УНИВЕРСИТЕТ ИМЕНИ СУЛЕЙМАНА ДЕМИРЕЛЯ

Жапаров М. К.

Learning QML

Учебно-методическое пособие

Каскелен, 2015

УДК 004. 4(075.8)

ББК 32.973.26-018 я73

Z 11

Рекомендовано к изданию Учебно-методическим советом Университета имени Сулеймана Демиреля протокол №1 от 25.08.2015 г.

Рецензенты**:**

Кожахметов К., PhD, ассистент профессора, Казахстанско-Британский технический университет

Богданчиков А. Н. PhD, Университет имени Сулеймана Демиреля

Zhaparov M.K.

Z11 Learning QML: Methodical-educational manual. Zhaparov M.K. — Kaskelen, Suleyman Demirel University, 2015. — 65 p. — In English.

ISBN 978-601-7537-25-8

Учебно- методическое пособие «Microlearning in Programming Language C++:» предназначена для бакалавров по специальностям 5B03000 — «Информационные Системы» и 5B040000 — «Вычислительная Техника и Программное Обеспечение».

Methodical-educational manual « Microlearning in Programming Language C++» is designed for bachelors of specialties 5B030000 — “Information Systems” and 5B040000 — “Computing Systems and Software”.

УДК 004. 4(075.8)

ББК 32.973.26-018 я73

ISBN 978-601-7537-25-8

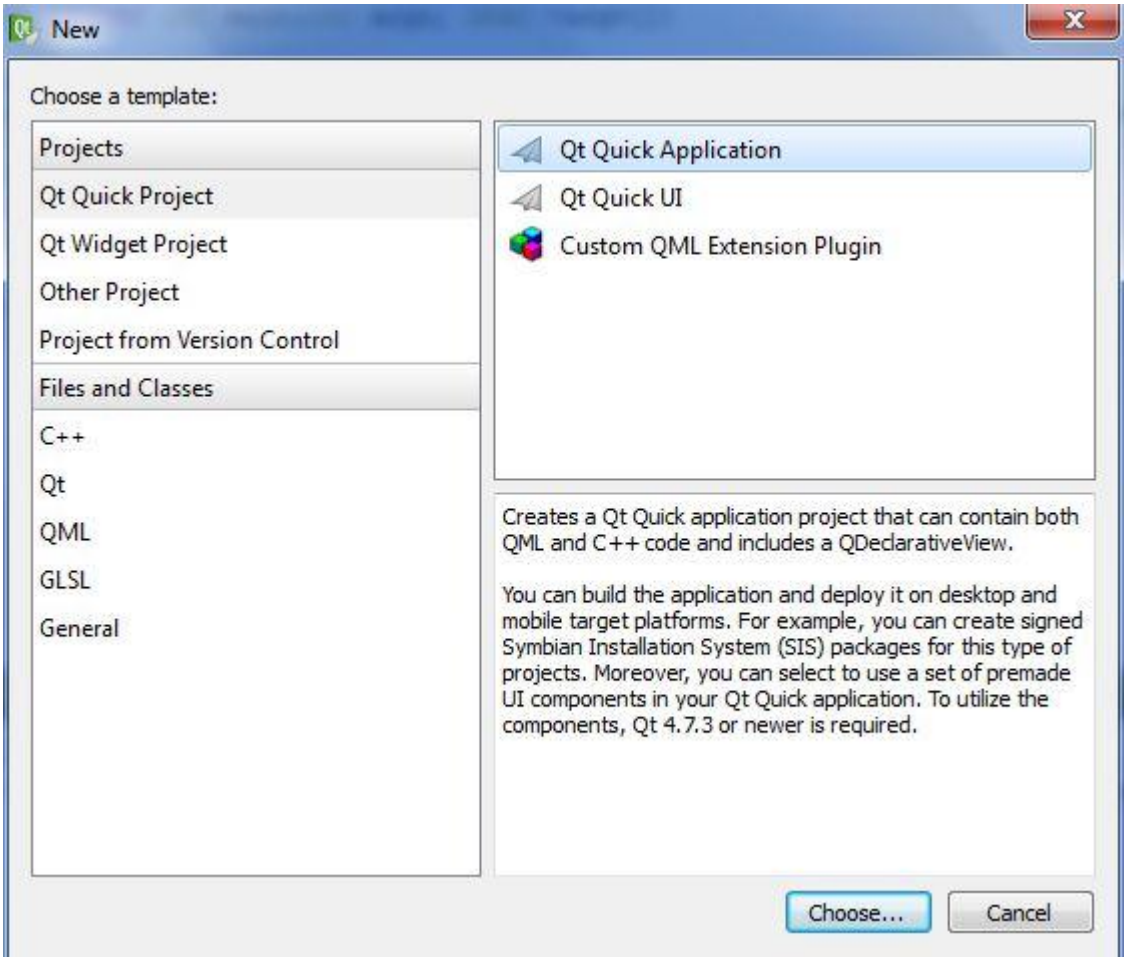
CONTENTS

Tutorial #1

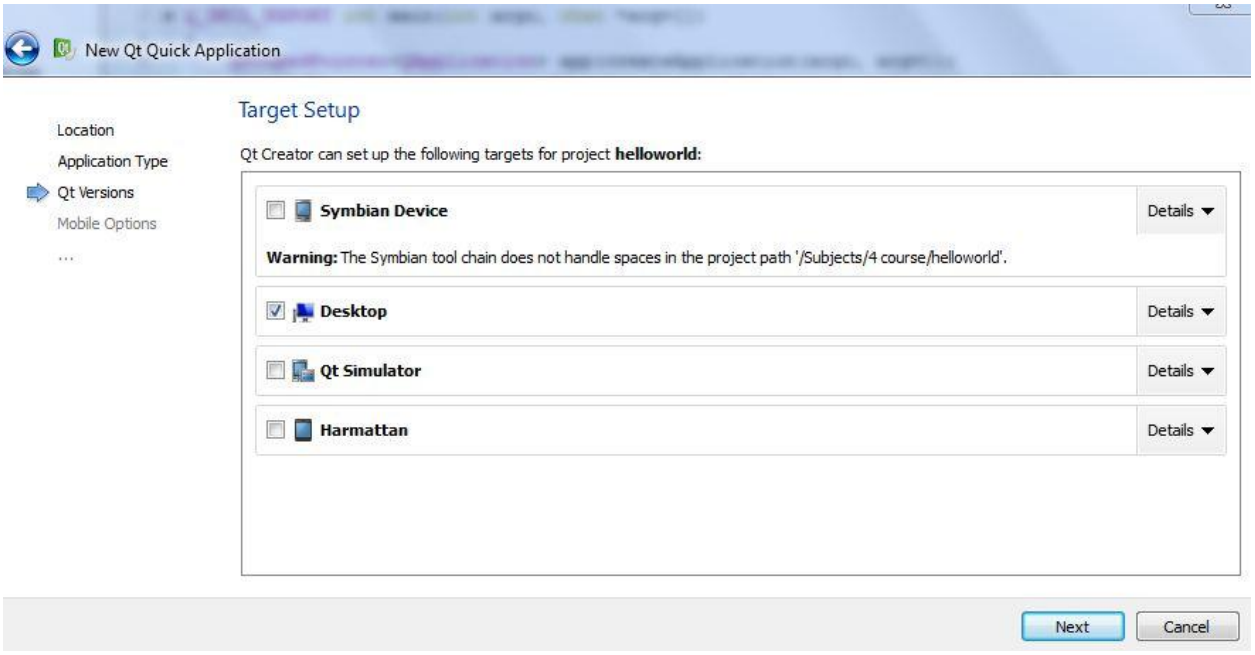
"Hello World!"

Our first tutorial in QML programming as usual we will start with "Hello World!" project

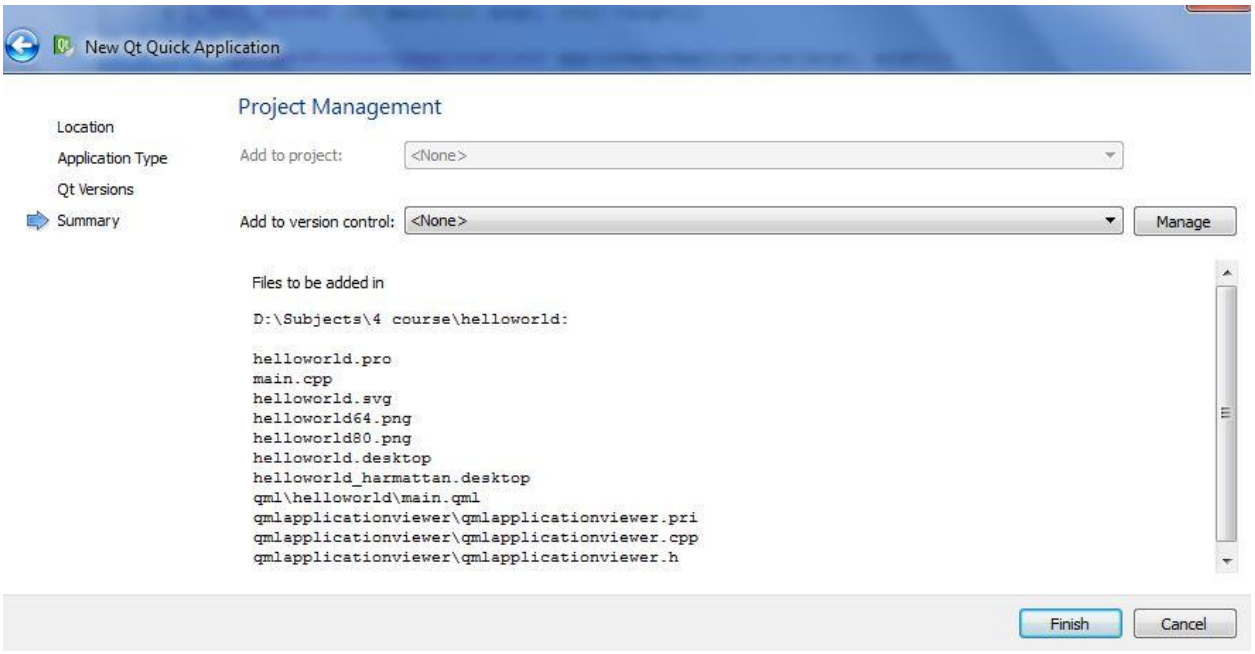
After the installing Qt, we will create a new project. As shown below first we will choose Qt Quick Project and then Qt Quick Application



