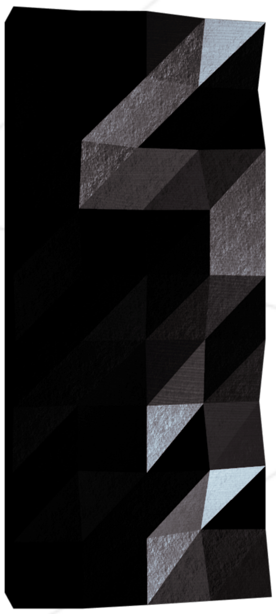

Custom items and components

Composing QML UIs

Qt training training@qt.io
42 Pedago Staff pedago@42.fr

Summary: Custom items can be created by composing them out of existing QML types. QML files, called components are the building blocks of a QML UI.

Contents

Chapter I

General instructions

Unless explicitely specified, the following rules will apply every day of this Piscine.

- This subject is the one and only trustable source. Don't trust any rumor.
- This subject can be updated up to one hour before the turn-in deadline.
- The assignments in a subject must be done in the given order. Later assignments won't be rated unless all the previous ones are perfectly executed.
- Be careful about the access rights of your files and folders.
- You must follow the `turn-in process` for each assignment. The url of your `GIT` repository for this day is available on your intranet.
- Your assignments will be evaluated by your Piscine peers.
- In addition to your peers evaluation, a program called the "Moulinette" will also evaluate your assignments. Fully automated, The Moulinette is tough and unforgiving in its evaluations. As a consequence, it is impossible to bargain your grade with it. Uphold the highest level of rigor to avoid unpleasant surprises.
- You must not leave in your turn-in repository any file other than the ones explicitly requested by the assignments.
- You have a question? Ask your left neighbor. Otherwise, try your luck with your right neighbor.
- Every technical answer you might need is available in the `mans` or in the Qt Documentation, which is available both in `QtCreator` IDE and on `http://doc.qt.io`.
- Remember to use the Piscine forum of your intranet and also Slack!
- You must read the examples thoroughly. They can reveal requirements that are not obvious in the assignment's description.
- Use the latest Qt version. Qt version 5.10 or newer is recommended.

- Many Qt classes are available as standard C++ classes. Qt classes should be preferred to standard classes, as you are supposed to learn Qt.
- Basic Qt coding style should be used, so read `http://wiki.qt.io/Qt_Coding_Style` before writing any assignments.
- Deprecated macros, functions or classes must not be used.
- Any build system, such as `qmake`, `cmake` or Qt Build System, can be used in the assignments. Instructions are based on `qmake`.
- Any editor or IDE, supporting Qt, can be used. However, `QtCreator` usage is strongly recommended.
- By Thor, by Odin! Use your brain!!!

Chapter II

Assignment 00: Custom item

Create a re-usable notification item, which contains at least a `Text` item. The parent item may change the border colour and text of the notification. The item should have a gradient background and scalable text. An example item is shown below.

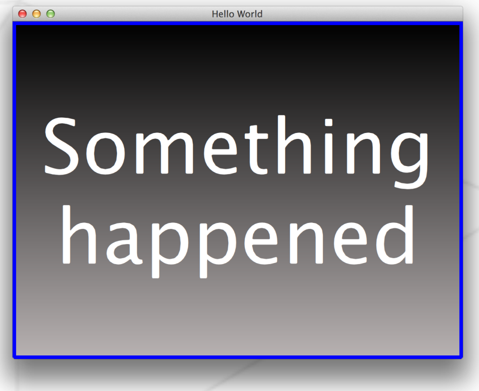

Figure II.1: Re-usable notification item

Chapter III

Assignment 01: Properties, signals, and methods

Extend the notification from the previous assignment.

- Add a custom signal with three arguments to the notification: mouse x and y coordinates and the mouse event itself. Emit the signal in `MouseArea` inside the notification. Implement a signal handler in the same place, where you instantiate the notification. The signal handler should just log the values of the arguments to the debug console.
- Add a method, which can be called to show the notification. By default the notification should not be visible. When the method is called, the method argument sets the notification text, sets the notification opacity to 0.7, and start the timer. When the timer expires, the opacity is set to 0.2. The notification should be visible only, if the opacity is greater than 0.2. In Windows 10, it may occur than the opacity value 0.2 is too small and the result is a blank, non-responsive window. If this occurs, increase the minimum opacity value to 0,5, for example.
- After the window is created, call your method to show the notification.

Chapter IV

Assignment 02: Animations

Continue with the application of the previous assignment. Animate the opacity change. The change should take 2 seconds. Implement a one second rotation animation for the text item. Start the animation, when the notification is clicked.

