# Lucrarea 1

# Noţiuni introductive de Qt

**Qt**[1] este un mediu de dezvoltare care include biblioteci de clase C++ şi unelte de dezvoltare de aplicaţii pentru diverse sisteme de operare: Microsoft Windows, MAC OS sau Linux.

Bibliotecile **Qt** cuprind peste 400 de clase C++, care încapsulează infrastructura necesară pentru dezvoltarea de aplicaţii. **Qt** API[2] include clase pentru realizarea de interfeţe grafice utilizator, pentru programarea în reţea, baze de date sau integrare OpenGL[3].

## 1.1 Clasa `QApplication`

Clasa `QApplication` controlează şi gestionează interfaţa grafică a unei aplicaţii. Conţine bucla principală de evenimente a aplicaţiei, în care sunt procesate evenimentele provenind de la sistemul de ferestre. De asemenea gestionează configurările aplicaţiei, fazele de iniţializare şi terminare a aplicaţiei.

Orice aplicaţie **Qt** este un obiect `QApplication` indiferent de numărul de ferestre ale aplicaţiei. Obiectul este accesibil prin apelarea funcţiei membre `instance()`, care returnează un pointer către acest obiect.

## 1.2 Clasa `QImage`

Clasa `QImage` pune la dispoziţie o reprezentare independentă de platformă a unei imagini, care permite accesul direct la valorile pixelilor. **Qt** oferă patru clase pentru manipularea imaginilor: `QImage`, `QPixmap`, `QBitmap` şi `QPicture`.

Clasa `QImage` este proiectată şi optimizată pentru operaţii de intrare/ieşire şi pentru acces direct la pixelii imaginii, pe când clasa `QPixmap` este proiectată şi optimizată pentru vizualizarea imaginilor pe ecran. Clasa `QBitmap`

[1] Trolltech, http://www.trolltech.com
[2] Application Program Interface.
[3] Limbaj de grafică 3D dezvoltat de firma Silicon Graphics.

moşteneşte clasa `Pixmap` şi este folosită pentru imagini binare, iar clasa `QPicture` este un "paint device". În continuare ne vom concentra atenţia asupra clasei `QImage`.

Clasa `QImage` poate manipula o serie de formate de imagine, care includ fie imagini monocrome, fie reprezentate pe 8 biţi sau 32 biţi. Clasa pune la dispoziţie o colecţie de funcţii care pot fi folosite pentru obţinerea de informaţii despre imagine sau care permit anumite transformări ale imaginii.

### 1.2.1 Manipularea imaginilor

Clasa `QImage` oferă câteva moduri de a încărca o imagine dintr-un fişier: la instanţierea obiectului QImage sau folosind funcţiile `loadFromData()` sau `load()`, care vor fi apelate după crearea obiectului `QImage`. Pentru a salva un obiect de tip `QImage` se foloseşte funcţia `save()`.

Următorul exemplu ilustrează folosirea funcţiei `load()`:

```
QImage image;
QString filename = QFileDialog :: getOpenFileName();
image.load( filename, 0 );
```

Lista completă a formatelor de fişiere cunoscute este disponibilă prin intermediul a două metode: `QImageReader::supportedImageFormats()` şi `QImageWriter::supportedImageFormats()`. Implicit, **Qt** suportă următoarele formate de fişiere:

| **Format** | **Descriere** | **Operaţii** |
|---|---|---|
| BMP | Windows Bitmap | Read/Write |
| GIF | Graphic Interchange Format (opţional) | Read |
| JPG | Joint Photographic Experts Group | Read/Write |
| JPEG | Joint Photographic Experts Group | Read/Write |
| PNG | Portable Network Graphics | Read/Write |
| PBM | Portable Bitmap | Read |
| PGM | Portable Graymap | Read |
| PPM | Portable Pixmap | Read/Write |
| XBM | X11 Bitmap | Read/Write |
| XPM | X11 Pixmap | Read/Write |

Tabelul 1.1: Formate de fişiere imagine suportate de **Qt**.

### 1.2.2 Atribute ale imaginii

`QImage` oferă o colecţie de funcţii pentru obţinerea de informaţii despre atributele imaginii, cum ar fi geometria imaginii sau informaţii despre paleta de culori.

| Atribute | Funcţii |
|---|---|
| Geometrie | Funcţiile `size()`, `width()`, `height()`, `dotsPerMeterX()` şi `dotsPerMeterY()` oferă informaţii despre dimensiunile şi aspectul imaginii. |
| Culoare | Culoarea unui pixel poate fi aflată specificând coordonatele pixelului funcţiei `pixel()`, care returnează culoarea ca o valoare `QRbg`. În cazul imaginilor monocrome sau cu nivele de gri, funcţiile `numColors()` şi `colorTable` oferă informaţii despre componentele de culoare ale imaginii. |
| Nivel jos | Funcţia `depth()` returnează numărul de biţi pe care este reprezentată valoarea unui pixel: 1 (imagini monocrome), 8 sau 32. Funcţiile `format()`, `bytesPerLine()` şi `numBytes()` oferă informaţii despre modul de stocare a imaginii. |

Tabelul 1.2: Funcţii de aflare a atributelor unei imagini.

### 1.2.3 Manipularea pixelilor

În cazul în care culorile pixelilor sunt stocate pe 32 biţi, funcţia `setPixel` se poate utiliza pentru modificarea valorii unui pixel specificat prin coordonatele sale. Noua sa valoare este un cvadruplu ARGB[4], fiind unul din argumentele funcţiei. Pentru a specifica valoarea unui pixel în modelul RGB se foloseşte funcţia `qRgb` care returnează un obiect QRgb.

În continuare este prezentat un exemplu de folosire a funcţiei `setPixel`. Se crează mai întâi un obiect `QImage`, reprezentând o imagine de dimensiune 3 x 3.

```
QImage image(3, 3, QImage::Format_RGB32);
QRgb value;

value = qRgb(189, 149, 39); // 0xffbd9527
image.setPixel(1, 1, value);

value = qRgb(122, 163, 39); // 0xff7aa327
image.setPixel(0, 1, value);
image.setPixel(1, 0, value);
```

[4]Alpha Red Green Blue.

```
value = qRgb(237, 187, 51); // 0xffedba31
image.setPixel(2, 1, value);
```

### 1.2.4 Formate de imagine

Fiecare pixel stocat într-un obiect de tip `QImage` este reprezentat ca un întreg. Dimensiunea acestui întreg depinde de format. **Qt** suportă imagini monocrome, reprezentate pe 1 bit şi imagini reprezentate pe 8 sau 32 biţi.

Imaginile monocrome sunt stocate folosind indici de 1 bit într-o tabelă de culoare cu doar două intrări (culori). În funcţie de ordinea de stocare a biţilor, big endian sau little endian, diferenţiem două tipuri de imagini monocrome.

Imaginile reprezentate pe 8 biţi sunt stocate folosind indici de 8 biţi într-o tabelă de culoare, având un byte per pixel. Tabela de culoare este un obiect `QVector<QRgb>`.

Imaginile reprezentate pe 32 biţi nu au tabelă de culoare, fiecare pixel conţinând o valoare `QRgb`. Cea mai răspândită modalitate de a reprezenta tripletul RGB pe 32 biţi este următoarea: 0xffRRGGBB, în care se folosesc câte 8 biţi pentru fiecare componentă.

Formatul unei imagini se poate determina folosind funcţia `format()`.

Pentru a converti o imagine într-un nou format se foloseşte funcţia `convertToFormat()`, care acceptă ca argument noul format al imaginii.