QT grafičko korisničko sučelje u Pythonu

Komaromi, Siniša

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

QT GRAFIČKO KORISNIČKO SUČELJE U PYTHONU

Završni rad

Siniša Komaromi

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 06.09.2017.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Siniša Komaromi
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3653, 23.07.2014.
OIB studenta:	81678646504
Mentor:	Doc.dr.sc. Irena Galić
Sumentor:	Hrvoje Leventić
Sumentor iz tvrtke:	
Naslov završnog rada:	QT grafičko korisničko sučelje u Pythonu
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	06.09.2017.
Datum potvrde ocjene Odbora:	11.09.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 12.09.2017.

Ime i prezime studenta:	Siniša Komaromi
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3653, 23.07.2014.
Ephorus podudaranje [%]:	1%

Ovom izjavom izjavljujem da je rad pod nazivom**: QT grafičko korisničko sučelje u Pythonu**

izrađen pod vodstvom mentora Doc.dr.sc. Irena Galić

i sumentora Hrvoje Leventić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

U ovome radu predstavljen je Qt, radni okvir (engl. *framework*), biblioteka koja olakšava razvoj sučelja računalnih programa. Isprva je zamišljen za rad s programskim jezikom C++, no primjenom jezičnih poveznica (engl. *language bindings*) omogućena je izrada sučelja i za programe napisane u drugim programskim jezicima, bez potrebe znanja rada u jeziku u kojem je biblioteka izvorno napisana. Poveznica sadrži i pruža na uporabu razvojnom programeru preddefinirane objekte, metode ili funkcije kojima se, u ovom slučaju, definiraju uvjeti interakcije, oblikuju izgled i raspored sučelja. Dio jezične poveznice u odredišnom jeziku napisan je u programeru poznatom jeziku, a uključivanjem pojedinih njezinih elemenata pozivaju se ekvivalenti iz izvornog koda biblioteke. Time je postignuto da programer ne mora poznavati jezik u kojem je izvorno napisana biblioteka, već programer posredstvom poveznice definira, primjerice, sučelje, koristeći izraze njemu poznatog jezika. U ovome radu u svrhu demonstracije Qt biblioteke izrađen je i prikazan primjer programa napisanog u Pythonu koji omogućava prikaz učitanih slika, njihovu izmjenu primjenom skriptiranih filtera i pregled obrađene slike. Filteri su, kao i sam programski primjer, napisani u Pythonu, a za obradu slike koristi se biblioteka OpenCV preko pripadne poveznice za Python.

Pored Qt-a, primjeri višeplatformskih biblioteka za izradu sučelja su GTK+, Kivy i wxWidgets. Većina navedenih biblioteka pruža približno jednako opsežan skup mogućnosti pri izradi programa kao i Qt te su u gotovo istoj mjeri prema kriteriju vremena razvoja zrele, no zbog različitih načina programske implementacije pojavnih i funkcionalnih svojstava sučelja, svaka biblioteka ima svoju prednost u odnosu na druge, bilo to olakšano programiranje sučelja uvođenjem nestandardnih proširenja korištenom temeljnom programskom jeziku, kao što je to slučaj s Qt-om, ili obratno, korištenje isključivo standardnog jezika uz pojednostavljen pristup ostvarenju interaktivnosti sučelja (wxWidgets), ili pak razvoj bez opterećenja razvojnog programera prilagođavanjem programa za rad na više različitih hardverskih platformi (stolna računala te pametni telefoni i tableti na drugoj strani) korištenjem isključivo onih elemenata sučelja koji se mogu primijeniti na svim ciljanim platformama (Kivy). Zbog jednostavnosti razvoja, zrelosti poveznice za Python i cilja ostvarenja tradicionalnog sučelja aplikacije za stolna računala u ovom radu odabrana je Qt biblioteka.

Glavni dio rada podijeljen je u tri poglavlja. U poglavlju Qt pružen je teorijski uvod u Qt biblioteku, gdje se ukratko opisuju povijest nastanka, osnovna svojstva, tri temeljna obilježja Qt-a i PySide, biblioteka (jezična poveznica) koja omogućuje uporabu Qt-a u programskom jeziku

Python. U poglavlju Usporedba s drugim bibliotekama opisane su biblioteke GTK+, Kivy i wxWidgets, adekvatne zamjene Qt-u te istaknuta određena svojstva koja dijele Qt i dana biblioteka ili čine razliku između njih. Poglavlje Python program za obradu slika sadrži primjer programa napisanoga u Pythonu sa sučeljem ostvarenim pomoću Qt biblioteke koji služi prikazu i obradi slika.

1.1. Zadatak završnog rada

Zadatak ovog rada je opisati Qt biblioteku i jezičnu poveznicu koja omogućuje uporabu Qt biblioteke u programskom jeziku Python te usporediti Qt s drugim višeplatformskim bibliotekama za izradu sučelja. Potrebno je izraditi program za prikaz učitanih slika, omogućiti njihovu obradu filterima i pregled izmijenjene slike.

2. QT

Qt je višeplatformski radni okvir koji se koristi za razvoj aplikacija. Može se pokretati na raznim programskim i sklopovskim platformama bez ili uz manje izmjene temeljnog Qt koda, pri čemu su sposobnosti i brzina takve aplikacije identični aplikaciji izvorno napisanoj za određenu ciljanu platformu. Brojne organizacije temelje svoje programe na Qt-u. Primjeri takvih organizacija su AMD (Advanced Micro Devices), DreamWorks, Samsung, Siemens i Electronic Arts. U aplikacije koje koriste Qt biblioteku ubrajaju se:

- EAGLE, aplikacija za automatizirano projektiranje elektroničkih sklopova (engl. *electronic design automation*, skraćeno EDA) s podrškom za skriptiranje [1, 2],
- FreeMat, okruženje i programski jezik za obradu numeričkih podataka otvorenog koda, funkcionalno sličan MATLAB-u [3],
- Google Earth,
- Maya, Autodeskov program za izradu trodimenzionalne računalne grafike,
- QGIS, aplikacija otvorenog koda za rukovanje geografskim podacima u GIS-u (geografski informacijski sustav) [4],
- Skype te
- VLC, program otvorenog koda za prikazivanje i emitiranje multimedijskih sadržaja [5, 6].

2.1. Povijest

Izvorni autori Qt biblioteke su Haavard Nord, glavni izvršni upravitelj tadašnje tvrtke Trolltech, i Eirik Chambe-Eng, njezin predsjednik. 1991., Nord je počeo s pisanjem razreda (klasa) koje su naposljetku postale Qt-om. Sljedeće godine Chambe-Eng osmislio je koncept signala i slotova, jednostavnu, ali djelotvornu paradigmu za razvoj grafičkih korisničkih sučelja (engl. *graphical user interface*, skraćeno GUI), koju su preuzeli nekolicina drugih skupova alata (engl. *toolkit*) za izradu sučelja. Ime Qt odabrano je jer se Nordu svidio izgled slova Q u Emacsu, programu za uređivanje teksta, dok slovo t označava *toolkit*, po uzoru na Xt (X toolkit) [6]. Prvi ugovor za razvoj programa na temelju Qt biblioteke sklopljen je s norveškom tvrtkom Metis u travnju 1995. zahvaljujući vezi uspostavljenoj preko jednog od Nordovih profesora na sveučilištu. U tom razdoblju Trolltech je zaposlio Arnta Gulbrandsena, koji je za svog šestogodišnjeg boravka u Trolltechu razvio i implementirao inovativan sustav dokumentacije te općenito doprinio razvoju Qt-a. 20. svibnja 1995. Qt biblioteka je po prvi put objavljena javnosti, noseći oznaku Qt 0.90. U

ožujku 1996. Europska svemirska agencija (engl. *European Space Agency*, skraćeno ESA) je kupovinom deset licenci za komercijalnu uporabu Qt-a postala drugim kupcem Qt biblioteke. Potaknuti još jednim sklopljenim ugovorom, Nord i Chambe-Eng zapošljavaju još jednog razvojnog programera te u rujnu 1996. izlazi prva inačica Qt-a, Qt 1.0. U listopadu 1996. Matthias Ettrich započinje projekt razvoja vlastitog radnog okruženja (engl. *desktop environment*, skraćeno DE) za Linux operacijski sustav pod nazivom KDE (isprva osmišljen kao kratica od Kool Desktop Environment, no kasnije je odlučeno da slovo K ne označava ništa određeno) [7, 8]. Kao temelj KDE-a Ettrich je odabrao Qt, čime Qt postaje *de facto* standardom za razvoj korisničkih sučelja programa za Linux pisanih u jeziku C++. 1998. Ettrich se pridružuje tvrtki Trolltech. U tijeku idućeg desetljeća izlaze druga (lipanj 1999.), treća (2001.) i četvrta inačica Qt biblioteke (ljeto 2005.), a komercijalni korisnici broje se u tisućama [9, str. xv-xvii]. Peta inačica objavljena je u prosincu 2012. godine [10].

2.2. Svojstva

Qt se koristi za razvoj višeplatformskih aplikacija i korisničkih sučelja, no može se koristiti i za razvoj programa koji nemaju grafičko korisničko sučelje, poput alata i konzola s tekstualnim sučeljem, namijenjenih za uporabu na poslužiteljima. Peta inačica (Qt 5.9) podržava rad na Linux, Windows i macOS operacijskim sustavima, operacijskim sustavima ugradbenih platformi (Linux, QNX i INTEGRITY) te operacijskim sustavima pametnih telefona i tableta (Android, iOS i UWP - Universal Windows Platform) [11]. U mogućnosti Qt-a koje nisu strogo vezane uz sučelje ubrajaju se pristup SQL bazama podataka, podrška za tumačenje XML i JSON zapisa, rad s dretvama (engl. *threads*) i mrežom. Primjer programa bez GUI-ja izrađenog pomoću biblioteke Qt je Cutelyst, radni okvir za razvoj internetskih aplikacija. Programi s GUI-jem izrađenim pomoću Qt-a imaju sučelje koje u potpunosti izgleda kao sučelje aplikacija izvorno napisanih za određeni operacijski sustav ili radno okruženje. Prema tome, Qt spada u skupove alata za izradu sučelja (engl. *widget toolkit*), biblioteke koje pružaju niz grafičkih upravljačkih elemenata (engl. *widgets*) koji sačinjavaju grafičko sučelje programa [12].