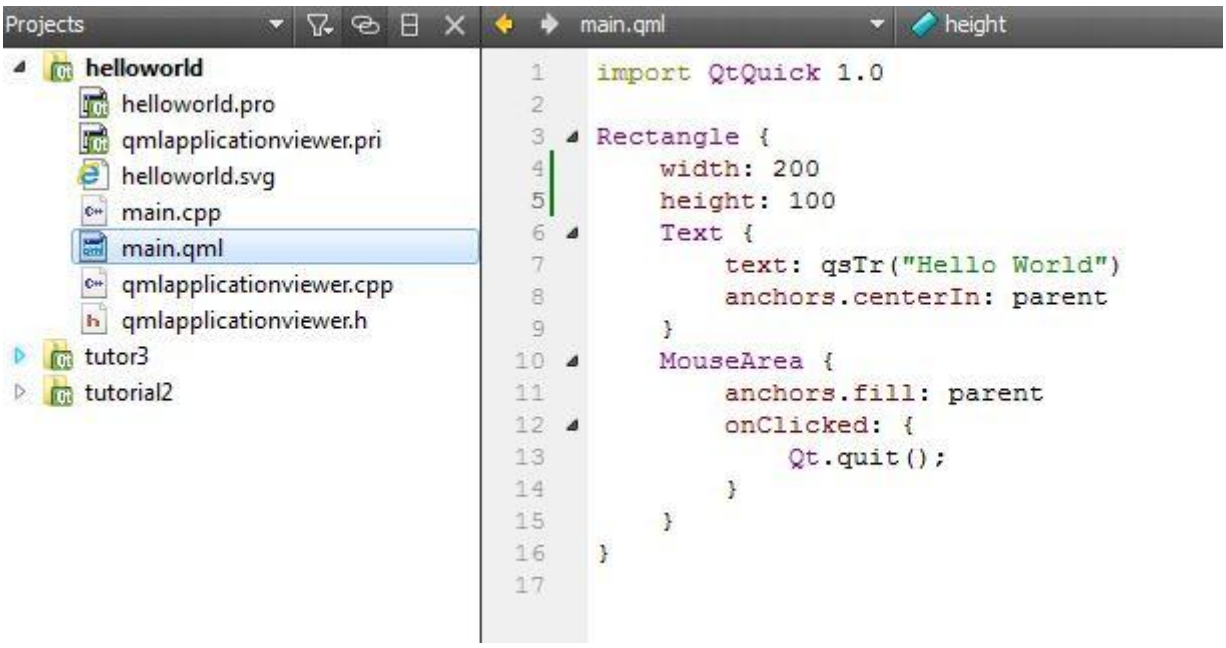
Then choose a desktop, as you remember this is for our output



and finish it…



In our project open file main.qml. This file is our main QML file where will placed our main window.

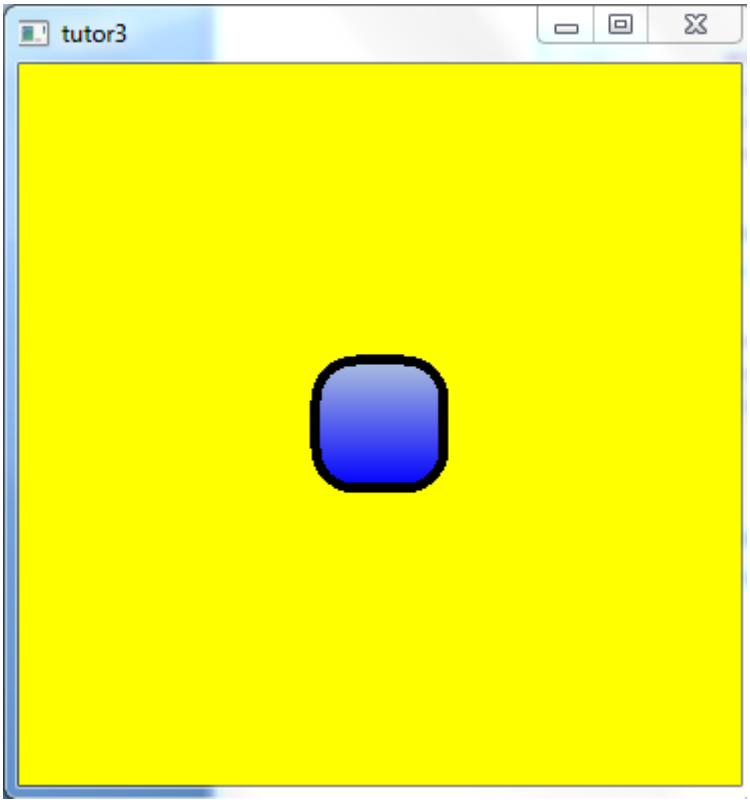


Ok, let's run it and see the result.

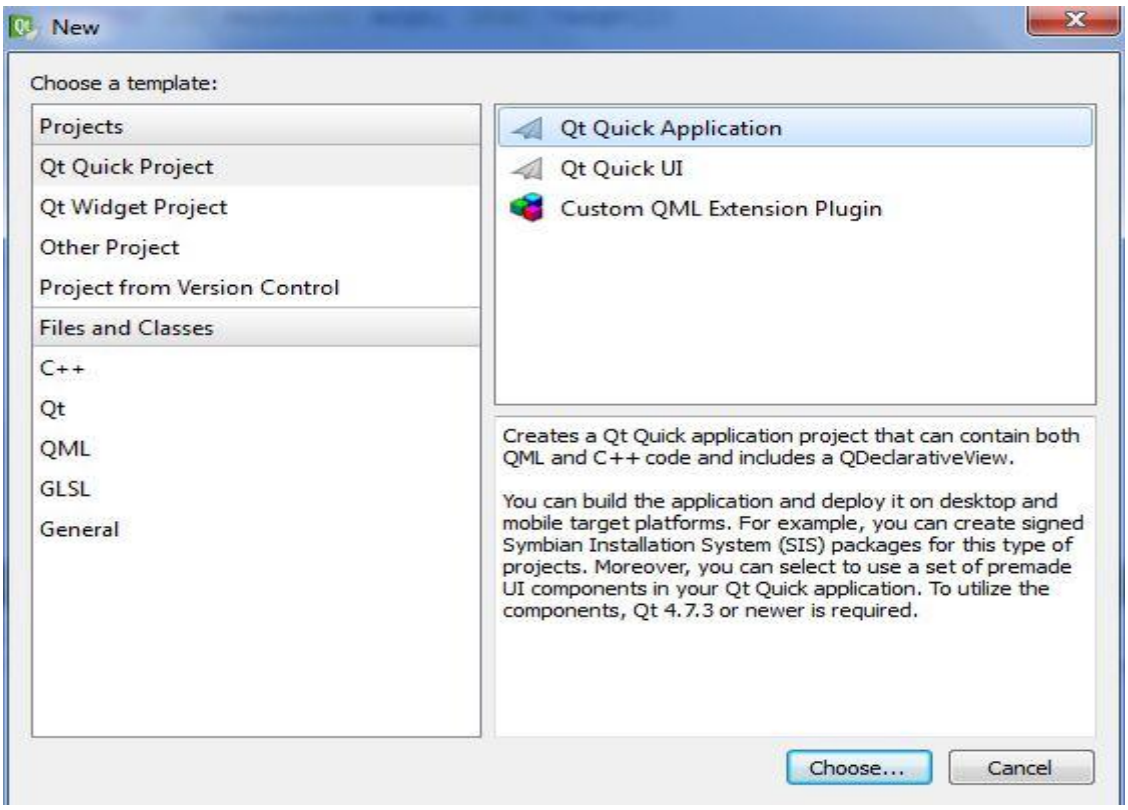
Tutorial #2

“Rectangles”

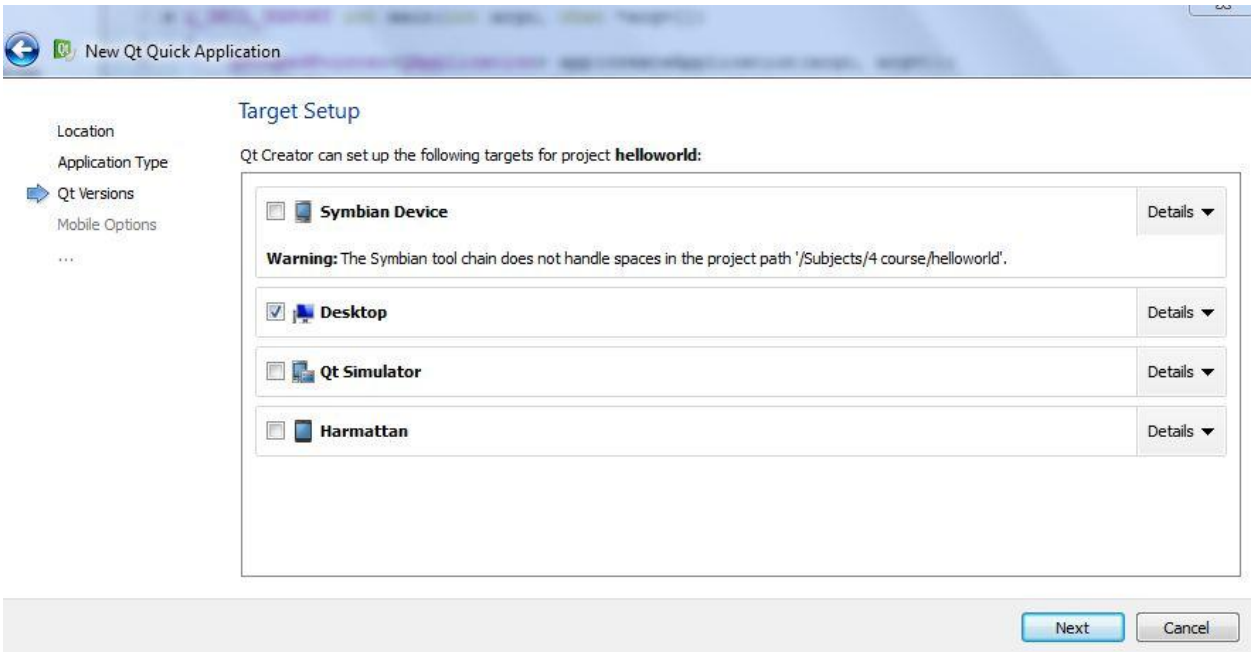
In this tutorial we will create 2 rectangles:



Ok, let’s start. First of all create new Qt Quick application

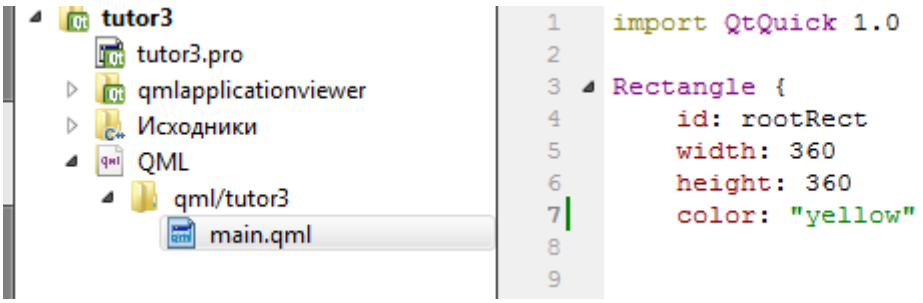


Then choose there a desktop



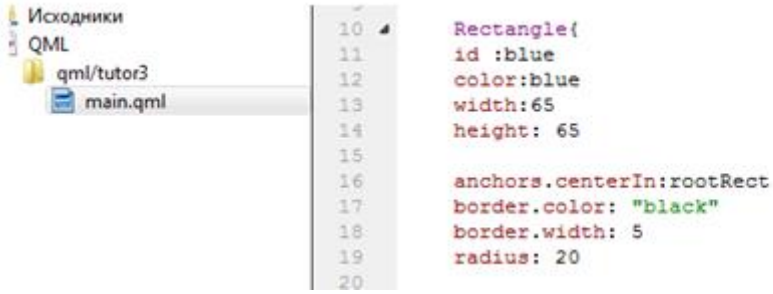
And then Next – Next - Next and Finish it.

