagnostic Log | Resource Inspector | Texture Viewer | Pipeline State | BenchmarkDemo [PID 11196] | BenchmarkDemo [PID 10552] | BenchmarkDemo [PID 7076]

Channels RGBA R G B A Subresource Sample Sample 0 Slice/Face Actions

Zoom 1:1 Fit 80% Overlay None Range 0.00 1.00

Cur Output 0 - Renderbuffer 2088

Inputs Outputs



FB0 Renderbuffer 2088

Mesh Viewer

Controls Sync Views Row Offset 0 Instance 0 View 0

VS Input

VTX	IDX	attr_textan			attr_uv0
0	0	0.83312	0.2153	-0.50947	0.33655

VS Output | GS/DS Output

VTX	IDX	gl_Position			
0	0	-26.96503	61.15884	847.95856	84

Preview

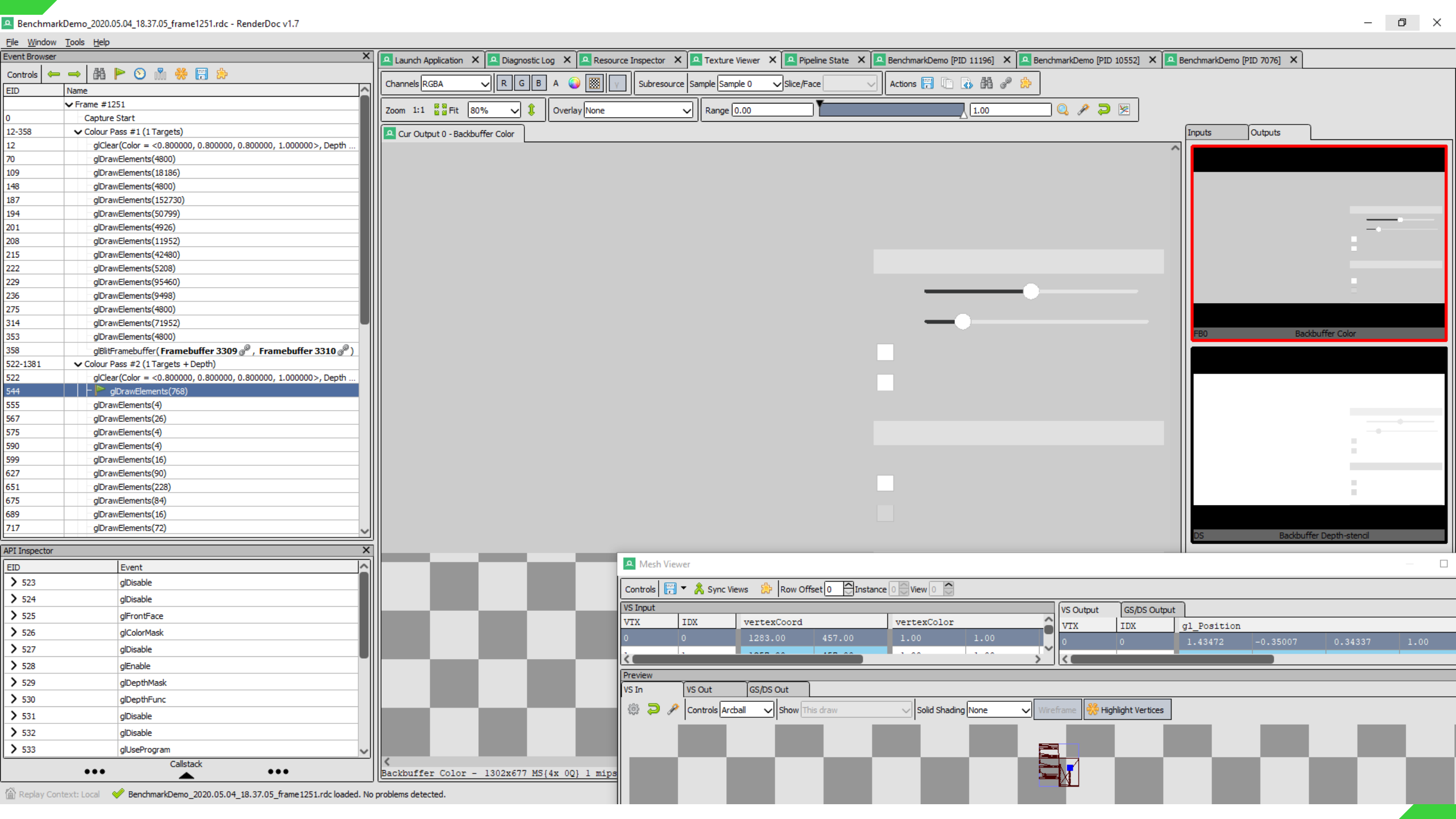
VS In | VS Out | GS/DS Out

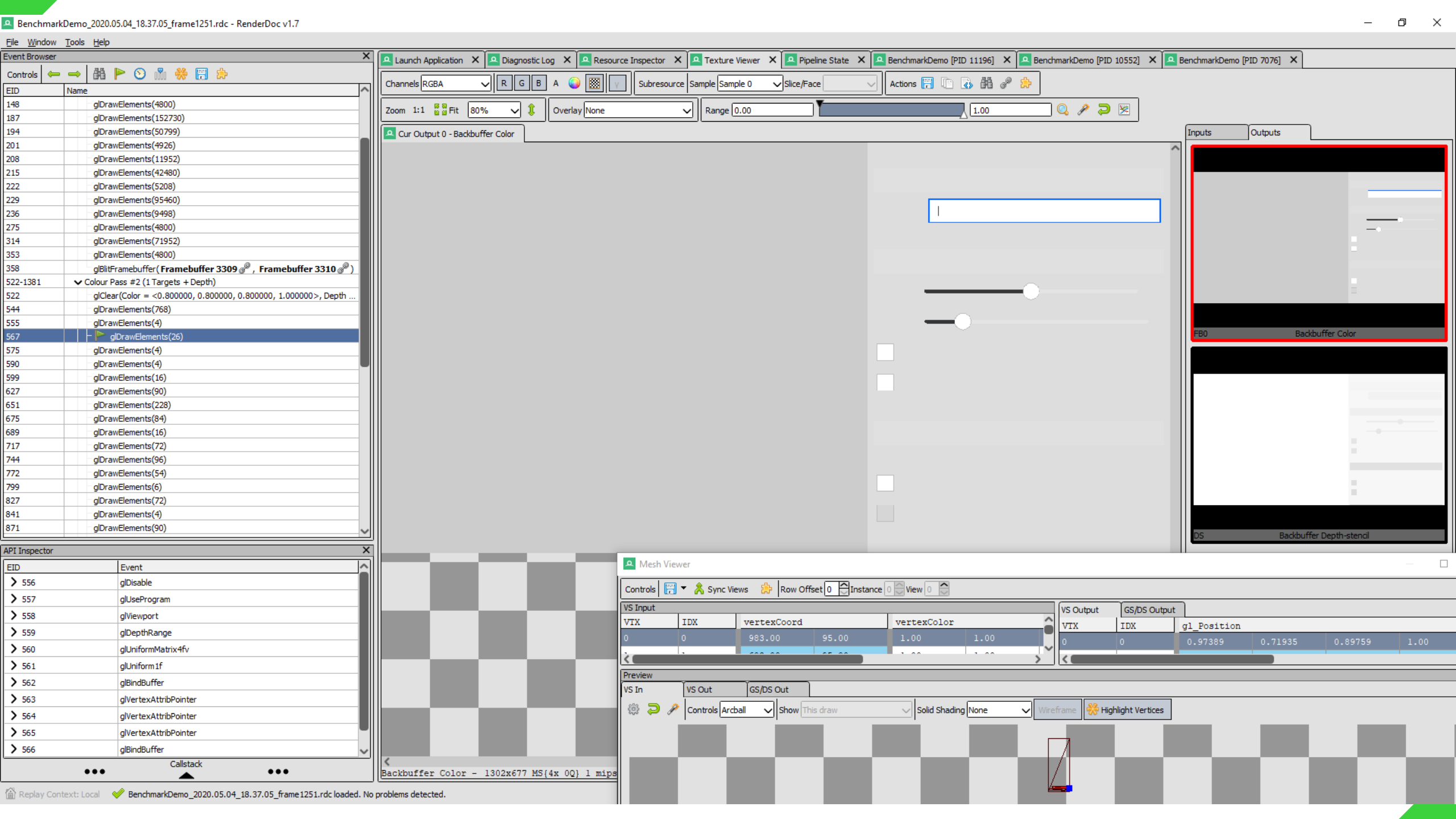
Controls Arcball Show This draw Solid Shading None Wireframe Highlight Vertices

Renderbuffer 2088 - 1302x677 MS{4x 0Q} 1 mips - R8G

Replay Context: Local

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc loaded. No problems detected.


BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc - RenderDoc v1.7
File Window Tools Help
Event Browser
Controls
EID Name
Frame #1251
0 Capture Start
12-358 Colour Pass #1 (1 Targets)
12 glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
70 glDrawElements(4800)
109 glDrawElements(18186)
148 glDrawElements(4800)
187 glDrawElements(152730)
194 glDrawElements(50799)
201 glDrawElements(4926)
208 glDrawElements(11952)
215 glDrawElements(42480)
222 glDrawElements(5208)
229 glDrawElements(95460)
236 glDrawElements(9498)
275 glDrawElements(4800)
314 glDrawElements(71952)
353 glDrawElements(4800)
358 glBlitFramebuffer(Framebuffer 3309 , Framebuffer 3310)
522-1381 Colour Pass #2 (1 Targets + Depth)
522 glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
544 glDrawElements(768)
555 glDrawElements(4)
567 glDrawElements(26)
575 glDrawElements(4)
590 glDrawElements(4)
599 glDrawElements(16)
627 glDrawElements(90)
651 glDrawElements(228)
675 glDrawElements(84)
689 glDrawElements(16)
717 glDrawElements(72)
API Inspector
EID Event
523 glDisable
524 glDisable
525 glFrontFace
526 glColorMask
527 glDisable
528 glEnable
529 glDepthMask
530 glDepthFunc
531 glDisable
532 glDisable
533 glUseProgram
Callstack
Launch Application
Diagnostic Log
Resource Inspector
Texture Viewer
Pipeline State
BenchmarkDemo [PID 11196]
BenchmarkDemo [PID 10552]
BenchmarkDemo [PID 7076]
Channels RGBA R G B A
Subresource Sample Sample 0 Slice/Face
Actions
Zoom 1:1 Fit 80%
Overlay None
Range 0.00 1.00
Cur Output 0 - Backbuffer Color
Inputs
Outputs
FB0 Backbuffer Color
DS Backbuffer Depth-stencil
Backbuffer Color - 1302x677 MS{4x 0Q} 1 mips
Mesh Viewer
Controls Sync Views Row Offset 0 Instance 0 View 0
VS Input
VTX IDX vertexCoord vertexColor
0 0 1283.00 457.00 1.00 1.00
VS Output GS/DS Output
VTX IDX gl_Position
0 0 1.43472 -0.35007 0.34337 1.00
Preview
VS In VS Out GS/DS Out
Controls Arcball Show This draw Solid Shading None Wireframe Highlight Vertices
Replay Context: Local
BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc loaded. No problems detected.


BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc - RenderDoc v1.7
File Window Tools Help
Event Browser
Controls
EID Name
148 glDrawElements(4800)
187 glDrawElements(152730)
194 glDrawElements(50799)
201 glDrawElements(4926)
208 glDrawElements(11952)
215 glDrawElements(42480)
222 glDrawElements(5208)
229 glDrawElements(95460)
236 glDrawElements(9498)
275 glDrawElements(4800)
314 glDrawElements(71952)
353 glDrawElements(4800)
358 glBlitFramebuffer(Framebuffer 3309, Framebuffer 3310)
522-1381 Colour Pass #2 (1 Targets + Depth)
522 glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
544 glDrawElements(768)
555 glDrawElements(4)
567 glDrawElements(26)
575 glDrawElements(4)
590 glDrawElements(4)
599 glDrawElements(16)
627 glDrawElements(90)
651 glDrawElements(228)
675 glDrawElements(84)
689 glDrawElements(16)
717 glDrawElements(72)
744 glDrawElements(96)
772 glDrawElements(54)
799 glDrawElements(6)
827 glDrawElements(72)
841 glDrawElements(4)
871 glDrawElements(90)
API Inspector
EID Event
556 glDisable
557 glUseProgram
558 glViewport
559 glDepthRange
560 glUniformMatrix4fv
561 glUniform1f
562 glBindBuffer
563 glVertexAttribPointer
564 glVertexAttribPointer
565 glVertexAttribPointer
566 glBindBuffer
Callstack
Launch Application
Diagnostic Log
Resource Inspector
Texture Viewer
Pipeline State
BenchmarkDemo [PID 11196]
BenchmarkDemo [PID 10552]
BenchmarkDemo [PID 7076]
Channels RGBA R G B A
Subresource Sample Sample 0 Slice/Face
Actions
Zoom 1:1 Fit 80%
Overlay None
Range 0.00 1.00
Cur Output 0 - Backbuffer Color
Inputs
Outputs
FB0 Backbuffer Color
DS Backbuffer Depth-stencil
Backbuffer Color - 1302x677 MS{4x 0Q} 1 mips
Mesh Viewer
Controls Sync Views Row Offset 0 Instance 0 View 0
VS Input
VTX IDX vertexCoord vertexColor
0 0 983.00 95.00 1.00 1.00
VS Output GS/DS Output
VTX IDX gl_Position
0 0 0.97389 0.71935 0.89759 1.00
Preview
VS In VS Out GS/DS Out
Controls Arcball Show This draw Solid Shading None Wireframe Highlight Vertices
Replay Context: Local
BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc loaded. No problems detected.

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc - RenderDoc v1.7

File Window Tools Help

Event Browser

Controls

EID	Name
148	glDrawElements(4800)
187	glDrawElements(152730)
194	glDrawElements(50799)
201	glDrawElements(4926)
208	glDrawElements(11952)
215	glDrawElements(42480)
222	glDrawElements(5208)
229	glDrawElements(95460)
236	glDrawElements(9498)
275	glDrawElements(4800)
314	glDrawElements(71952)
353	glDrawElements(4800)
358	glBlitFramebuffer(**Framebuffer 3309** , **Framebuffer 3310**)
522-1381	Colour Pass #2 (1 Targets + Depth)
522	glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
544	glDrawElements(768)
555	glDrawElements(4)
567	glDrawElements(26)
575	glDrawElements(4)
590	glDrawElements(4)
599	glDrawElements(16)
627	glDrawElements(90)
651	glDrawElements(228)
675	glDrawElements(84)
689	glDrawElements(16)
717	glDrawElements(72)
744	glDrawElements(96)
772	glDrawElements(54)
799	glDrawElements(6)
827	glDrawElements(72)
841	glDrawElements(4)
871	glDrawElements(90)

API Inspector

EID	Event
576	glEnable
577	glBlendFuncSeparate
578	glBlendEquationSeparate
579	glDepthMask
580	glUseProgram
581	glUniformMatrix4fv
582	glActiveTexture
583	glBindTexture
584	glUniform1i
585	glBindBuffer
586	glVertexAttribPointer

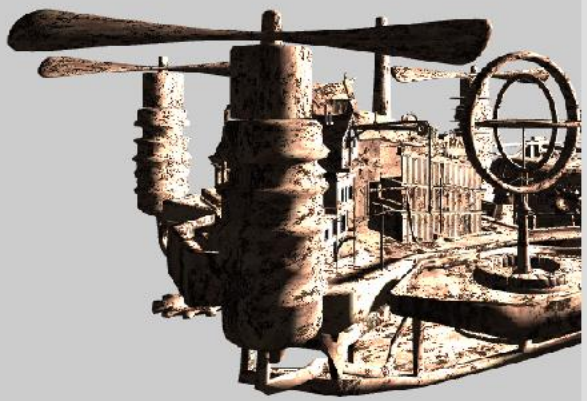
Callstack

Launch Application | Diagnostic Log | Resource Inspector | Texture Viewer | Pipeline State | BenchmarkDemo [PID 11196] | BenchmarkDemo [PID 10552] | BenchmarkDemo [PID 7076]

Channels RGBA R G B A | Subresource Sample Sample 0 Slice/Face | Actions

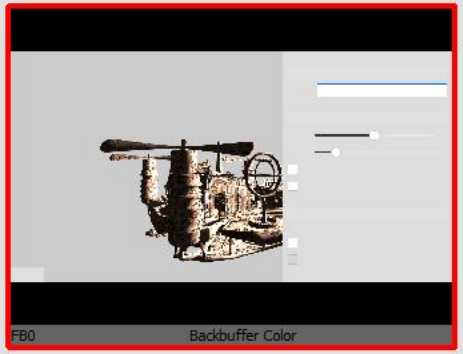
Zoom 1:1 Fit 80% | Overlay None | Range 0.00 1.00

Cur Output 0 - Backbuffer Color



Backbuffer Color - 1302x677 MS{4x 0Q} 1 mips

Inputs | Outputs



FB0 Backbuffer Color

DS Backbuffer Depth-stencil

Mesh Viewer

Controls | Sync Views | Row Offset 0 | Instance 0 | View 0

VS Input

VTX	IDX	qt_VertexPosition		qt_VertexTexCoord	
0	0	0.00	0.00	0.00	1.00

VS Output | GS/DS Output

VTX	IDX	gl_Position			
0	0	-1.00	1.00	0.99398	1.00

Preview

VS In | VS Out | GS/DS Out

Controls Arcball | Show This draw | Solid Shading None | Wireframe | Highlight Vertices

Replay Context: Local

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc loaded. No problems detected.

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc - RenderDoc v1.7

File Window Tools Help

Event Browser

Controls

EID	Name
148	glDrawElements(4800)
187	glDrawElements(152730)
194	glDrawElements(50799)
201	glDrawElements(4926)
208	glDrawElements(11952)
215	glDrawElements(42480)
222	glDrawElements(5208)
229	glDrawElements(95460)
236	glDrawElements(9498)
275	glDrawElements(4800)
314	glDrawElements(71952)
353	glDrawElements(4800)
358	glBlitFramebuffer(**Framebuffer 3309** , **Framebuffer 3310**)
522-1381	Colour Pass #2 (1 Targets + Depth)
522	glClear(Color = <0.800000, 0.800000, 0.800000, 1.000000>, Depth ...
544	glDrawElements(768)
555	glDrawElements(4)
567	glDrawElements(26)
575	glDrawElements(4)
590	glDrawElements(4)
599	glDrawElements(16)
627	glDrawElements(90)
651	glDrawElements(228)
675	glDrawElements(84)
689	glDrawElements(16)
717	glDrawElements(72)
744	glDrawElements(96)
772	glDrawElements(54)
799	glDrawElements(6)
827	glDrawElements(72)
841	glDrawElements(4)
871	glDrawElements(90)

API Inspector

EID	Event
628	glBlendColor
629	glUniformMatrix4fv
630	glUniform2fv
631	glUniform4fv
632	glUniform1f
633	glUniform1f
634	glUniform1f
635	glUniform4fv
636	glUniformMatrix4fv
637	glUniform2fv
638	glUniform4fv

Callstack

Launch Application | Diagnostic Log | Resource Inspector | Texture Viewer | Pipeline State | BenchmarkDemo [PID 11196] | BenchmarkDemo [PID 10552] | BenchmarkDemo [PID 7076]

Channels RGBA R G B A Subresource Sample Sample 0 Slice/Face Actions

Zoom 1:1 Fit 80% Overlay None Range 0.00 1.00

Cur Output 0 - Backbuffer Color

60 FPS (16.669ms)
Sync: 0.014ms
Render: 0.340ms

Inputs Outputs

FB0 Backbuffer Color

DS Backbuffer Depth-stencil

Backbuffer Color - 1302x677 MS{4x 0Q} 1 mips

Mesh Viewer

Controls Sync Views Row Offset 0 Instance 0 View 0

VS Input

VTX	IDX	vCoord		tCoord	
0	0	4.59787	31.29905	271.00	2.00

VS Output GS/DS Output

VTX	IDX	gl_Position			
0	0	-0.99294	0.90754	0.9759	1.00

Preview

VS In VS Out GS/DS Out

Controls Arcball Show This draw Solid Shading None Wireframe Highlight Vertices

Replay Context: Local

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc loaded. No problems detected.