Qt je napisan u standardnom C++ jeziku, no uvodi koncept signala i slotova. Tako je ostvareno olakšano rukovanje događajima (engl. *events*), a time i razvoj grafičkih i poslužiteljskih aplikacija koje primaju skup podataka o događaju koji svaka aplikacija obrađuje na specifičan način. Dio Qt-a je i Qt Quick, radni okvir koji za definiranje izgleda i interakcije sa sučeljem koristi QML (Qt Meta Language ili Qt Modeling Language), deklarativni skriptni jezik s podrškom za

proširivanje JavaScript kodom [13, 14]. Koristeći Qt Quick omogućen je ubrzani razvoj aplikacija (engl. *rapid application development*, skraćeno RAD) za prijenosne uređaje, no za bolje performance poželjno je pisanje u jeziku izvorno zamišljenom za pisanje aplikacija na određenom operacijskom sustavu.

Qt se može koristiti i kao dio mnogih drugih programskih jezika koristeći jezične poveznice koje implementiraju dio ili cijeli skup mogućnosti Qt-a. Za četvrtu i petu inačicu Qt-a, Qt 4 i 5, razvijene su poveznice s jezicima C#, Haskell, Python, Ruby i nekolicini drugih jezika [15, 16].

2.3. Temeljna obilježja

Qt obilježavaju tri osnovne značajke: potpuna apstrakcija od grafičkog sučelja, signali i slotovi i metaobjektni prevoditelj.

2.3.1. Apstrakcija od grafičkog sučelja

Qt je izrađen tako da koristi vlastiti pokretački stroj za crtanje (engl. *paint engine*), kod za prikaz elemenata sučelja na zaslonu, koji oponaša izgled platforme na kojoj je pokrenut. Time je prijenos aplikacija među platformama znatno olakšan jer se prilično malen broj klasa uistinu oslanja na ciljanu platformu, a razvojni programer je gotovo u potpunosti apstrahiran od izrade grafičkog sučelja. Isprva to oponašanje nije bilo savršeno, no s novijim inačicama uvodi se uporaba sučelja za razvoj aplikacija (engl. *application programming interface*, API) koje pruža platforma. Njima Qt upućuje upite vezane uz dimenzije sučelja i zahtjeve za crtanje većine elemenata sučelja, čime su odstupanja svedena na minimum. Neke platforme, kao što su MeeGo i KDE, koriste Qt kao svoj API za oblikovanje sučelja, čime se uklanja potreba oponašanja odredišne platforme. S druge strane, postoje biblioteke poput wxWidgetsa, koje umjesto oponašanja koriste skupove alata za izradu sučelja koje pruža odredišna platforma [6].

2.3.2. Signali i slotovi

Signali i slotovi su jezične strukture uvedene u Qt-u radi uspostavljanja komunikacije između objekata. Omogućavaju lakše praćenje stanja objekata bez potrebe pisanja programskih predložaka (engl. *boilerplate code*) [6]. Predstavljaju temeljno svojstvo koje razlikuje Qt od ostalih

radnih okvira. Koncept se temelji na tome da elementi sučelja mogu odašiljati signale s podacima o događaju koje upravljački dio programa prima pomoću posebnih funkcija koje se nazivaju slotovima. Stariji *toolkiti* ovakvu komunikaciju uspostavljali su koristeći koncept koji ima engl. naziv *callback*. U jeziku C++ on se može implementirati pomoću pokazivača na funkciju, a uz STL (Standard Template Library) biblioteku moguća je i implementacija primjenom funkcijskih objekata (engl. *functors* ili *function objects*). U Pythonu se ostvaruje jednostavnim predavanjem funkcije kao argumenta drugoj funkciji, bez uključivanja dodatnih biblioteka [17]. Temeljni nedostatak korištenja *callback* funkcija je što nije osiguran poziv funkcije s ispravnim argumentima, odnosno nisu osigurani ispravni tipovi podataka (engl. *type safety*). Također, funkcija koja obrađuje događaj usko je povezana s pridruženim pokazivačem na funkciju jer mora znati koju *callback* funkciju poziva.

Sustav signala i slotova u suglasnosti je s načinom izrade grafičkih korisničkih sučelja. Signal se odašilje kada dođe do određenog događaja. Qt-ovi *widgeti* imaju velik broj preddefiniranih signala, no moguće je definirati vlastite signale u izvedenim *widgetima*. Slot je funkcija pozvana kako bi pružila odziv na određeni signal. Kao i u slučaju signala, Qt posjeduje brojne preddefinirane slotove, no po potrebi je moguće opisati vlastite. Mehanizam signala i slotova osmišljen je tako da osigurava ispravnost tipova podataka. Ispravnost tipova podataka osigurana je zahtjevom da se potpis signala mora podudarati sa slotom primateljem. Potpis (engl. *signature*) ovdje označava dio podataka u deklaraciji funkcije koji prevoditelj koristi za razrješavanje preopterećenja: tipovi parametara, pripadnost funkcije klasi, mijenja li funkcija predani objekt, mijenja li se predani objekt neovisno o funkciji, smije li objekt pripadne klase biti izmijenjen i mijenja li se objekt pripadne klase neovisno o promatranoj funkciji [18, 19, 20]. Pritom čak slot može imati kraći potpis od signala jer može odbaciti dodatne argumente. Budući da su potpisi signala i slota međusobno kompatibilni, prevoditelj može biti korišten u svrhu prepoznavanja razlika u tipovima podataka. Signali i slotovi povezani su slabom vezom, odnosno klasa koja odašilje signal ne vodi računa o slotovima koji primaju signal.

Signali i slotovi mogu se koristiti i u slučajevima kada GUI nije prisutan. Primjeri su obavijesti o događajima vezanima uz asinkrone ulaze i izlaze (priključne točke ili engl. *sockets*, cijevi za komunikaciju ili engl. *pipes*, serijski uređaji i sl.) ili rukovanje događajima vezanima uz istek roka odziva (engl. *timeout events*). Sustav signala i slotova jednostavan je za uporabu jer rad temelji na Qt-ovom metaobjektnom prevoditelju koji automatski generira potrebnu infrastrukturu [21, 22].

2.3.3. Metaobjektni prevoditelj

Metaobjektni prevoditelj, nazvan moc (kao kratica od engl. *Meta Object Compiler*), je alat, generator koda koji omogućuje ostvarivanje mogućnosti koje izvorno nisu dostupne kao dio standardnog C++ jezika: signali i slotovi, unutarnji uvid (engl. *introspection*) i asinkrono pozivanje funkcija. Kako bi to postigao, moc stvara dodatni C++ kod koji bilo koji standardni C++ prevoditelj može prevesti. moc čita izvorne C++ datoteke i u njima traži deklaracije klasa koje sadrže odgovarajuću makronaredbu. Za svaku takvu klasu stvara novu C++ izvornu datoteku koja sadrži pripadni metaobjektni kod. C++ datoteke koje generira moc potrebno je prevesti i povezati (uspostaviti engl. *link*) s implementacijom pripadne klase, ili uključiti (engl. *include* naredbama) u izvornu datoteku klase. moc se obično ne poziva ručno, već ga automatski poziva sustav za prevođenje programa. Prema tome, za rad moc-a nije potrebna intervencija programera. Uz moc, Qt koristi i generator koda pod nazivom uic (*User Interface Compiler*). uic pretvara opis sučelja u obliku XML datoteke u C++ kod koji pri pokretanju stvara prozor s prethodno opisanim elementima [6, 23].

2.4. PySide

PySide je jezična poveznica Pythona s Qt višeplatformskim *toolkitom* za izradu sučelja. Jedna je od mogućih alternativa standardnoj Tkinter biblioteci. Tkinter, čije ime je skraćenica za Tk interface, poveznica je Pythona s *toolkitom* Tk i smatra se *de facto* standardom za izradu sučelja aplikacija u Pythonu [24]. Kao i Qt, PySide spada u besplatni softver. Projekt razvoja PySide biblioteke nastao je kao posljedica neuspjeha u pregovorima Nokije s britanskom tvrtkom Riverbank Computing oko licenciranja PyQt biblioteke, također poveznice Qt-a s Pythonom, pri čemu je zahtjev Nokije bio da PyQt bude licenciran i pod LGPL licencom pored postojeće GPLv3 i komercijalne licence [25, 26]. LGPL licenca omogućuje primjenu PySide biblioteke u razvoju besplatnih programa otvorenog koda, ali i komercijalnog softvera [27]. Primjer aplikacije koja koristi PySide biblioteku je FreeCAD, višeplatformska aplikacija za izradu trodimenzionalnih modela objekata s podrškom za naknadnu prilagodbu modela izmjenom njegovih parametara [28, 29].

Za uspostavljanje poveznice s Pythonom u početku je bio korišten Boost.Python iz Boost skupa biblioteka za C++, no kasnije je u sklopu PySide projekta razvijen vlastiti generator poveznica nazvan Shiboken kako bi se smanjila veličina binarnih datoteka i zauzeće memorije

[30]. U aktivnom razvoju je nadogradnja PySide biblioteke koja uvodi podršku za petu inačicu Qt-a, dok trenutna stabilna inačica PySidea podržava četvrtu inačicu (Qt 4.6, 4.7 i 4.8) [31]. Nova, nadograđena inačica PySide biblioteke ima projektni naziv PySide2, a razvoj je započeo tijekom 2014. godine [32]. PySide podržava rad na Linux, Mac OS X, Windows i Maemo operacijskim sustavima, dok je podrška za Android i Symbian u postupku uspostavljanja [33].

PySide biblioteka sastoji se od sljedećih 15 modula:

- QtCore, pruža jezgrenu funkcionalnost nevezanu za korisničko sučelje,
- QtGui, proširuje QtCore modul s mogućnostima vezanima uz GUI,
- QtDeclarative, deklarativni radni okvir za izradu visoko dinamičnih, korisnički definiranih sučelja,
- QtHelp, pruža klase za uključivanje internetske dokumentacije u aplikacije,
- QtMultimedia, pruža multimedijsku funkcionalnost na najnižoj razini,
- QtNetwork, pruža klase koje omogućuju pisanje TCP/IP klijenata i poslužitelja,
- QtOpenGL, pruža klase koje olakšavaju uporabu OpenGL-a u aplikacijama koje koriste Qt,
- QtScript, pruža klase koje uvode podršku za skriptiranje u aplikacijama temeljenima na Qt-u,
- QtScriptTools, pruža dodatne komponente za aplikacije koje koriste skriptiranje u Qt-u,
- QtSql, olakšava integraciju baza podataka u aplikacije napisanima pomoću Qt-a,
- QtSvg, pruža klase za prikaz sadržaja SVG datoteka,
- QtUiTools, pruža klase za rukovanje prozorima izrađenima u Qt Designeru,
- QtXml, pruža čitač i pisač toka za XML dokumente,
- QtWebkit, pruža pogonski stroj za pristup Internetu te
- Phonon, višeplatformski medijski radni okvir koji omogućuje uporabu zvukovnih i slikovnih zapisa u aplikacijama koje su temeljene na Qt-u [34].

