

Archived:Adding Options menu, panes, and icon to a Qt for Symbian application



Archived: This article is [archived](#) because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template `{{ReviewForRemovalFromArchive|user=~~~~~|write your reason here}}`.

Qt Quick should be used for all UI development on mobile devices. The approach described in this article (using C++ for the Qt app UI) is deprecated.

Overview

Standard Qt applications can be compiled for Symbian devices. This code snippet demonstrates how to modify the source code so that when building for the Symbian platform, the application has the following Symbian style elements:

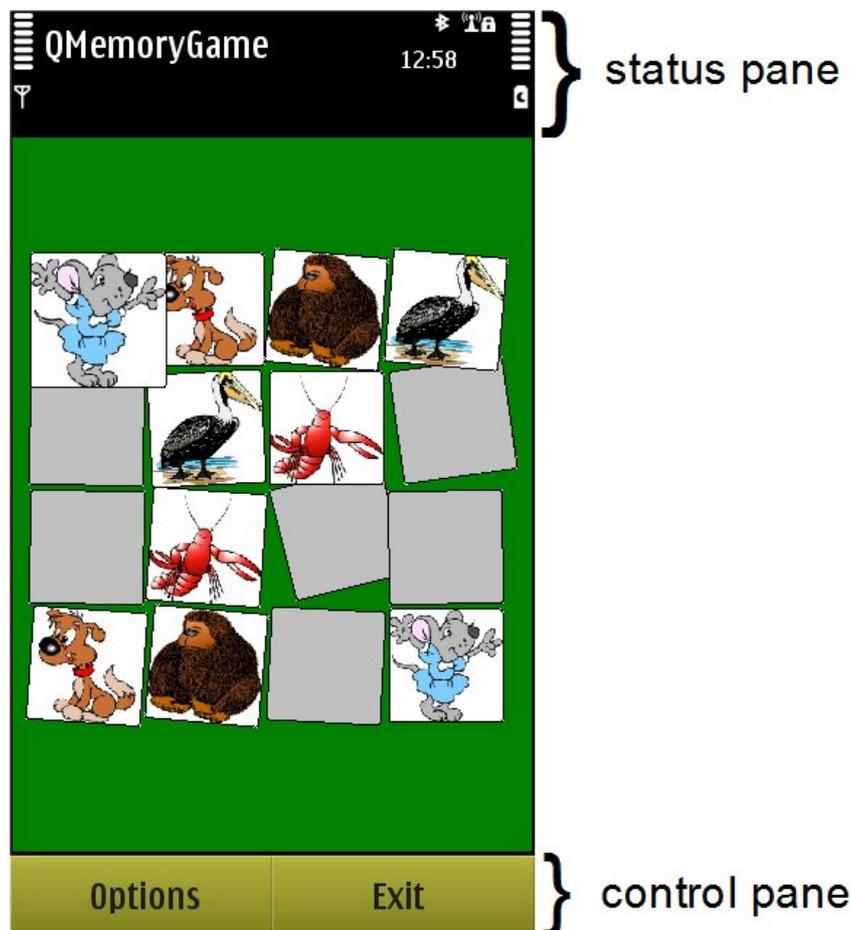
- Status pane
- Control pane
- Application menu (Options & Back)
- Application icon

Note: In order to use this code, you need to have Qt installed on your platform.

Preconditions

None

Showing status pane and control pane



Call `QMainWindow::showMaximized()` to display the application status and control panes. The application already has these panes, all you need to do is arrange the space for them to appear.

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    MyMainWindow *mainwindow = new MyMainWindow;
```

```
mainwindow->showMaximized();

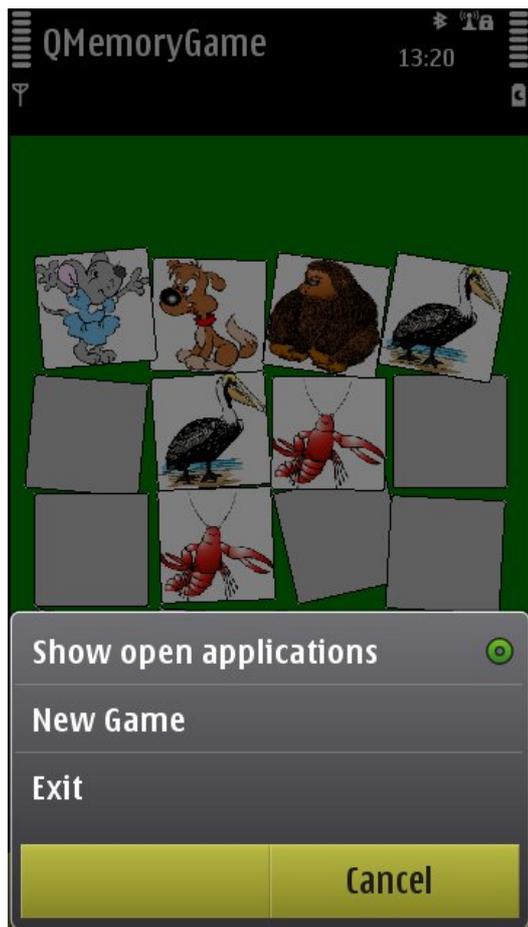
ret = app.exec();
delete mainwindow;
return ret;
}
```

Defining application icon

This section explains how to set up the SVG icon for the application. Add the following Symbian C++ section to the Qt project file (.pro). The example icon.svg is in the same folder as the project file.

```
symbian {
    ICON = icon.svg
}
```

Creating Options menu



Menus can be created into QMainWindow, QDialog or into QWidget in Symbian. QMainWindow has predefined Symbian CBA buttons (**Options/Exit**), but for QDialog or QWidget you need to define them.

When Qt menus and actions are populated into a menu bar, they appear in the **Options** menu and can be accessed using the left softkey. Note that in Qt, you can add menu actions directly to the menu bar because a Symbian **Options** menu item has already been created. On other Qt platforms, you first have to create the **Options** menu item and after that add the menu actions to the menu.

Defining menus in QMainWindow

```
this->menuBar()->addAction("Menu item", this, SLOT(someSlot()));
```

The someSlot() slot is called when an item is selected from the **Options** menu.

Defining menus in QDialog or QWidget

Remember that Symbian CBA buttons have to be defined when using QDialog or QWidget.

```
// Instance variable for header
QMenu* m_menu;
```

```
// Create menu
m_menu = new QMenu(this);
m_menu->addAction("Menu item", this, SLOT(someSlot()));

// Create Options CBA
QAction *optionsAction = new QAction("Options", this);
// Set defined menu into Options button
optionsAction->setMenu(m_menu);
optionsAction->setSoftKeyRole(QAction::PositiveSoftKey);
addAction(optionsAction);

// Create Exit CBA
QAction *backSoftKeyAction = new QAction(QString("Exit"), this);
backSoftKeyAction->setSoftKeyRole(QAction::NegativeSoftKey);
// Exit button closes the application
QObject::connect(backSoftKeyAction, SIGNAL(triggered()),
QApplication::instance(), SLOT(quit()));
addAction(backSoftKeyAction);

// or Create Back CBA
// Back button closes the QDialog
// QAction *backSoftKeyAction = new QAction(QString("Back"), this);
// backSoftKeyAction->setSoftKeyRole(QAction::NegativeSoftKey);
// QObject::connect(backSoftKeyAction, SIGNAL(triggered()),
// this, SLOT(close()));
// addAction(backSoftKeyAction);
```

Postconditions

The Qt application contains an application icon, Status and Control panes, and an **Options** menu.

See also

- QMenu
- QMainWindow
- QDialog