Now we will create our first rectangle:

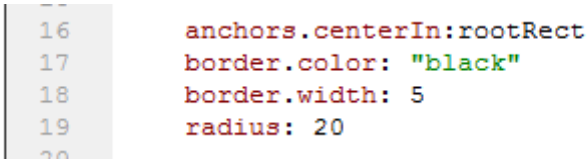


The id of our rectangle is rootRect, by default it's width and height is 360 and we've colored it to the yellow.

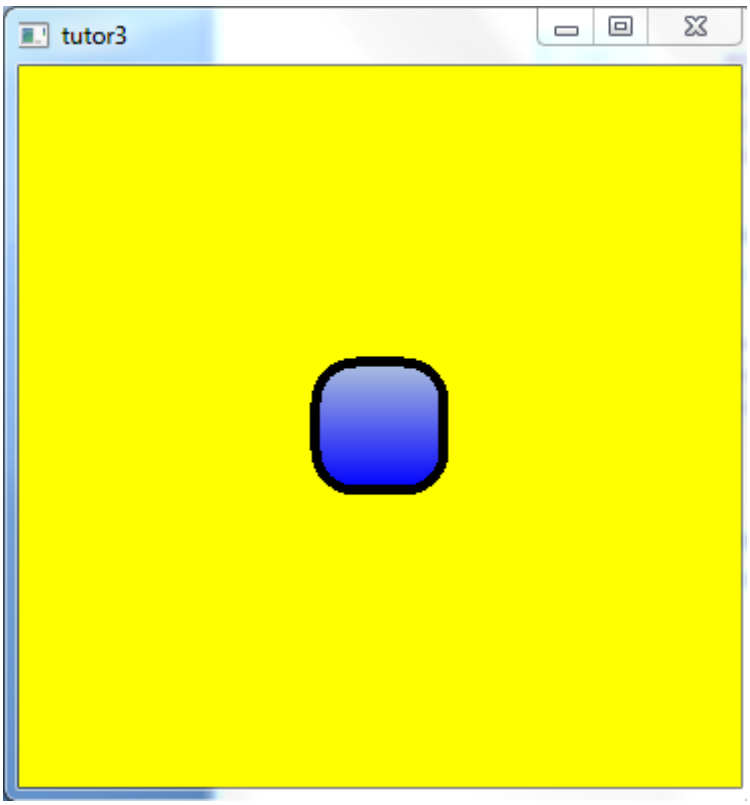
Now we will create second small rectangle to inside the yellow one:



There



Ok, let's run it and see the result:



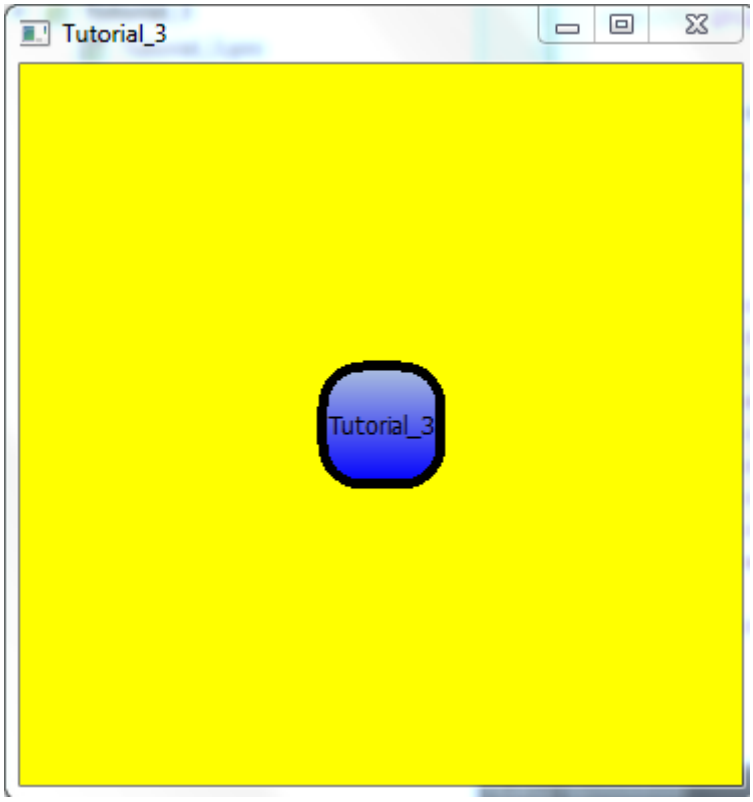
Extra exercises:

1. Type some text inside little rectangle
2. Change the gradient color

Tutorial #3

“Text”

In this tutorial we will paste the text to our second tutorial:



Ok, let’s start with copying all code from second tutorial. Then just add the given code inside our Blue rectangle:

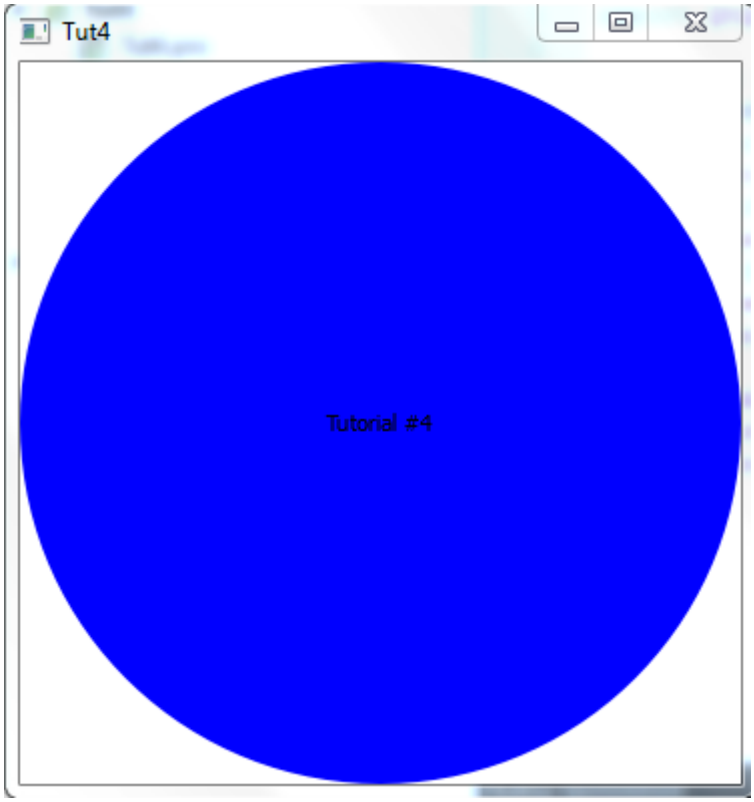
```
Text{
id:text
text:"Tutorial_3"
font.pixelSize: 12
anchors.centerIn: blue
}
```

That’s all, you can run it!!!

Tutorial #4

“State and Transition”

In this tutorial we will learn “Transitions in QML”. Transitions it means do some animations, some changes with our created objects (texts, shapes and so on). Let’s create a new project. Our project will be simple; we will create a rectangle (circle) then do with it some animation:



There is a code of creating our rectangle:

```
1  import QtQuick 1.0
2
3  Rectangle {
4      id:rectangle
5      width: 360
6      height: 360
7      color:"blue"
8      radius: 180
9      Text {
10         text: qsTr("Tutorial #4")
11         anchors.centerIn: parent
12     }
13     MouseArea {
14         id:mouseArea
15         anchors.fill: parent
16     }
```

Then we will add there some animations: this animation will change our rectangle's position:

```
17         states:State {
18                name: "changed"; when: mouseArea.pressed
19                PropertyChanges {target: rectangle; x: 180; y: 180}
20         }
21             transitions: Transition {
22                 NumberAnimation { properties:"x, y"; easing.type: Easing.InOutQuad  }
23
24             }
25                     }
```

On following code we create a new position of our rectangle after pressing in it: there our x and y positions will change from 360 to 180(half of rectangle will hide):

```
17         states:State {
18                name: "changed"; when: mouseArea.pressed
19                PropertyChanges {target: rectangle; x: 180; y: 180}
20         }
```

Then in transitions we specifies change of the rectangle between the default and the "changed" state, any changes to the `x` and `y` properties should be animated, using an **`Easing.InOutQuad`**.

```
21                transitions: Transition {
22                    NumberAnimation { properties:"x, y"; easing.type: Easing.InOutQuad  }
23
24                }
25                        }
```

Ok guys, that is all. You can run it and see the result.

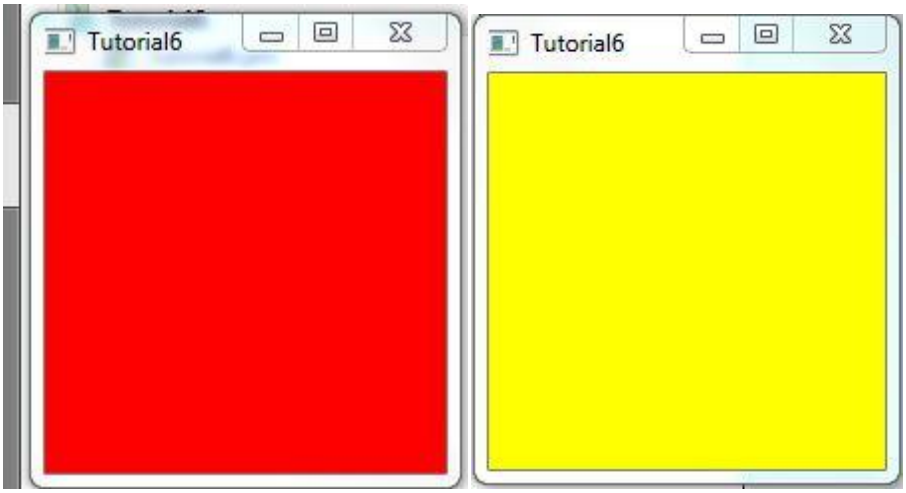
Task

Redo this task's transition where in output rectangle will move to the right sight for 1 second.

Tutorial #5

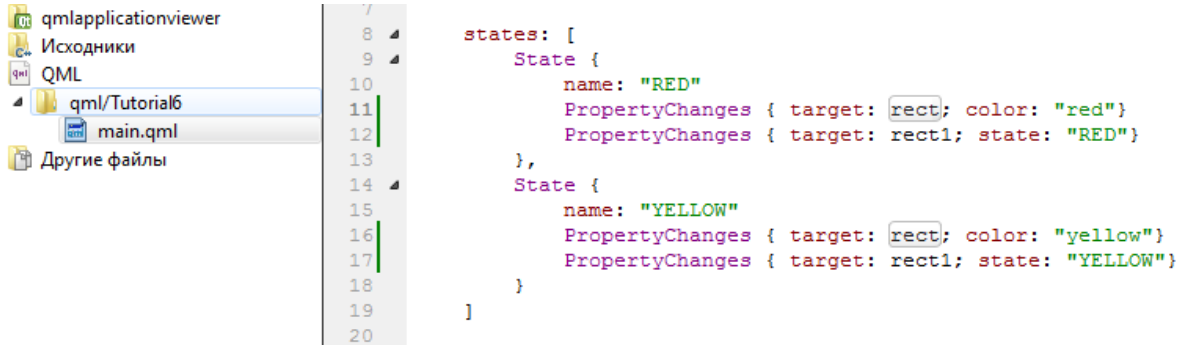
"States Part #1"

In this tutorial we will do the project related with States only for a good understanding. Tutorial will be related with 2 colors (red, yellow) where we can change them by clicking.