Chapter V

Assignment 03: States

Implement a light switch using states. In the figure below, the switch is on the left and the light rectangle on the right. When the switch is clicked, the light grey handle should be animated to the bottom and the light colour should be animated to red. Implement the animations using states and reversible transitions. Define a minimum number of states needed to implement the required functionality.

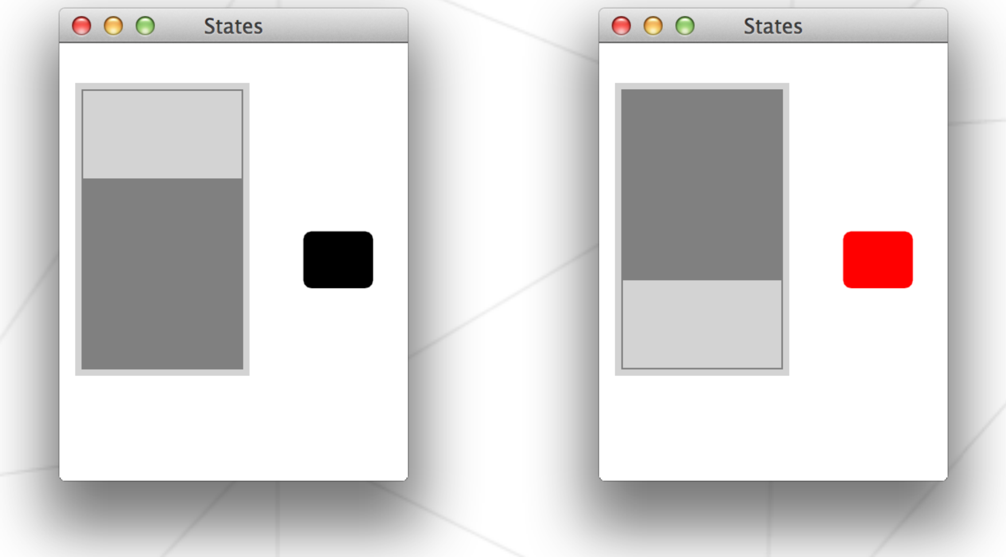

Figure V.1: Light switch

Chapter VI

Assignment 04: Modules

Let's get back to the notification item. Re-structure the notification application in such a way that the notification qml file is located in its own folder. Give the notification item some version number and used versioned import to import the folder to your `main.qml` file.

Chapter VII

Assignment 05: Component

Create a Qt Quick 2 project. Use an `Instantiator` type to create 100 rectangles as shown below.

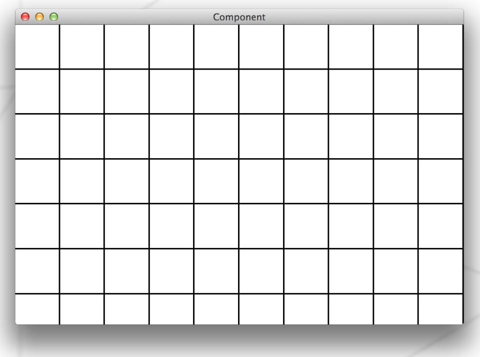

Figure VII.1: 100 instances of a component

- The window background should be black.
- The rectangles should be white.
- Position the rectangles into a grid. Use a few pixels spacing between rectangles. The grid should have ten columns.
- The rectangles should scale according to the window size. The height may equal to width to make scaling easier.

Chapter VIII

Assignment 06: Property scope and context

Let's extend the grid in the previous assignment to make it a bit more usable.

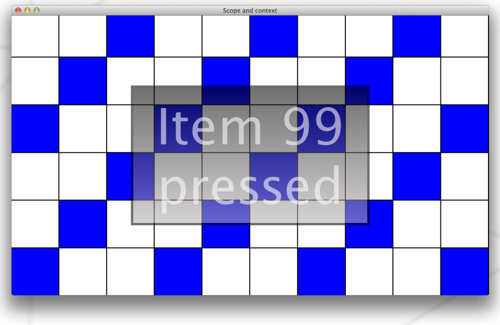

Figure VIII.1: Modified grid of rectangles

- `Flickable` places its children on a surface that can be dragged and flicked, causing the view onto the child items to scroll. This allows you to show large numbers of child items. Put the grid inside `Flickable` so that the user can flick up and down to access any rectangle item. Obviously no magic numbers can be used and any rectangle must be accessible.
- Change the colour of every third rectangle to blue. You must implement the colour change in the grid. Any colour assignment in the rectangles themselves, except the initial white colour, will result to the failure.

`Component scope includes the component's root object's properties.`

- Use the notification to show, which rectangle is clicked. The notification should be positioned in the middle of the window. The size must be large enough for the notification text to be visible.

Instantiator may expose useful context properties in the sub-context, where each rectangle is created.