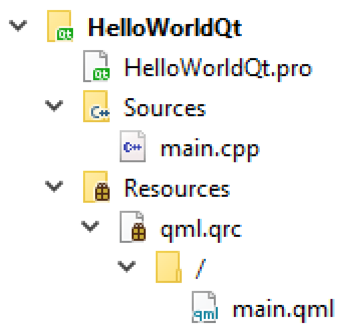
Author: dailey.dai@openthinks.com

Qt Quick

Qt Quick project



main.cpp

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>

int main(int argc, char *argv[])
{
#if defined(Q_OS_WIN)
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
#endif

    QGuiApplication app(argc, argv);

    QQmlApplicationEngine engine;
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
    if (engine.rootObjects().isEmpty())
        return -1;

    return app.exec();
}
```

This *main.cpp* defined a instance of QML engine(QQmlApplicationEngine) then use it to load *main.qml* ; at the end, start Qt main loop by *app.exec()* .

main.qml

QML include two parts:

1. import statement
2. QML object tree

```
import QtQuick 2.9
import QtQuick.Window 2.2

Window {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")
}
```

import statement

`import QtQuick 2.2` means import QtQuick modules and basic types like `Text`, `Rectangle`,`Item`,`Row`, could be used in this QML file. `import QtQuick.Window 2.1` import Window modules, and type `Window` could be used in QML file

QML object tree

```
import QtQuick 2.9
import QtQuick.Window 2.2

Window {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    MouseArea{
        anchors.fill: parent;
        onClicked: {
            Qt.quit();
        }
    }

    Text{
        text:qsTr("Hello Qt Quick App");
        anchors.centerIn: parent;
    }
}
```

`MouseArea` and `Text` are embeded in `Window`; that mean `Window` is a root and parent object, and `MouseArea` and `Text` are child objects.

default property

TODO

QQuickView

main.cpp could change as below:

```
#include <QGuiApplication>
#include <QQuickView>

int main(int argc,char *argv[])
{
 QGuiApplication app(argc,argv);
 QQuickVIew viewer;
 viewer.setResizeMode(QQuickView::SizeRootObjectToView);
 viewer.setSource(QUrl("qrc:///main.qml"));
 viewer.show();
 return app.exec();
}
```

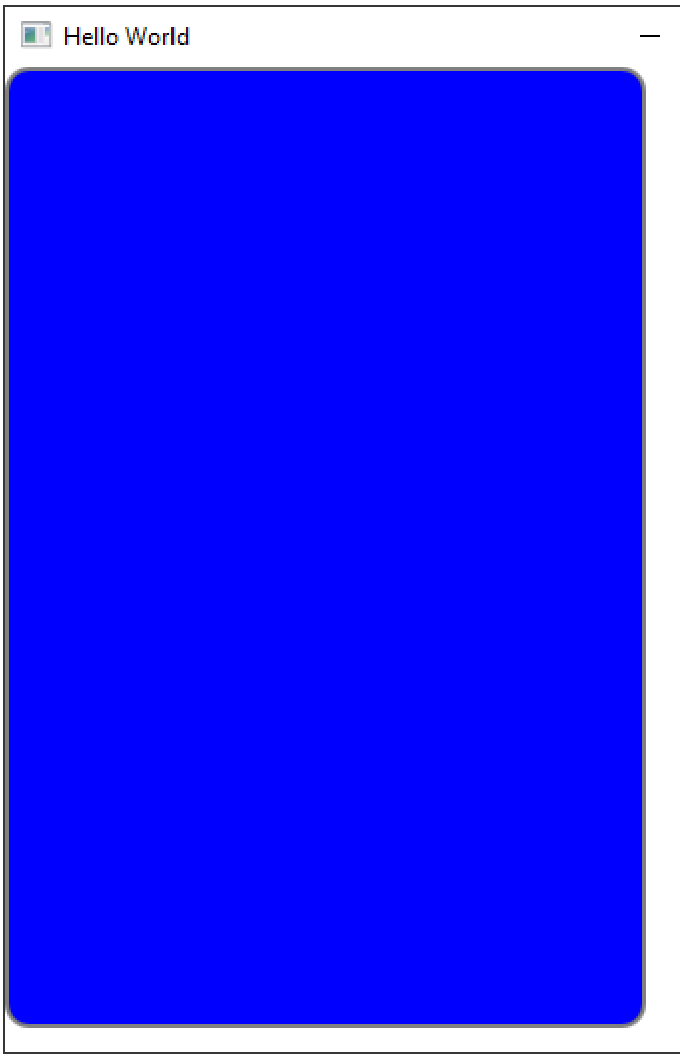
Start Qt Quick App mode:

1. QQmlApplicationEngine with Window
2. QQuickView with Item - could not control window size,icon,title

Basic Qt Quick Elements

Rectangle