3. USPOREDBA S DRUGIM BIBLIOTEKAMA

Pored Qt-a, za izradu sučelja računalnih programa koriste se, između ostalih, biblioteke GTK+, Kivy i wxWidgets.

3.1. GTK+

GTK+ je višeplatformski skup alata za izradu grafičkih korisničkih sučelja napisan u jeziku C. Prikladan je za uporabu u različite svrhe, od razvoja aplikacija jednokratne namjene do cjelovitih programskih paketa [35]. Uz Qt, jedan je od najrasprostranjenijih *toolkita* za Wayland i X11 prozorske sustave (engl. *windowing system*) *Unix-like* operacijskih sustava [36, 37]. Koristi se u brojnim višeplatformskim aplikacijama (AbiWord, Chromium, GIMP, Inkscape, Pidgin) i radnim okruženjima za *Unix-like* operacijske sustave (GNOME, Unity, Cinnamon, MATE, XFCE) [38, 39, 40, 41, 42, 43]. Obilježavaju ga svojstva kao što su podudarnost izgleda i funkcionalnosti s platformom na kojoj je pokrenut, podrška za teme i dvosmjerni tekst, dretvena sigurnost (engl. *thread safety*), objektno orijentirani pristup, internacionalizacija i lokalizacija, pristupačnost (engl. *accessibility*) i podrška za UTF-8 zapis [44].

GTK+ projekt započeo je pod nazivom GTK (GIMP Toolkit). Izvorno je osmišljen kao zamjena za Motif *toolkit* koji je bio odabran kao temelj GIMP-a (GNU Image Manipulation Program), besplatnog programa otvorenog koda za uređivanje i naknadno prepravljanje slika, crtanje i druge slične zadatke, funkcionalno usporedivog s Adobeovim Photoshopom [40]. Peter Mattis, jedan od izvornih autora, započeo je rad na GTK *toolkitu* zbog nezadovoljstva Motif *toolkitom*, ali i radi upoznavanja s načinom razvoja *toolkita*. Mattis je bio usredotočen na razvoj *toolkita,* dok je Spencer Kimball, drugi izvorni autor iz takozvanog S&P (Spencer and Peter) dvojca, radio na usavršavanju funkcionalnosti samog GIMP-a [45]. S inačicom 0.60 Motif je u GIMP-u uspješno zamijenjen GTK-om. Potom je GTK napisan iznova s unaprijeđenom objektno-orijentiranom osnovom kao GTK+, koji uz doradu objektno-orijentiranog pristupa uvodi mehanizam sa signalima sličan Qt-ovom, umjesto prethodnog mehanizma temeljenog na *callback* funkcijama [46]. GTK+ uveden je u uporabu s inačicom 0.99 GIMP-a. Nakon što su Mattis i Kimball prestali doprinositi razvoju GIMP-a i GTK+-a, nadzor nad razvojem preuzima organizacija GNOME Foundation koja GTK+ koristi kao dio svog GNOME radnog okruženja. S drugom inačicom GTK+ *toolkita* poboljšani su prikaz teksta i pristupačnost, API je proširen te je obavljen prelazak na Unicode uvođenjem UTF-8 zapisa. U trećoj inačici izmijenjen je način

rukovanja ulaznim uređajima, dodana je podrška za uređivanje izgleda sučelja koristeći teme napisane sintaksom sličnom CSS-ovoj (*Cascading Style Sheets*) te mogućnost dohvata informacija o drugim istovremeno aktivnim GTK+ aplikacijama.

GTK+ biblioteka sadrži skup grafičkih upravljačkih elemenata kojeg u inačici 3.13.3 čini 203 aktivna *widgeta* i 37 *widgeta* nadomještenima boljim rješenjima (engl. *deprecated*). Koristi GObject, odnosno GLib sustav objekata kako bi se ostvarila objektna orijentiranost u C jeziku. Iako je primarno razvijen za prozorske sustave temeljene na X11 i Waylandu, GTK+ podržava i rad na drugim platformama, kao što su to Windows (posredstvom Windowsovog GDI API-ja), macOS (posredstvom Quartza). Podržan je i HTML5 prikaz u preglednicima koristeći Broadway pozadinsku komponentu (engl. *back-end*) GTK+-a. Izgled nacrtanih grafičkih upravljačkih elemenata moguće je izmijeniti primjenom različitih pokretačkih strojeva za prikaz (engl. *display engine*). Za ponašanje izgleda *widgeta* platforme na kojoj je pokrenut, GTK+ koristi nekoliko različitih pokretačkih strojeva za prikaz. Grafički upravljački elementi su počevši od inačice 2.8 većinom crtani koristeći biblioteku Cairo. Cairo je biblioteka otvorenog koda koja razvojnim programerima pruža API koji je temeljen na vektorskoj računalnoj grafici i neovisan o vrsti uređaja. Koristi sklopovsko ubrzanje kada god je to moguće [47]. Budući da se temelji na vektorskoj grafici, pruža visoku kvalitetu prikaza i ispisa [48]. Od treće inačice crtanje sučelja u cijelosti obavlja Cairo. Uvođenjem Cairo biblioteke olakšan je istovremeni razvoj GTK+ biblioteke za više različitih platformi [49, 50].

Kao i Qt, GTK+ ima jezične poveznice za brojne jezike, uključujući C++, C#, Javu, Perl, Python i mnoge druge [51].

3.2. Kivy

Kivy je višeplatformska biblioteka otvorenog koda za razvoj mobilnih aplikacija i drugog aplikacijskog softvera s prirodnim korisničkim sučeljem (engl. NUI - *natural user interface*) koji koristi istovremeni dodirni unos s više prstiju (engl. *multitouch*). Pristaše NUI pristupa izradi sučelja računalnih aplikacija predstavljaju ga kao sljedeći korak u evoluciji sučelja nakon GUI-ja, kao što je to bio prelazak sa sučelja naredbenog retka (engl. CLI - *command-line interface*) na GUI. Oslanja se na izravnu i intuitivnu interakciju, dok se GUI temelji na metaforama i istraživanju [52]. Kivy je napisan najvećim dijelom u Pythonu, a manjim u Cythonu. Korištenjem Cythona omogućeno je brže izvođenje vremenski zahtjevnih dijelova biblioteke [53]. Za razliku od Qt i

GTK+ biblioteka, Kivy ne posjeduje jezične poveznice prema drugim jezicima, što omogućuje njegov usredotočen razvoj za programski jezik Python, pritom uzimajući u obzir isključivo programerska načela Pythona. Prva inačica Kivy radnog okvira postala je dostupna javnosti u veljači 2011. godine. Podržava rad na Android, iOS, Linux, OS X i Windows platformama te Raspberry Pi računalima s jednom pločom (engl. SBC - *single-board computer*) [54]. Nasljednik je PyMT projekta, koji je, kao i Kivy, radni okvir za izradu hardverski ubrzanih aplikacijskih sučelja s podrškom za višedodirni unos [55]. Razvojni programeri koji su predvodili razvoj PyMT-a započeli su rad na Kivy biblioteci pa je aktivan razvoj PyMT-a zaustavljen [53]. Za nove projekte stoga je poželjno kao temelj koristiti Kivy biblioteku. Pri razvoju aplikacija u Kivyju od osobite su koristi njegovi sestrinski projekti:

- Plyer, platformski neovisan omotač u Pythonu za API-je pojedinih platformi,
- Pyjnius, Python modul za pristup Java klasama Android API-ja [56],
- Pyobjus, Python modul za pristup Objective-C klasama iOS API-ja [57],
- Python for Android, skup razvojnih alata (engl. *toolchain*) za prevođenje i izradu paketa s aplikacijama u Pythonu za Android,
- Kivy iOS, skup razvojnih alata za izradu paketa za iOS,
- Kivy Designer, alat za oblikovanje izgleda sučelja Kivy programa [58],
- KivEnt, pogonski stroj za igre izrađene pomoću Kivyja,
- Kivy Garden, skup *widgeta* i biblioteka koje korisnici izrađuju i održavaju [59],
- KivyPie, distribucija Linuxa temeljena na Raspbianu (operacijski sustav temeljen na Debianu prilagođen za Raspberry Pi računala) koja sadrži najnoviju inačicu Kivy radnog okvira [60], i nekolicina drugih.

Kivy obilježava opsežna podrška za ulazne uređaje: miš, tipkovnicu, TUIO (Tangible User Interface Objects) protokol, a, ovisno o operacijskom sustavu i višedodirne događaje unosa [61]. TUIO je radni okvir koji opisuje zajednički protokol i API za opipljive višedodirne površine. Omogućuje odašiljanje opisa interaktivnih površina, uključujući dodire i stanja dodirljivih (opipljivih, engl. *tangible*) predmeta [62]. Kivy se oslanja na grafičku biblioteku koja koristi ugradbenu ES (Embedded Systems) varijantu OpenGL biblioteke. Za prikaz sučelja na zaslonu koriste se objekti spremnika vrhova (engl. VBO - *vertex buffer object*) OpenGL-a [63, 64]. Objekt spremnika vrhova je svojstvo OpenGL-a koje pruža način za spremanje podataka o vrhovima trodimenzionalnog objekta (položaj u prostoru, vektor normale, boja i sl.) na grafičkoj kartici računala. Omogućuje značajno ubrzanje u radu jer su podaci spremljeni u memoriju grafičke kartice umjesto u glavnoj memoriji računala [65]. Kivy dolazi s velikim brojem grafičkih

upravljačkih elemenata koji podržavaju višedodirni unos te posredničkim jezikom pod nazivom Kv koji omogućuje izradu vlastitih *widgeta*. Slično Qt-ovom QML jeziku, Kv omogućuje opis strukture cjelokupnog sučelja i povezivanje s Python kodom koji rukuje interakcijom korisnika sa sučeljem [66]. Kv, kao jezik za definiranje sučelja, na jednostavan način odvaja dizajn sučelja od definiranja načina rada aplikacije te tako olakšava provođenje načela razdvajanja područja (engl. *separation of concerns*) [67]. Razdvajanje područja predstavlja načelo koje doprinosi boljem razumijevanju, izgradnji i upravljanju složenim sustavima podjelom računalnog programa na zasebne dijelove koji se bave različitim problemima [68].