First of all let's create our red rectangle:

Now we must create our "RED" and "YELLOW" States where in first State we say to change the color to red when it will be "RED" State, and as the same in the second one also to change the color of our rectangle to yellow when it will be "YELLOW" State:



Ok, then we should create our MouseArea for changing our colors:

```
23          MouseArea {
24              anchors.fill: parent
25              onClicked: {
26                  if (rect.state == "RED")
27                      rect.state = "YELLOW"
28                  else
29                      rect.state = "RED"
30              }
31          }
```

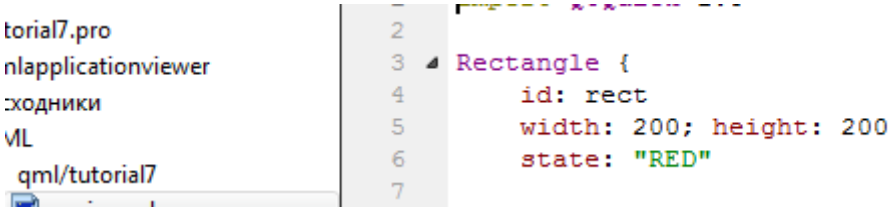
That’s all. You can run it and view the result.

Tutorial #6

"States Part #2"

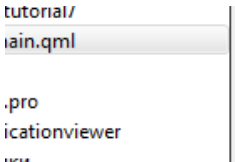
Our 6 th tutorial is same as the previous one. But there will be three colors (red, yellow, green).

Ok, let's start it with creating main rectangle:




```
2
3   Rectangle {
4       id: rect
5       width: 200; height: 200
6       state: "RED"
7
```

Then we should create our states:

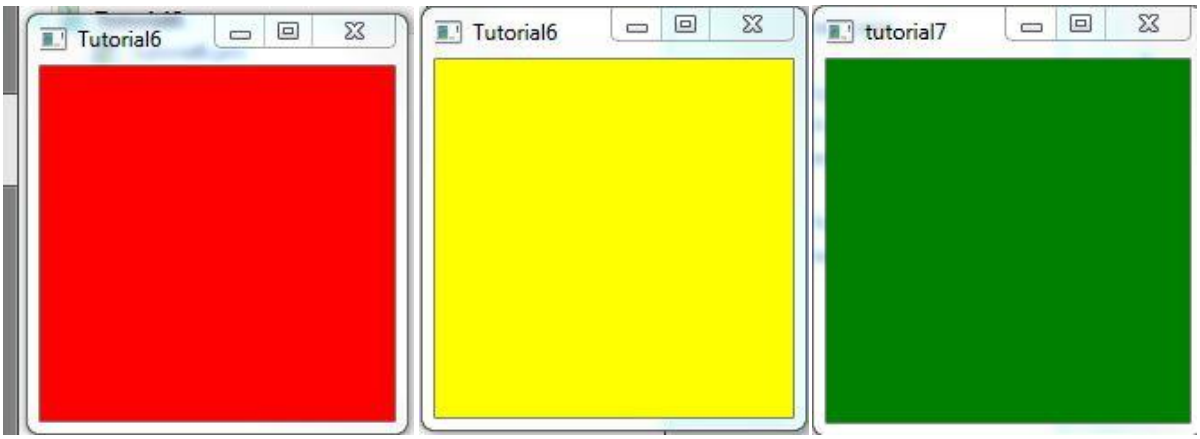



```
7
8   states: [
9       State {
10          name: "RED"
11          PropertyChanges { target: rect; color: "red"}
12          PropertyChanges { target: flag; state: "RED"}
13      },
14      State {
15          name: "YELLOW"
16          PropertyChanges { target: rect; color: "yellow"}
17          PropertyChanges { target: flag; state: "YELLOW"}
18      },
19
20      State {
21          name: "GREEN"
22          PropertyChanges { target: rect; color: "green"}
23          PropertyChanges { target: flag; state: "GREEN"}
24      }
25  ]
```

And finally let's create our MouseArea for changing our red rectangle's color:

```
28
29          MouseArea {
30              anchors.fill: parent
31              onClicked: {
32                  if (rect.state == "RED")
33                      rect.state = "YELLOW"
34                  else if(rect.state == "YELLOW")
35                      rect.state = "GREEN"
36
37                      else
38                      rect.state = "RED"
39
```

To see our result just run it and it will be like that:



Tutorial #7

"Traffic Light"

In this tutorial we will create a ***"Traffic light".*** Ok, let's start with creating a new project. Then create a Main Rectangle and then inside it second Red one and the State's name of this rectangle will be "RED":

```
Rectangle{
    id: mainRect
    width:600
    height: 600


Rectangle {
    id: rect
    width: 150; height: 150
    radius: 150
    x:200
    y:100
    state: "RED"
```

Then we should create our States:

```
    states: [
        State {
            name: "YELLOW"
            when: mouseArea.pressed
            PropertyChanges { target: rect; color: "yellow"; x:200; y:250}

        },
        State {
            name: "GREEN"
            when: rect.state == "YELLOW"
            PropertyChanges { target: rect; color: "green"; x:200; y:350}

        },

        State {
            name: "RED"
            //when:rect.state == "GREEN"
            PropertyChanges { target: rect; color: "red"; x:200; y:100}

        }
    ]
```

There in first State we have created when the color of our Red rectangle will be Yellow and the coordinates will change down. There we said that when we press our Red rectangle this rectangle's color will change to Yellow and the coordinate will change like in code:

```
15
16      states: [
17          State {
18              name: "YELLOW"
19              when: mouseArea.pressed
20              PropertyChanges { target: rect; color: "yellow"; x:200; y:250}
```

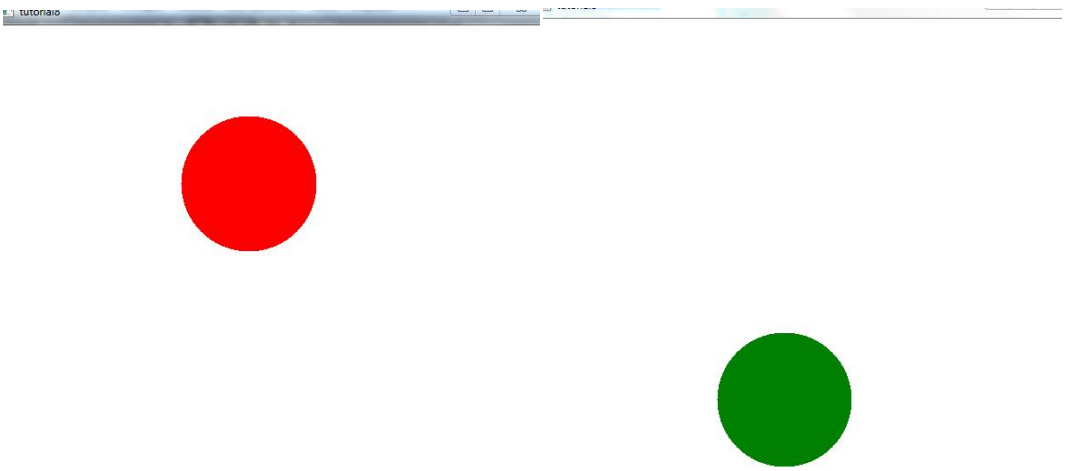
After finishing with States we should create Transitions:

```
48          transitions: [
49              Transition {
50                  from: "RED"; to: "YELLOW"
51                  PropertyAnimation { target: rect
52                      }
53                  ColorAnimation { from: "red"; to: "yellow"; duration:10}
54
55              },
56              Transition {
57                  from: "YELLOW"; to: "GREEN"
58                  PropertyAnimation { target: rect
59                      }
60                  ColorAnimation { from: "yellow"; to: "green"; duration: 200 }
61              },
62               Transition {
63                   from: "GREEN"; to: "RED"
64                   PropertyAnimation { target: rect
65                           }
66                  ColorAnimation { from: "green"; to: "red"; duration: 10000 }
67              }
68
69          ]
```

On the first transition when our State will change from “RED” to “YELLOW” our color will also change as the same and time duration for it will be 10m seconds:

```
49              Transition {
50                  from: "RED"; to: "YELLOW"
51                  PropertyAnimation { target: rect
52                      }
53                  ColorAnimation { from: "red"; to: "yellow"; duration:10}
54
```

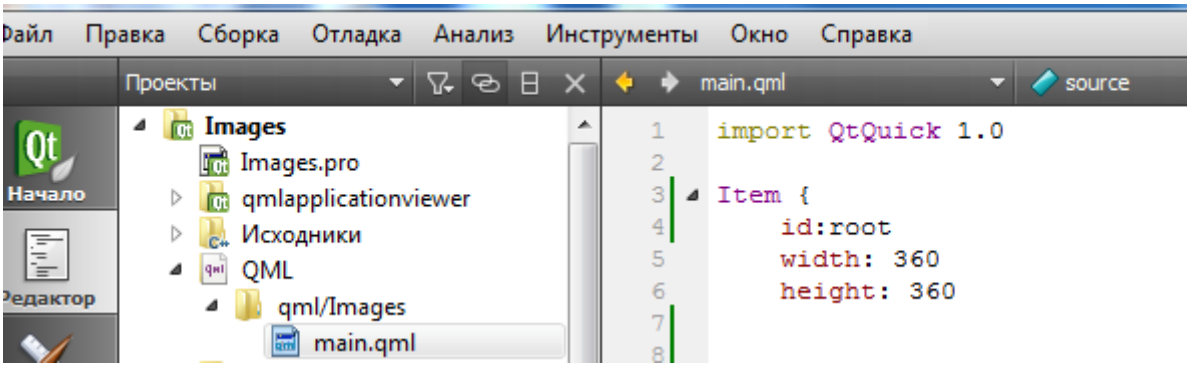
After finishing of creating the transitions you may run your project and see the result. After clicking the red one there will appear the yellow circle and then green one:



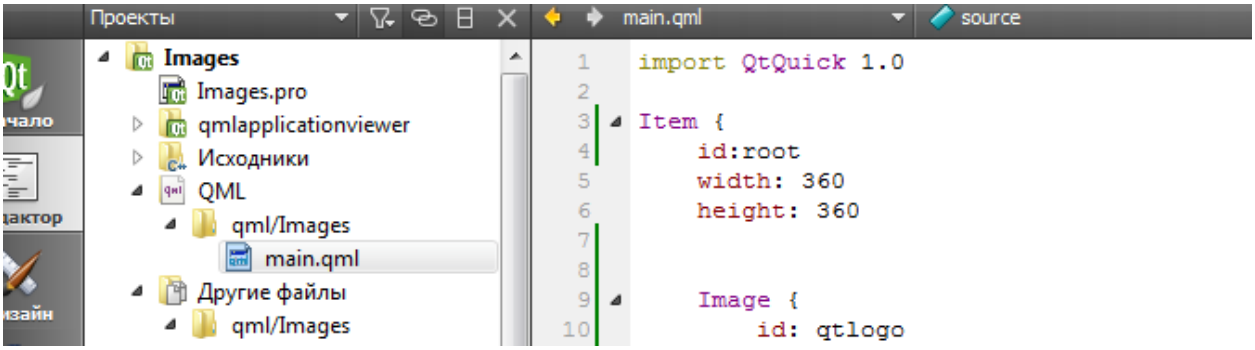
Tutorial #8

"Images"