```
import QtQuick 2.9
import QtQuick.Window 2.2
Window {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    Rectangle{
        width: 320;
        height: 480;
        color: "blue";
        border.color: "#808080";
        border.width: 2;
        radius: 12;
    }
}
```



Color

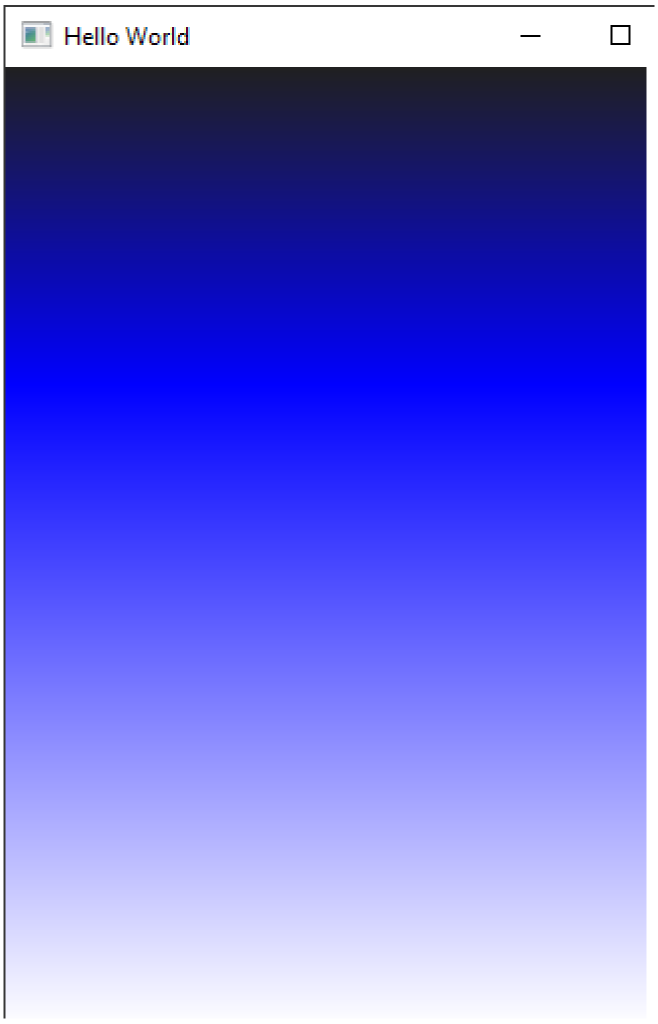
In QML color value could be below:

1. color name - blue,red
2. #RRGGBB
3. #AARRGGBB
4. Qt.rgba(0.8,0.6,0.4,1.0)

```
Rectangle{
 width:100;
 height:100;
 color:"red";
 //color:"#00AA00";
 //color:"#800000B0";
 //color:Qt.rgba(0.8,0.6,0.4,1.0);
 Component.onCompleted:console.log(color.r,color.g,color.b,color.a)
}
```

Gradient

```
import QtQuick 2.9
import QtQuick.Window 2.2

Window {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    Rectangle{
        width: 320;
        height: 480;
        gradient: Gradient{
            GradientStop{position: 0.0;color: "#202020";}
            GradientStop{position: 0.33;color: "blue";}
            GradientStop{position: 1.0;color: "#FFFFFF";}
        }
    }
}
```

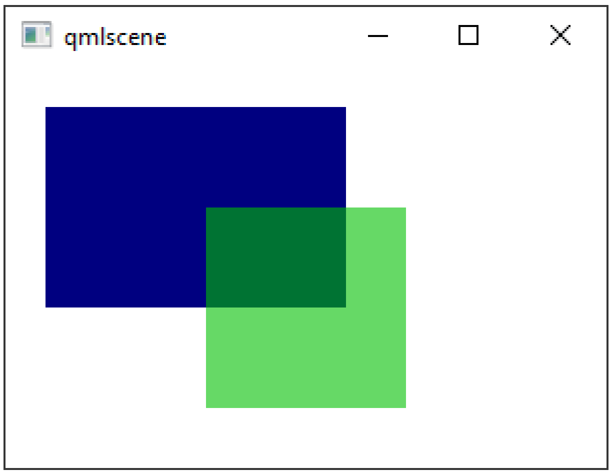


Item

Item is all visible elements base type/class in Qt Quick. It contains some common properties:

