GUI aplikacije i PyQt

Slajdovi za predmet Osnove programiranja

Katedra za informatiku, Fakultet tehničkih nauka, Novi Sad

2022.

Ciljevi

- GUI = graphical user interface
- savlađivanje osnovnih koncepata pisanja GUI aplikacija

GUI biblioteke i Python

- pisanje programa koji ima GUI od nule je ogroman posao
- puno istih funkcija bismo morali da pišemo iznova za svaki program
- $\Rightarrow$ možemo da koristimo biblioteku klasa
- za Python ih ima mnogo:
  - TkInter
  - wxPython
  - PyQt
  - PyGTK
  - ...

PyQt

- Qt je cross-platform biblioteka za C++
  - Windows, Linux, Mac OSX, Symbian, Android, iOS
- PyQt je Python omotač oko Qt biblioteke

Instalacija PyQt

```
pip install pyqt5 pyqt5-tools
```

Struktura GUI aplikacije

- bez obzira na izabranu GUI biblioteku, svaki program ima sličnu strukturu
- program se ne izvršava linearno (od početka prema kraju, bez pauze), već se izvršava samo u pojedinim trenucima

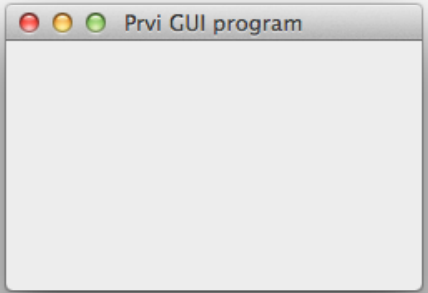
1. inicijalizacija programa prilikom pokretanja
   - inicijalizuju se interne strukture podataka
2. prepuštanje kontrole OS-u i čekanje na događaje
   - mouse click, key press, touch, swipe, ...
3. obrada nastalog događaja i prelazak na 2

Prva GUI aplikacija

```
import sys
from PyQt5 import QtGui, QtWidgets

def main():
    # inicijalizacija
    app = QtWidgets.QApplication(sys.argv)
    w = QtWidgets.QWidget()
    w.resize(250, 150)
    w.move(300, 300)
    w.setWindowTitle('Prvi GUI program')
    w.show()
    # ovde je petlja za obradu događaja
    sys.exit(app.exec_())

if __name__ == '__main__':
```

Struktura GUI aplikacije

- prozor već ima dosta funkcionalnosti (close, resize)
- nismo morali da pišemo sve od početka

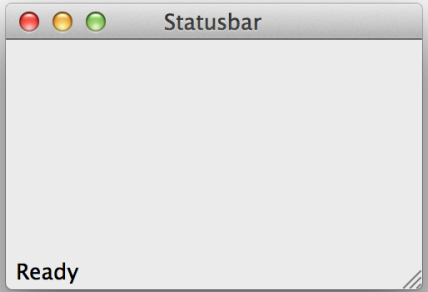


Dodajemo status bar

```
class Example(QtWidgets.QMainWindow):
    def __init__(self):
        super(Example, self).__init__()
        self.initUI()

    def initUI(self):
        # ovde se uključuje status bar
        self.statusBar().showMessage('Ready')
        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('Statusbar')
        self.show()

def main():
    app = QtWidgets.QApplication(sys.argv)
    ex = Example()
```

Dodajemo status bar 2

- tekst u status baru se definiše pomoću `setMessage`

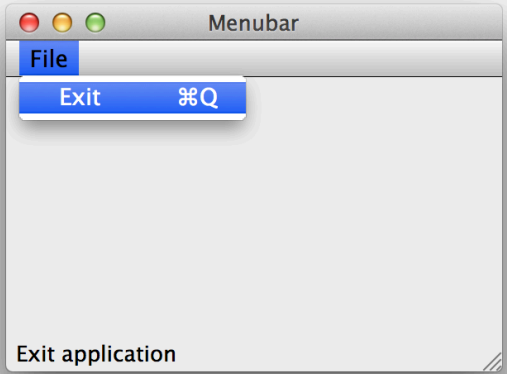


Dodajemo meni

```
class Example(QtWidgets.QMainWindow):
    def initUI(self):
        exitAction = QtWidgets.QAction(QtGui.QIcon('exit.png'), '&Exit', s
        exitAction.setShortcut('Ctrl+Q')
        exitAction.setStatusTip('Exit application')
        exitAction.triggered.connect(QtWidgets.qApp.quit)

        self.statusBar().showMessage('Ready')
        menubar = self.menuBar()
        fileMenu = menubar.addMenu('&File')
        fileMenu.addAction(exitAction)

        self.setGeometry(300, 300, 300, 200)
        self.setWindowTitle('Menubar')
        self.show()
```

