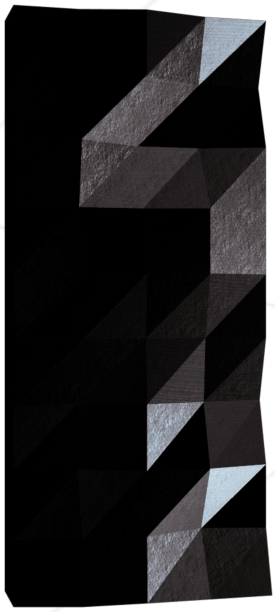

Image Sharing Application

Rush01

Staff 42 bocal@staff.42.fr

Summary: This document contains the subject for rush 01 of 42's Qt pool.

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

Day-specific rules

- The UI must be based on Qt Quick. It is possible to use HTML5 and OpenGL as well.
- Any Qt modules, except widgets, can be used.
- 3rd party libraries may be used.

Chapter III

Foreword

Something clever should be written here.

Chapter IV

Subject

Implement an image sharing application using Qt Quick and QML. There should be three ways to add images to the application.

- User picks up image files from the file system.
- User takes images with a camera.
- User browser images downloaded from https://www.flickr.com/. Selected image references are stored by the application.

Flickr should be a natural place to share the images, but let's use traditional sockets this time. So, several applications can be connected to a UI-less server, which just broadcasts shared images to all connected clients.

The essentials requirements for the game are:

- Get images from files, Flickr, and camera. The user must be able to choose, which files to pick from the file system or Flickr. The camera image can be also ignored, if the user does not like it.
- At least two views or application pages: The basic view, showing images, which the user has selected from the above services and the shared view, showing user's images and images, shared by someone else. Obviously, you need UI elements for browsing image files, taking a photo, and so on.
- Images can be organised freely in the basic view. All images do not need to be shown at the same time. If the user clicks on the image, the image is shown in a larger format with date, owner, owner picture, and location details, if available. There should be also a way to delete images from the model.
- Image data must be stored in a persistent model.
- The user must be able to pick up an image and stream it to the server. The server address may be the loop address or other hard-coded IP address.
- The user must be notified about arriving images. The images will be shown in the shared view, where the user can select images, which he or she wants to store into

the persistent image model. Other images are destroyed, when the user leaves the shared view.

- Obviously, some kind of an indicator is needed to show the number of clients, connected to the server.

IV.1 Score

Rather than implementing the best image sharing application in the world, the goal is to write good QML code. The following aspects affect your score.

- Clear separation of UI and business logic
- Use of QML files, i.e., how the applications is strctured into a set of QML files
- Quality of QML components: re-usability, use of properties, bindings, signals
- C++ and QML separation and integration
- Use of model/view
- Fluid user experience, animations, effects

If you have spare time to implement even more features, the following list provides ideas to get more bonuses. However, keep in mind that we'll consider them useless if you haven't done the other stuff before.

- All eye candy
- Use of cloud or database
- Image editor, possibility to modify images with all kinds of effects