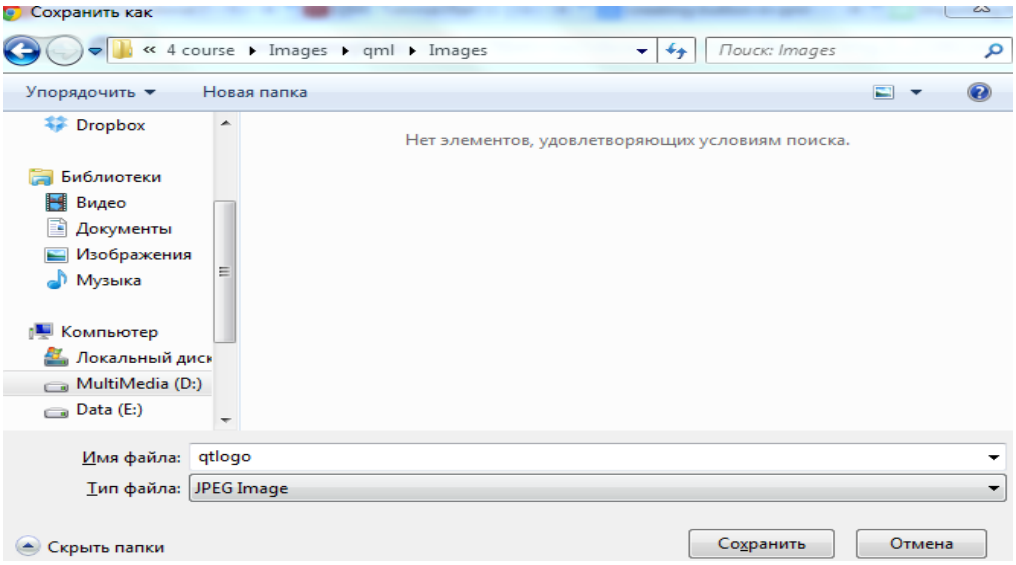
In this tutorial we will learn about images: how to display them. Ok, let's start with creating a new project with name "**Images**". Then create an Item with width and height:



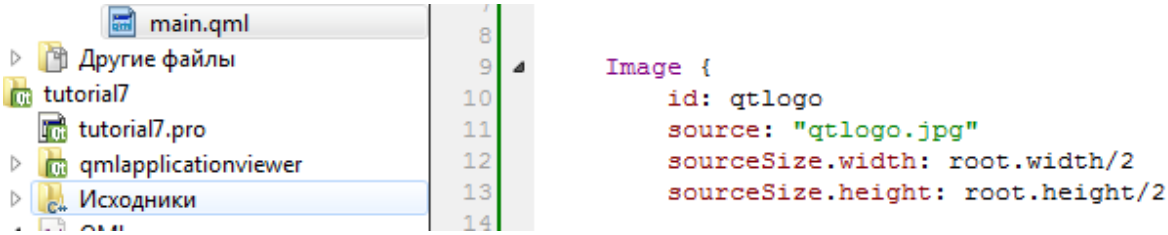
After this inside an Item we will create an Image:



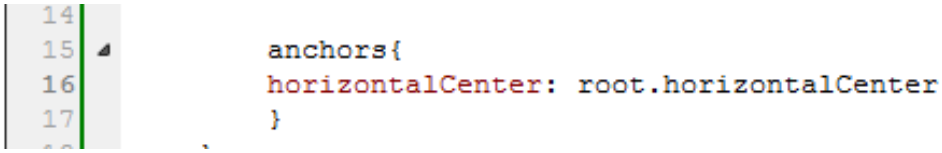
Now we download an image of QtLogo from the internet and will save it into your project's qml Images folder, like on Qt create a .qrc file then add there an image:



Then for displaying an image we will write:



For displaying an image on the center we should write:



Ok ☺ We have finished, let's run it and see the result:

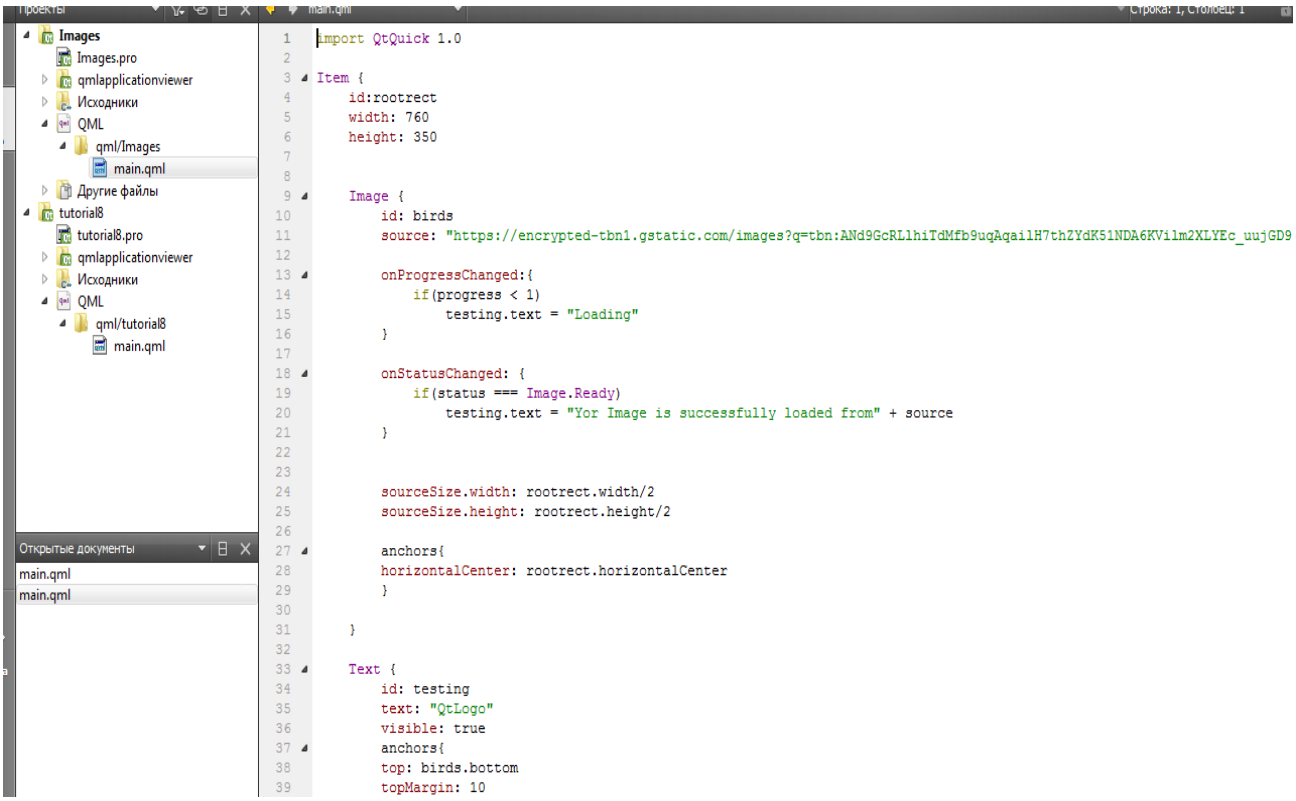
Task:

Write a text also on the top of window under the picture.

Tutorial #9

“Image form Internet page”

In this tutorial we will load an image from the internet. Create a new project and you can copy a main file from preview tutorial. Then change and add there some things:



As you see we change there the name of our Image an on the source you should write the URL name of your image:

```
source: "https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcRLlhiTdMfb9uqAqailH7thZYdK51NDA6KVilm2XLYEc_uujGD9"
```

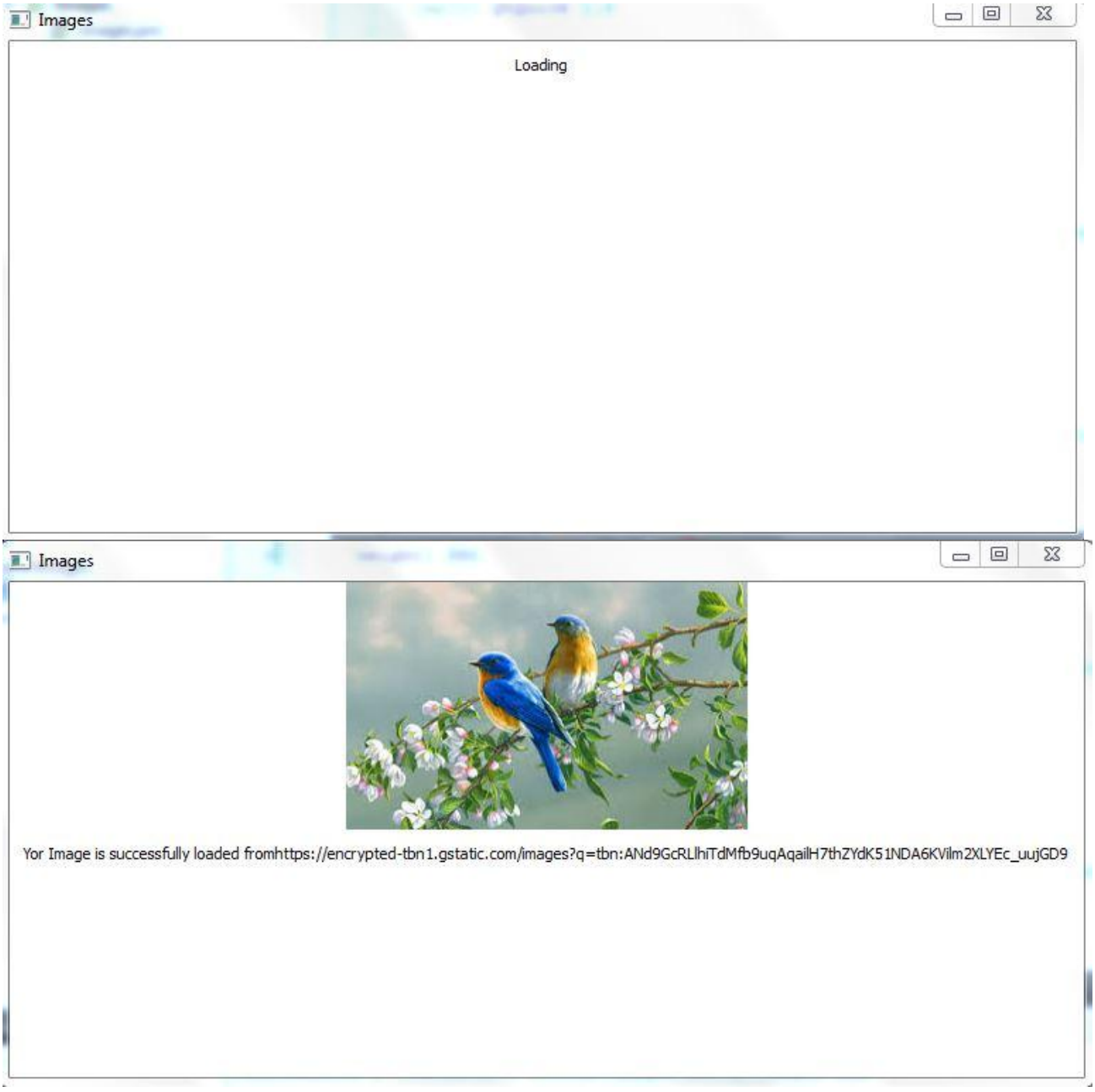
Also we add there onProgress() and onStatus() methods. Firstly let’s describe onProgress() method. There we write a code of: before an image will load from the internet, on window will be a word “Loading“:

```
onProgressChanged:{
    if(progress < 1)
        testing.text = "Loading"
}
```

