QQuickWidget Class

QQuickWidget 类提供了用于显示 Qt Quick 用户界面的小部件。更多的...

Header:	**#include < QQuickWidget>**
CMake:	find_package(Qt6 REQUIRED COMPONENTS QuickWidgets) target_link_libraries(mytarget PRIVATE Qt6::QuickWidgets)
qmake:	QT += quickwidgets
Inherits:	QWidget

- 所有成员的列表，包括继承的成员

公共类型

enum	**ResizeMode { SizeViewToRootObject, SizeRootObjectToView }**
enum	**Status** { Null, Ready, Loading, Error }

特性

- **resizeMode** : ResizeMode
- **source** : QUrl
- **status** : const Status

公共函数

	QQuickWidget(QWidget *parent = nullptr)
	QQuickWidget(QQmlEngine **engine*, **QWidget** parent*)
	QQuickWidget(const QUrl &*source*, QWidget **parent* = nullptr)
virtual	**~QQuickWidget**() override
QQmlEngine *	**engine**() const
QList	**errors**() const

	QQuickWidget(QWidget **parent* = nullptr)
QSurfaceFormat	**format**() const
QImage	**grabFramebuffer**() const
QSize	**initialSize**() const
QQuickWindow *	**quickWindow**() const
QQuickWidget::ResizeMode	**resizeMode**() const
QQmlContext *	**rootContext**() const
QQuickItem *	**rootObject**() const
void	**setClearColor**(const QColor &*color*)
void	**setFormat**(const QSurfaceFormat &*format*)
void	**setResizeMode**(QQuickWidget::ResizeMode)
QUrl	**source**() const
QQuickWidget::Status	**status**() const

公共槽

void	**setSource(const QUrl &*url*)**

信号

void	**sceneGraphError(QQuickWindow::SceneGraphError *error*, const QString &*message*)**
void	**statusChanged**(QQuickWidget::Status *status*)

重载的函数

virtual void	**dragEnterEvent(QDragEnterEvent **e*) override**
virtual void	**dragLeaveEvent**(QDragLeaveEvent **e*) override
virtual void	**dragMoveEvent**(QDragMoveEvent **e*) override
virtual void	**dropEvent**(QDropEvent **e*) override
virtual bool	**event**(QEvent **e*) override

virtual void	**dragEnterEvent(QDragEnterEvent *e) override**
virtual void	**focusInEvent**(QFocusEvent *event) override
virtual bool	**focusNextPrevChild**(bool *next*) override
virtual void	**focusOutEvent**(QFocusEvent *event) override
virtual void	**hideEvent**(QHideEvent *) override
virtual void	**keyPressEvent**(QKeyEvent *e) override
virtual void	**keyReleaseEvent**(QKeyEvent *e) override
virtual void	**mouseDoubleClickEvent**(QMouseEvent *e) override
virtual void	**mouseMoveEvent**(QMouseEvent *e) override
virtual void	**mousePressEvent**(QMouseEvent *e) override
virtual void	**mouseReleaseEvent**(QMouseEvent *e) override
virtual void	**paintEvent**(QPaintEvent *event) override
virtual void	**showEvent**(QShowEvent *) override
virtual void	**wheelEvent**(QWheelEvent *e) override

详细说明

这是一个方便的包装QQuickWindow当给定主源文件的 URL 时，它将自动加载并显示 QML 场景。或者，您可以使用实例化您自己的对象QQmlComponent并将它们放置在手动设置的 QQuickWidget 中。

典型用法：

```
QQuickWidget *view = new QQuickWidget;
view->setSource(QUrl::fromLocalFile("myqmlfile.qml"));
view->show();
```

要接收与使用 QQuickWidget 加载和执行 QML 相关的错误，您可以连接到statusChanged() 信号和监视器QQuickWidget::Error。错误可通过QQuickWidget::errors()。

QQuickWidget 还管理视图和根对象的大小。默认情况下，resizeMode是SizeViewToRootObject，这将加载组件并将其大小调整为视图的大小。或者，resizeMode可以设置为SizeRootObjectToView这会将视图大小调整为根对象的大小。

性能考虑因素

QQuickWidget 是使用的替代方法QQuickView和QWidget::createWindowContainer()。对堆叠顺序的限制不适用，这使得 QQuickWidget 成为更灵活的替代方案，其行为更像普通的小部件。

然而，上述优点是以牺牲性能为代价的：

- 不像QQuickWindow和QQuickView，QQuickWidget 涉及至少一个针对屏幕外颜色缓冲区（通常是 2D 纹理）的附加渲染通道，然后绘制纹理四边形。这意味着负载增加，尤其是 GPU 的片段处理。
- 使用 QQuickWidget 禁用threaded render loop在所有平台上。这意味着线程渲染的一些好处，例如Animator类和垂直同步驱动的动画将不可用。