3.3. wxWidgets

wxWidgets je besplatan skup alata, biblioteka otvorenog koda za izradu grafičkih korisničkih sučelja za višeplatformske aplikacije napisana u programskom jeziku C++. Kao i u slučaju Qt-a, programski kod za definiranje korisničkog sučelja pomoću wxWidgetsa moguće je prevesti i pokrenuti na različitim računalnim platformama bez ili uz manje izmjene koda [69].

Projekt wxWidgets započeo je Julian Smart na institutu AIAI (Artificial Intelligence Applications Institute) na sveučilištu u Edinburghu, 1992. godine pod imenom wxWindows. Projekt je nastao jer su sve ostale tadašnje biblioteke bile preskupe za eksperimentalni metaCASE program nazvan Hardy na kojem je Smart radio. MetaCASE (kratica od engl. *computer-aided software engineering*) alat predstavlja vrstu aplikacijskog softvera koji pruža mogućnost definiranja jedne ili više metoda modeliranja, jezika ili načina označavanja za uporabu u procesu razvoja programa [70, 71]. Hardy je bio zamišljen za rad na Windows i Unix radnim stanicama temeljenima na X-u pa je za ime projekta odabran naziv wxWindows, gdje slovo w označava Windows operacijski sustav, a x X prozorski sustav. Projekt je započeo s podrškom za XView i MFC 1.0 biblioteke, a AIAI je dopustio njegovu objavu na Internetu [72]. U ožujku 1999. izlazi druga inačica wxWindows biblioteke. Zbog spora s Microsoftovim ogrankom u Ujedinjenom Kraljevstvu oko zaštitnog znaka (engl. *trademark*) vezanog uz riječ Windows u imenu projekta, wxWindows 2004. godine preimenovan je u wxWidgets. U studenom 2013. godine izlazi i treća inačica biblioteke wxWidgets.

Koristi se u brojnim većim organizacijama, kao što su Xerox, AMD, Lockheed Martin, NASA i Središte za analize u mornarici američke vlade (engl. CNA - *Center for Naval Analyses*) [73]. Pored toga, koristi se u javnom sektoru, obrazovanju, brojnim projektima otvorenog koda, a

koriste ga i individualni programeri. Primjeri rasprostranjenijih aplikacija koje koriste wxWidgets su:

- 0 A.D. - besplatna igra otvorenog koda po uzoru na Age of Empires [74],
- Audacity - višeplatformski program za uređivanje zvučnih zapisa,
- AVG AntiVirus, program za zaštitu od virusa za Windows i Linux operacijske sustave [75, str. 601],
- Code::Blocks i CodeLite - besplatna višeplatformska razvojna okruženja (engl. IDE - *integrated development environment*) otvorenog koda [76, 77] te
- FileZilla - besplatna FTP aplikacija koja se sastoji od klijentskog i poslužiteljskog dijela, smatra se najpopularnijim FTP klijentom [78, 79].

wxWidgets podržava rad na raznim operacijskim sustavima, uključujući Microsoftov Windows, macOS (koristeći Cocoa API), iOS (Cocoa Touch) te Linux i Unix operacijske sustave (X11 sustav te *toolkite* Motif, Qt i GTK+) [80]. Posebna varijanta za ugradbene sustave, nazvana wxEmbedded, je u razvoju [81]. Za postizanje kompatibilnosti s navedenim platformama u sklopu wxWidgets projekta razvijeni su programski prijenosi (engl. *ports*) koji opisuju grafičke upravljačke elemente wxWidgets biblioteke koristeći biblioteke ili API-je odredišnih operacijskih sustava. Implementacija za Windows operacijske sustave počevši od Windowsa XP naziva se wxMSW. Za Linux i Unix operacijske sustave razvijeni su wxX11, wxMotif, wxQt i wxGTK, za macOS wxOSX/Cocoa, a za iOS wxiOS [82].

Pored osnovnih svojstava vezanih uz razvoj korisničkog sučelja, wxWidgets sadrži sloj za međuprocesnu komunikaciju (engl. IPC - *inter-process communication*), podršku za internacionalizaciju, rad s pisačima i *socketima* te višenitno programiranje [83]. Za razliku od Qt-a ne oponaša sučelje odredišne platforme, već koristi elemente sučelja koje pruža sama platforma kada god je to moguće, što pruža poboljšanje u brzini rada programa. Također, wxWidgets ne koristi mehanizam signala i slotova koji uvodi nestandardna proširenja jeziku C++, već način rukovanja događajima prepušta na izbor programeru [84]. Razvoj wxWidgets aplikacija moguć je koristeći višeplatformska razvojna okruženja Code::Blocks (s wxSmith proširenjem ili engl. *plug-inom*), CodeLite (wxCrafter), Eclipse (CDT), NetBeans, wxHatch i SPE (kratica za Stani's Python Editor). Za razvoj na Windowsu moguće je koristiti Borland C++ Builder (s wxForms proširenjem), Dev-C++ (postoji i wxDev-C++, proširena varijanta s ugrađenim programom za oblikovanje wxWidgets prozora), Microsoft Visual C++, VisualWX, Philasmicos Entwickler

Studio i Zeus IDE. Na Linuxu mogu se koristiti Anjuta, CodeForge i KDevelop razvojna okruženja [85].

Kao i Qt i GTK+ biblioteke, wxWidgets ima jezične poveznice za rad u programskim jezicima kao što su C, Java, PHP, Python i Ruby [86, 69].

4. PYTHON PROGRAM ZA OBRADU SLIKE

U ovome poglavlju predstavljen je primjer programa napisanog u Pythonu koristeći biblioteku Qt, a namijenjen je prikazu i obradi slike na Linux operacijskim sustavima. Uz manje prilagodbe moguća je potpuna funkcionalnost programa i na Windowsu. Sučelje je izrađeno koristeći biblioteku Qt posredstvom jezične poveznice PySide, a učitavanje slike i rukovanje unutar programa vrši se pomoću biblioteke OpenCV. Prikazani su i objašnjeni sučelje i ključni dijelovi izvornog koda programa te način izrade filtera, također pisanih u Pythonu, pomoću kojih se vrši obrada slike.

OpenCV, punim imenom Open Source Computer Vision Library, biblioteka je otvorenog koda namijenjena za primjenu u područjima računalnog vida (engl. *computer vision*) i strojnog učenja (engl. *machine learning*). U korisnike OpenCV biblioteke, između ostalih, ubrajaju se tvrtke Google, Microsoft, Intel, IBM i Honda te brojne razvojne tvrtke (engl. *startups*) [87]. Ima široki raspon primjena, a neki od primjera su prepoznavanje provala na snimkama nadzornih kamera, nadzor opreme za rudarenje i brzo prepoznavanje lica. U sklopu ovog rada koristi se za učitavanje, spremanje i prikaz slika u radnoj memoriji te po potrebi primjenu ugrađenih metoda pri pisanju filtera za obradu slike. OpenCV je izvorno napisan u jeziku C++, no zahvaljujući jezičnoj poveznici može se primijeniti u sklopu programa napisanih u Pythonu [88].

4.1. Filteri

Za razumijevanje načina rada cjelokupnog programa, potrebno je najprije predstaviti način pisanja filtera, i to zaglavlja i glavne funkcije filtera kako bi se objasnila njihova uloga u sučelju programa i kako utječu na njegov rad.

Kako bi uspješno prošao postupak provjere ispravnosti funkcije *parse_fhead* bez obustavljanja te potom postao uključenim u glavni program, svaki filter mora imati:

- naveden naziv funkcije, koja mora postojati u datoteci filtera i imati dva argumenta; te
- Python kod i zaglavlje bez pogrešaka u sintaksi.

Kako bi se osigurala obrada isključivo tekstualnih datoteka s Python kodom, obrađuju se samo one datoteke koje završavaju .py nastavkom (engl. *extension*).

Zaglavlje s opisom filtera smije se nalaziti u bilo kojem dijelu datoteke s programskim kodom filtera, bio to početak, kraj, ili u području između. Kako bi dio programa namijenjen za tumačenje zaglavlja prepoznao početak zaglavlja u filteru, ono mora započeti retkom *## begin fhead*, a završiti s *## end fhead*, gdje je *fhead* kratica od *filter header*, što u prijevodu označava zaglavlje filtera. Prije dvostruke oznake za komentar (*##*) moguće je dodati proizvoljan broj razmaka. Oznaka za komentar i prva sljedeća ključna riječ (*begin* ili *end*) mogu biti razmaknuti jednim ili više razmaka, no nije obvezno. Tako su, na primjer, jednako ispravni *##begin fhead* i *## begin fhead*. Međutim, ključne riječi koje slijede nužno je odvojiti razmakom. Broj ovih razmaka prepušta se na izbor autoru filtera, no mora biti prisutan barem jedan znak razmaka kako bi se spriječilo tumačenje dviju uzastopnih ključnih riječi u retku kao jedne. Izostavljanjem razmaka nastaje nova ključna riječ nepoznata kodu za tumačenje koja se smatra neispravnom sintaksom.

Svaki sljedeći redak unutar zaglavlja mora započeti jednostrukom oznakom komentara (#). Nakon toga može biti navedena neka od ključnih riječi kojima se opisuje filter. Takav redak piše se prema sintaksi *# ključna_riječ parametri*. Dopušteni su prazni redovi, no i oni moraju započeti jednostrukim znakom komentara. Kao i u slučaju oznaka početka i kraja zaglavlja, i ovdje vrijedi pravilo da redak može započeti proizvoljnim brojem razmaka, nakon kojeg slijedi prva ključna riječ koja smije biti spojena sa znakom komentara, dok ključne riječi i parametri koji se navode kasnije u retku moraju biti odvojeni najmanje jednim razmakom. Proizvoljni razmaci i položaj zaglavlja u datoteci uvedeni su s namjerom povećanja fleksibilnosti, odnosno stilske slobode autora pri razvoju filtera, a pri tome je zanemarivo povećana složenost programskog koda zaduženog za tumačenje zaglavlja filtera.

