KREIRANJE JEDNOSTAVNE APLIKACIJE ZA OBRADU SLIKA KORIŠĆENJEM QT FRAMEWORK-A

MILICA STANKOVIĆ 1986

ŠTA JE QT?

- Qt (*kjut*) je višeplatformski C++ razvojni okvir za kreiranje grafičkih korisničkih informacija
- Podržava desktop, mobilne i embeded uređaje
- Pruža širok spektar vidžeta i alata za kreiranje nativnih aplikacija
- Široko korišćen u industriji za aplikacije koje zahtevaju visoke performanse

PRIBAVLJANJE I INSTALACIJA QT-A

- Posetite web stranu: https://www.qt.io/download
- Preuzmite ***Qt Online Installer*** za Vaš operativni sistem
- Napravite Qt nalog (besplatan za open-source korišćenje)
- Pokrenite instalaciju i izaberite komponente: *Qt version* (na primer Qt6.x ili Qt5.x) i *Desktop development kits*
- Instalirajte *Qt Creator IDE*
- Pokrenite *Qt Creator* i napravite novi projekat

GLAVNE KARAKTERISTIKE PROJEKTA

- Učitavanje, pamćenje, promena veličine i sečenje slike
- Rotacija i filteri
- Undo/redo funkcionalnost
- Zumiranje i skalianje slike na prozor
- *Drag and drop* i metadata
- Prečice na tastaturi

.PRO FAJL

- .pro fajl prestavlja fajl koji koristi *qmake* (bild sistem u Qt frameqork-u) za definisanje kako projekat treba da bude izbildovan
- Sadrži sve neophodne informacije kao što su:
  - Koji Qt moduli treba da budu korišćeni (*widget*-i, *gui*, *network*)
  - Koji *.cpp* i *.h* fajlovi treba da budu deo projekta
  - Postavke kompajlera i konfiguracioni flegovi
  - Resursi, prevođenja i UI fajlovi

.PRO FAJL

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++17

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    cropDialog.cpp \
    historyPanel.cpp \
    imageArea.cpp \
    imageProcessor.cpp \
    main.cpp \
    mainwindow.cpp \
    metadataDialog.cpp \
    resizeDialog.cpp

HEADERS += \
    cropDialog.h \
    historyPanel.h \
    imageArea.h \
    imageProcessor.h \
    mainwindow.h \
    metadataDialog.h \
    resizeDialog.h

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

```

QDIALOG

- **QDialog** je popup prozor ili sekundarni prozor aplikacije, koji se tipično koristi za:
  - Pribavljanje korisničkog unosa (ime, fajl, veličina ili slično)
  - Prikazivanje poruke
  - Prikazivanje i omogućavanje dodatnih podešavanja ili opcija
  - Prikazivanje fokusiranog UI koji blokira glavni prozor (modal)
- **Qdialog** je osnovna klasa za proizvoljne dijaloge, ali postoje i ugrađene nasleđene komponente:
  - **QFileDialog, QColorDialog, QFontDialog**

RAD SA FAJLOVIMA

- **QFileDialog** je standardni Qt dijalog koji omogućava korisnicima da izaberu fajlove ili direktorijume iz fajl sistema
- Koristi se za otvaranje, čuvanje ili biranje fajlova ili direktorijuma u nativnom ili proizvoljnom prozoru
- Često korišćenje ugrađene funkcije:
  - getOpenFileName()
  - getSaveFileName()
  - getOpenFileNames()
  - getExistingDirectory()

RAD SA FAJLOVIMA (UČITAVANJE SLIKE)

```
void MainWindow::openImage() {
    QString fileName = QFileDialog::getOpenFileName(this, "Open Image", QString(),
                                                    "Images (*.png *.jpg *.bmp *.jpeg *.gif *.tiff)");

    if (fileName.isEmpty()) return;

    QImage loaded(fileName);
    if (loaded.isNull()) {
        QMessageBox::warning(this, "Open Image", "Failed to load image.");
        return;
    }
    updateImage(loaded);
    updateWindowTitle();
}
```

RAD SA FAJLOVIMA (ČUVANJE SLIKE)