注意：避免打电话winId() 在 QQuickWidget 上。此函数会触发本机窗口的创建，从而导致性能下降并可能出现渲染故障。QQuickWidget 的全部目的是在没有单独的本机窗口的情况下渲染快速场景，因此应始终避免使其成为本机小部件。

图形API支持

QQuickWidget 可与 Qt Quick 支持的所有 3D 图形 API 以及`software`后端一起使用。然而，其他后端（例如 OpenVG）不兼容，尝试构建 QQuickWidget 将导致问题。

覆盖平台的默认图形 API 的方式与QQuickWindow和QQuickView: 要么通过调用QQuickWindow::setGraphicsApi() 早在构造第一个 QQuickWidget 之前，或者通过设置`QSG_RHI_BACKEND`环境变量。

注意：一个顶层窗口只能使用一个图形API进行渲染。例如，尝试使用 Vulkan 和QOpenGLWidget在同一顶级窗口的小部件层次结构中，将会出现问题，并且其中一个小部件将不会按预期呈现。

场景图和上下文持久性

QQuickWidget 荣誉 QQuickWindow::isPersistentSceneGraph()，意味着应用程序可以通过调用来决定 QQuickWindow::setPersistentSceneGraph() 在从返回的窗口上quickWindow() 函数 - 让场景图节点和其他 Qt Quick 场景相关资源在小部件隐藏时被释放。默认情况下启用持久性，就像QQuickWindow。

当使用 OpenGL 运行时，QQuickWindow还提供了禁用持久 OpenGL 上下文的可能性。当前，QQuickWidget 会忽略此设置，并且上下文始终是持久的。因此，隐藏小部件时 OpenGL 上下文不会被破坏。仅当小部件被销毁或小部件重新成为另一个顶级小部件的子层次结构时，上下文才会被销毁。然而，一些应用程序，特别是那些由于在 Qt Quick 场景中执行自定义 OpenGL 渲染而拥有自己的图形资源的应用程序，可能希望禁用后者，因为它们可能不准备在将 QQuickWidget 移动到另一个窗口。此类应用程序可以设置 QCoreApplication::AA_ShareOpenGLContexts 属性。有关资源初始化和清理细节的讨论，请参阅QOpenGLWidget文档。

注意： QQuickWidget 对其内部 OpenGL 上下文提供的细粒度控制比QOpenGLWidget，并且存在细微的差异，最值得注意的是，无论 QCoreApplication::AA_ShareOpenGLContexts 是否存在，禁用持久场景图都会导致窗口更改时破坏上下文。

局限性

将其他小部件放在下面并使 QQuickWidget 透明不会导致预期的结果：下面的小部件将不可见。这是因为实际上 QQuickWidget 是在所有其他常规非 OpenGL 小部件之前绘制的，因此透明类型的解决方案是不可行的。其他类型的布局，例如在 QQuickWidget 之上放置小部件，将按预期运行。

当绝对必要时，可以通过设置来克服此限制Qt::WA_AlwaysStackOnTopQQuickWidget 上的属性。但请注意，这会破坏堆叠顺序。例如，在 QQuickWidget 之上不可能有其他小部件，因此它只能在需要半透明 QQuickWidget 且其他小部件在下面可见的情况下使用。

仅当同一窗口内的 QQuickWidget 下面有其他小部件时，此限制才适用。使窗口半透明，其他应用程序和桌面在后台可见，是通过传统方式完成的：设置Qt::WA_TranslucentBackground在顶层窗口中，请求一个 Alpha 通道，并将 Qt Quick Scenegraph 的透明颜色更改为Qt::transparent通过setClearColor()。

Tab 键处理

按下该[TAB]键后，QQuickWidget 内的项目将获得焦点。如果该项目可以处理[TAB]按键，则焦点将在该项目内相应地改变，否则焦点链中的下一个小部件将获得焦点。

也可以看看Exposing Attributes of C++ Types to QML,Qt Quick Widgets Example， 和QQuickView。

会员类型文档

enum QQuickWidget::ResizeMode

该枚举指定如何调整视图大小。

持续的	价值	描述
QQuickWidget::SizeViewToRootObject	0	视图随 QML 中的根项调整大小。
QQuickWidget::SizeRootObjectToView	1	视图会自动将根项目的大小调整为视图的大小。

enum QQuickWidget::Status

指定加载状态QQuickWidget。

持续的	价值	描述
QQuickWidget::Null	0	这QQuickWidget没有源集。
QQuickWidget::Ready	1	这QQuickWidget已加载并创建 QML 组件。
QQuickWidget::Loading	2	这QQuickWidget正在加载网络数据。
QQuickWidget::Error	3	发生一个或多个错误。称呼errors() 检索错误列表。