Podržane ključne riječi za opis filtera su:

- *funcname*, ključna riječ pomoću koje se navodi ime glavne funkcije filtera;
- *name*, ključna riječ kojom se navodi puni naziv filtera za uporabu kroz glavno sučelje programa;
- *desc*, ključna riječ koja navodi opis filtera koji se pojavljuje u statusnoj traci (engl. *status bar*) pri prelasku mišem preko danog filtera u izborniku *Filters*; te
- *arg*, ključna riječ kojom se navodi parametar filtera, njegova svojstva i gradbeni element u dijaloškom okviru za specificiranje svih parametara filtera.

Jedina ključna riječ od navedenih koju svaki filter mora posjedovati unutar zaglavlja za uspješno uključivanje u glavni program je *funcname*. Bez navođenja naziva glavne funkcije nije

moguće potvrditi postoji li, ima li primjeren broj argumenata, a onemogućen je i njezin dohvat po imenu u obliku funkcijskog objekta za pozivanje u glavnom programu. Glavna funkcija u filteru mora imati dva argumenta, gdje je prvi slika u obliku NumPy matrice koju glavni program predaje funkciji na obradu, a drugi je Python rječnik (engl. *dictionary*) s argumentima čije nazive ključeva i tipove autor navodi u zaglavlju pomoću ključne riječi *arg*. Naziv glavne funkcije navodi se poslije ključne riječi *funcname* u obliku niza znakova (engl. *string*) omeđenih navodnicima ili apostrofima. Prema tome, naredba za navođenje naziva glavne funkcije ima oblik *# funcname "ime_funkcije"*. Razmaci u nazivu funkcije nisu dozvoljeni, što je u sukladnosti s pravilima programskog jezika Python [89]. Ispravno oblikovano zaglavlje s osnovnim skupom naredbi ima strukturu kao na slici 4.1.

```
1   ##begin fhead
2   #funcname "ime_funkcije"
3   ##end fhead
```

Sl. 4.1. *Ispravno oblikovano zaglavlje s osnovnim skupom naredbi, razmaci i prazni redovi svedeni na minimum*

Za bolju čitljivost moguće je dodati proizvoljan broj razmaka i praznih redova, pritom ne utječući na ispravnost koda, pa prethodni primjer poprima oblik kao sa slike 4.2.

```
1   ## begin fhead
2    #
3    # funcname "ime_funkcije"
4    #
5   ## end fhead
```

Sl. 4.2. *Ispravno oblikovano zaglavlje s minimalnim skupom naredbi, dodani razmaci i prazni redovi*

Greške u sintaksi zaglavlja ili ostatku koda filtera, odsustvo funkcije, funkcija s neprimjerenim brojem argumenata i izostavljanje retka s navodom naziva glavne funkcije imaju za posljedicu preventivni prekid tumačenja filtera i njegovo zanemarivanje do sljedećeg pokretanja programa. Bez ulaska u zaglavlje postavljanje vrijednosti naziva funkcije, kao ni vrijednosti koje ostale ključne riječi predstavljaju nije moguće.

Za razliku od ključne riječi *funcname*, preostale opisne ključne riječi nisu obvezne, no poboljšavaju prikaz filtera u glavnom sučelju programa ili njegove mogućnosti prilagođavanja dodavanjem mogućnosti korisničkog unosa dodatnih parametara. Uklanjanjem obveze specificiranja ostalih ključnih riječi smanjen je broj zahtjeva koje autor filtera mora ispuniti s

ciljem olakšavanja razvoja filtera. Tako se navođenjem naziva filtera ključnom riječi *name* njegov naziv u sučelju postavlja na vrijednost koju navodi autor, no ako se i ne navede naziv filtera, postupak tumačenja filtera se ne obustavlja, već kod za tumačenje naziv u sučelju postavlja na naziv filterske datoteke. Ključna riječ *name* koristi se prema sintaksi *# name "naziv filtera u sučelju"*, slično ključnoj riječi *funcname*, s tim da su razmaci u parametru ovdje dozvoljeni.

Budući da se radi o tekstualnom opisu rada filtera, za ključnu riječ *desc* vrijede ista pravila o razmacima kao i za *name*, a sintaksa je oblika *# desc "opis filtera"*. Izostavljanjem ove ključne riječi funkcionalnost samog filtera nije umanjena, no njegov opis neće biti definiran, a zbog toga ni prikazan u statusnoj traci prilikom prelaska miša preko pripadajuće stavke u izborniku *Filters*. Međutim, zbog pristupačnosti krajnjim korisnicima, poželjno je pružiti opsežniji opis koji daje bolji uvid u rad filtera, kao što bi i informativan naziv filtera doprinio ukupnoj upotrebljivosti filtera.

Postoje filteri kojima se dodavanjem mogućnosti korisničkog unosa parametara funkcionalnost ne povećava u zamjetnoj mjeri. Primjer takvog filtera je filter koji određuje negativ trenutno otvorene slike, odnosno komplementira iznose jakosti crvene, zelene i plave boje svakog piksela u slici. Kod takvih filtera ključnu riječ *arg* nema potrebe navoditi u zaglavlju, a svoju funkciju obavljaju bez prikazivanja dijaloškog okvira prilikom njihova odabira iz izbornika *Filters*. Premda ne zahtijevaju dodatne argumente, glavna funkcija i dalje mora imati dva argumenta, s tim da je u ovom slučaju drugi argument prazan rječnik. S druge strane, postoje filteri poput Gaussovog. Gaussov filter se u obradi slika može primijeniti za njihovo izglađivanje, a za rad su mu tada potrebna dva broja, gdje prvi predstavlja vodoravnu, a drugi okomitu jačinu izglađenosti slike. Budući da o korisničkom unosu ovisi jačina izglađenosti, ovdje je potrebno korisniku omogućiti unos spomenutih vrijednosti putem dijaloškog okvira. Za ovakve i slične slučajeve, u kodu za tumačenje uvedena je podrška za argumente koji mogu biti predstavljeni jednom od sljedeće tri vrste *widgeta*:

- *spinbox*, polje za unos brojčanog iznosa;
- *dropdown*, padajući izbornik s nekoliko prethodno definiranih izbora; te
- *checkbox*, element sučelja kod kojega se u slučaju pritiska na njega u glavnu funkciju prosljeđuje binarnu vrijednost istine i obratno.

Widgeti u dijaloškom okviru pojavljuju se po redu po kojem su argumenti definirani u zaglavlju. Ukoliko po završetku unosa parametara korisnik potvrdi unos pritiskom na tipku dijaloškog okvira za potvrđivanje, uneseni podaci se obrađuju i stvara rječnik s parovima ključeva,

čije nazive određuje autor, i pripadnih vrijednosti. Potom se poziva glavna funkcija filtera kojoj se stvoreni rječnik predaje kao drugi argument. Ukoliko korisnik završi unos parametara bez pravilne potvrde, uneseni podaci se odbacuju i ne dolazi do poziva povezane funkcije.

Polje za unos brojčanog iznosa određuje se sintaksom *# arg "naziv_u_rječniku" spinbox tip_podatka [gornja_granica, donja_granica] "naziv u sučelju"*. Prvi parametar, naziv u rječniku, obvezan je i odnosi se na naziv ključa u rječniku te ne smije sadržavati razmake. Idući parametar, tip podatka, također je obvezan, a određuje tip korisničkog unosa koji može biti cijeli broj (engl. *integer*) ili broj s pomičnim zarezom (engl. *floating-point number*). Ukoliko je očekivana vrsta korisničkog unosa cijeli broj, za tip podatka navodi se ključna riječ *int*, a ako se očekuje decimalan broj, onda je ključna riječ *float*. Gornja i donja granica su parametri koji nisu obvezni, a pomoću njih moguće je odrediti najmanju ili najveću vrijednost koju korisnik može unijeti, ili oboje. Ukoliko se za tip podatka navede *int*, a jedna ili obje granice izražene su u obliku decimalnih brojeva, znamenke iza decimalnog zareza se odbacuju. Uz cjelobrojne i decimalne konstante podržane su ključne riječi *inf* i *-inf*, čijim navođenjem se granica postavlja na vrijednost 2^{31} - 1, odnosno -2^{31}, maksimalne pozitivne i negativne vrijednosti za cjelobrojne podatke u biblioteci PySide, budući da se pohranjuju u obliku 32 bita s jednim bitom za označavanje predznaka (engl. *signed*) [90]. Za uspješno postavljanje granica polja gornja granica mora biti veća od donje. Ukoliko nema potrebe za ograničavanjem korisničkog unosa moguće je u potpunosti izostaviti parametre vezane uz granice, pa naredba tada ima oblik *# arg "naziv_u_rječniku" spinbox tip_podatka "naziv u sučelju"*. Na kraju se nalazi parametar koji, kao ni prethodni nije obvezan, a određuje tekst labele pored polja u dijaloškom okviru. Ukoliko se naziv u sučelju ne navede, kod za tumačenje postavlja naziv u sučelju na vrijednost odabranu za naziv ključa u rječniku.

Naredba za definiranje padajućeg izbornika ima oblik *# arg "naziv_u_rječniku" dropdown ["Izbor 1", "Izbor 2", ... , "Izbor n"] "naziv u sučelju"*. Popis izbora u padajućem izborniku omeđenih uglatim zagradama obvezan je parametar za ovaj *widget*, a mora biti najmanje dva izbora. Kao i svi dosadašnji nizovi znakova, tekstovi izbora mogu biti omeđeni navodnicima ili apostrofima. Pravila vezana za nazive u rječniku i sučelju navedena za polje za unos brojčanog iznosa vrijede i za ovaj *widget* kao i za *checkbox*.