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc - RenderDoc v1.7

File Window Tools Help

Event Browser

Controls

EID	Name
651	glDrawElements(228)
675	glDrawElements(84)
689	glDrawElements(16)
717	glDrawElements(72)
744	glDrawElements(96)
772	glDrawElements(54)
799	glDrawElements(6)
827	glDrawElements(72)
841	glDrawElements(4)
871	glDrawElements(90)
899	glDrawElements(138)
923	glDrawElements(108)
947	glDrawElements(102)
961	glDrawElements(4)
991	glDrawElements(120)
1019	glDrawElements(228)
1043	glDrawElements(96)
1067	glDrawElements(36)
1081	glDrawElements(4)
1111	glDrawElements(108)
1131	glDrawElements(4)
1161	glDrawElements(108)
1189	glDrawElements(108)
1213	glDrawElements(84)
1227	glDrawElements(10)
1257	glDrawElements(126)
1285	glDrawElements(90)
1299	glDrawElements(4)
1329	glDrawElements(144)
1357	glDrawElements(84)
1381	glDrawElements(1446)
1383	SwapBuffers(**Backbuffer Color**)

API Inspector

EID	Event
1358	glBlendColor
1359	glUniformMatrix4fv
1360	glUniform2fv
1361	glUniform4fv
1362	glUniform1f
1363	glUniform1f
1364	glUniform1f
1365	glUniform4fv
1366	glUniformMatrix4fv
1367	glUniform2fv
1368	glUniform4fv

Callstack

Launch Application | Diagnostic Log | Resource Inspector | Texture Viewer | Pipeline State | BenchmarkDemo [PID 11196] | BenchmarkDemo [PID 10552] | BenchmarkDemo [PID 7076]

Channels RGBA R G B A | Subresource Sample Sample 0 Slice/Face | Actions

Zoom 1:1 Fit 80% | Overlay None | Range 0.00 1.00

Cur Output 0 - Backbuffer Color

60 FPS (16.669ms)
Sync: 0.014ms
Render: 0.340ms

Model Options
Model: High (100k)
# of models 1
Light Options
Light Type: Point
Brightness 500
# of lights 1
Shadows
IBL lighting
Render Mode Options
Render Mode: Offscreen
Environment Options
Ambient Occlusion
Skybox
Anti-Aliasing Options

Start Measuring

Backbuffer Color - 1302x677 MS{4x 0Q} 1 mips

Inputs | Outputs

FB0 Backbuffer Color

DS Backbuffer Depth-stencil

Mesh Viewer

Controls Sync Views Row Offset 0 Instance 0 View 0

VS Input

VTX	IDX	vCoord		tCoord	
0	0	1039.01782	15.43967	954.00	2.00

VS Output | GS/DS Output

VTX	IDX	gl_Position			
0	0	1.05994	0.95439	0.57831	1.00

Preview

VS In | VS Out | GS/DS Out

Controls Arcball | Show This draw | Solid Shading None | Wireframe | Highlight Vertices

Replay Context: Local

BenchmarkDemo_2020.05.04_18.37.05_frame1251.rdc loaded. No problems detected.

Most of this is in 5.15. What's cooking for 6.0 then?

- Can the scene graphs, the renderers, some APIs be made more unified?
  - Even though on the QML API level it is quite nice already, see QQuickItem-in-3D-node example earlier.
  - Investigations on-going in many different areas.
  - Not going to cover this today.

- **What about Vulkan, Metal, Direct3D, WebGPU, etc.?**
  - And on a related node: **shading languages**?
  - Not just an internal implementation detail:
    - Qt Quick ShaderEffect, custom Qt Quick materials, custom 3D materials, post-processing effects all consume application-provided shader code.
    - Some Qt users integrate their own 3D rendering code with Qt Quick and co. (and that code is today OpenGL, tomorrow perhaps Vulkan, Metal, etc.)

Qt Quick (3D) on Vulkan, Metal, D3D11, and OpenGL

- Qt 5.14 and 5.15 ship with a preview of Qt Quick's new graphics API independent rendering engine.
  - Can be opted in via environment variables or C++ APIs.
- No more direct OpenGL and GLSL / GLSL ES shader code sprinkled all around.
- Graphics API abstraction is provided by the QRhi class and its backends.
  - part of the "Qt RHI" (Qt Rendering Hardware Interface) component living in Qt Gui module

```
c:/work/qtbase_dev/src/gui/rhi:
total used in directory 1377 available 644192552
drwxrwxrwx  1 lagoc lagoc  12288 05-04 10:35 ..
drwxrwxrwx  1 lagoc lagoc  12288 05-04 10:35 .
-rw-rw-rw-  1 lagoc lagoc   7482 03-12 11:25 cs_tdr_p.h
-rw-rw-rw-  1 lagoc lagoc 221699 05-02 13:08 qrhi.cpp
-rw-rw-rw-  1 lagoc lagoc 170875 05-04 10:35 qrhid3d11.cpp
-rw-rw-rw-  1 lagoc lagoc   2556 03-12 11:25 qrhid3d11_p.h
-rw-rw-rw-  1 lagoc lagoc  27539 05-02 13:08 qrhid3d11_p_p.h
-rw-rw-rw-  1 lagoc lagoc 171397 05-02 13:08 qrhigles2.cpp
-rw-rw-rw-  1 lagoc lagoc   2784 03-12 11:25 qrhigles2_p.h
-rw-rw-rw-  1 lagoc lagoc  33280 05-02 13:08 qrhigles2_p_p.h
-rw-rw-rw-  1 lagoc lagoc 149896 05-02 13:08 qrhimetal.mm
-rw-rw-rw-  1 lagoc lagoc   2649 03-12 11:25 qrhimetal_p.h
-rw-rw-rw-  1 lagoc lagoc  17613 05-02 13:08 qrhimetal_p_p.h
-rw-rw-rw-  1 lagoc lagoc  26182 05-02 13:08 qrhinull.cpp
-rw-rw-rw-  1 lagoc lagoc   2273 03-12 11:25 qrhinull_p.h
-rw-rw-rw-  1 lagoc lagoc  12159 05-02 13:08 qrhinull_p_p.h
-rw-rw-rw-  1 lagoc lagoc  51918 05-02 13:08 qrhi_p.h
-rw-rw-rw-  1 lagoc lagoc  20239 05-02 13:08 qrhi_p_p.h
-rw-rw-rw-  1 lagoc lagoc  22675 03-12 11:25 qrhiprofiler.cpp
-rw-rw-rw-  1 lagoc lagoc   3679 03-12 11:25 qrhiprofiler_p.h
-rw-rw-rw-  1 lagoc lagoc   4865 03-12 11:25 qrhiprofiler_p_p.h
-rw-rw-rw-  1 lagoc lagoc 287634 05-04 10:35 qrhivulkan.cpp
-rw-rw-rw-  1 lagoc lagoc   3217 03-12 11:25 qrhivulkanext_p.h
-rw-rw-rw-  1 lagoc lagoc   2989 03-12 11:25 qrhivulkan_p.h
-rw-rw-rw-  1 lagoc lagoc  37586 05-02 13:08 qrhivulkan_p_p.h
-rw-rw-rw-  1 lagoc lagoc  22041 05-04 10:35 qshader.cpp
-rw-rw-rw-  1 lagoc lagoc  54163 05-04 10:35 qshaderdescription.cpp
-rw-rw-rw-  1 lagoc lagoc  10298 03-12 11:25 qshaderdescription_p.h
-rw-rw-rw-  1 lagoc lagoc   3794 03-12 11:25 qshaderdescription_p_p.h
-rw-rw-rw-  1 lagoc lagoc   7382 04-27 12:15 qshader_p.h
-rw-rw-rw-  1 lagoc lagoc   3230 03-12 11:25 qshader_p_p.h
-rw-rw-rw-  1 lagoc lagoc   1159 03-12 11:25 rhi.pri
```

Qt Rendering Hardware Interface

- QRhi and related structs is the main interface
  - This is what Qt Quick and Quick 3D will use instead of QOpenGLContext, the QOpenGL wrappers, and calling OpenGL functions directly.
- Has backends for
  - Vulkan (1.0) (shaders: SPIR-V 1.0)
  - D3D11 (11.1) (shaders: HLSL Shader Model 5.0)
  - Metal (1) (shaders: MSL 1.2 (or 2.0 for some things))
  - OpenGL (ES) 2/3 (needs 4.3/ES3.1 for compute) (shaders: various GLSL versions)
- This is not a 100% Vulkan/Metal/whatever wrapper.
  - To unleash the full, unrestricted feature set of Vulkan/Metal/whatever, use Vulkan/Metal/whatever directly.
- Private API in Qt 6.0.
  - To be decided if/when/how this changes in 6.x and beyond.