Then in onStatus() method after appearing of image on the window there will appear some text:

```
        onStatusChanged: {
            if(status === Image.Ready)
                testing.text = "Yor Image is successfully loaded from" + source
        }
```

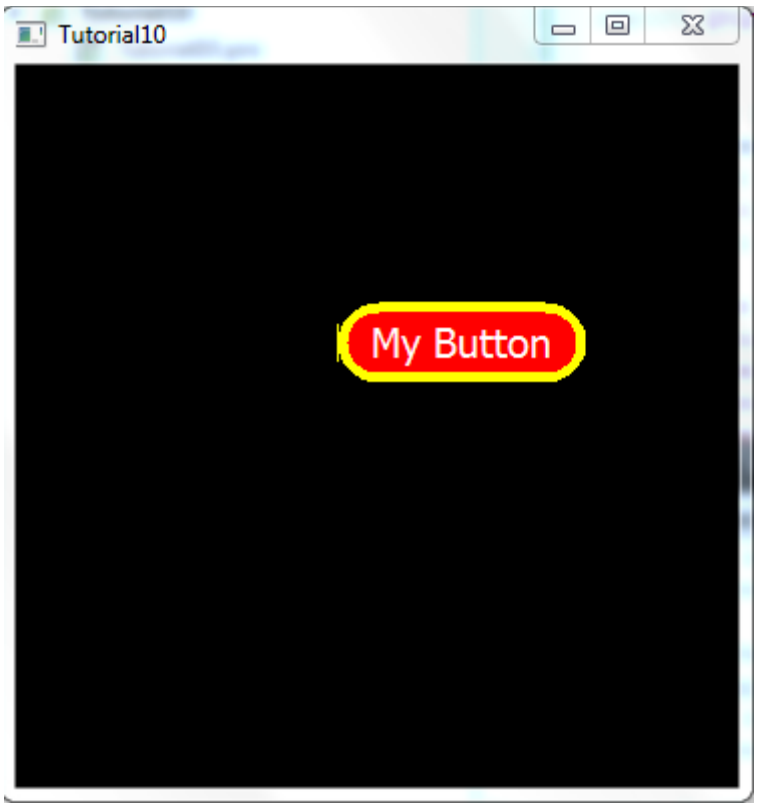
We have finished ☺ You can run and see the result:



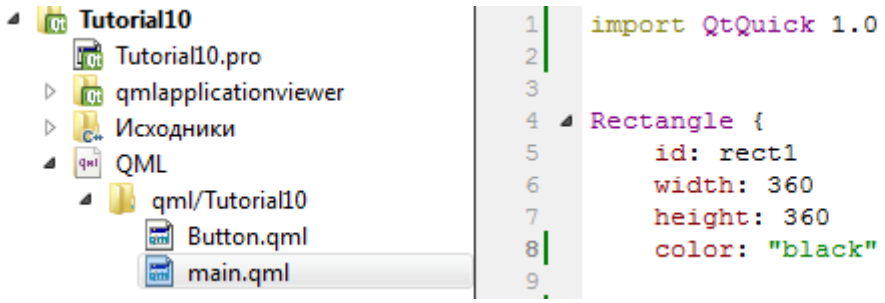
Tutorial #10

"Button Part#1"

We will create the Qml button in this tutorial. And this will be our output, when we enter the button it will increase:

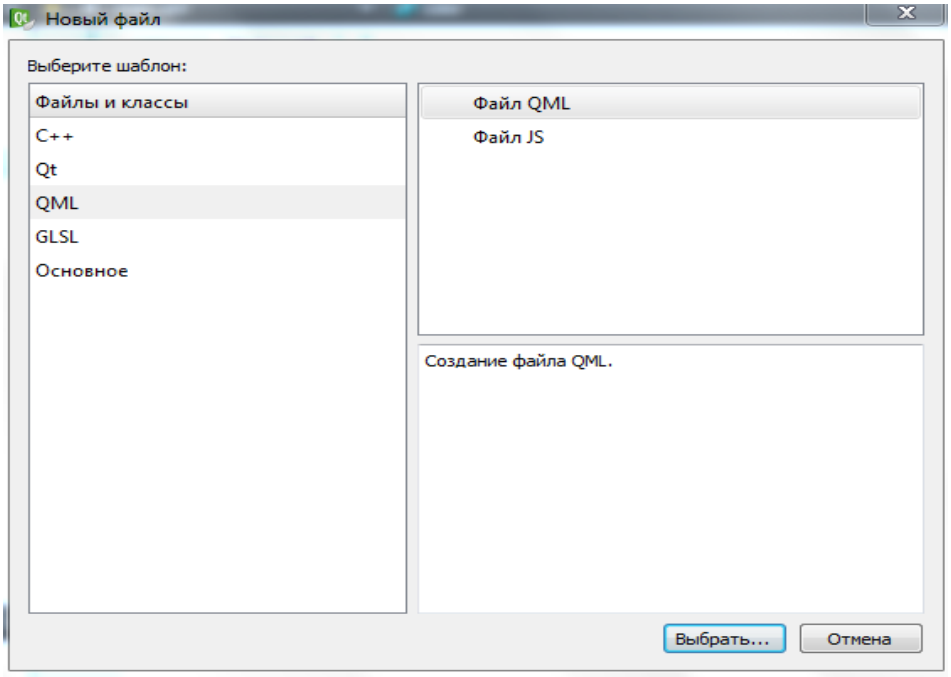


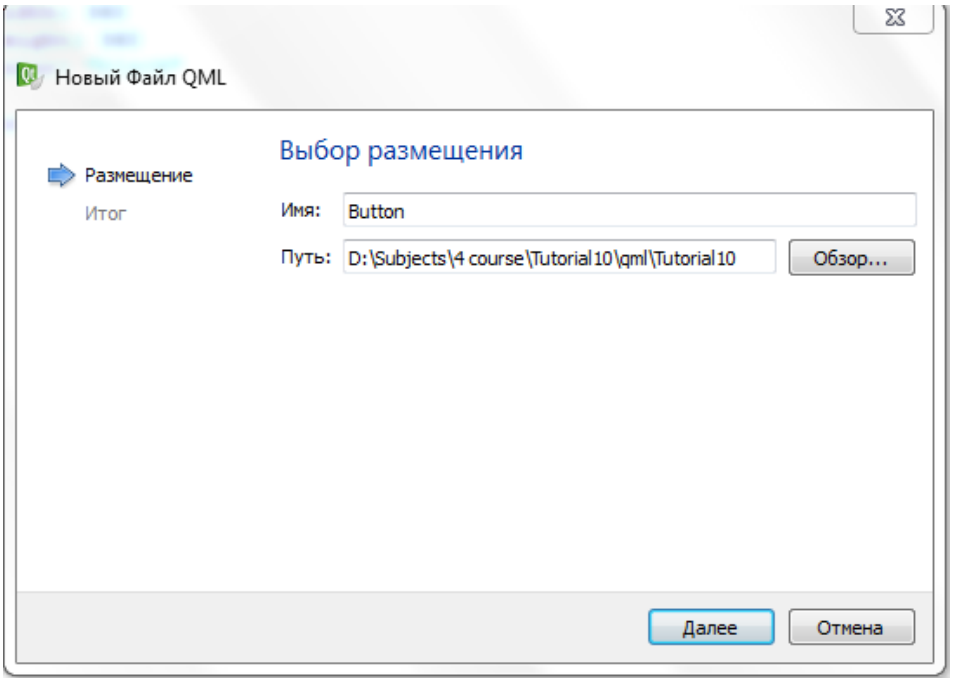
In our main.qml file we will create a rectangle:

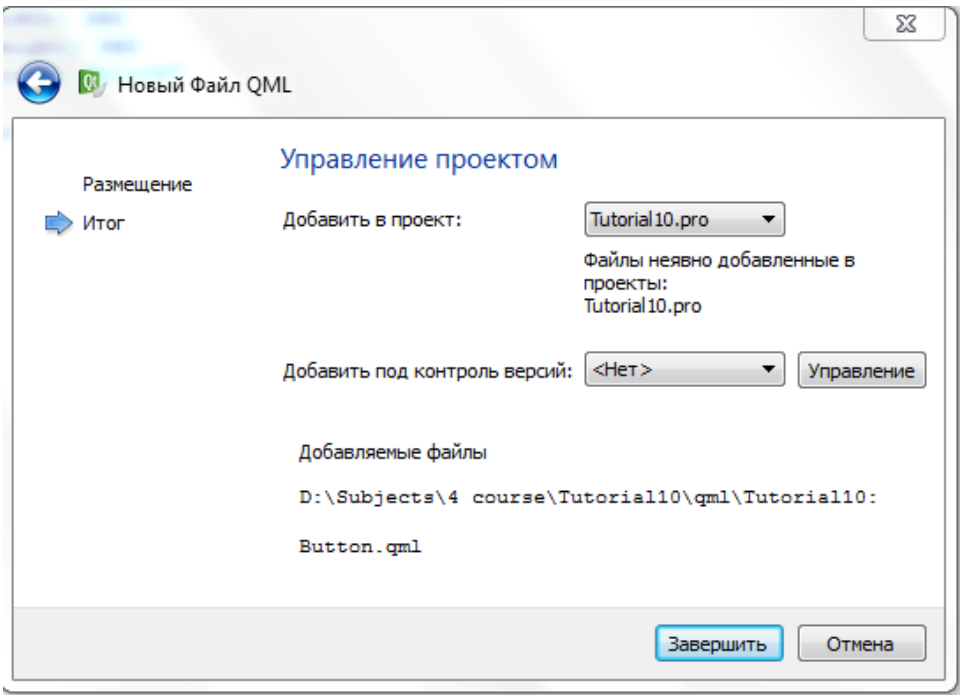



```
import QtQuick 1.0


Rectangle {
    id: rect1
    width: 360
    height: 360
    color: "black"

```

Then in our project by clicking right button of mouse we should create a new Qml file named as Button.qml:







