What’s new in Qt 3D

Dr Sean Harmer
Contents

• New in Qt 3D with Qt 5.9
• New in Qt 3D with Qt 5.10
• For the future
New in Qt 3D with Qt 5.9
Physics Based Rendering

- PBR rendering gives much nicer results
- Based on the physics of how light interacts with matter
- Control with intuitive properties
- See previous talks for details
- Two new materials in Qt 5.9:
  - QMetalRoughMaterial
  - QTexturedMetalRoughMaterial
- New light type:
  - QEnvironmentLight
- Also, QSkyBox
Painted Textures

• Have legacy painter code you want to use in 3D?
• Needs integration to be useable with QTexture2D
• Provided by QPaintedTextureImage:
  • Inherit from it
  • Override the paint() function
  • Use like any other texture image
  • From C++ or QML
Embed Qt Quick 2 in Qt 3D

• Do you want to embed Qt Quick 2 within 3D?
• Scene2D element is provided by the QtQuick.Scene2D module
• Takes a Qt Quick Item as child which will be your whole 2D scene
• Renders the 2D scene into a RenderTargetOutput controlled by the output property
• Resulting texture can be used by any material
• The entities property allows to declare on which entities the texture will be used
  • Necessary for mouse event handling
  • Requires PickingSettings.TrianglePicking to be set to have the triangle information
Level of Detail Support

- Scenes often contain complex objects
- Such objects are expensive to display
- Does it still make sense if they are far from the camera?
- With level of detail management, simpler objects can be displayed instead
- This feature is provided with LevelOfDetail and LevelOfDetailLoader
- Switch based upon distance from camera or projected screen size
Text: 3D Geometric

- Generating geometry out of text is done with ExtrudedTextGeometry or ExtrudedTextMesh
- Can be used like any other Geometry or GeometryRenderer
- The font and text are controlled using properties
- The depth of the extrusion is controlled with the depth property
Text: 2D Distance Field

- Based on distance fields – just like Qt Quick 2
- Provided by Text2DEntity
  - It’s an Entity as it provides geometry, material and transform
  - Just place it in your object tree
- font, color and text are controlled using properties
- Size of the surface on which the text is rendered can be controlled via width and height
Frame Graph Nodes - RenderCapture

- Allows to create “screenshots” of the scene rendering
- Allows to debug complex multi-pass rendering
  - One can save as an image one of the intermediate steps
- RenderCapture is a FrameGraphNode
- Each time a capture is needed, a call to requestCapture() is necessary
- Requests are processed asynchronously
Input Axis Accumulators

- Axis provides floating point user input
  - From mouse, keyboard, joystick etc
- We only have access to the instantaneous axis value
- Forces us to use imperative code on the main thread:
  - Typically increment a value based on the axis position
  - Need frame time delta and then integrate axis value
- AxisAccumulator does this for you without the need for main thread callbacks
- Can treat the axis value as a *velocity* or an *acceleration*
- Integration over time performed on backend and property update sent to object
New in Qt 3D with Qt 5.10
Skeletal Animations

- Allows animating parts of an Entity
  - Either organic or rigid body
- Builds on the key frame animation foundation from Qt 5.9
- New renderer types: Armature, Skeleton, SkeletonLoader, Joint
- New animation type: SkeletonMapping
- Works with animation blend trees (much fun)
- Shameless plug:

  Come see my talk tomorrow at 13:30:
  “Breathing Life Into Your Applications”
  Be there or be two triangles arranged into a square!
Shader Graphs

• Materials tend to suffer from combinatorial explosion
• Each input may be constant (uniform) or texture
• Many variations on a theme
• People are (sometimes) scared of GLSL
• Shader graphs provide higher level abstraction
• Work with the concepts
• Let node writers and engine worry about the details
Shader Graphs

worldPos

subtract

normalize

eyePos

normalize

Metal/Rough

worldNormal

baseColor

metalness

roughness

fragColor
Shader Graphs
Shader Graphs

- Private API in QtGui
  - Potential to be shared by Qt Quick 2 in the future
- One public type in Qt 3D: ShaderBuilder
- We provide default set of node prototypes
- User and we can make shader graphs
- ShaderBuilder consumes graph and generates GLSL shader program
- Already in use with PBR and Phong materials in Qt 5.10
- Initially concentrating on Fragment Shaders
- Later expanding to other shader stages: vertex, tessellation, geometry, compute.
Sprite Sheets

• Provides the ability to use an image from a tile within a texture based upon a provided index
• SpriteGrid provides an interface to tessellate a 2D texture into a regular grid
  • Control with rows and columns properties
• SpriteSheet contains SpriteItems describing your own custom texture atlas layout.
• Set the currentIndex property to switch between sprites
• QAbstractSpriteSheet calculates a texture transform matrix you can then pass to TextureMaterial.
Helpers

• QCamera::viewAll() command adjust camera position to fit whole scene into view.
• QMesh and QTextureLoader now support remote urls
• Points and lines can now be picked by ObjectPicker in addition to bounding spheres and triangles
• Optional support for SIMD instructions in the Qt 3D renderer
  • Currently supports SSE2 and AVX2
  • Configure time option --qt3d-simd-sse2 (default) --qt3d-simd-avx2
  • Will be extended to other aspects over time
Frame Graph Nodes

• Improved Layer filtering support
• Layers are now optionally recursive!
• ProximityFilter frame graph node
  • Only render entities close to another (e.g. the camera)
• FramebufferBlit frame graph node
  • Copy a rect from one RenderTarget to another on the GPU
  • Set sampling options
  • Useful for many rendering algorithms or resolving multisampled textures
For the Future
Future Work

• Better documentation
  • Task oriented help topics

• More examples
  • Need solution for large assets

• Performance

• Bug fixing

• Improve existing aspects

• VR/AR support

• Vulkan support
Future Work

• Tooling:
  • Texture compression
  • Shader graph designer
  • Animation blend tree designer
  • Debug and profiling tools

• Better feedback

• Engine introspection
Summary

• Good set of features
• Focus now on stability, performance, convenience
Thank you for listening!
Any questions?

https://www.kdab.com
Sean Harmer <sean.harmer@kdab.com>