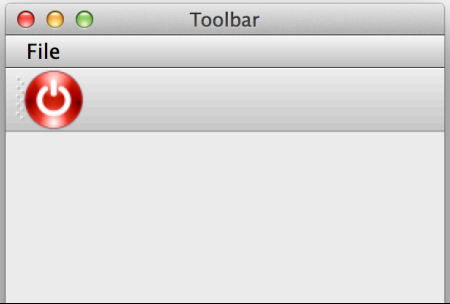
Dodajemo meni 2

- klik na stavku menija ili Ctrl-Q ili Alt-F,E zatvaraju program



Dodajemo toolbar

```
class Example(QtWidgets.QMainWindow):
    def initUI(self):
        ...
        self.toolbar = self.addToolBar('Exit')
        self.toolbar.addAction(exitAction)
```



Raspored komponenti na prozoru

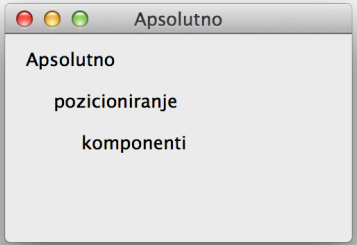
- komponente (widgets) se mogu rasporediti na više načina

1. apsolutno pozicioniranje (piksel koordinate)
2. neki drugi način

Apsolutno pozicioniranje

```
class Example(QtWidgets.QMainWindow):
    def initUI(self):
        ...
        lbl1 = QtWidgets.QLabel('Apsolutno', self)
        lbl1.move(15, 10)
        lbl2 = QtWidgets.QLabel('pozicioniranje', self)
        lbl2.move(35, 40)
        lbl3 = QtWidgets.QLabel('komponenti', self)
        lbl3.move(55, 70)
        self.show()
```

Apsolutno pozicioniranje 2

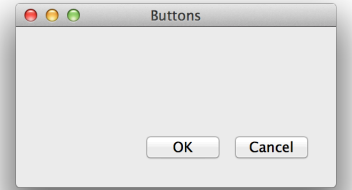


- teško je prilagoditi prikaz različitim uslovima
  - različiti OS
  - različita veličina osnovnog fonta

Pozicioniranje komponenti: Box Layout

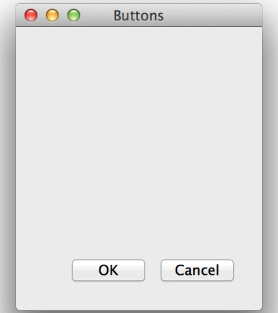
```
class Example(QtWidgets.QMainWindow):
    def initUI(self):
        ...
        okButton = QtWidgets.QPushButton("OK")
        cancelButton = QtWidgets.QPushButton("Cancel")
        hbox = QtWidgets.QHBoxLayout()
        hbox.addStretch(1)
        hbox.addWidget(okButton)
        hbox.addWidget(cancelButton)
        vbox = QtWidgets.QVBoxLayout()
        vbox.addStretch(1)
        vbox.addLayout(hbox)
        self.setLayout(vbox)
```

Pozicioniranje komponenti: Box Layout 2



- OK i Cancel su u horizontalnom boxu, gurnuti u desno
- horizontalni box je u vertikalnom boxu, gurnut na dole

Pozicioniranje komponenti: Box Layout 3

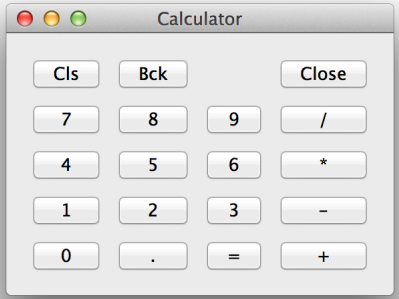


- isti odnos je i nakon promene veličine prozora

Pozicioniranje komponenti: QGridLayout

```
class Example(QtWidgets.QMainWindow):
    def initUI(self):
        ...
        grid = QtWidgets.QGridLayout()
        self.setLayout(grid)
        names = ['Cls', 'Bck', '', 'Close',
                 '7', '8', '9', '/',
                 '4', '5', '6', '*',
                 '1', '2', '3', '-',
                 '0', '.', '=', '+']
        positions = [(i,j) for i in range(5) for j in range(4)]
        for position, name in zip(positions, names):
            if name == '':
                continue
            button = QtWidgets.QPushButton(name)
            grid.addWidget(button, *position)
```