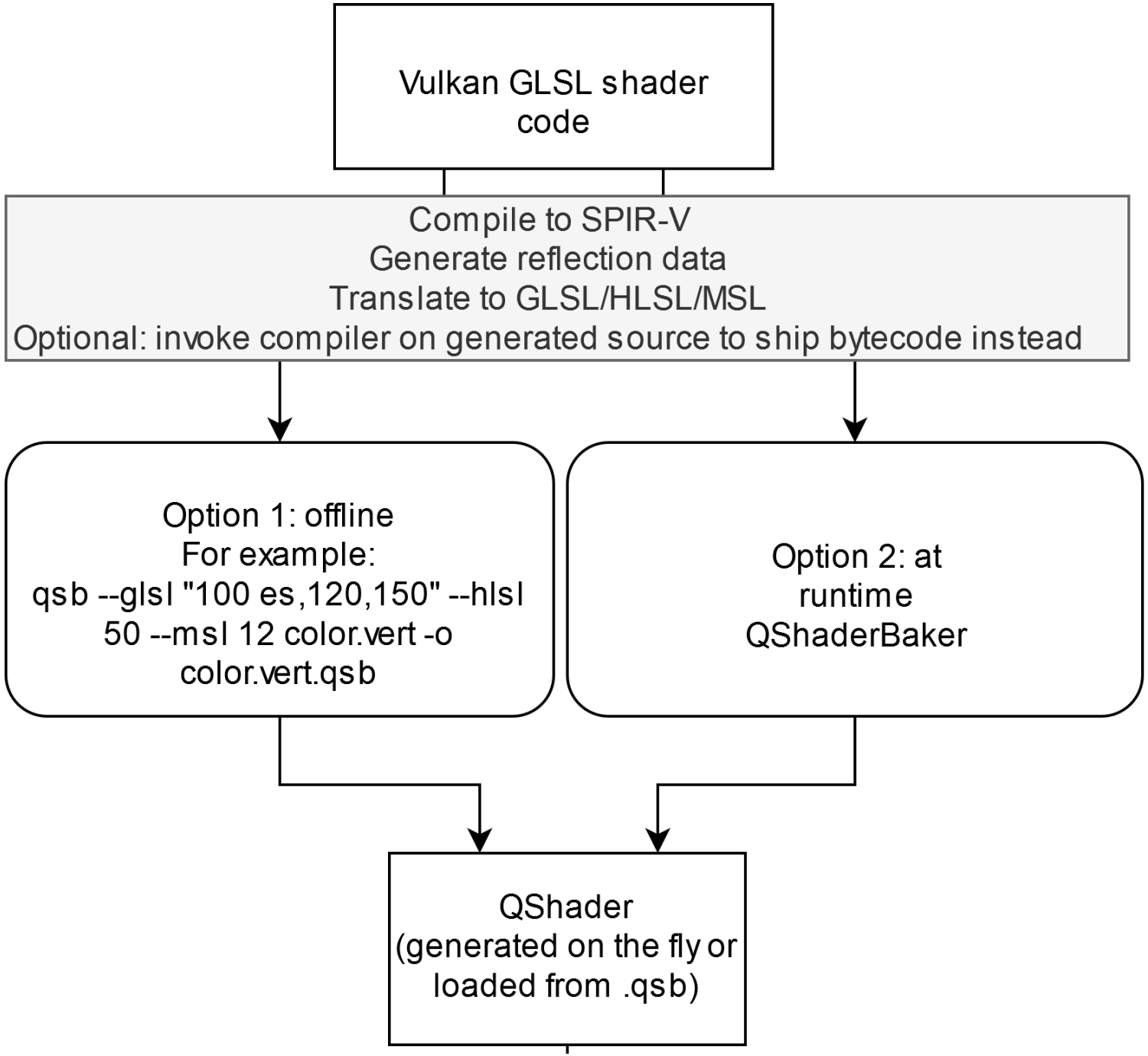
Qt Rendering Hardware Interface

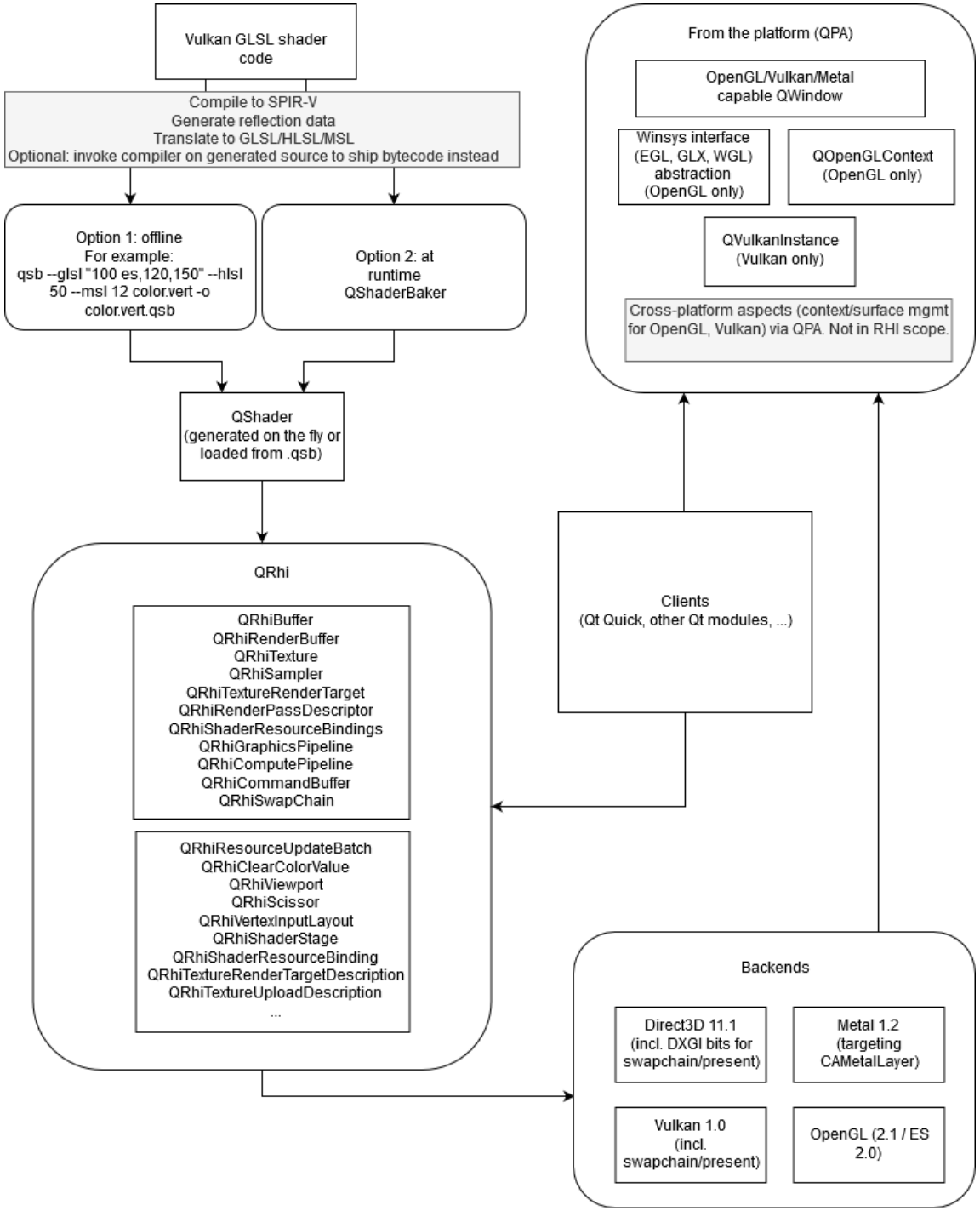
- Command buffer oriented API.
- Resource updates (upload, copy, readback) and render/compute pass recording are separate phases.
  - Think MTLBlitCommandEncoder vs. MTLRenderCommandEncoder or the Vulkan renderpass concept.
- Vertex, fragment, compute shader stages.
  - No geometry or tessellation
- Attempts to enable multiple frames in flight (reduce pipeline stalls)
  - No low-level operations such as map/unmap. Instead, one enqueues upload/update/copy operations on a *resource update batch*, that is then committed at latest when starting to record a render/compute pass.
  - Handle host visible buffers smarter than just a dumb 1:1 QRhiBuffer – VkBuffer/MTLBuffer/… mapping.
    - Some backends internally double/triple buffer such native objects, transparently to the API client.
  - No headache with resources being still in-use: a QRhi* C++ object can be destroyed safely once a frame has been submitted, even if the underlying native resource is still in use by an in-flight frame.
- No barriers / explicit synchronization. No subpasses. No multi queue (yet).

Qt Quick (3D) on Vulkan, Metal, D3D11, and OpenGL

- New shader pipeline
  - Qt Shader Tools module
    - not shipping with Qt in 5.14/5.15, but will be part of Qt 6.0
  - Vulkan-compatible GLSL as input.
  - Compile to SPIR-V (via glslang).
  - Then get reflection info and translate to GLSL/HLSL/MSL (via SPIRV-Cross)
  - Ideally done offline, or at build time at latest.
    - Not at runtime.
    - Move towards offline asset (this case, shader) conditioning as much as possible.

```
C:\>qsb
Usage: qsb [options] file
Qt Shader Baker (using QShader from Qt 6.0.0)

Options:
  -?, -h, --help               Displays help on commandline options.
  --help-all                   Displays help including Qt specific options.
  -v, --version                Displays version information.
  -b, --batchable              Also generates rewritten vertex shader for Qt
                               Quick scene graph batching.
  --zorder-loc <location>      The extra vertex input location when rewriting
                               for batching. Defaults to 7.
  -g, --glsl <versions>        Comma separated list of GLSL versions to
                               generate. (for example, "100 es,120,330")
  -l, --hlsl <versions>        Comma separated list of HLSL (Shader Model)
                               versions to generate. F.ex. 50 is 5.0, 51 is 5.1.
  -m, --msl <versions>         Comma separated list of Metal Shading Language
                               versions to generate. F.ex. 12 is 1.2, 20 is 2.0.
  -o, --output <filename>      Output file for the shader pack.
  -c, --fxc                    In combination with --hlsl invokes fxc to store
                               DXBC instead of HLSL.
  -t, --metallib               In combination with --msl builds a Metal library
                               with xcrun metal(lib) and stores that instead of
                               the source.
  -D, --define <name[=value]>  Define macro
  -p, --per-target             Enable per-target compilation. (instead of
                               source->SPIRV->targets, do source->SPIRV->target
                               separately for each target)
  -d, --dump                   Switches to dump mode. Input file is expected to
                               be a shader pack.
  -x, --extract <what>         Switches to extract mode. Input file is expected
                               to be a shader pack. Result is written to the
                               output specified by -o. Pass -b to choose the
                               batchable variant.
                               <what>=reflect|spirv.<version>|glsl.<version>|...

Arguments:
  file                         Vulkan GLSL source file to compile
```


Vulkan GLSL shader code
Compile to SPIR-V
Generate reflection data
Translate to GLSL/HLSL/MSL
Optional: invoke compiler on generated source to ship bytecode instead
Option 1: offline
For example:
qsb --glsl "100 es,120,150" --hlsl 50 --msl 12 color.vert -o color.vert.qsb
Option 2: at runtime
QShaderBaker
QShader
(generated on the fly or loaded from .qsb)