```
x,y,z,width,height,anchors,opacity,clip,scale,smooth,enabled,visible,state,children,transit
ions
```

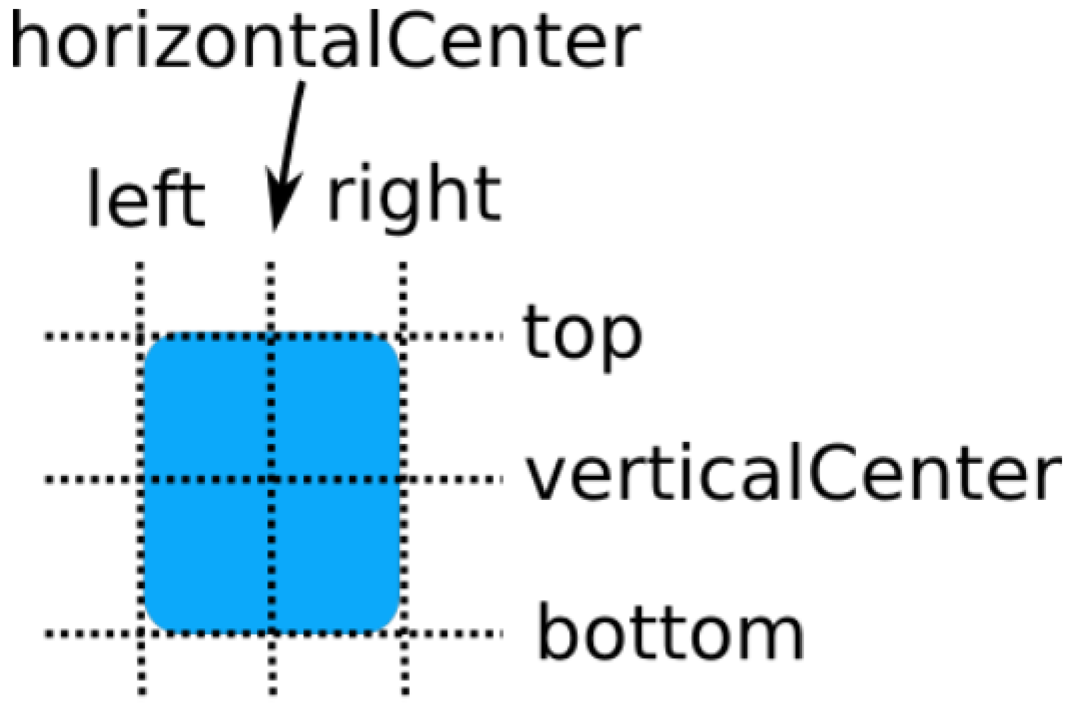
```
import QtQuick 2.0

Item {
 width:300;
 height:200;