Pozicioniranje komponenti: QGridLayout 2



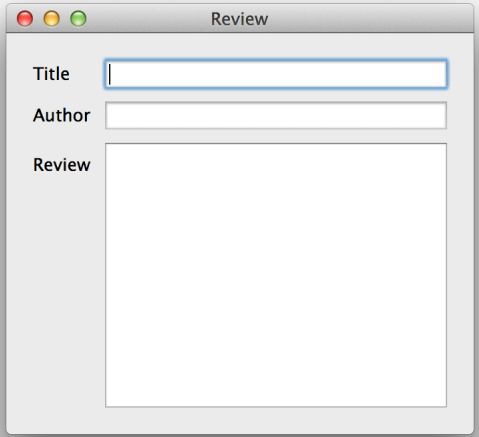
- raspored je otporan na promenu veličine prozora

Pozicioniranje komponenti: QGridLayout 3

- komponente mogu da se prostiru preko više kolona ili redova

```
class Example(QtGui.QMainWindow):
    def initUI(self):
        ...
        title = QtWidgets.QLabel('Title'); author = QtWidgets.QLabel('Author')
        review = QtWidgets.QLabel('Review'); titleEdit = QtWidgets.QLineEdit()
        authorEdit = QtWidgets.QLineEdit(); reviewEdit = QtWidgets.QTextEdit()
        grid = QtWidgets.QGridLayout()
        grid.setSpacing(10)
        grid.addWidget(title, 1, 0)
        grid.addWidget(titleEdit, 1, 1)
        grid.addWidget(author, 2, 0)
        grid.addWidget(authorEdit, 2, 1)
        grid.addWidget(review, 3, 0)
        # reviewEdit se prostire preko 5 redova
        grid.addWidget(reviewEdit, 3, 1, 5, 1)
        self.setLayout(grid)
```

Pozicioniranje komponenti: QGridLayout 4



Događaji u korisničkom interfejsu

- za svaki događaj vezane su tri stvari
  - izvor: objekat čije stanje se menja
  - objekat: enkapsulira promenu stanja
  - odredište: želi da bude obavešten o događaju
- signal se emituje kada se desi događaj
- slot se poziva kada se desi događaj

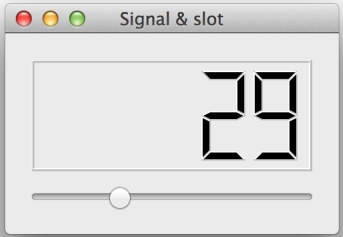
```
button.clicked.connect(self.onClick)
```

- izvor: `button`
- signal: `clicked`
- slot: `onClick()`

Signal/slot primer

```
class Example(QtWidgets.QWidget):
    def initUI(self):
        ...
        lcd = QtWidgets.QLCDNumber(self)
        sld = QtWidgets.QSlider(QtCore.Qt.Horizontal, self)
        vbox = QtWidgets.QVBoxLayout()
        vbox.addWidget(lcd)
        vbox.addWidget(sld)
        self.setLayout(vbox)
        # source: sld
        # signal: valueChanged
        # slot: lcd.display
        sld.valueChanged.connect(lcd.display)
```

Signal/slot primer 2



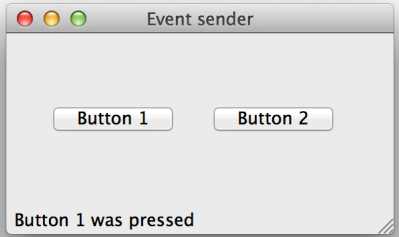
Redefinisanje postojeće obrade događaja

```
class Example(QtWidgets.QWidget):
    # nasleđena metoda, poziva se na pritisak tastera
    # redefinišemo je u našem nasledniku QWidget-a
    def keyPressEvent(self, e):
        if e.key() == QtCore.Qt.Key_Escape:
            self.close()
```