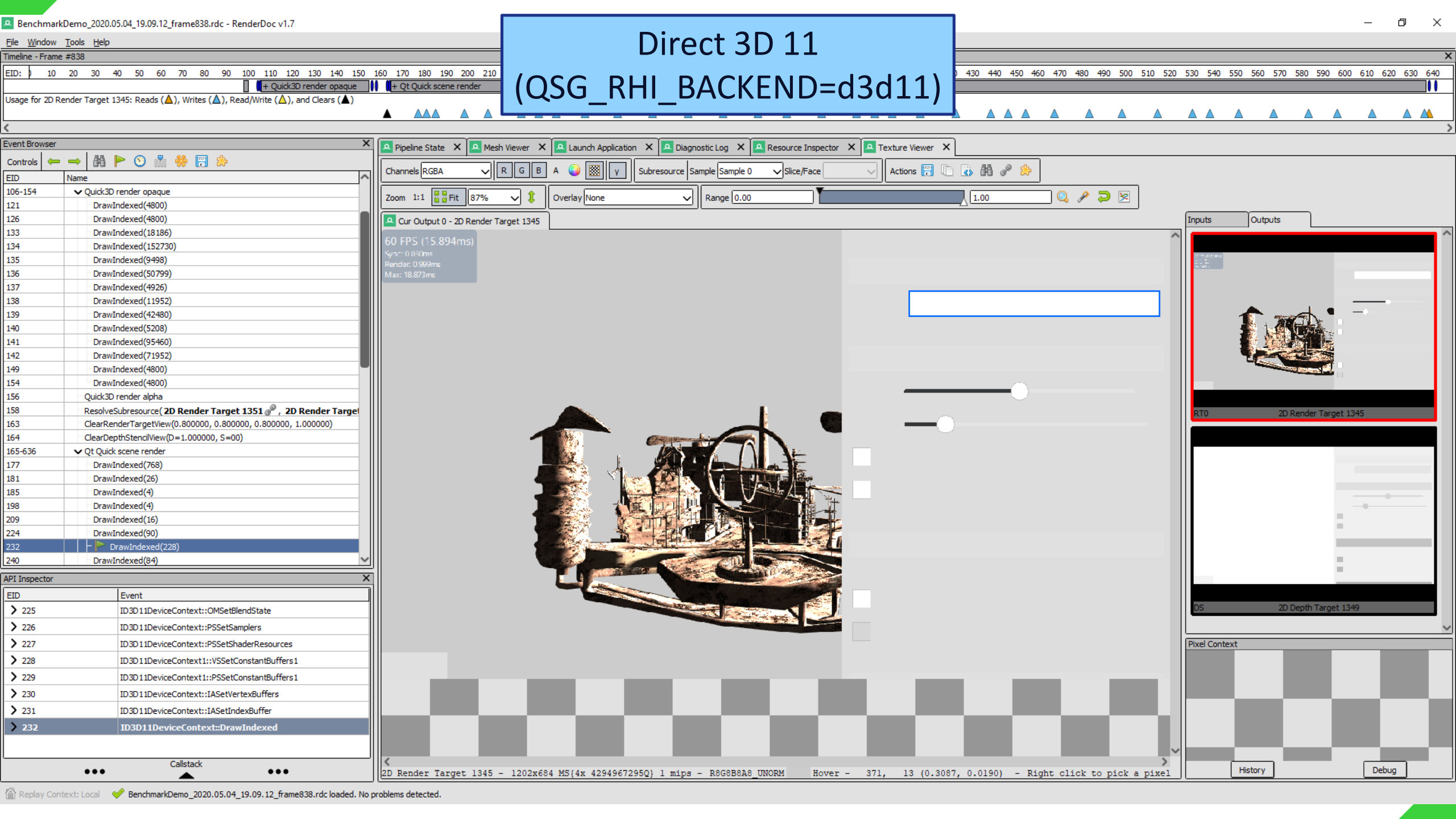

Vulkan GLSL shader code
Compile to SPIR-V
Generate reflection data
Translate to GLSL/HLSL/MSL
Optional: invoke compiler on generated source to ship bytecode instead
Option 1: offline
For example:
qsb --glsl "100 es,120,150" --hlsl 50 --msl 12 color.vert -o color.vert.qsb
Option 2: at runtime
QShaderBaker
QShader
(generated on the fly or loaded from .qsb)
From the platform (QPA)
OpenGL/Vulkan/Metal capable QWindow
Winsys interface (EGL, GLX, WGL) abstraction (OpenGL only)
QOpenGLContext (OpenGL only)
QVulkanInstance (Vulkan only)
Cross-platform aspects (context/surface mgmt for OpenGL, Vulkan) via QPA. Not in RHI scope.
Clients
(Qt Quick, other Qt modules, ...)
QRhi
QRhiBuffer
QRhiRenderBuffer
QRhiTexture
QRhiSampler
QRhiTextureRenderTarget
QRhiRenderPassDescriptor
QRhiShaderResourceBindings
QRhiGraphicsPipeline
QRhiComputePipeline
QRhiCommandBuffer
QRhiSwapChain
QRhiResourceUpdateBatch
QRhiClearColorValue
QRhiViewport
QRhiScissor
QRhiVertexInputLayout
QRhiShaderStage
QRhiShaderResourceBinding
QRhiTextureRenderTargetDescription
QRhiTextureUploadDescription
...
Backends
Direct3D 11.1 (incl. DXGI bits for swapchain/present)
Metal 1.2 (targeting CAMetalLayer)
Vulkan 1.0 (incl. swapchain/present)
OpenGL (2.1 / ES 2.0)

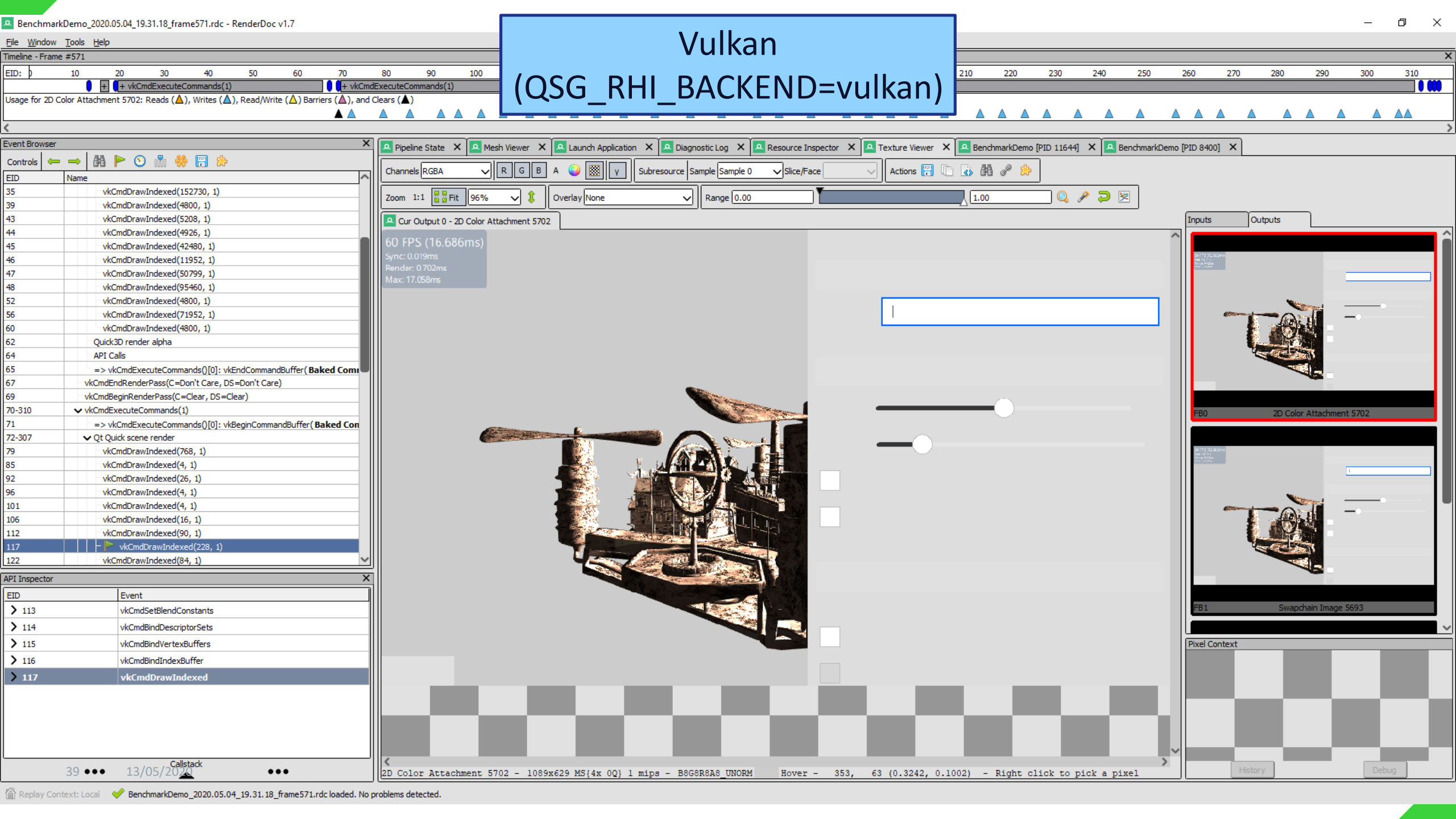
```
void HelloWindow::customInit()
{
    m_vbuf.reset(m_rhi->newBuffer(QRhiBuffer::Immutable, QRhiBuffer::VertexBuffer, sizeof(vertexData)));
    m_vbuf->build();
    m_vbufReady = false;

    m_ubuf.reset(m_rhi->newBuffer(QRhiBuffer::Dynamic, QRhiBuffer::UniformBuffer, 68));
    m_ubuf->build();

    m_srb.reset(m_rhi->newShaderResourceBindings());
    m_srb->setBindings({
            QRhiShaderResourceBinding::uniformBuffer(0,
                                                     QRhiShaderResourceBinding::VertexStage
                                                     | QRhiShaderResourceBinding::FragmentStage,
                                                     m_ubuf.get())
    });
    m_srb->build();

    m_ps.reset(m_rhi->newGraphicsPipeline());

    QRhiGraphicsPipeline::TargetBlend premulAlphaBlend;
    premulAlphaBlend.enable = true;
    m_ps->setTargetBlends({ premulAlphaBlend });

    const QShader vs = getShader(QLatin1String(":/color.vert.qsb"));
    if (!vs.isValid())
        qFatal("Failed to load shader pack (vertex)");
    const QShader fs = getShader(QLatin1String(":/color.frag.qsb"));
    if (!fs.isValid())
        qFatal("Failed to load shader pack (fragment)");

    m_ps->setShaderStages({
        { QRhiShaderStage::Vertex, vs },
        { QRhiShaderStage::Fragment, fs }
    });

    QRhiVertexInputLayout inputLayout;
    inputLayout.setBindings({
        { 5 * sizeof(float) }
    });
    inputLayout.setAttributes({
        { 0, 0, QRhiVertexInputAttribute::Float2, 0 },
        { 0, 1, QRhiVertexInputAttribute::Float3, 2 * sizeof(float) }
    });

    m_ps->setVertexInputLayout(inputLayout);
    m_ps->setShaderResourceBindings(m_srb.get());
    m_ps->setRenderPassDescriptor(m_rp.get());

    m_ps->build();
}

// called once per frame
void HelloWindow::customRender()
{
    QRhiResourceUpdateBatch *u = m_rhi->nextResourceUpdateBatch();
    if (!m_vbufReady) {
        m_vbufReady = true;
        u->uploadStaticBuffer(m_vbuf.get(), vertexData);
    }
    m_rotation += 1.0f;
    QMatrix4x4 mvp = m_proj;
    mvp.rotate(m_rotation, 0, 1, 0);
    u->updateDynamicBuffer(m_ubuf.get(), 0, 64, mvp.constData());
    m_opacity += m_opacityDir * 0.005f;
    if (m_opacity < 0.0f || m_opacity > 1.0f) {
        m_opacityDir *= -1;
        m_opacity = qBound(0.0f, m_opacity, 1.0f);
    }
    u->updateDynamicBuffer(m_ubuf.get(), 64, 4, &m_opacity);

    QRhiCommandBuffer *cb = m_sc->currentFrameCommandBuffer();
    const QSize outputSizeInPixels = m_sc->currentPixelSize();

    cb->beginPass(m_sc->currentFrameRenderTarget(), QColor::fromRgbF(0.4f, 0.7f, 0.0f, 1.0f), { 1.0f, 0 }, u);

    cb->setGraphicsPipeline(m_ps.get());
    cb->setViewport({ 0, 0, float(outputSizeInPixels.width()), float(outputSizeInPixels.height()) });
    cb->setShaderResources();

    const QRhiCommandBuffer::VertexInput vbufBinding(m_vbuf.get(), 0);
    cb->setVertexInput(0, 1, &vbufBinding);
    cb->draw(3);

    cb->endPass();
}
```

