Разбор QML-программ

Пример 1. «clocks»

QT += qml quick

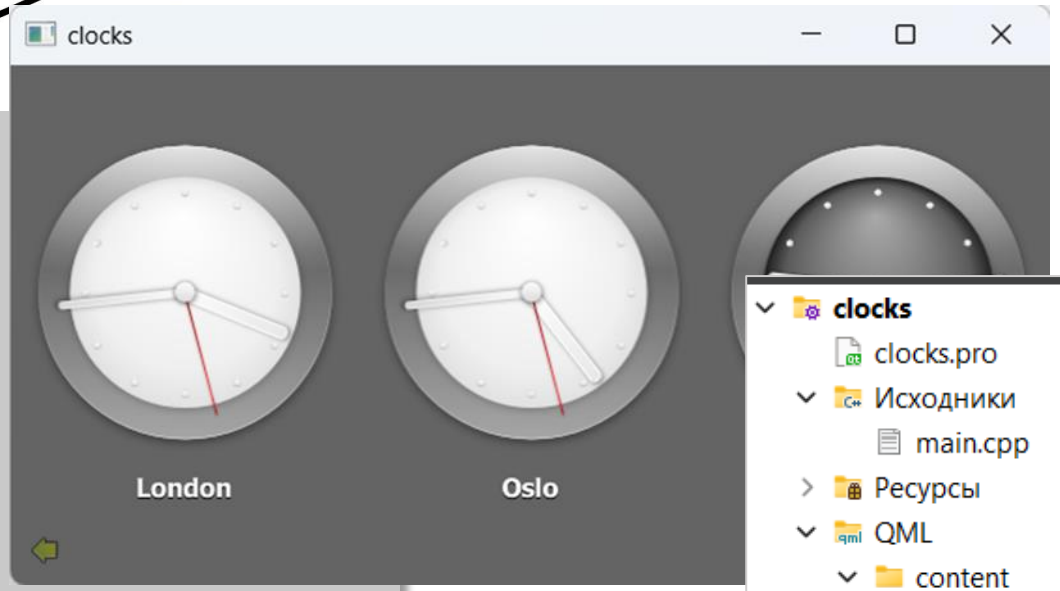
main.cpp

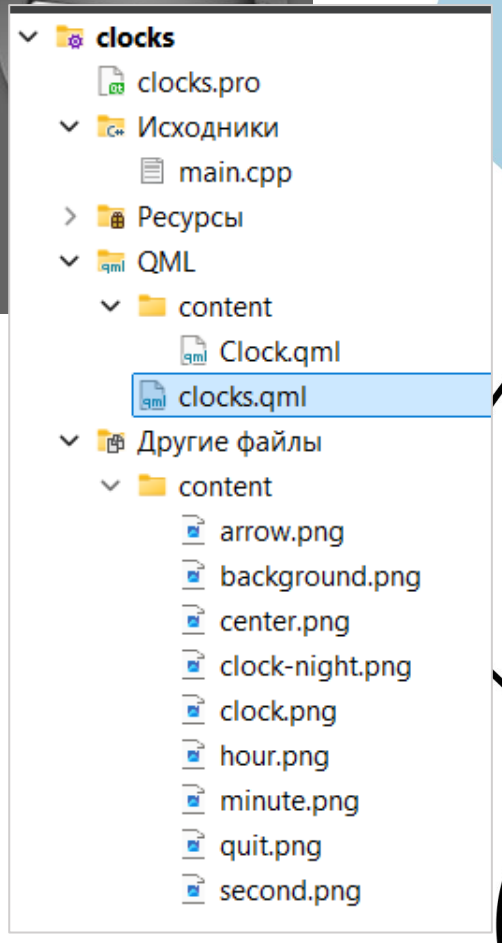
```
#include <QQmlEngine>
#include <QQmlFileSelector>
#include <QQuickView>

int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
    QCoreApplication::setOrganizationName("QtExamples");

    QGuiApplication app(argc, argv);

    QQuickView view;
    view.connect(view.engine(), &QQmlEngine::quit, &app, &QCoreApplication::quit);
    view.setSource(QUrl("qrc:/demos/clocks/clocks.qml"));
    if (view.status() == QQuickView::Error)
        return -1;
    view.setResizeMode(QQuickView::SizeRootObjectToView);
    view.show();
    return app.exec();
}
```