Informacije o izvoru događaja

```
class Example(QtWidgets.QMainWindow):
    def initUI(self):
        btn1 = QtWidgets.QPushButton("Button 1", self)
        btn1.move(30, 50)
        btn2 = QtWidgets.QPushButton("Button 2", self)
        btn2.move(150, 50)
        btn1.clicked.connect(self.buttonClicked)
        btn2.clicked.connect(self.buttonClicked)
        self.statusBar()
        ...
    def buttonClicked(self):
        # zbog koga sam pozvan?
        sender = self.sender()
        self.statusBar().showMessage(sender.text() +
            ' was pressed')
```

Informacije o izvoru događaja 2



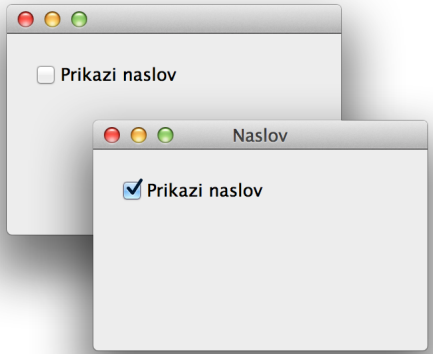
Definisanje nove vrste događaja

```
class Example(QtWidgets.QMainWindow):
    # closeApp je novi signal
    closeApp = QtCore.pyqtSignal()

    def initUI(self):
        # poveži closeApp sa close() slotom
        # close zatvara aplikaciju
        self.closeApp.connect(self.close)
        ...

    def mousePressEvent(self, event):
        # emituj događaj kad se klikne mišem
        self.closeApp.emit()
```

QCheckBox

```
class Example(QtWidgets.QMainWindow):
    def initUI(self):
        cb = QtWidgets.QCheckBox('Prikazi naslov', self)
        cb.move(20, 20)
        cb.toggle()
        cb.stateChanged.connect(self.changeTitle)
        ...

    def changeTitle(self, state):
        if state == QtCore.Qt.Checked:
            self.setWindowTitle('Naslov')
        else:
            self.setWindowTitle('')
```

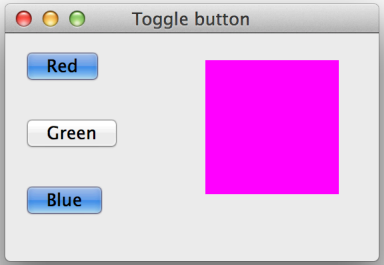
QCheckBox 2



QPushButton

```
class Example(QtWidgets.QMainWindow):
    def initUI(self):
        redb = QtWidgets.QPushButton('Red', self)
        redb.setCheckable(True)  # dugme sa dva stanja!
        redb.move(10, 10)
        # signal prenosi bool parametar!
        redb.clicked[bool].connect(self.setColor)
        ...
        self.square = QtWidgets.QFrame(self)
        self.square.setGeometry(150, 20, 100, 100)
        ...
    def setColor(self, pressed):
        source = self.sender()
        if pressed:
            ...
```

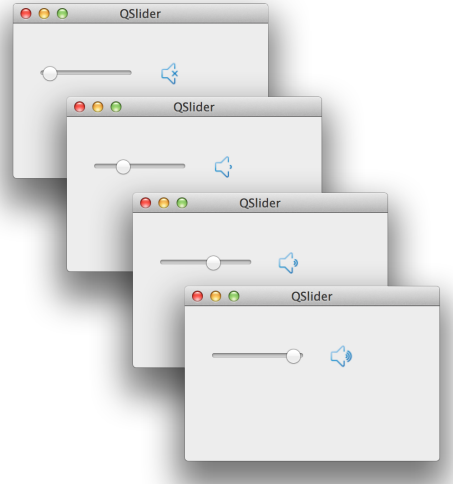
QPushButton 2



QSlider

```
class Example(QtWidgets.QMainWindow):
    def initUI(self):
        sld = QtWidgets.QSlider(QtCore.Qt.Horizontal, self)
        sld.setFocusPolicy(QtCore.Qt.NoFocus)
        sld.setGeometry(30, 40, 100, 30)
        # signal prenosi int parametar!
        sld.valueChanged[int].connect(self.changeValue)
        ...
        self.label = QtWidgets.QLabel(self)
        self.label.setPixmap(QtGui.QPixmap('mute.png'))
        self.label.setGeometry(160, 40, 80, 30)
        ...
    def changeValue(self, value):
        if value == 0:
            self.label.setPixmap(QtGui.QPixmap('mute.png'))
        elif value > 0 and value <= 30:
            self.label.setPixmap(QtGui.QPixmap('min.png'))
        elif value > 30 and value < 80:
            self.label.setPixmap(QtGui.QPixmap('mid.png'))
        else:
            self.label.setPixmap(QtGui.QPixmap('max.png'))
```