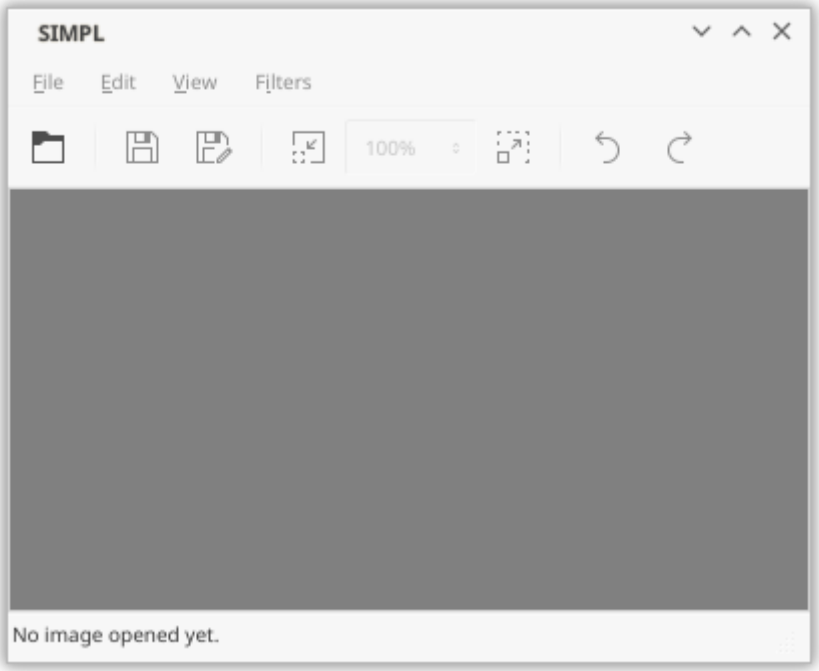
Checkbox je najjednostavniji widget za definiranje od navedenih, a definira se naredbom *# arg "naziv_u_rječniku" checkbox "naziv u sučelju"*.

Kako ne bi došlo do pogrešaka u radu filtera, autor treba voditi računa da pristupa ključevima rječnika navedenima u zaglavlju. Pored navedenih pravila nema ograničenja vezanih

uz način pisanja programskog koda filtera, ali tip podatka koji filterska funkcija vraća mora također biti NumPy matrica. Ukoliko povratni tip podatka ne odgovara navedenom, glavni program smatra da je došlo do nepoznate pogreške prilikom izvršavanja filtera.

4.2. Sučelje

Pri pokretanju programa korisniku se pojavljuje glavni prozor sličan onome na slici 4.3.



Sl. 4.3. *Glavno sučelje programa*

Na raspolaganju se nalaze četiri izbornika: *File*, *Edit*, *View* i *Filters*. U izborniku *File* nalaze se uobičajene stavke vezane uz rukovanje datotekom: otvaranje postojeće slikovne datoteke (*Open*), spremanje izmijenjene slike preko postojeće (*Save*), spremanje otvorene slike neovisno o izmjenama s nazivom datoteke i tipom slike prema korisničkom izboru (*Save As…*) te stavka *Quit* za kraj rada. Ukoliko je slika izmijenjena, korisniku se postavlja upit želi li izmijenjenu sliku spremiti, odbaciti izmjene ili obustaviti zatvaranje programa. Isti dijaloški okvir pojavljuje se i pritiskom na tipku za zatvaranje glavnog prozora programa. Program utvrđuje je li trenutna slika spremljena provjerom odgovarajuće pohranjene binarne vrijednosti u objektu iz razreda *ImageHistoryEntry* smještenim pod indeksom trenutnog položaja u povijesti izmjena slike. Sadržaj razreda *ImageHistoryEntry* prikazan je na slici 4.4.

```
class ImageHistoryEntry:
    def __init__(self, n, i, s):
        self.name = n
        self.image = i
        self.saved = s
```

Sl. 4.4. *Razred ImageHistoryEntry*

Svaki objekt iz razreda *ImageHistoryEntry* čine tri atributa: naziv unosa u povijest izmjena (*name*), NumPy matrica koja sadrži sliku (*image*) i binarna vrijednost koja govori je li slika iz promatranog objekta podudarna sa zadnje pohranjenom slikom (*saved*). Naziv unosa u povijest izmjena može biti neki od filtera ili *Original image* ukoliko se radi o izvornoj slici, prvom unosu u polje izmjena smješteno u glavni razred programa, *QSMain*.

Slijedi izbornik *Edit* koji se sastoji od stavke za poništavanje posljednje promjene (*Undo ime filtera*, ili samo *Undo* ako ne postoji prethodna izmjena) i stavke za ponovnu primjenu poništene izmjene (*Redo ime filtera* ili *Redo* ako nema kasnije izmjene). Svakim poništavanjem ili ponovnim primjenjivanjem poništene izmjene mijenja se trenutni indeks u polju izmjena, a pritom se i prikazana slika zamjenjuje onom koja se nalazi u novoizabranom *ImageHistoryEntry* objektu.

Izbornik *View* se također sastoji od samo dvije opcije. Prva opcija (*Zoom out*) obavlja smanjenje, a druga (*Zoom in*) povećanje prikaza slike. Pritom se također mijenja trenutni indeks, no ovaj se odnosi na trenutno odabrani faktor povećanja, odnosno smanjenja slike. U glavnom razredu *QSMain* polje vrijednosti povećanja slike definirano je u konstruktoru naredbom sa slike 4.5. Faktori su odabrani po uzoru na program za obradu slika GIMP [91].

```
self.zoom_levels = [8, 4, 2, 1, 0.667, 0.5, 0.333, 0.25, 0.182, 0.125]
```

Sl. 4.5. *Polje faktora povećanja slike u razredu QSMain*

Svaka od vrijednosti u polju pomnožena sa 100 predstavlja vrijednost omjera trenutno prikazane veličine i stvarne veličine slike u postocima. Stoga pri pokretanju konstruktor indeks trenutnog faktora povećanja *zoom_idx* postavlja na 3, odnosno 100%. Pomicanjem unaprijed ili unazad kroz popis faktora povećanja stavkom za smanjenje ili povećanje prikaza slike mijenja se trenutni indeks i prikaz osvježava umanjenom, odnosno uvećanom varijantom slike.

Većina stavki iz prethodno navedenih izbornika dostupna je i u obliku gumba na alatnoj traci za bolju pristupačnost. Iznimka je padajući izbornik s postotnim iznosima povećanja slike

pomoću kojega je moguć izbor željenog povećanja bez višestrukog biranja stavki za promjenu veličine prikaza slike kako bi se postigla željena veličina.

Zadnji u nizu izbornika je *Filters*. Ovaj izbornik popunjava se pomoću metode *populate_filters_menu* razreda *QSMain*, a prikazana je na slici 4.6.

```
    def populate_filters_menu(self):
        fpathl = glob.glob(self.abs_path('filters/*.py'))

        fcount = 0

        for f in fpathl:
            fhead = parse_fhead(f)
            if fhead is not None:
                fcount += 1

                fs = os.path.split(f)[1]
                fs = fs.rsplit('.', 1)[0]
                mod_name = string.join(["filters-", fs], '')

                mod = imp.load_source(mod_name, f)
                funcs = inspect.getmembers(mod, inspect.isfunction)

                filter_fobj = None

                for ftuple in funcs:
                    if ftuple[0] == fhead.funcname:
                        filter_fobj = ftuple[1]
                        break

                tmp = FilterEntry(filter_fobj, fhead)
                self.filters.append(tmp)

                if fhead.args is not None:
                    filter_name = string.join(['&', fhead.name, '...'], '')
                else:
                    filter_name = string.join(['&', fhead.name], '')
                filter_action = QtGui.QAction(filter_name, self)

                if fhead.desc != '':
                    filter_action.setStatusTip(fhead.desc)
                filter_action.setObjectName(str(fcount - 1))
                filter_action.triggered.connect(self.filter_handler)

                self.menu_filters.addAction(filter_action)

        if fcount == 0:
            no_filters_action = QtGui.QAction("&No filters available", self)
            no_filters_action.setEnabled(False)
            self.menu_filters.addAction(no_filters_action)
```

Sl. 4.6. *Metoda populate_filters_menu*

Prvi korak u popunjavanju izbornika *Filters* je pomoću metode *glob* istoimenog modula pronaći sve filterske datoteke koje se nalaze u podmapi *filters* i završavaju nastavkom *.py* [92]. Kao što je spomenuto u potpoglavlju 4.1., datoteke koje ne završavaju ovim nastavkom se ni ne obrađuju jer se u njima ne očekuje Python kod s opisom filtera. Time je izbjegnuto nenužno obrađivanje binarnih datoteka koje bi se mogle također pronaći u ovoj podmapi. Kako bi se osigurao ispravan rad programa i u slučajevima kada se radna mapa (engl. *working directory*) ne podudara s mapom u kojoj se nalazi sam program, kao argument metodi *glob* predaje se povratna vrijednost metode *abs_path* razreda *QSMain*. Metoda *abs_path* dodaje apsolutnu putanju do mape u kojoj se glavni program nalazi kao predmetak (prefiks) predanoj relativnoj putanji. Ukoliko postoje datoteke koje odgovaraju navedenim uvjetima, metoda *glob* vraća polje nizova znakova s apsolutnim putanjama mogućih filterskih datoteka. Popis filterskih datoteka obrađuje se u *for* petlji. Prvi korak u *for* petlji je predavanje putanje iz trenutne iteracije funkciji *parse_fhead* koja, ukoliko filter ispunjava sve prethodno navedene uvjete, vraća objekt razreda *FilterMeta* (Sl.4.7.) s podacima o filteru prikupljenima iz njegovog zaglavlja.

```
1  class FilterMeta:
2      def __init__(self, _name = "", _desc = "", _funcname = "", _args = None):
3          self.name = _name
4          self.desc = _desc
5          self.funcname = _funcname
6          self.args = _args
```

Sl. 4.7. *Razred FilterMeta*

Razred *FilterMeta* sadrži osnovne podatke o filteru: njegovo ime (*name*), opis (*desc*), naziv glavne funkcije (*funcname*) i polje s argumentima (*args*).

Ukoliko vraćeni objekt nije neodređenog tipa (*None*), prema funkciji *parse_fhead* filter je napisan dovoljno ispravnim kodom za uključivanje u glavni program. Za uključivanje je najprije potrebno učitati programsku datoteku filtera kao modul, što se ostvaruje pomoću metode *load_source* modula *imp* [93]. Metoda *load_source* prima dva argumenta, gdje je prvi naziv modula, a drugi putanja k datoteci s izvornim kodom filtera [94]. Za naziv modula bira se naziv filterske datoteke bez datotečnog nastavka s dodanim predmetkom *filters-*. Kako bi gubici u brzini rada programa bili što manji, u ovoj metodi, kao i u većem dijelu ostatka programa koristi se metoda *join* modula *string* za spajanje više nizova znakova u jedan, koja je prema [95] rješenje s najmanjim gubicima u brzini u usporedbi s drugim pristupima spajanju nizova znakova. Nakon određivanja naziva, filterska datoteka uključuje se kao modul iz kojega se izdvajaju uređeni parovi

imena funkcija i pripadajućih funkcijskih objekata [96]. Nakon što je u polju uređenih parova pronađena odgovarajuća funkcija prema nazivu, pohranjuje se u obliku funkcijskog objekta u objekt klase *FilterEntry* zajedno s podacima koji opisuju filter. Stvoreni objekt dodaje se na kraj polja filtera u glavnom razredu za kasnije pozivanje u njegovoj metodi *filter_handler*.