Qt Quick 3D in Qt 6.0

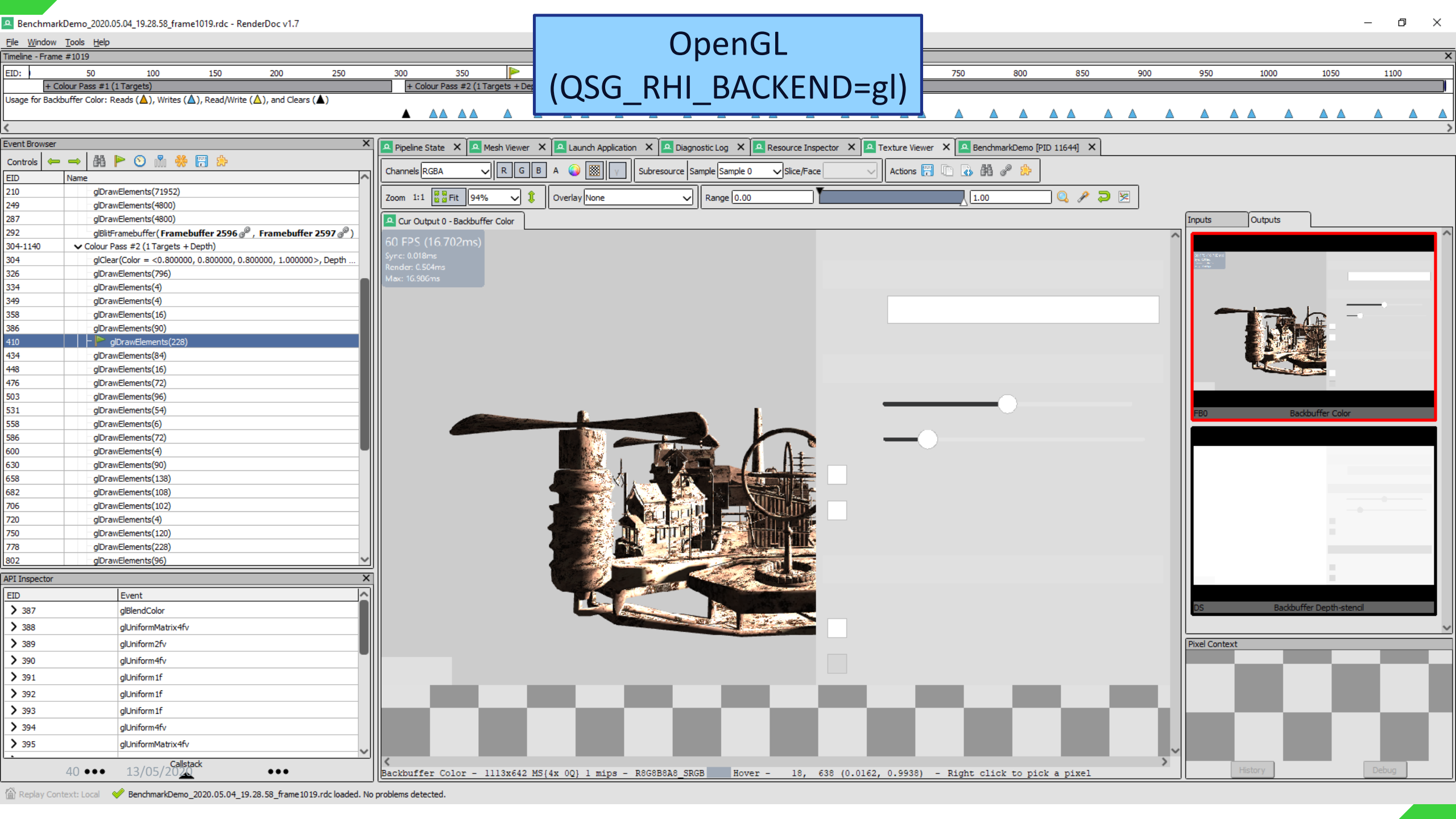
- Qt Quick 3D is fully supported as of Qt 5.15.
  - Only OpenGL, however.
- Qt 6.0 will introduce a QRhi-based renderer for Qt Quick 3D as well.
  - Currently in fairly usable state in the ‘dev’ branch already.

Direct 3D 11 (QSG_RHI_BACKEND=d3d11)

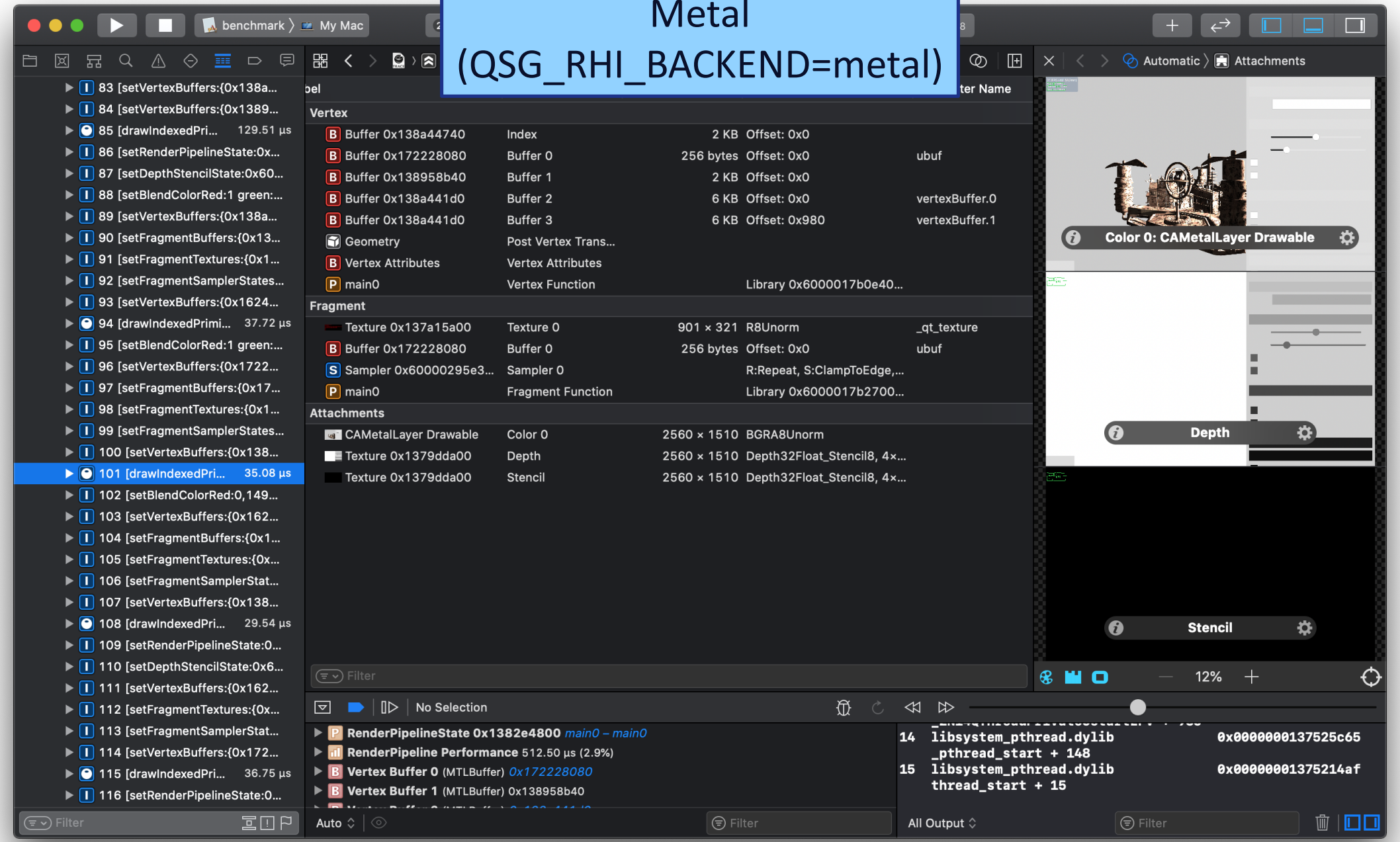



BenchmarkDemo_2020.05.04_19.31.18_frame571.rdc - RenderDoc v1.7
File Window Tools Help
Vulkan
(QSG_RHI_BACKEND=vulkan)
Timeline - Frame #571
Usage for 2D Color Attachment 5702: Reads (), Writes (), Read/Write () Barriers (), and Clears ()
Event Browser
Controls
EID Name
35 vkCmdDrawIndexed(152730, 1)
39 vkCmdDrawIndexed(4800, 1)
43 vkCmdDrawIndexed(5208, 1)
44 vkCmdDrawIndexed(4926, 1)
45 vkCmdDrawIndexed(42480, 1)
46 vkCmdDrawIndexed(11952, 1)
47 vkCmdDrawIndexed(50799, 1)
48 vkCmdDrawIndexed(95460, 1)
52 vkCmdDrawIndexed(4800, 1)
56 vkCmdDrawIndexed(71952, 1)
60 vkCmdDrawIndexed(4800, 1)
62 Quick3D render alpha
64 API Calls
65 => vkCmdExecuteCommands()[0]: vkEndCommandBuffer(Baked Comm
67 vkCmdEndRenderPass(C=Don't Care, DS=Don't Care)
69 vkCmdBeginRenderPass(C=Clear, DS=Clear)
70-310 vkCmdExecuteCommands(1)
71 => vkCmdExecuteCommands()[0]: vkBeginCommandBuffer(Baked Con
72-307 Qt Quick scene render
79 vkCmdDrawIndexed(768, 1)
85 vkCmdDrawIndexed(4, 1)
92 vkCmdDrawIndexed(26, 1)
96 vkCmdDrawIndexed(4, 1)
101 vkCmdDrawIndexed(4, 1)
106 vkCmdDrawIndexed(16, 1)
112 vkCmdDrawIndexed(90, 1)
117 vkCmdDrawIndexed(228, 1)
122 vkCmdDrawIndexed(84, 1)
API Inspector
EID Event
113 vkCmdSetBlendConstants
114 vkCmdBindDescriptorSets
115 vkCmdBindVertexBuffers
116 vkCmdBindIndexBuffer
117 vkCmdDrawIndexed
Callstack
Pipeline State
Mesh Viewer
Launch Application
Diagnostic Log
Resource Inspector
Texture Viewer
BenchmarkDemo [PID 11644]
BenchmarkDemo [PID 8400]
Channels RGBA
Subresource Sample Sample 0 Slice/Face
Actions
Zoom 1:1 Fit 96%
Overlay None
Range 0.00 1.00
Cur Output 0 - 2D Color Attachment 5702
60 FPS (16.686ms)
Sync: 0.019ms
Render: 0.702ms
Max: 17.058ms
Inputs
Outputs
FB0 2D Color Attachment 5702
FB1 Swapchain Image 5693
Pixel Context
History
Debug
2D Color Attachment 5702 - 1089x629 MS{4x 0Q} 1 mips - B8G8R8A8_UNORM
Hover - 353, 63 (0.3242, 0.1002) - Right click to pick a pixel
Replay Context: Local
BenchmarkDemo_2020.05.04_19.31.18_frame571.rdc loaded. No problems detected.