Пример 1. «clocks»

clock.qml

```
import QtQuick 2.15
import "content" as Content
Rectangle {
    id: root
    width: 640; height: 320;   color: "#646464"
    ListView {
        id: clockview
        anchors.fill: parent
        orientation: ListView.Horizontal
        cacheBuffer: 2000 // размер области для хранения делег.
        snapMode: ListView.SnapOneItem// поведение прокрутки
        highlightRangeMode: ListView.ApplyRange
        delegate: Content.Clock { city: cityName; shift: timeShift }
        model:  . . . }  // ListModel place
    Image {
        anchors.left: parent.left
        anchors.bottom: parent.bottom
        anchors.margins: 10
        source: "content/arrow.png"
        rotation: -90
        opacity: clockview.atXBeginning ? 0 : 0.5
        Behavior on opacity { NumberAnimation { duration: 500 } }
    }
    . . .  // another arrow place
}
```

```
ListModel {
        ListElement { cityName: "New York"; timeShift: -4 }
        ListElement { cityName: "London"; timeShift: 0 }
        ListElement { cityName: "Oslo"; timeShift: 1 }
        ListElement { cityName: "Mumbai"; timeShift: 5.5 }
        ListElement { cityName: "Tokyo"; timeShift: 9 }
        ListElement { cityName: "Brisbane"; timeShift: 10 }
        ListElement { cityName: "Los Angeles"; timeShift: -8 }
    }
```

```
Image {
        anchors.right: parent.right
        anchors.bottom: parent.bottom
        anchors.margins: 10
        source: "content/arrow.png"
        rotation: 90
        opacity: clockview.atXEnd ? 0 : 0.5
        Behavior on opacity { NumberAnimation { duration: 500 } }
    }
```

Пример 1. «clocks»

Content/Clock.qml

```
import QtQuick 2.15
Item {
   id : clock
   width: {
      if (ListView.view && ListView.view.width >= 200)
         return ListView.view.width / Math.floor(ListView.view.width / 200.0);
      else  return 200;
   }
   height: {
      if (ListView.view && ListView.view.height >= 240)
         return ListView.view.height;
      else   return 240;
    }
    . . . // property
   function timeChanged() { . . .}
   Timer { . . . }
   Item {
      anchors.centerIn: parent
      width: 200; height: 240
      Image { id: background; source: "clock.png"; visible: clock.night == false }
      Image { source: "clock-night.png"; visible: clock.night == true }
      Image { . . . }
      Image { . . . }
      Image { . . . }
      Image { anchors.centerIn: background; source: "center.png“ }
      Text { . . . }
   }
}
```

```
property alias city: cityLabel.text
property int hours
property int minutes
property int seconds
property real shift
property bool night: false
property bool internationalTime: true //Unset for local time
```

```
function timeChanged() {
    var date = new Date;
    hours = internationalTime ? date.getUTCHours() +
                    Math.floor(clock.shift) : date.getHours()
    night = ( hours < 7 || hours > 19 )
    minutes = internationalTime ? date.getUTCMinutes() +
                    ((clock.shift % 1) * 60) : date.getMinutes()
    seconds = date.getUTCSeconds();
  }
```

```
Timer {
    interval: 100  // mc
     running: true; repeat: true;
    onTriggered: clock.timeChanged()
  }
```

Пример 1. «clocks»

```
Text {
        id: cityLabel
        y: 210;
        anchors.horizontalCenter: parent.horizontalCenter
        color: "white"
        font.family: "Helvetica"
        font.bold: true; font.pixelSize: 16
        style: Text.Raised; styleColor: "black"
    }
```

Clock.qml

```
Image {
        x: 97.5; y: 20
        source: "second.png"
        transform: Rotation {
            id: secondRotation
            origin.x: 2.5; origin.y: 80;
            angle: clock.seconds * 6
            Behavior on angle {
                SpringAnimation { spring: 2; damping: 0.2; modulus: 360 }
            }
        }
    }
```

```
Image {
        x: 92.5; y: 27
        source: "hour.png"
        transform: Rotation {
            id: hourRotation
            origin.x: 7.5; origin.y: 73;
            angle: (clock.hours * 30) + (clock.minutes * 0.5)
            Behavior on angle {
                SpringAnimation { spring: 2; damping: 0.2; modulus: 360 }
            }
        }
}
```

```
Image {
        x: 93.5; y: 17
        source: "minute.png"
        transform: Rotation {
            id: minuteRotation
            origin.x: 6.5; origin.y: 83;
            angle: clock.minutes * 6
            Behavior on angle {
                SpringAnimation { spring: 2; damping: 0.2; modulus: 360 }
            }
        }
    }
```