Then we should create there 2 Rectangles and write there:


```
import QtQuick 1.0
Rectangle{

    id: button
    color:"black"
    property string backColor
    property string borderColor
    property int borderWidth
    property int radius



Rectangle{
    id: rect2

    width: parent.height
    height: parent.width
    color: button.backColor
    anchors.centerIn: rect1
    radius: button.radius
    border.color: button.borderColor
    border.width: button.borderWidth

```

And then in main.qml file create a **Button** and give the values to variables:

```
    Button {
        width: rect1.width/10
        height: 120
        anchors.centerIn: rect1
        backColor:"red"
        borderColor:"yellow"
        borderWidth:5
        radius:25
    }

```

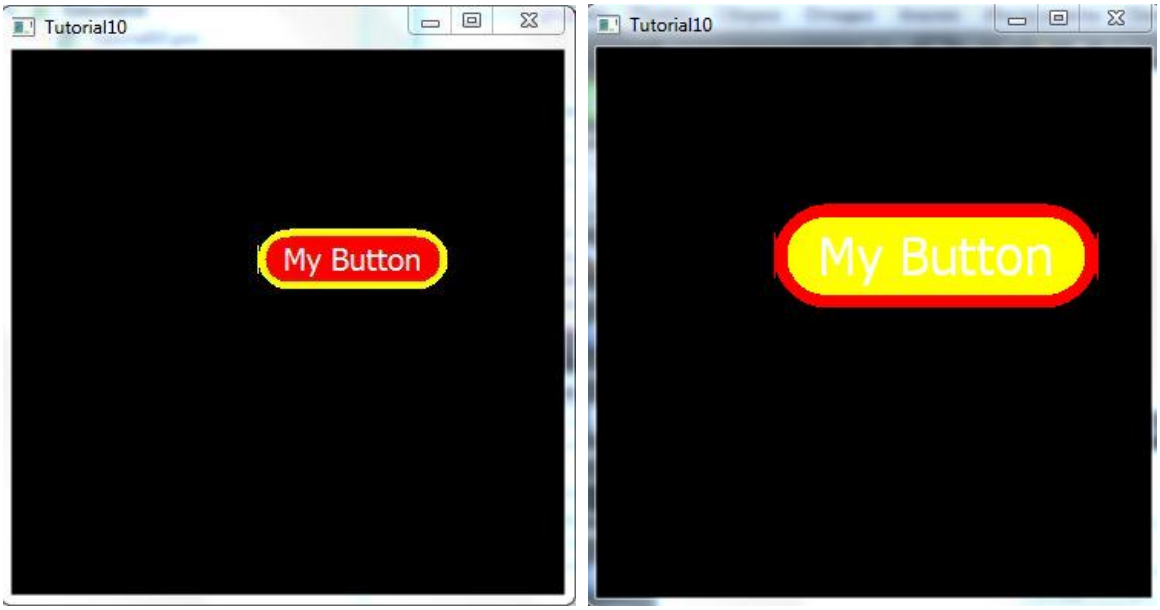
After this in MouseArea(Button.qml):

```
    MouseArea {
        id: rect2MouseArea
        anchors.fill: rect2
        hoverEnabled: true

        onEntered:{
            rect2.scale = 1.7
            rect2.color = "yellow"
            rect2.border.color = "red"

        }
        onExited:{
            rect2.scale = 1
            rect2.color = "red"
            rect2.border.color = "yellow"

        }

        onClicked: {
            console.log("MouseArea clicked")
        }
    }
```

By ending we should create a Text in our Button.qml file and write there:

```
    Text{
    id: text
    font.pixelSize: Math.round(rect1.height/18)
    anchors.centerIn: rect2MouseArea
    text: "My Button"
    color: "white"
    wrapMode: Text.WordWrap
    }
```

Ok we have finished our task ☺ You can save and run it:

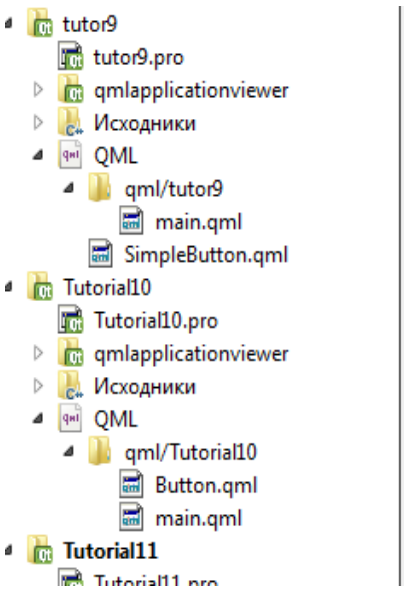

Tutorial10
My Button
Tutorial10
My Button

Tutorial #11

"Button Part#2"

In out 11th tutorial we will work with buttons and images. When we will press the "Cat" button there will appear the first image and then like this by pressing the second button the second button:

Ok, let's start our coding. Like in previous tutorial create Button.qml file. In main file you should right:




```
1   import QtQuick 1.0
2
3
4   Rectangle {
5       id: rect1
6       width: 360
7       height: 360
8       color: "yellow"
9
10      Button {
11          x: 149
12          y: 125
13          width: rect1.width/10
14          height: 110
15          anchors.verticalCenterOffset: 0
16          anchors.horizontalCenterOffset: -6
17          anchors.centerIn: rect1
18
19      }
```

And in this rect1 create Images:

```
    Image {
        id: image
        visible: false
        x: 171
        y: 87
        width: 67
        height: 65
        anchors.verticalCenterOffset: -60
        anchors.horizontalCenterOffset: 25
        anchors.centerIn: rect1
        source: "E:\\1.jpg"
    }

    Image{
        id: image1
        x: 171
        y: 179
        width: 67
        height: 64
        anchors.centerIn: mouseArea2
        source: "E:\\2.jpg"
        anchors.horizontalCenterOffset: 26
        visible: false
        anchors.verticalCenterOffset: -9
    }
```

In Button.qml file firstly create rectangle for our buttons:

```
import QtQuick 1.0


Item{


Rectangle{
    id: rect2
    x: -83
    y: -5
    width: 200
    height: 200
    color: "yellow"
```

Then respectively Mouse Areas for them:

```
    MouseArea {
        id: mouseArea1
        x: 13
        y: 0
        width: 51
        height: 5
        onClicked: image.visible = true


    }
    MouseArea{
        id: mouseArea2
        x: 18
        y: 64
        width: 45
        height: 34
        onClicked: image1.visible = true
    }
```

Then the Texts and there is not in code the second one you should write it by yourself:

```
Text{
id: text
x: 25
y: 10
width: 47
height: 27
font.pixelSize: Math.round(rect1.height/18)
anchors.centerIn: mouseArea1
text: "Cat"
anchors.verticalCenterOffset: -14
anchors.horizontalCenterOffset: 1
color: "black"
wrapMode: Text.WordWrap
}
```

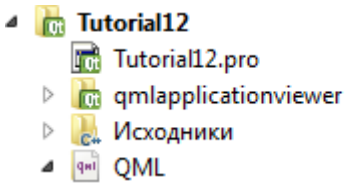
And that is all. You can run it and see the result.

Tutorial #12

Using of Row and Column

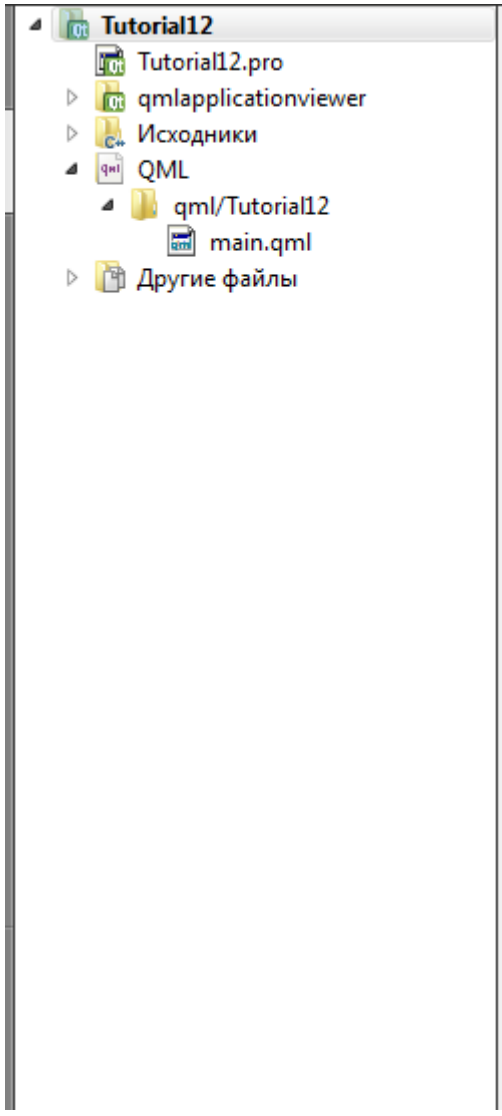
Our this tutorial is very simple☺ We will place rectangles in one row and column.

For it First of all as always we will create a main Rectangle:



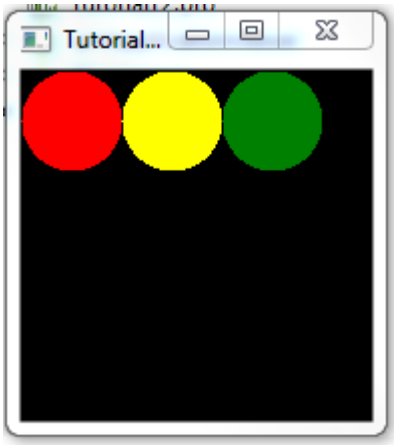

```
import QtQuick 1.0
Rectangle{
    width:175
    height: 175
    color:"black"
```

Then Create 3 rectangles to one row By this way:




```
import QtQuick 1.0
Rectangle{
    width:175
    height: 175
    color:"black"

    Row{

Rectangle {
    width: 50
    height: 50
    color:"red"
    radius:100

}

Rectangle {
    width: 50
    height: 50
    color:"yellow"
    radius:100

}

Rectangle {
    width: 50
    height: 50
    color:"green"
    radius:100

}
}
}
```

That is all☺ This is very simple. Now just run it:




```
3       width:175
4       height: 175
5       color:"black
6
7       Row{
8
9   Rectangle {
10      width: 50
11      height: 50
12      color:"red"
13      radius:100
14
15  }
16
17  Rectangle {
18      width: 50
```

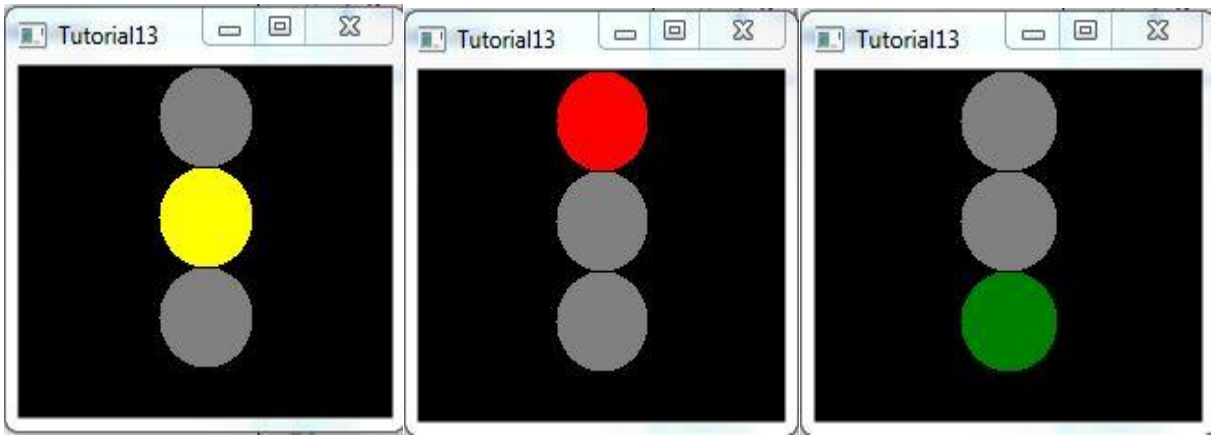
Task:

By this way place these rectangles to one column.