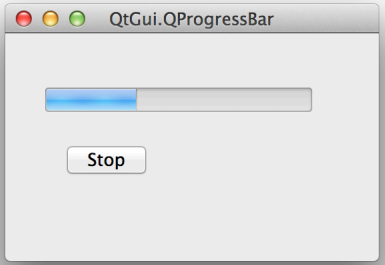
QSlider 2



QProgressBar

```
class Example(QtWidgets.QWidget):
    def initUI(self):
        self.pbar = QtWidgets.QProgressBar(self)
        self.pbar.setGeometry(30, 40, 200, 25)
        self.btn = QtWidgets.QPushButton('Start', self)
        self.btn.move(40, 80)
        self.btn.clicked.connect(self.doAction)
        self.timer = QtCore.QBasicTimer()
        self.step = 0
        ...
    def timerEvent(self, e):
        if self.step >= 100:
            self.timer.stop()
            self.btn.setText('Finished')
            return
        self.step = self.step + 1
        self.pbar.setValue(self.step)

    def doAction(self):
        if self.timer.isActive():
            self.timer.stop()
            self.btn.setText('Start')
        else:
            self.timer.start(100, self)
            self.btn.setText('Stop')
```

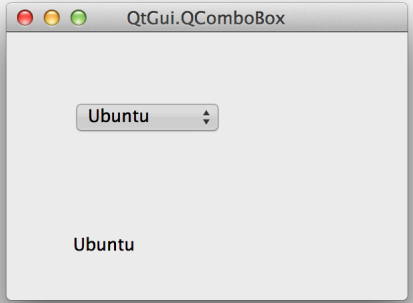
QProgressBar 2



QComboBox

```
class Example(QtWidgets.QWidget):
    def initUI(self):
        self.lbl = QtWidgets.QLabel("Ubuntu", self)
        combo = QtWidgets.QComboBox(self)
        combo.addItem("Ubuntu")
        combo.addItem("Mandriva")
        combo.addItem("Fedora")
        combo.addItem("Red Hat")
        combo.addItem("Gentoo")
        combo.activated[str].connect(self.onActivated)
        ...
    def onActivated(self, text):
        self.lbl.setText(text)
        self.lbl.adjustSize()
```

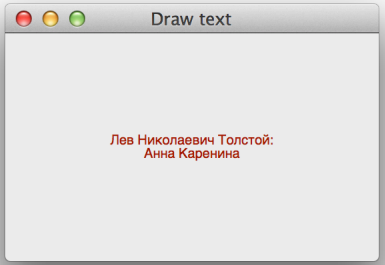
QComboBox 2



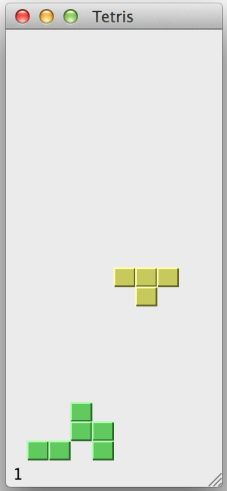
Pravljenje sopstvene komponente

```
class Example(QtWidgets.QWidget):
    def initUI(self):
        self.text = u'\u041b\u0435...'

    def paintEvent(self, event):
        qp = QtGui.QPainter()
        qp.begin(self)
        self.drawText(event, qp)
        qp.end()

    def drawText(self, event, qp):
        qp.setPen(QtGui.QColor(168, 34, 3))
        qp.setFont(QtGui.QFont('Decorative', 10))
        qp.drawText(event.rect(), QtCore.Qt.AlignCenter,
            self.text)
```

Pravljenje sopstvene komponente 2



Primer: Tetris



Tetris: 4 klase

- `Tetris`
  - inicijalizuje igru
- `Board`
  - sadrži logiku igre
- `Tetrominoe`
  - sadrži imena svih delova
- `Shape`
  - implementira ponašanje jedne figure