```
void MainWindow::saveImage() {
    if (currentImage.isNull()) return;
    QString fileName = QFileDialog::getSaveFileName(this, "Save Image", QString(),
                                                    "PNG (*.png);;JPEG (*.jpg *.jpeg);;BMP (*.bmp)");
    if (fileName.isEmpty()) return;

    if (!currentImage.save(fileName)) {
        QMessageBox::warning(this, "Save Image", "Failed to save image.");
    }
}
```

PROIZVOLJNI DIJALOG

```
ResizeDialog::ResizeDialog(int currentWidth, int currentHeight, QWidget *parent)
    : QDialog(parent),
    aspectRatio(double(currentWidth) / currentHeight)
{
    widthLineEdit = new QLineEdit(QString::number(currentWidth), this);
    heightLineEdit = new QLineEdit(QString::number(currentHeight), this);
    aspectRatioCheckBox = new QCheckBox("Lock Aspect Ratio", this);
    aspectRatioCheckBox->setChecked(true);

    widthLineEdit->setValidator(new QIntValidator(1, 10000, this));
    heightLineEdit->setValidator(new QIntValidator(1, 10000, this));

    // Layout
    auto layout = new QVBoxLayout(this);
    auto wLayout = new QHBoxLayout();
    wLayout->addWidget(new QLabel("Width:", this));
    wLayout->addWidget(widthLineEdit);
    layout->addLayout(wLayout);

    auto hLayout = new QHBoxLayout();
    hLayout->addWidget(new QLabel("Height:", this));
    hLayout->addWidget(heightLineEdit);
    layout->addLayout(hLayout);

    layout->addWidget(aspectRatioCheckBox);

    // Buttons
    auto buttons = new QHBoxLayout();
    QPushButton *okBtn = new QPushButton("OK", this);
    QPushButton *cancelBtn = new QPushButton("Cancel", this);
    buttons->addWidget(okBtn);
    buttons->addWidget(cancelBtn);
    layout->addLayout(buttons);

    connect(okBtn, &QPushButton::clicked, this, &ResizeDialog::accept);
    connect(cancelBtn, &QPushButton::clicked, this, &ResizeDialog::reject);

    connect(widthLineEdit, &QLineEdit::textChanged, this, &ResizeDialog::onWidthChanged);
    connect(heightLineEdit, &QLineEdit::textChanged, this, &ResizeDialog::onHeightChanged);
}
```

QIMAGE

- **QImage** je Qt komponenta koja se koristi za manipulaciju slika
- Najčešće operacije za koje se koristi su:
  - Učitavanje slika sa diska
  - Postavljanje filtera na sliku
  - Izmena pojedinačnih piksela
  - Čuvanje slika na disku
- Nezavisna od GUI i optimizovana za editovanje slika, ne za prikaz

QIMAGE (UČITAVANJE I ČUVANJE SLIKE)

```
QImage loaded(fileName);
if (loaded.isNull()) {
    QMessageBox::warning(this, "Open Image", "Failed to load image.");
    return;
}
```

```
if (!currentImage.save(fileName)) {
    QMessageBox::warning(this, "Save Image", "Failed to save image.");
}
```

QIMAGE (FILTERI)

- Pomoću *QImage::Format* jednostavna primena osnovnih filtera poput *grayscale*
- Ugrađena funkcija *convertToFormat* menja format slike njegovim prosleđivanjem

```
QImage ImageProcessor::toGrayscale(const QImage &image) {
    return image.convertToFormat(QImage::Format_Grayscale8);
}
```

QIMAGE (SKALIRANJE SLIKE)

- *scaled* funkcija se koristi za promenu veličine slike preko zadate širine i visine, a opciono čuvajući njene proporcije i izbor kvaliteta/brzine transformacije

```
QImage resized = currentImage.scaled(dlg.getNewWidth(), dlg.getNewHeight(),
                                     Qt::IgnoreAspectRatio, Qt::SmoothTransformation);
```

QTRANSFORM

- **Qtransform** je komponenta u Qt-u koja se koristi za geometrijske transformacije 2D objekata, kao što su *QImage*, *QPixmap*, *QPainterPath*…
- Predstavlja 3x3 matricu koja se koristi za:
  - Rotaciju
  - Skaliranje
  - Translaciju
  - Smicanje

QTRANSFORM (PRIMER)

```
void MainWindow::rotate90() {
    if (currentImage.isNull()) return;