Nakon pohrane filterske funkcije i popunjavanja popisa filtera potrebno je definirati pripadajuću stavku kako bi se ona mogla dodati u izbornik *Filters*. Za početak se nazivu filtera u sučelju dodaje znak *&* (engl. *ampersand*) kao predmetak, čime se omogućava pokretanje filtera iz izbornika i pomoću tipkovnice jednim pritiskom na početno slovo imena na tipkovnici, u slučaju kada to slovo ne dijeli s drugim filterima u izborniku. Ukoliko filter prima argumente, za rad filtera potreban je dodatan korisnički unos, pa se na kraju naziva dodaje trotočje u skladu s konvencijama o korisničkom iskustvu (engl. *user experience*, skraćeno UX) [97]. Slijedi izrada stavke za izbornik *Filters* pomoću konstruktora razreda *QAction* modula *QtGui*. Kreiranoj stavci dodaje se opis ukoliko je on naveden prethodno u zaglavlju. Metodom *setObjectName* naziv objekta stavke postavlja se na indeks koji odgovara rednom broju u polju filtera u glavnom razredu kako bi se kasnije u funkciji *filter_handler* jednostavnije odredilo koji filter je korisnik odabrao iz izbornika [98]. Naposljetku se stvoreni *QAction* dodaje u izbornik *Filters*.

Postupak se ponavlja za ostale datoteke u podmapi. Na kraju postupka, izbornik *Filters* trebao bi biti popunjen svim ispravnim filterima koji su pronađeni u podmapi. Ukoliko niti jedan ispravan filter nije pronađen, u izbornik se dodaje stavka koja korisniku govori da nema filtera dostupnih za uporabu (*No filters available*). Ovu stavku korisnik ne može pokrenuti jer ima isključivo informativnu svrhu.

Kada korisnik odabere jedan od filtera iz izbornika *Filters*, poziva se metoda *filter_handler* glavnog razreda, prikazana na slici 4.8.

```
    def filter_handler(self):
        s = int(self.sender().objectName())
        f = self.filters[s]
        n = f.filter_name
        fo = f.function_obj
        a = f.filter_args

        if a is not None:
            k = QSFilterDialog(n, a, self).run()
        else:
            k = []

        if k is not None:
            try:
                i = fo(self.cur_img, k)
            except Exception as e:
                self.show_err_dlg(
                    string.join(
                        ["An error has occurred while running ", n,
                         ". Please try adjusting the parameters or contact the "
                         "filter author."], ''
                    ),
                    str(e)
                )
            else:
                if not isinstance(i, ndarray):
                    self.show_err_dlg(
                        string.join(
                            [n,
                             " did not return a valid image object. Please try "
                             "adjusting the parameters or contact the filter "
                             "author."], ''
                        )
                    )
                else:
                    if self.image_history_idx != len(self.image_history) - 1:
                        self.image_history = \
                            self.image_history[:self.image_history_idx + 1]

                    self.image_history.append(ImageHistoryEntry(n, i, False))
                    self.image_history_idx = len(self.image_history) - 1
                    self.cur_img = i

                    self.update_window()
```

Sl. 4.8. *Metoda filter_handler*

Kako bi se povezala odabrana stavka iz izbornika Filters s odgovarajućim filterom u polju *FilterEntry* objekata glavnog razreda, potrebno je najprije iz pozivateljskog objekta dohvatiti naziv (*objectName*) prethodno postavljen u metodi *populate_filters_menu*. Naziv objekta je indeks odgovarajućeg *FilterEntry* objekta u polju glavnog razreda u tekstualnom obliku, pa ga je potrebno pretvoriti u cjelobrojni podatak. Nakon dohvaćanja *FilterEntry* objekta na specificiranom indeksu, ukoliko su za filter navedeni argumenti, instancira se objekt razreda *QSFilterDialog* (Sl.4.9.).

```
class QSFilterDialog(QtGui.QDialog):
    def __init__(self, name, args, parent = None):
        super(QSFilterDialog, self).__init__(parent)

        self.setWindowTitle(name)
        self.setModal(True)

        self.args = args
        self.widgets = []
        self.lyt = QtGui.QFormLayout()

        self.lyt.setSizeConstraint(QtGui.QLayout.SizeConstraint.SetFixedSize)

        i = 0
        for a in self.args:
            if a.ui_widget == ArgUIWidget.QSpinBox:
                w = QtGui.QSpinBox(self)
            elif a.ui_widget == ArgUIWidget.QDoubleSpinBox:
                w = QtGui.QDoubleSpinBox(self)
            elif a.ui_widget == ArgUIWidget.QComboBox:
                w = QtGui.QComboBox(self)
                for v in a.string_values:
                    w.addItem(v)
            elif a.ui_widget == ArgUIWidget.QCheckBox:
                w = QtGui.QCheckBox(self)

            w.setObjectName(a.key_name)

            if a.ui_widget == ArgUIWidget.QSpinBox \
                    or a.ui_widget == ArgUIWidget.QDoubleSpinBox:
                if a.limit_lower is not None:
                    w.setMinimum(a.limit_lower)
                else:
                    minv = -2**31
                    w.setMinimum(minv)

                if a.limit_upper is not None:
                    w.setMaximum(a.limit_upper)
                else:
                    maxv = 2**31 - 1
                    w.setMaximum(maxv)

            self.widgets.append(w)
            self.lyt.addRow(string.join([a.gui_name, ':'], ''), self.widgets[i])
            i += 1

        self.btns = QtGui.QDialogButtonBox(
            QtGui.QDialogButtonBox.Ok | QtGui.QDialogButtonBox.Cancel
        )
        self.btns.accepted.connect(self.accept)
        self.btns.rejected.connect(self.reject)
        self.lyt.addRow(self.btns)
        self.setLayout(self.lyt)
```

Sl. 4.9. *Razred QSFilterDialog, konstruktor*

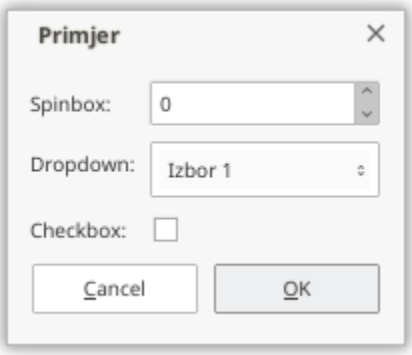
QSFilterDialog je klasa izvedena iz razreda *QDialog* PySide biblioteke iz modula *QtGui*. U konstruktoru ovog razreda dinamički se izrađuje sučelje dijaloškog okvira za unos korisničkih parametara za filtere u skladu s predanim argumentima. Parametri koji se predaju konstruktoru ovog razreda su naslov dijaloškog okvira, popis argumenata i nadređeni *QWidget*. Za naslov dijaloškog okvira metoda *filter_handler* konstruktoru predaje naziv filtera iz dohvaćenog *FilterEntry* objekta. Na temelju popisa argumenata u dijaloški okvir dodaju se *widgeti* navedeni u popisu. Svaki *widget* dobiva svoj naziv u skladu s rječničkim ključem kojega je autor naveo u zaglavlju, te, ukoliko se radi o *spinbox widgetu*, postavljaju gornje i donje granice, ako su one navedene. Na labelu pokraj *widgeta* piše se naziv u sučelju u skladu s povratnom informacijom funkcije *parse_fhead*. Treći argument moguće je izostaviti bez neposrednog prestanka rada programa, no bez navođenja nadređenog *QWidgeta* moguća su curenja memorije (engl. *memory leaks*) te se narušava modalna priroda dijaloških okvira [99, 100].

Na slici 4.10. prikazan je primjer filtera sa zaglavljem koji ima nekoliko različitih argumenata navedenih sa specificiranim nazivima u sučelju. Funkcija *primjer* uvedena je zbog cjelovitosti primjera, ali osobitu funkcionalnost nema, već samo vraća primljenu sliku u neizmijenjenom obliku.

```
1   ## begin fhead
2    #
3    # name     "Primjer"
4    # funcname "primjer"
5    # desc     "Primjer zaglavlja"
6    #
7    # arg "p_spinbox"  spinbox int [-500, 1500] "Spinbox"
8    # arg "p_dropdown" dropdown ["Izbor 1", "Izbor 2", "Izbor 3"] "Dropdown"
9    # arg "p_checkbox" checkbox "Checkbox"
10   #
11  ## end fhead
12
13  def primjer(img, args):
14      return img
```

Sl. 4.10. *Primjer jednostavnog filtera*

Slika 4.11. prikazuje izgled dijaloškog okvira izrađenog prema zaglavlju napisanog filtera.



Sl. 4.11. *Dijaloški okvir za unos korisničkih podataka*

Po završetku rada konstruktora, metodi *filter_handler* vraća se *QSFilterDialog* objekt na kojem se odmah pokreće metoda *run* (Sl.4.12.).

```
1       def run(self):
2           r = self.exec_()
3           if r == QtGui.QDialog.Accepted:
4               d = {}
5
6               i = 0
7               for w in self.widgets:
8                   arg_uiw = self.args[i].ui_widget
9                   if arg_uiw == ArgUIWidget.QSpinBox \
10                          or arg_uiw == ArgUIWidget.QDoubleSpinBox:
11                      v = w.value()
12                  elif arg_uiw == ArgUIWidget.QComboBox:
13                      v = w.currentText()
14                  elif arg_uiw == ArgUIWidget.QCheckBox:
15                      v = w.isChecked()
16
17                  n = w.objectName()
18                  d[n] = v
19
20                  i += 1
21
22              return d
23          else:
24              return None
```

Sl. 4.12. *Razred QSFilterDialog, metoda run*

Metoda *run* prikazuje izrađeni dijaloški okvir te ukoliko korisnik prihvati unesene podatke pritiskom na tipku za potvrdu unosa, iz njega se u *for* petlji prikupljaju podaci u rječnik koji se na kraju rada vraća metodi *filter_handler*. Slika 4.13. prikazuje sadržaj rječnika za prethodno opisani primjer filtera uz podatke sa slike 4.11.

```
{
    u'p_dropdown': u'Izbor 1',
    u'p_spinbox': 0,
    u'p_checkbox': False
}
```

Sl. 4.13. *Rječnik s podacima prikupljenima iz dijaloškog okvira s pretprošle slike*

Prekidom rada dijaloškog okvira na bilo koji drugi način uneseni podaci se odbacuju i program nastavlja s radom bez poziva filtera.