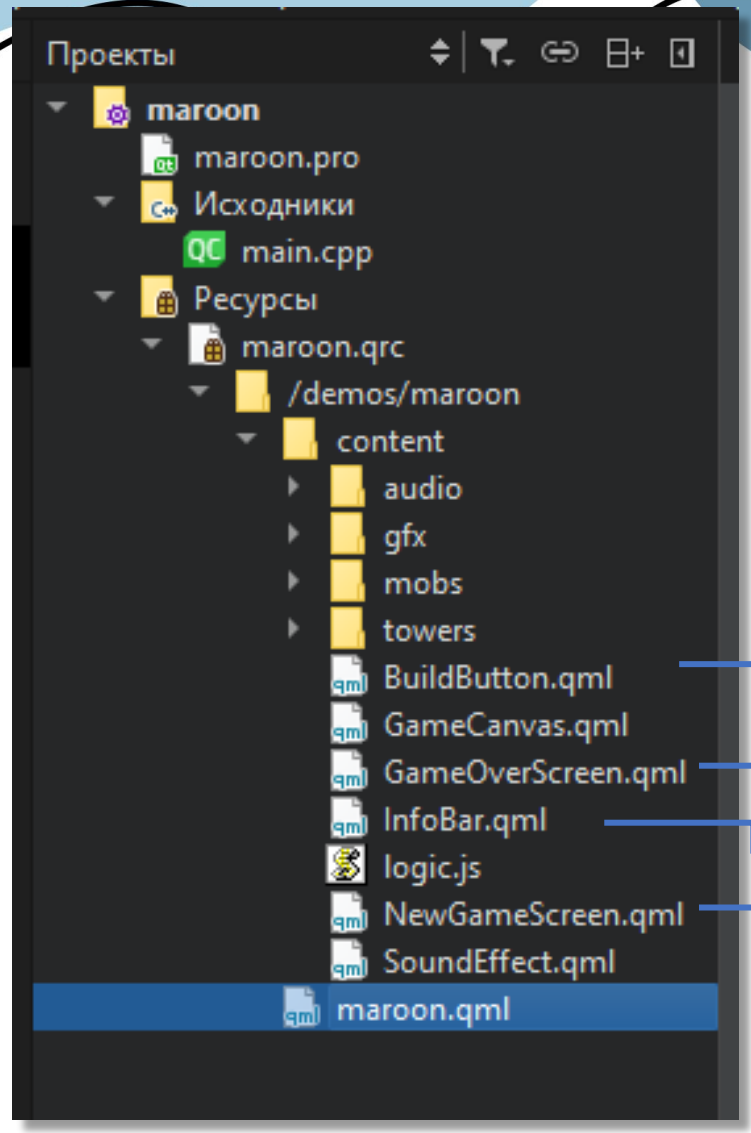
Пример 2. «maroon»

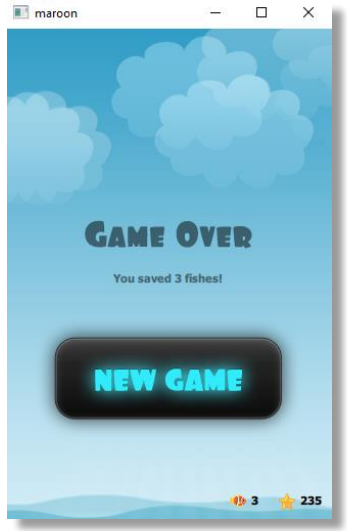

```
#include <QGuiApplication>
#include <QQmlEngine>
#include <QQmlFileSelector>
#include <QQuickView>

int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
    QCoreApplication::setOrganizationName("QtExamples");
    QGuiApplication app(argc, argv);

    QQuickView view;
    view.connect(view.engine(), &QQmlEngine::quit, &app, &QCoreApplication::quit);
    view.setSource(QUrl("qrc:/demos/maroon/maroon.qml"));
    if (view.status() == QQuickView::Error)
        return -1;
    view.setResizeMode(QQuickView::SizeRootObjectToView);
    view.show();
    return app.exec();
}
```

Пример 2. «maroon»