财产文件

resizeMode : ResizeMode

确定视图是否应调整窗口内容的大小。

如果该属性设置为SizeViewToRootObject（默认），视图大小调整为 QML 中根项的大小。

如果该属性设置为SizeRootObjectToView，视图会自动将根项目的大小调整为视图的大小。

不管这个属性如何，视图的 sizeHint 都是根项的初始大小。但请注意，由于 QML 可能会动态加载，因此该大小可能会发生变化。

访问功能：

QQuickWidget::ResizeMode	resizeMode() const
void	**setResizeMode**(QQuickWidget::ResizeMode)

也可以看看initialSize()。

source : QUrl

该属性保存 QML 组件源的 URL。

确保提供的 URL 完整且正确，特别是使用QUrl::fromLocalFile() 从本地文件系统加载文件时。

注意：设置源 URL 将导致 QML 组件被实例化，即使 URL 与当前值相比没有变化。

访问功能：

QUrl	source() const
void	**setSource**(const QUrl &*url*)

[read-only]status : const Status

组件的电流status。

访问功能：

QQuickWidget::Status	status() const

通知器信号：

void	**statusChanged**(QQuickWidget::Status *status*)

void **statusChanged(QQuickWidget::Status *status*)**

成员函数文档

*[explicit]QQuickWidget::QQuickWidget(QWidget *parent = nullptr)*

使用给定的值构造一个 QQuickWidget*parent*。默认值为*parent*是 0。

*QQuickWidget::QQuickWidget(QQmlEngine **engine***, **QWidget** parent*)*

使用给定的 QML 构造一个 QQuickWidget*engine*和*parent*。

注意：在这种情况下，QQuickWidget 不拥有给定的*engine*目的; 调用者有责任销毁引擎。如果*engine*在视图之前被删除，status（） 将返回QQuickWidget::Error。

也可以看看Status,status（），和errors()。

*[explicit]QQuickWidget::QQuickWidget(const QUrl &source, QWidget *parent = nullptr)*

使用给定的 QML 构造一个 QQuickWidget*source*和*parent*。默认值为*parent*是 0。

[override virtual]QQuickWidget::~QQuickWidget()

摧毁了QQuickWidget。

*[override virtual protected]void QQuickWidget::dragEnterEvent(QDragEnterEvent *e)*

重新实现：QWidget::dragEnterEvent（QDragEnterEvent *事件）。

*[override virtual protected]void QQuickWidget::dragLeaveEvent(QDragLeaveEvent *e)*

重新实现：QWidget::dragLeaveEvent（QDragLeaveEvent *事件）。

*[override virtual protected]void QQuickWidget::dragMoveEvent(QDragMoveEvent *e)*

重新实现：QWidget::dragMoveEvent（QDragMoveEvent *事件）。

*[override virtual protected]void QQuickWidget::dropEvent(QDropEvent *e)*

重新实现：QWidget::dropEvent（QDropEvent *事件）。

*QQmlEngine *QQuickWidget::engine() const*

返回一个指向QQmlEngine用于实例化 QML 组件。

QList[QQmlError](https://doc-qt-io.translate.goog/qt-6/qqmlerror.html?_x_tr_sl=auto&_x_tr_tl=zh-CN&_x_tr_hl=zh-CN&_x_tr_pto=wapp) QQuickWidget::errors() const

返回上次编译或创建操作期间发生的错误列表。当状态不是Error，返回一个空列表。

也可以看看status。

*[override virtual protected]bool QQuickWidget::event(QEvent *e)*

重新实现：QWidget::event（QEvent *事件）。

*[override virtual protected]void QQuickWidget::focusInEvent(QFocusEvent *event)*

重新实现：QWidget::focusInEvent（QFocusEvent *事件）。

[override virtual protected]bool QQuickWidget::focusNextPrevChild(bool next)

重新实现：QWidget::focusNextPrevChild（接下来是布尔值）。

*[override virtual protected]void QQuickWidget::focusOutEvent(QFocusEvent *event)*

重新实现：QWidget::focusOutEvent（QFocusEvent *事件）。

QSurfaceFormat QQuickWidget::format() const

返回实际的表面格式。

如果小部件尚未显示，则返回请求的格式。

也可以看看setFormat()。

QImage QQuickWidget::grabFramebuffer() const

渲染一帧并将其读回到图像中。

注意：这是一个潜在昂贵的操作。

*[override virtual protected]void QQuickWidget::hideEvent(QHideEvent *)*

重新实现：QWidget::hideEvent（QHideEvent *事件）。

QSize QQuickWidget::initialSize() const

返回根对象的初始大小。

如果resizeMode是SizeRootObjectToView，根对象的大小将调整为视图的大小。此函数返回根对象在调整大小之前的大小。

*[override virtual protected]void QQuickWidget::keyPressEvent(QKeyEvent *e)*

重新实现：QWidget::keyPressEvent（QKeyEvent *事件）。

*[override virtual protected]void QQuickWidget::keyReleaseEvent(QKeyEvent *e)*

重新实现：QWidget::keyReleaseEvent（QKeyEvent *事件）。

*[override virtual protected]void QQuickWidget::mouseDoubleClickEvent(QMouseEvent *e)*