    QTransform transform;
    transform.rotate(90);
    QImage rotated = currentImage.transformed(transform);
    updateImage(rotated);
}
```

QACTION

- **QAction** predstavlja akciju od strane korisnika, kao što je otvaranje, čuvanje, undo i slično
- Uglavnom se koristi za:
  - Meni
  - Toolbar
  - Prečice sa tastature
- S obrizom na to da se ponaša kao apstrakcija, moguće je napraviti jednu akciju i koristiti je za više korisničnih unosa

QACTION (MENI)

```
// Edit menu actions
QAction *resizeAction = new QAction(tr("&Resize"), this);
QAction *cropAction = new QAction(tr("&Crop"), this);
QAction *rotateAction = new QAction(tr("&Rotate 90 degrees"), this);
QAction *rotateCustomAction = new QAction(tr("Rotate custom angle"), this);
QAction *undoAction = new QAction(tr("&Undo"), this);
QAction *redoAction = new QAction(tr("&Redo"), this);

editMenu->addAction(resizeAction);
editMenu->addAction(cropAction);
editMenu->addAction(rotateAction);
editMenu->addAction(rotateCustomAction);
editMenu->addSeparator();
editMenu->addAction(undoAction);
editMenu->addAction(redoAction);
```

QACTION (PREČICE NA TASTATURI)

```
loadAction->setShortcut(QKeySequence::Open);           // Ctrl+O
saveAction->setShortcut(QKeySequence::Save);           // Ctrl+S
undoAction->setShortcut(QKeySequence::Undo);           // Ctrl+Z
redoAction->setShortcut(QKeySequence::Redo);           // Ctrl+Y or Ctrl+Shift+Z
zoomInAction->setShortcut(QKeySequence("Ctrl++"));     // Ctrl + Plus
zoomOutAction->setShortcut(QKeySequence("Ctrl+-"));    // Ctrl + Minus
grayscaleAction->setShortcut(QKeySequence("Ctrl+G")); // Ctrl + G
invertAction->setShortcut(QKeySequence("Ctrl+I"));     // Ctrl + I
```

QACTION (POVEZIVANJE AKCIJE)

```
// Connect actions to slots (implement these slots in MainWindow)
connect(loadAction, &QAction::triggered, this, &MainWindow::openImage);
connect(saveAction, &QAction::triggered, this, &MainWindow::saveImage);
connect(saveAsAction, &QAction::triggered, this, &MainWindow::saveImageAs);
connect(zoomInAction, &QAction::triggered, this, &MainWindow::zoomIn);
connect(zoomOutAction, &QAction::triggered, this, &MainWindow::zoomOut);
connect(exitAction, &QAction::triggered, this, &QMainWindow::close);
```

DRAG & DROP

- Kako bi se omogućio *drag&drop*, potrebno je pozvati *setAcceptDrops* funkciju u konstruktoru glavnog prozora
- Ugrađeni event-i **QDragEnterEvent** i **QDropEvent**

```
void MainWindow::dragEnterEvent(QDragEnterEvent *event) {
    if (event->mimeData()->hasUrls()) {
        event->acceptProposedAction();
    }
}

void MainWindow::dropEvent(QDropEvent *event) {
    auto urls = event->mimeData()->urls();
    if (!urls.isEmpty()) {
        QString filePath = urls.first().toLocalFile();
        QImage loaded(filePath);
        if (!loaded.isNull()) {
            updateImage(loaded);
        }
    }
}
```

UNDO/REDO

- Qt omogućava jednostavno izvođenje *undo* i *redo* operacija zahvaljujući ugrađenoj strukturi **QStack**
- Kreiranje dva steka – jedan za undo i drugi za redo operaciju
- Pre primene operacije (na primer *grayscale*) pozovemo ugrađenu funkciju *push* na undo stek i obrišemo redo stek
- Lako povezivanje sa prečicama na tastaturi (*Ctrl+Z, Ctrl+Y*)

UNDO-/REDO

```
void MainWindow::undo() {
    if (undoStack.isEmpty()) return;
    redoStack.push(currentImage);
    currentImage = undoStack.pop();
    imageArea->setImage(currentImage);
    imageArea->resize(currentImage.size());
    updateWindowTitle();
}

void MainWindow::redo() {
    if (redoStack.isEmpty()) return;
    undoStack.push(currentImage);
    currentImage = redoStack.pop();
    imageArea->setImage(currentImage);
    imageArea->resize(currentImage.size());
    updateWindowTitle();
}
```

HVALA NA PAŽNJI!