Пример 2. «maroon»

maroon.qml

```
import QtQuick 2.0
import QtQuick.Particles 2.0
import "content"
import "content/logic.js" as Logic

Item {
    id: root
    width: 320
    height: 480
    property var gameState
    property bool passedSplash: false
  //  Image_background *
  //  Column_Screen **
    property int countdown: 10
   // Timer{ . . . }***
    states:   . . .
    transitions: . . .
Component.onCompleted: gameState =
            Logic.newGameState(canvas);
}
```

```
*
Image {
        source:"content/gfx/background.png"
        ors.bottom: view.bottom
    ParticleSystem {
        id: particles
        anchors.fill: parent
        ImageParticle {
            id: bubble;     anchors.fill: parent
            source: "content/gfx/catch.png"
            opacity: 0.25
        }
        Wander {  xVariance: 25;  pace: 25;     }// изменение траектории в сек
        Emitter {
            width: parent.width
            height: 150
            anchors.bottom: parent.bottom
            anchors.bottomMargin: 3
            startTime: 15000
            emitRate: 2
            lifeSpan: 15000
            acceleration: PointDirection{ y: -6; xVariation: 2; yVariation: 2 }
            size: 24;   sizeVariation: 16
        }
    }
}
```

```
***
Timer {
        id: gameStarter
        interval: 4000
        running: false
        repeat: false
        onTriggered: Logic.startGame(canvas);
    }
```

Пример 2. «maroon»

```
**
Column {
    id: view
    y: -(height - 480) ;  width: 320
    GameOverScreen { gameCanvas: canvas }
    Item {
        id: canvasArea
        width: 320;  height: 480

        ROWs_WAVE . . .

        IMAGEs_SUNLIGHT . . .

       Image { source: "content/gfx/grid.png";  opacity: 0.5
}
      . . . ->

}
```

```
Row {
        height: childrenRect.height
        Image {  id: wave;  y: 30;  source:"content/gfx/wave.png" }
        Image {  y: 30;  source: "content/gfx/wave.png"   }
        NumberAnimation on x { from: 0; to: -(wave.width);
                               duration: 16000; loops: Animation.Infinite }
        SequentialAnimation on y {
          loops: Animation.Infinite
          NumberAnimation { from: y - 2; to: y + 2; duration: 1600;
                            easing.type: Easing.InOutQuad }
          NumberAnimation { from: y + 2; to: y - 2; duration: 1600;
                            easing.type: Easing.InOutQuad }
        }
    }
 Row {
      opacity: 0.5
      Image {  id: wave2;  y: 25; source: "content/gfx/wave.png" }
      Image {  y: 25;   source: "content/gfx/wave.png"  }
      NumberAnimation on x { from: -(wave2.width); to: 0; duration: 32000;
                             loops: Animation.Infinite }
      SequentialAnimation on y {
          loops: Animation.Infinite
          NumberAnimation { from: y + 2; to: y - 2; duration: 1600;
                            easing.type: Easing.InOutQuad }
          NumberAnimation { from: y - 2; to: y + 2; duration: 1600;
                            easing.type: Easing.InOutQuad }
        }
      }
```

Пример 2. «maroon»

```
Image {
        source: "content/gfx/sunlight.png"
        opacity: 0.02;  y: 0
        anchors.horizontalCenter: parent.horizontalCenter
        transformOrigin: Item.Top
        SequentialAnimation on rotation {
            loops: Animation.Infinite
            NumberAnimation { from: -10; to: 10; duration: 8000; easing.type: Easing.InOutSine }
            NumberAnimation { from: 10; to: -10; duration: 8000; easing.type: Easing.InOutSine }
        }
    }
Image {
        source: "content/gfx/sunlight.png"
        opacity: 0.04;  y: 20
        anchors.horizontalCenter: parent.horizontalCenter
        transformOrigin: Item.Top
        SequentialAnimation on rotation {
            loops: Animation.Infinite
            NumberAnimation { from: 10; to: -10; duration: 8000; easing.type: Easing.InOutSine }
            NumberAnimation { from: -10; to: 10; duration: 8000; easing.type: Easing.InOutSine }
        }
    }
```

Пример 2. «maroon»

```
    -> . . .
      GameCanvas {
          id: canvas
          anchors.bottom: parent.bottom
          anchors.bottomMargin: 20
          x: 32;          focus: true
      }
      InfoBar { anchors.bottom: canvas.top; anchors.bottomMargin: 6; width: parent.width }
      //3..2..1..go
      Timer {
          id: countdownTimer
          interval: 1000
          running: root.countdown < 5
          repeat: true
          onTriggered: root.countdown++
      }
      Repeater {
          model: ["content/gfx/text-blank.png", "content/gfx/text-3.png", "content/gfx/text-2.png", "content/gfx/text-1.png", "content/gfx/text-go.png"]
          delegate: Image {
              visible: root.countdown <= index
              opacity: root.countdown == index ? 0.5 : 0.1
              scale: root.countdown >= index ? 1.0 : 0.0
              source: modelData
              Behavior on opacity { NumberAnimation {} }
              Behavior on scale { NumberAnimation {} }
          }
      }
    }
    NewGameScreen {   onStartButtonClicked: root.passedSplash = true    }
}
```