重新实现：QWidget::mouseDoubleClickEvent（QMouseEvent *事件）。

*[override virtual protected]void QQuickWidget::mouseMoveEvent(QMouseEvent *e)*

重新实现：QWidget::mouseMoveEvent（QMouseEvent *事件）。

*[override virtual protected]void QQuickWidget::mousePressEvent(QMouseEvent *e)*

重新实现：QWidget::mousePressEvent（QMouseEvent *事件）。

*[override virtual protected]void QQuickWidget::mouseReleaseEvent(QMouseEvent *e)*

重新实现：QWidget::mouseReleaseEvent（QMouseEvent *事件）。

*[override virtual protected]void QQuickWidget::paintEvent(QPaintEvent *event)*

重新实现：QWidget::paintEvent（QPaintEvent *事件）。

*QQuickWindow *QQuickWidget::quickWindow() const*

返回屏幕外QQuickWindow该小部件使用它来驱动 Qt Quick 渲染。如果您想使用，这很有用QQuickWindow当前未公开的 APIQQuickWidget，例如连接到QQuickWindow::beforeRendering() 信号以便在 Qt Quick 自己的渲染下绘制本机 OpenGL 内容。

警告：请谨慎使用该函数的返回值。特别是，切勿试图展示QQuickWindow，并在使用其他时要非常小心QWindow-仅限 API。

警告：在屏幕外窗口的生命周期内可能会被删除（并重新创建）QQuickWidget，特别是当小部件移动到另一个小部件时QQuickWindow。如果您需要知道窗户何时被更换，请连接到其destroyed（） 信号。

*QQmlContext *QQuickWidget::rootContext() const*

该函数返回上下文层次结构的根。每个 QML 组件都在一个实例中实例化QQmlContext。QQmlContext' 对于将数据传递到 QML 组件至关重要。在 QML 中，上下文按层次结构排列，并且该层次结构由QQmlEngine。

*QQuickItem *QQuickWidget::rootObject() const*

返回视图的根item。可以为空，当setSource() 还没有被调用，如果是用broken调用的话QtQuick代码或同时QtQuick内容正在创建中。

[signal]void QQuickWidget::sceneGraphError(QQuickWindow::SceneGraphError error, const QString &message)

当一个*error*发生在场景图初始化期间。

如果应用程序希望以自定义方式处理错误（例如 OpenGL 上下文创建失败），则应连接到此信号。当没有插槽连接到信号时，行为会有所不同：快速将打印*message*，或显示消息框，然后终止应用程序。

该信号将从 GUI 线程发出。

也可以看看QQuickWindow::sceneGraphError()。

void QQuickWidget::setClearColor(const QColor &color)

设置清除*color*。默认情况下，这是不透明的颜色。

为了得到半透明的QQuickWidget，调用此函数*color*设置Qt::transparent，设置Qt::WA_TranslucentBackground顶级窗口上的 widget 属性，并通过以下方式请求 alpha 通道setFormat()。

也可以看看QQuickWindow::setColor()。

void QQuickWidget::setFormat(const QSurfaceFormat &format)

设置曲面*format*用于此小部件使用的上下文和屏幕外表面。

当需要请求给定 OpenGL 版本或配置文件的上下文时，调用此函数。深度、模板和 Alpha 缓冲区的大小会自动处理，无需明确请求。

也可以看看QWindow::setFormat(),QWindow::format（），和format()。

[slot]void QQuickWidget::setSource(const QUrl &url)

将源设置为*url*，加载 QML 组件并实例化它。

确保提供的 URL 完整且正确，特别是使用QUrl::fromLocalFile() 从本地文件系统加载文件时。

使用相同的 URL 多次调用此方法将导致 QML 组件被重新实例化。

注意：属性的 Setter 函数source。

也可以看看source()。

*[override virtual protected]void QQuickWidget::showEvent(QShowEvent *)*

重新实现：QWidget::showEvent（QShowEvent *事件）。

QUrl QQuickWidget::source() const

返回源 URL（如果已设置）。

注意：属性源的 Getter 函数。

也可以看看setSource()。

[signal]void QQuickWidget::statusChanged(QQuickWidget::Status status)

当组件的电流*status*变化。

注意：属性的通知程序信号status。

*[override virtual protected]void QQuickWidget::wheelEvent(QWheelEvent *e)*

重新实现：QWidget::wheelEvent（QWheelEvent *事件）。