 Rectangle{
   x:20;y:20;width:150;height:100;color:"#000080";z:0.5;
 }

 Rectangle{
   x:100;y:70;width:100;height:100;color:"#00c000";z:1;opacity: 0.6;
 }
}
```



anchors

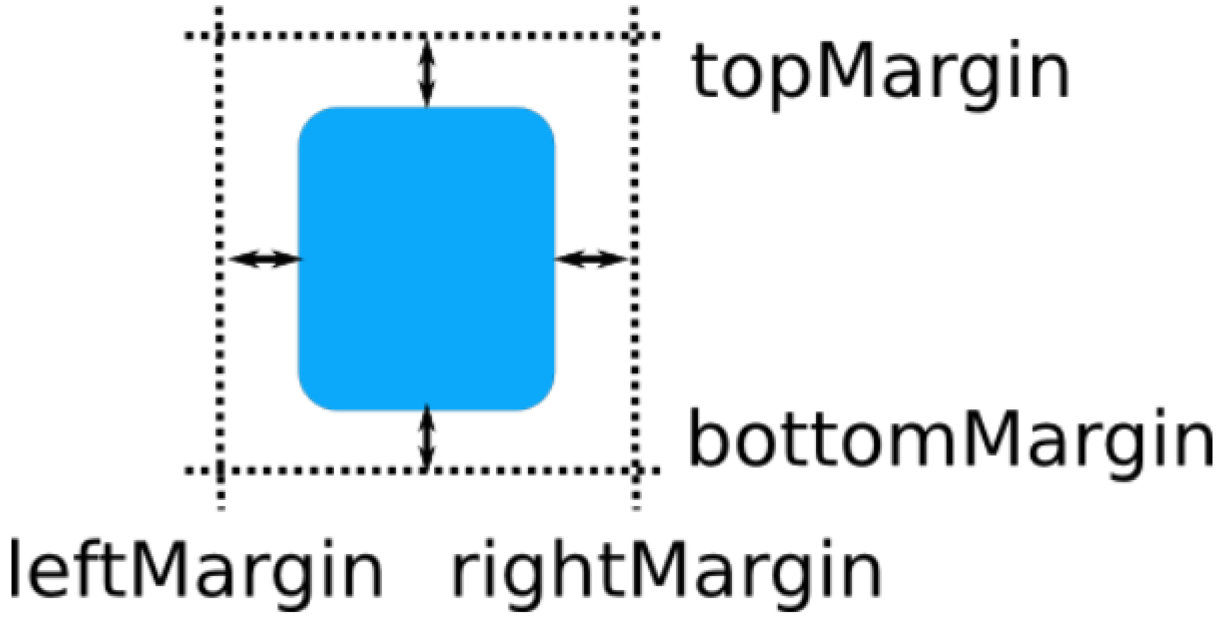
All `Item` include 7 invisible anchor lines:

- left
- horizontalCenter
- top
- bottom
- right
- verticalCenter
- baseline



Also could be set margin for anchors:

- topMargin
- bottomMargin
- leftMargin
- rightMargin



Keys

All visual primitives support key handling via the Keys attached property. Keys can be handled via the onPressed and onReleased signal properties. The signal properties have a KeyEvent parameter, named event which contains details of the event. If a key is handled event.accepted should be set to true to prevent the event from propagating up the item hierarchy.

```
Item {
     anchors.fill: parent
     focus: true
     Keys.onPressed: {
         if (event.key == Qt.Key_Left) {
             console.log("move left");
             event.accepted = true;
         }
     }
 }

 Item {
     anchors.fill: parent
     focus: true
     Keys.onLeftPressed: console.log("move left")
 }
```

Text

Text items can display both plain and rich text. For example, red text with a specific font and size can be defined like this:

```
Text {
      text: "Hello World!"
      font.family: "Helvetica"
      font.pointSize: 24
      color: "red"
  }
```

Rich text is defined using HTML-style markup:

```
Text {
     text: "<b>Hello</b> <i>World!</i>"
 }
```

`Text` include some style for text style:

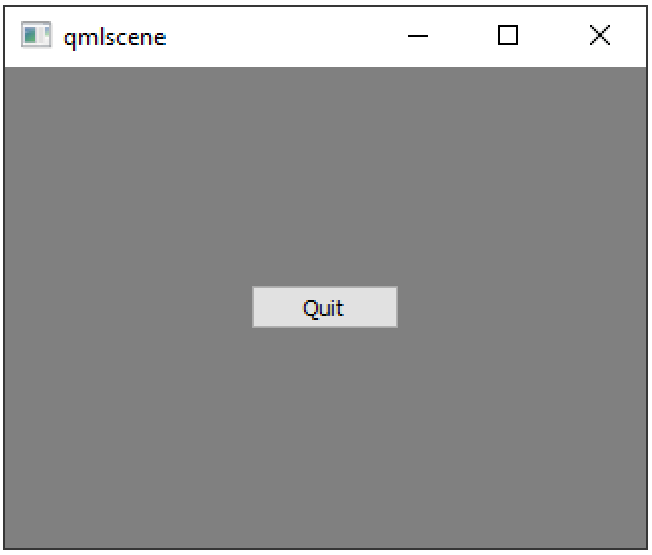
- Text.Normal
- Text.Outline
- Text.Raised
- Text.Sunken

```
  Row {
      Text { font.pointSize: 24; text: "Normal" }
      Text { font.pointSize: 24; text: "Raised"; style: Text.Raised; styleColor: "#AAAAAA"
}
      Text { font.pointSize: 24; text: "Outline";style: Text.Outline; styleColor: "red" }
      Text { font.pointSize: 24; text: "Sunken"; style: Text.Sunken; styleColor: "#AAAAAA"
}
  }
```

Button

The push button is perhaps the most commonly used widget in any graphical user interface. Pushing (or clicking) a button commands the computer to perform some action or answer a question. Common examples of buttons are OK, Apply, Cancel, Close, Yes, No, and Help buttons.