OpenGL (QSG_RHI_BACKEND=gl)



Metal (QSG_RHI_BACKEND=metal)



Qt Quick (3D) on Vulkan, Metal, D3D11, and OpenGL

Qt 6.0 will flip the switch.

› Qt Quick defaults to QRhi-based rendering.

› So does Qt Quick 3D.

› The direct OpenGL code path is removed in both.

› ANGLE is no longer distributed with Qt.

Rendering via the Qt Rendering Hardware Interface

From Qt 5.14 onwards, the default adaptation gains the option of rendering via a graphics abstraction layer, the Qt Rendering Hardware Interface (RHI), provided by the QtGui module. When enabled, no direct OpenGL calls are made. Rather, the scene graph renders by using the APIs provided by the abstraction layer, which is then translated into OpenGL, Vulkan, Metal, or Direct 3D calls. Shader handling is also unified by writing shader code once, compiling to SPIR-V, and then translating to the language appropriate for the various graphics APIs.

To enable this instead of directly using OpenGL, the following environment variables can be used:

Environment Variable	Possible Values	Description
QSG_RHI	1	Enables rendering via the RHI. The targeted graphics API is chosen based on the platform, unless overridden by QSG_RHI_BACKEND. The defaults are currently Direct3D 11 for Windows, Metal for macOS, OpenGL elsehwere.
QSG_RHI_BACKEND	vulkan, metal, opengl, d3d11	Requests the specific RHI backend.
QSG_INFO	1	Like with the OpenGL-based rendering path, setting this enables printing system information when initializing the Qt Quick scene graph. This can be very useful for troubleshooting.
QSG_RHI_DEBUG_LAYER	1	Where applicable (Vulkan, Direct3D), enables the graphics API implementation's debug and/or validation layers, if available.
QSG_RHI_PREFER_SOFTWARE_RENDERER	1	Requests choosing an adapter or physical device that uses software-based rasterization. Applicable only when the underlying API has support for enumerating adapters (for example, Direct3D or Vulkan), and is ignored otherwise.

Applications wishing to always run with a single given graphics API, can request this via C++ as well. For example, the following call made early in main(), before constructing any QQuickWindow, forces the use of Vulkan (and will fail otherwise);

```
QQuickWindow::setSceneGraphBackend(QSGRendererInterface::VulkanRhi);
```

See QSGRendererInterface::GraphicsApi. The enum values ending in Rhi are equivalent in effect to running with both QSG_RHI and QSG_RHI_BACKEND set.

Rendering via the Qt Rendering Hardware Interface

From Qt 5.14 onwards, the default adaptation gains the option of rendering via a graphics abstraction layer, the Qt Rendering Hardware Interface (RHI), provided by the QtGui module. When enabled, no direct OpenGL calls are made. Rather, the scene graph renders by using the APIs provided by the abstraction layer, which is then translated into OpenGL, Vulkan, Metal, or Direct 3D calls. Shader handling is also unified by writing shader code once, compiling to SPIR-V, and then translating to the language appropriate for the various graphics APIs.

To enable this instead of directly using OpenGL, the following environment variables can be used:

Environment Variable	Possible Values	Description
QSG_RHI	1	Enables rendering via the RHI. The targeted graphics API is chosen based on the platform, unless overridden by QSG_RHI_BACKEND. The defaults are currently Direct3D 11 for Windows, Metal for macOS, OpenGL elsehwere.
QSG_RHI_BACKEND	vulkan, metal, opengl, d3d11	Requests the specific RHI backend.
QSG_INFO	1	Like with the OpenGL-based rendering path, setting this enables printing system information when initializing the Qt Quick scene graph. This can be very useful for troubleshooting.
QSG_RHI_DEBUG_LAYER	1	Where applicable (Vulkan, Direct3D), enables the graphics API implementation's debug and/or validation layers, if available.
QSG_RHI_PREFER_SOFTWARE_RENDERER	1	Requests choosing an adapter or physical device that uses software-based rasterization. Applicable only when the underlying API has support for enumerating adapters (for example, Direct3D or Vulkan), and is ignored otherwise.

Applications wishing to always run with a single given graphics API, can request this via C++ as well. For example, the following call made early in main(), before constructing any QQuickWindow, forces the use of Vulkan (and will fail otherwise);

```
QQuickWindow::setSceneGraphBackend(QSGRendererInterface::VulkanRhi);
```

See QSGRendererInterface::GraphicsApi. The enum values ending in Rhi are equivalent in effect to running with both QSG_RHI and QSG_RHI_BACKEND set.

Consequences for applications

- Many Quick/Quick3D applications will just work
- Advanced use cases may need to do some migration.
  - typically ones that involve application-provided shader code, or involve OpenGL directly in some form
    - for example:
      - ShaderEffect, post-processing effects
      - custom scenegraph materials (QSGMaterialShader)
      - custom 3D materials
      - integrating custom (OpenGL) rendering
      - redirecting the Qt Quick output via QQuickRenderControl
    - certain features may stay tied to OpenGL, meaning they function only when the RHI backend is the OpenGL one
      - QQuickFramebufferObject
      - QQuickWidget? - TBD

ShaderEffect

```
import QtQuick 2.0

Rectangle {
    width: 200; height: 100
    Row {
        Image { id: img;
                sourceSize { width: 100; height: 100 } source: "qt-logo.png" }
        ShaderEffect {
            width: 100; height: 100
            property variant src: img
            vertexShader: "
                uniform highp mat4 qt_Matrix;
                attribute highp vec4 qt_Vertex;
                attribute highp vec2 qt_MultiTexCoord0;
                varying highp vec2 coord;
                void main() {
                    coord = qt_MultiTexCoord0;
                    gl_Position = qt_Matrix * qt_Vertex;
                }"
            fragmentShader: "
                varying highp vec2 coord;
                uniform sampler2D src;
                uniform lowp float qt_Opacity;
                void main() {
                    lowp vec4 tex = texture2D(src, coord);
                    gl_FragColor = vec4(vec3(dot(tex.rgb,
                                        vec3(0.344, 0.5, 0.156))),
                                             tex.a) * qt_Opacity;
                }"
        }
```

???

```
ShaderEffect {
    width: 160
    height: 160
    property variant source: theSource
    property real amplitude: 0.04 * wobbleSlider.value
    property real frequency: 20
    property real time: 0
    NumberAnimation on time { loops: Animation.Infinite; from: 0; to: Math.PI * 2; duration: 600 }
    fragmentShader: "qrc:/wobble.frag.qsb"
```

- The vertexShader and fragmentShader properties also accept a URL (as in local file or qrc URL) since Qt 5.8.
- The default approach in Qt 6 is to use the command-line tool (qsb) from Qt Shader Tools (or the build system, eventually) to condition shaders offline, ship the resulting .qsb shader pack file with the application, and reference that from ShaderEffect.
- It could be that some compatibility solutions will be provided just for OpenGL. TBD.
- To be evaluated if/to what extent runtime shader processing is promoted in Qt 6.0 and beyond.
- Similar questions arise for Qt Quick 3D custom materials and post-processing effects.
  - And internally for the built-in default/principled material.
  - Qt Quick 3D shader management story (for Qt 6) is under development right now (May 2020).

Qt Quick Materials – QSGMaterial, QSGMaterialShader

```
class Q_QUICK_EXPORT QSGMaterialShader
{
public:
    class Q_QUICK_EXPORT RenderState {
    public:
        inline bool isMatrixDirty() const { return m_dirty & DirtyMatrix; }
        ...
        QMatrix4x4 combinedMatrix() const;
        ...
    };

    virtual void updateState(const RenderState &state,
                             QSGMaterial *newMaterial,
                             QSGMaterial *oldMaterial);
    virtual char const *const *attributeNames() const = 0;

    virtual void initialize();
    virtual void activate();
    virtual void deactivate();

    void setShaderSourceFile(QOpenGLShader::ShaderType type, const QString &sourceFile);

    virtual const char *vertexShader() const;
    virtual const char *fragmentShader() const;
    virtual void compile();
```

QSGMaterial creates a QSGMaterialShader
-> suitable for direct OpenGL only

Qt 5.14/5.15: there is an alternative

```
class Q_QUICK_EXPORT QSGMaterialRhiShader
{
public:
    class Q_QUICK_EXPORT RenderState {
    public:
        inline bool isMatrixDirty() const { ... }
        ...
        QMatrix4x4 combinedMatrix() const;
        QByteArray *uniformData();
        ...
    };
    enum Flag {
        UpdatesGraphicsPipelineState = 0x0001
    };
    enum Stage {
        VertexStage,
        FragmentStage,
    };

    virtual bool updateUniformData(RenderState &state,
                                   QSGMaterial *newMaterial, QSGMaterial *oldMaterial);

    virtual void updateSampledImage(RenderState &state, int binding, QSGTexture **texture,
                                    QSGMaterial *newMaterial, QSGMaterial *oldMaterial);

    virtual bool updateGraphicsPipelineState(RenderState &state, GraphicsPipelineState *ps,
                                             QSGMaterial *newMaterial,QSGMaterial *oldMaterial);

    void setFlag(Flags flags, bool on = true);

    // filename is for a file containing a serialized QShader.
    void setShaderFileName(Stage stage, const QString &filename);
```

Qt 6.0: API break – the alternative becomes the one and only

```
class Q_QUICK_EXPORT QSGMaterialRhiShader
{
public:
    class Q_QUICK_EXPORT RenderState {
    public:
        inline bool isMatrixDirty() const { ... }
        ...
        QMatrix4x4 combinedMatrix() const;
        QByteArray *uniformData();
        ...
    };

    virtual bool updateUniformData(RenderState &state,
                                   QSGMaterial *newMaterial, QSGMaterial *oldMaterial);

    virtual void updateSampledImage(RenderState &state, int binding, QSGTexture **texture,
                                    QSGMaterial *newMaterial, QSGMaterial *oldMaterial);

    virtual bool updateGraphicsPipelineState(RenderState &state, GraphicsPipelineState *ps,
                                             QSGMaterial *newMaterial,QSGMaterial *oldMaterial);

    void setFlag(Flags flags, bool on = true);

    // filename is for a file containing a serialized QShader.
    void setShaderFileName(Stage stage, const QString &filename);
```

This becomes QSGMaterialShader.

OpenGL-specifics purged.

Strictly data oriented, no graphics API access.

Integrating custom Vulkan/Metal/D3D11/OpenGL rendering

- Underlay/overlay
  - QQuickWindow::beforeRendering(), afterRendering() signals have new siblings in 6.0:
    - beforeRenderPassRecording(), afterRenderPassRecording()
    - beforeFrameBegin(), afterFrameEnd()
  - Not necessarily relevant for OpenGL and D3D11, but will become important with some types of Vulkan and Metal rendering code.