Tutorial #13

“Key Navigation”

In this tutorial we will work with keys: when we will press up and down keys , there will changes the colors.



Ok guys, let’s start with creating a new project, then create there our main “Black” Rectangle:

```
1   import QtQuick 1.0
2   Rectangle{
3       width:200
4       height: 175
5
6       color:"black"
```

After this create a Column and inside this column you should create another rectangle with Key Navigations:

```
7   Column{
8
9   Rectangle {
10      id: redRect
11      width: 50
12      height: 50
13      color: activeFocus ? "red" : "grey"
14      radius:100
15      x:75
16      y:100
17      KeyNavigation.down: yellowRect
18      KeyNavigation.up: greenRect
19      focus: true
20
21  }
```

Then you must create another two rectangles an also give to them Key Navigations.

After finishing run it and see the result!

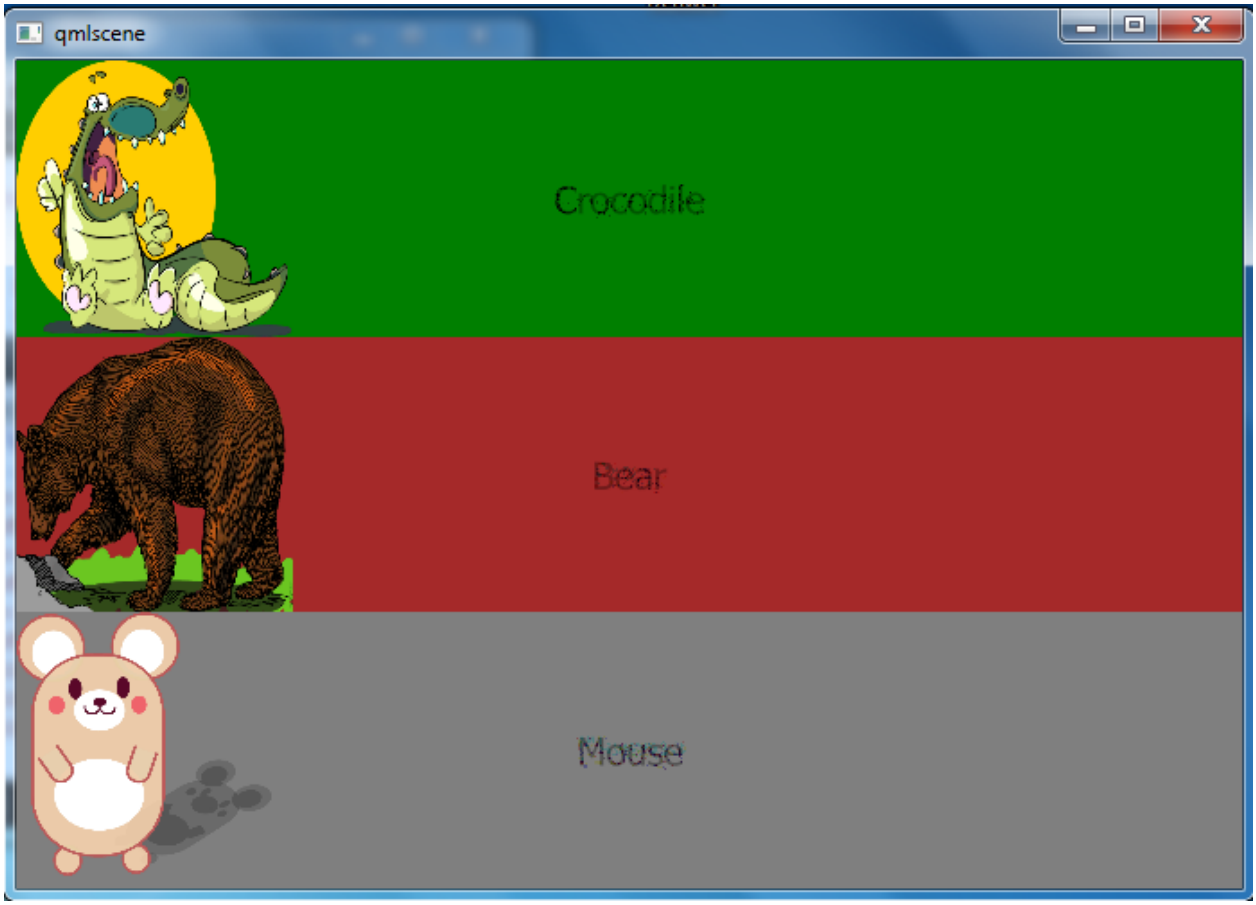
Task

Paste an image and try to move it to: right, left, up and down.

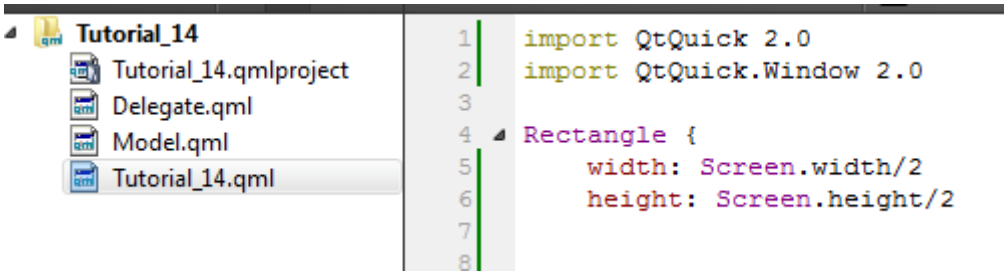
Tutorial #14

Model – View – Delegate

In this tutorial we will work with Model, View, Delegate elements. And by the ending our program will be like that:



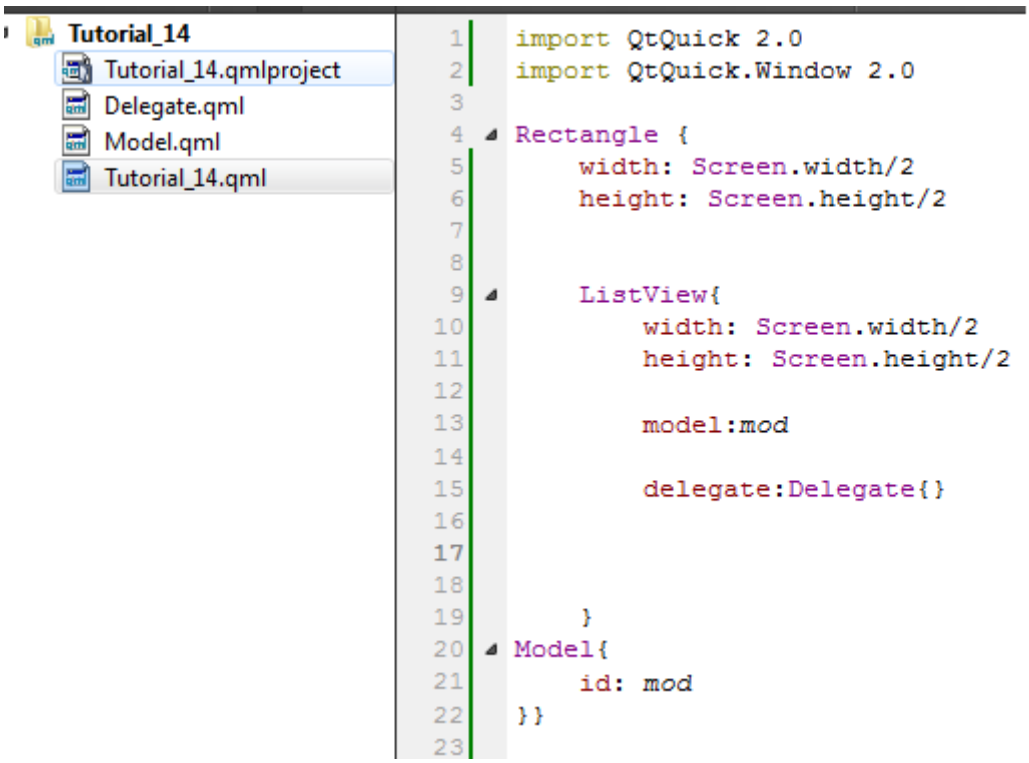
In this tutorial we need in version of Qt 5.0 creator. Let's start to write the code. First we will create a new project then write there:



Ok, then we need in Model, so we must create a file Model.qml and write there:

```
import QtQuick 2.0


ListModel{

    ListElement{
    name: "Crocodile"
    c:"green"
    s:"http://openclipart.org/people/Przylga/GreatSmileCrocodile.svg"
}
```

Like crocodile create there Bear and Mouse List Elements. After creating we need to write the View of this Lists in main file:




```
import QtQuick 2.0
import QtQuick.Window 2.0

Rectangle {
    width: Screen.width/2
    height: Screen.height/2


    ListView{
        width: Screen.width/2
        height: Screen.height/2

        model:mod

        delegate:Delegate{}



    }
Model{
    id: mod
}}
```

Then we should create our delegate file and create there an item:

```
import QtQuick 2.0
import QtQuick.Window 2.0

Item{

    width: Screen.width/2
    height: Screen.height/5
```

After, we should create the rectangles to each animal where they will locate:

```
Rectangle{
    id: mainRec
    width: Screen.width/2
    height: Screen.height/5
    color:c
```

And inside of this rectangle an image where our Delegate file will take the source from our Model file:

```
Image{
    source: s

    width: height
    height: Screen.height/5
}
```

And also the text:

```
    Text {
text: name
font.pixelSize: 21
anchors.centerIn: mainRec
    }
```

So, that is all, you can save it and see the result ☺

Task

Write the places where these animals lives **on the right side**.

Tutorial #15

“Video”

It’s our last tutorial on Qml ☺. You must have on this tutorial Qt 5 version because only there we have multimedia. Ok let’s start. It will be so easy for writing the code, but there will be some problem when you will run it.

Create a new project. Then on our main.qml file firstly create a rectangle:

```
import QtQuick 2.0
import QtMultimedia 5.0

Rectangle{
    width: 800
    height: 600
    color: "white"

```

After that in our rectangle we should create a Media Player:

```
    MediaPlayer {
        id: player
        source: "D://tutorial01.mp4"
        autoPlay: true
    }

```

Then finally create a Video output:

```

    VideoOutput {
        id: videoOutput
        source: player
        anchors.fill: parent
    }
}
```

That’s all. Thank you. You can run it and see the result ☺☺☺