```
import QtQuick 2.2
import QtQuick.Controls 1.2

Rectangle{
    width:320;
    height: 240;
    color: "gray";

    Button{
        text:"Quit";
        anchors.centerIn: parent;
        onClicked: {
            Qt.quit();
        }
    }
}
```



ButtonStyle

You can create a custom button by replacing the "background" delegate of the ButtonStyle with a custom design.

```
import QtQuick 2.2
import QtQuick.Controls 1.2
import QtQuick.Controls.Styles 1.2

Rectangle {
    width:300;
    height:200;
    Button{
        text:"Quit";
        anchors.centerIn: parent;
        style: ButtonStyle{
            background: Rectangle{
                implicitWidth: 70; // Defines the natural width or height of the Item if no
width or height is specified.
                implicitHeight: 25;
                border.width: control.pressed?2:1;
                border.color: (control.hovered || control.pressed)?"green":"#888888";
            }
        }
        onClicked:{
            Qt.quit();
        }
    }
}
```

Image

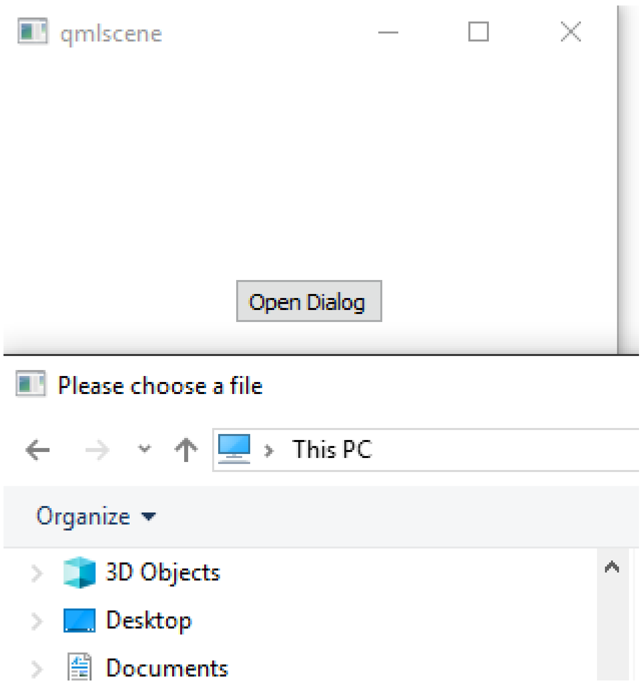
The Image type displays an image.Support JPG, PNG, BMP, GIF, SVG

```
import QtQuick 2.2
import QtQuick.Controls 1.2;

Rectangle{
width: 480;
height: 320;
color: "#121212";

    BusyIndicator{
        id:busy;
        running: false;
        anchors.centerIn: parent;
        z:2;
    }
    Text{
        id:stateLabel;
        visible: false;
        anchors.centerIn: parent;
        z:3;
    }
    Image{
        id:imageViewer;
        asynchronous: true;
        cache:false;
        anchors.fill: parent;
        fillMode:Image.PreserveAspectFit;
        onStatusChanged:{
            if(imageViewer.status===Image.Loading){
                busy.running=true;
                stateLabel.visible=false;
            }else if(imageViewer.status=== Image.Ready){
                busy.running=false;
            }else if(imageViewer.status=== Image.Error){
                busy.running=false;
                stateLabel.visible=true;
                stateLabel.text="Error";
            }
        }
    }

    Component.onCompleted: {
        imageViewer.source="https://www.v2ex.com/static/img/v2ex@2x.png";
    }
}
```

FileDialog

FileDialog provides a basic file chooser: it allows the user to select existing files and/or directories, or create new filenames. The dialog is initially invisible. You need to set the properties as desired first, then set visible to true or call open().

```
import QtQuick 2.2
import QtQuick.Controls 1.2
import QtQuick.Dialogs 1.0
Item{
  Button{
    text:"Open Dialog";
    anchors.centerIn: parent;
    onClicked: {
       fileDialog.open();
    }
  }

  FileDialog {
      id: fileDialog
      title: "Please choose a file"
      folder: shortcuts.home
      onAccepted: {
          console.log("You chose: " + fileDialog.fileUrls)
          Qt.quit()
      }
      onRejected: {
            console.log("Canceled")
            Qt.quit()
        }
    }
}
```


qmlscene
Open Dialog
Please choose a file
This PC
Organize
3D Objects
Desktop
Documents