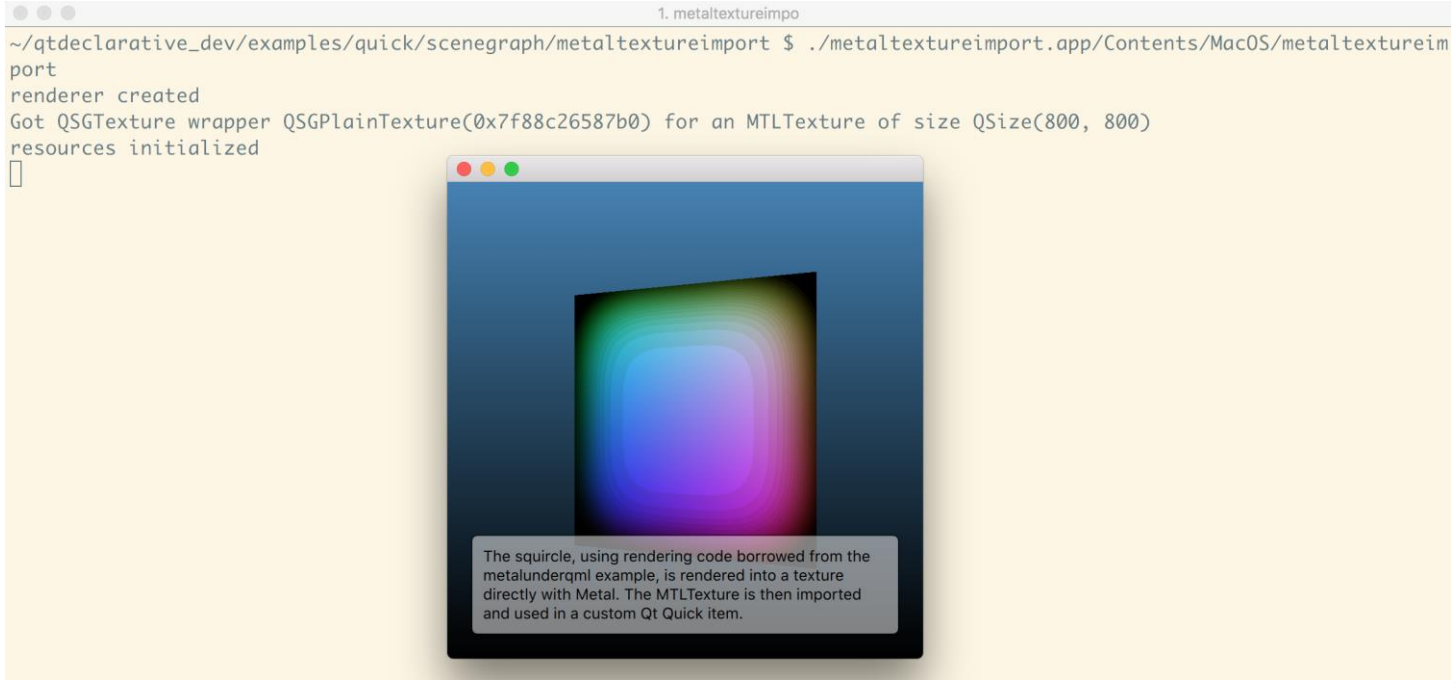
- Check under examples/quick/scenegraph
  - openglunderqml, d3d11underqml, metalunderqml, vulkanunderqml

To access the device, command buffer, etc. objects used by the scenegraph renderer, use QSGRendererInterface::getResource().

Constant	Value	Description
`QSGRendererInterface::DeviceResource`	0	The resource is a pointer to the graphics device, when applicable. For example, a `VkDevice *`, `MTLDevice *` or `ID3D11Device *`. Note that with Vulkan the returned value is a pointer to the VkDevice, not the handle itself. This is because Vulkan handles may not be pointers, and may use a different size from the architecture's pointer size so merely casting to/from `void *` is wrong.
`QSGRendererInterface::CommandQueueResource`	1	The resource is a pointer to the graphics command queue used by the scenegraph, when applicable. For example, a `VkQueue *` or `MTLCommandQueue *`. Note that with Vulkan the returned value is a pointer to the VkQueue, not the handle itself.
`QSGRendererInterface::CommandListResource`	2	The resource is a pointer to the command list or buffer used by the scenegraph, when applicable. For example, a `VkCommandBuffer *` or `MTLCommandBuffer *`. This object has limited validity, and is only valid while the scene graph is preparing the next frame. Note that with Vulkan the returned value is a pointer to the VkCommandBuffer, not the handle itself.
`QSGRendererInterface::PainterResource`	3	The resource is a pointer to the active QPainter used by the scenegraph, when running with the software backend.
`QSGRendererInterface::RhiResource`	4	The resource is a pointer to the QRhi instance used by the scenegraph, when applicable. This value was introduced in Qt 5.14.
`QSGRendererInterface::PhysicalDeviceResource`	5	The resource is a pointer to the pysical device object used by the scenegraph, when applicable. For example, a `VkPhysicalDevice *`. Note that with Vulkan the returned value is a pointer to the VkPhysicalDevice, not the handle itself. This value was introduced in Qt 5.14.
`QSGRendererInterface::OpenGLContextResource`	6	The resource is a pointer to the QOpenGLContext used by the scenegraph (on the render thread), when applicable. This value was introduced in Qt 5.14.
`QSGRendererInterface::DeviceContextResource`	7	The resource is a pointer to the device context used by the scenegraph, when applicable. For example, a `ID3D11DeviceContext *`. This value was introduced in Qt 5.14.
`QSGRendererInterface::CommandEncoderResource`	8	The resource is a pointer to the currently active render command encoder object used by the scenegraph, when applicable. For example, a `MTLRenderCommandEncoder *`. This object has limited validity, and is only valid while the scene graph is recording a render pass for the next frame. This value was introduced in Qt 5.14.
`QSGRendererInterface::VulkanInstanceResource`	9	The resource is a pointer to the QVulkanInstance used by the scenegraph, when applicable. This value was

Integrating custom Vulkan/Metal/D3D11/OpenGL rendering

- Item (as in custom QQuickItem) drawing a textured quad
  - QQuickFramebufferObject is a convenience for this, but will remain OpenGL-only.
  - Custom item with QSGTextureMaterial, or QSGSimpleTextureNode, etc. is fully possible
- QQuickWindow::createTextureFromId() is deprecated
  - Use createTextureFromNativeObject() instead
- Demonstrated by new examples in examples/quick/scenegraph
  - metaltextureimport
  - vulkantextureimport



Integrating custom Vulkan/Metal/D3D11/OpenGL rendering

- QSGRenderNode
  - Has a new virtual prepare() function. (optional)
  - Not much different otherwise.

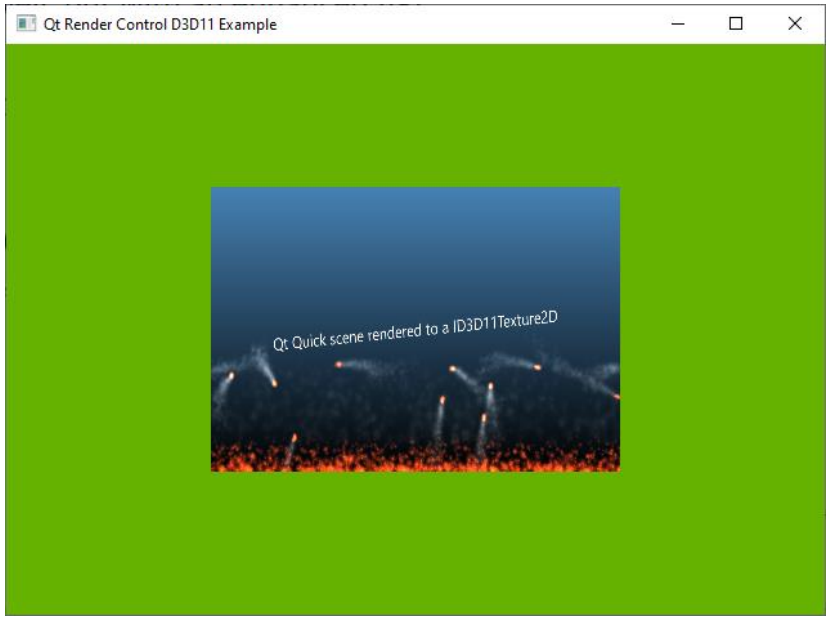
Subclassing QSGTexture

In the unlikely case of implementing a custom QSGTexture, be aware that:

- The interface changes.
  - No more virtuals like textureId(), bind(), updateBindOptions().
- New virtuals to be implemented.
  - comparisonKey(), commitTextureOperations(), etc.
  - May involve private API use (because QRhi is private).

QQuickRenderControl in Qt 6.0

- Redirect the output of Qt Quick, typically to target an OpenGL/Vulkan/Metal/… texture.
  - Then read it back (to generate image sequences), use it in an external 3D engine, VR, …
- Fully supported in 6.0 as well, but with an enhanced API.
- Now used in combination with the new QQuickRenderTarget, QQuickGraphicsDevice classes.
  - Consumed by QQuickWindow::setRenderTarget() and setGraphicsDevice()
  - Qt 5's OpenGL-specific overloads of setRenderTarget() are gone
- Some minor API changes
  - initialize(QOpenGLContext*) is replaced by initialize()
  - beginFrame(), endFrame(), setSamples()



QOpenGL*

- The already deprecated QGL classes, such as QGLWidget, are all gone.
- Most of the QOpenGL* classes live in the Qt OpenGL module in Qt 6.0.
  - not in Qt Gui
  - QT += opengl
- Only essentials and QPA-specifics stay in Qt Gui.
  - QOpenGLContext, cross-platform function resolvers
- QOpenGLWidget is in its own module now
  - QT += openglwidgets

Thank you!

- *A short survey & link to re-join the live event portal*
  https://www.qt.io/thankyousurvey
- Contact Qt at www.qt.io/contact-us/
- Download Qt at www.qt.io/download/
- Find out more in some of our recent blog posts:
  - https://www.qt.io/blog/2019/08/07/technical-vision-qt-6
  - https://www.qt.io/blog/2019/08/14/introducing-qt-quick-3d-high-level-3d-api-qt-quick
  - https://www.qt.io/blog/qt-quick-on-vulkan-metal-direct3d
  - https://www.qt.io/blog/qt-quick-on-vulkan-metal-and-direct3d-part-2
  - https://www.qt.io/blog/qt-quick-on-vulkan-metal-and-direct3d-part-3

Have more questions? The Live Chat with Qt Professional Services will be open during the live session on the live event portal.