Metoda *filter_handler* trenutnu sliku i primljeni rječnik predaje u pozivu glavnoj funkciji filtera. Po završetku rada filtera na kraj povijesti izmjena slike dodaje se novi objekt razreda *ImageHistoryEntry* kojemu se za naziv stavke predaje naziv filtera, slika je NumPy matrica koju vraća filter, a za parametar vezan uz podudarnost slike s onom pohranjenom na računalu predaje se *False*, jer izrađenu sliku nije moguće spremiti prije završetka metode. Potom se trenutna slika u memoriji postavlja na sliku koju je vratio filter nakon obrade i poziva metoda *update_window* za osvježavanje cjelokupnog glavnog prozora, pri čemu se ažuriraju naslov, izbornik *Edit* te trenutno prikazana slika u prozoru.

Ukoliko indeks koji upućuje na položaj u povijesti izmjena slike ne pokazuje na kraj, sve stavke u povijesti poslije trenutne odbacuju se, a potom se na kraj dodaje *ImageHistoryEntry* objekt. Metoda *filter_handler* poziva metodu *show_err_dlg* istog razreda, koja prikazuje dijaloški okvir s opisom pogreške, ukoliko filter tijekom rada podigne iznimku ili ako filter uspješno završi s radom, no povratni tip podatka nije NumPy matrica. Drugi argument metode *show_err_dlg* je namijenjen za predavanje detaljnijeg opisa pogreške, ako postoji. U slučaju pojave iznimke kao drugi argument ovoj metodi predaje se njezin tekstualni opis.

5. ZAKLJUČAK

U ovome završnom radu predstavljena je biblioteka Qt. Ukratko je opisana povijest razvoja, svojstva biblioteke i temeljna obilježja kojima se ističe u odnosu na ostala višeplatformska rješenja za izradu grafičkih sučelja programa. Qt ima jezične poveznice s brojnim programskim jezicima, no budući da je u ovome radu pažnja posvećena prvenstveno razvoju grafičkih aplikacija u programskom jeziku Python, opisana je jezična poveznica PySide koja omogućuje primjenu biblioteke Qt za izradu programa s grafičkim sučeljem u jeziku Python, odnosno omogućuje uporabu elemenata grafičkog sučelja koje pruža Qt u programima napisanima u Pythonu. Napravljena je usporedba s tri druge biblioteke otvorenog koda rasprostranjene u razvoju višeplatformskih aplikacija, pri čemu su istaknuta svojstva koja Qt dijeli s tim bibliotekama, kao i svojstva koja ih razlikuju.

Na kraju je predstavljen primjer programa napisanog u Pythonu čije grafičko sučelje se oslanja na biblioteku Qt posredstvom jezične poveznice PySide. Program služi pregledavanju i izmjenjivanju slika primjenom filtera napisanih u Pythonu. Način pisanja filtera osmišljen je tako da autori filtera ne moraju upoznavati način rada samog programa, već se mogu usredotočiti na razvoj samog filtera. Za uključivanje filtera u glavno sučelje programa i opisivanje parametara koje autor očekuje od korisnika osmišljena je sintaksa pomoću koje autor može opisati svoj filter demonstriranim skupom ključnih riječi. Filteri se opisuju u zaglavlju koje se smješta na proizvoljno mjesto u datoteke s programskim kodom filtera. Prikazan je izvorni kod i objašnjen način uključivanja filtera u glavno sučelje programa i komunikacija programa s filterima, a navedeni su i osnovni uvjeti koje filter mora ispuniti kako se napisani kod ne bi smatrao neispravnim.

U moguća poboljšanja programu ubrajaju se napredniji obilazak i prikaz povijesti izmjena slike, pregledavanje drugih slika iz mape u kojoj se trenutno otvorena slika nalazi koristeći strelice na tipkovnici ili u sučelju programa, umjesto ručnog otvaranja svake nove slike dijaloškim okvirom za odabir datoteke te podrška za rukovanje razmacima u nazivima ključeva argumenata u rječnicima.

LITERATURA

[1] Proprietary software that uses Qt [online], Wikimedia Foundation, 2014., dostupno na: https://en.wikipedia.org/wiki/Category:Proprietary_software_that_uses_Qt [16. 6. 2017.]

[2] EAGLE (program) [online], Wikimedia Foundation, 2005., dostupno na: https://en.wikipedia.org/wiki/EAGLE_(program) [16. 6. 2017.]

[3] FreeMat [online], Wikimedia Foundation, 2006., dostupno na: https://en.wikipedia.org/wiki/FreeMat [16. 6. 2017.]

[4] QGIS [online], Wikimedia Foundation, 2014., dostupno na: https://hr.wikipedia.org/wiki/QGIS [16. 6. 2017.]

[5] VLC [online], Wikimedia Foundation, 2010., dostupno na: https://hr.wikipedia.org/wiki/VLC [16. 6. 2017.]

[6] Qt (software) [online], Wikimedia Foundation, 2001., dostupno na: https://en.wikipedia.org/wiki/Qt_(software) [7. 6. 2017.]

[7] M. Ettrich, New Project: Kool Desktop Environment. Programmers wanted! [online], Google, 1996., dostupno na: http://groups.google.com/group/de.comp.os.linux.misc/msg/cb4b2d67ffc3ffce [16. 6. 2017.]

[8] KDE [online], Wikimedia Foundation, 2001., dostupno na: https://en.wikipedia.org/wiki/KDE [16. 6. 2017.]

[9] J. Blanchette, M. Summerfield, C++ GUI Programming with Qt 4, Prentice Hall, Upper Saddle River, 2006.

[10] Qt version history [online], Wikimedia Foundation, 2013., dostupno na: https://en.wikipedia.org/wiki/Qt_version_history [16. 6. 2017.]

[11] Supported Platforms [online], The Qt Company, dostupno na: http://doc.qt.io/qt-5/supported-platforms.html [16. 6. 2017.]

[12] Widget toolkit [online], Wikimedia Foundation, 2003., dostupno na: https://en.wikipedia.org/wiki/Widget_toolkit [16. 6. 2017.]

[13] Qt Quick [online], Wikimedia Foundation, 2010., dostupno na: https://en.wikipedia.org/wiki/Qt_Quick [16. 6. 2017.]

[14] QML [online], Wikimedia Foundation, 2010., dostupno na: https://en.wikipedia.org/wiki/QML [16. 6. 2017.]

[15] List of language bindings for Qt 4 [online], Wikimedia Foundation, 2013., dostupno na: https://en.wikipedia.org/wiki/List_of_language_bindings_for_Qt_4 [16. 6. 2017.]

[16] List of language bindings for Qt 5 [online], Wikimedia Foundation, 2013., dostupno na: https://en.wikipedia.org/wiki/List_of_language_bindings_for_Qt_5 [16. 6. 2017.]

[17] Callback (computer programming) [online], Wikimedia Foundation, 2003., dostupno na: https://en.wikipedia.org/wiki/Callback_(computer_programming) [16. 6. 2017.]

[18] intro [online], dostupno na: http://www.open-std.org/jtc1/sc22/open/n2356/intro.html [16. 6. 2017.]

[19] cv (const and volatile) type qualifiers [online], 2011., dostupno na: http://en.cppreference.com/w/cpp/language/cv [16. 6. 2017.]

[20] Const member functions in C++ [online], GeeksforGeeks, 2014., dostupno na: http://www.geeksforgeeks.org/const-member-functions-c/ [16. 6. 2017.]

[21] Signals & Slots [online], The Qt Company, dostupno na: http://doc.qt.io/qt-4.8/signalsandslots.html [7. 6. 2017.]

[22] Signals and slots [online], Wikimedia Foundation, 2005., dostupno na: https://en.wikipedia.org/wiki/Signals_and_slots [16. 6. 2017.]

[23] Why Does Qt Use Moc for Signals and Slots? [online], The Qt Company, dostupno na: http://doc.qt.io/qt-5/why-moc.html [7. 6. 2017.]

[24] Tkinter [online], Wikimedia Foundation, 2005., dostupno na: https://en.wikipedia.org/wiki/Tkinter [16. 6. 2017.]

[25] PyQt [online], Wikimedia Foundation, 2004., dostupno na: https://en.wikipedia.org/wiki/PyQt [16. 6. 2017.]

[26] What is PyQt? [online], Riverbank Computing, dostupno na: https://www.riverbankcomputing.com/software/pyqt/intro [16. 6. 2017.]

[27] PySide [online], Python Software Foundation, 2010., dostupno na: https://wiki.python.org/moin/PySide [16. 6. 2017.]

[28] PySide [online], 2009., dostupno na: https://www.freecadweb.org/wiki/index.php?title=PySide [16. 6. 2017.]

[29] FreeCAD: An open-source parametric 3D CAD modeler [online], dostupno na: https://www.freecadweb.org/ [16. 6. 2017.]

[30] M. Lira, Shiboken [online], 2009., dostupno na: https://setanta.wordpress.com/2009/08/31/shiboken/ [16. 6. 2017.]

[31] PySide FAQ [online], The Qt Company, 2015., dostupno na: https://wiki.qt.io/PySide_FAQ [16. 6. 2017.]

[32] PySide [online], The Qt Company, 2015., dostupno na: https://wiki.qt.io/PySide [16. 6. 2017.]

[33] PySide [online], Wikimedia Foundation, 2009., dostupno na: https://en.wikipedia.org/wiki/PySide [15. 6. 2017.]

[34] Overview [online], dostupno na: https://pyside.github.io/docs/pyside/ [15. 6. 2017.]

[35] The GTK+ Project [online], GNOME Foundation, dostupno na: https://www.gtk.org/ [16. 6. 2017.]

[36] Wayland (display server protocol) [online], Wikimedia Foundation, 2009., dostupno na: https://en.wikipedia.org/wiki/Wayland_(display_server_protocol) [16. 6. 2017.]

[37] X Window System [online], Wikimedia Foundation, 2001., https://en.wikipedia.org/wiki/X_Window_System [16. 6. 2017.]

[38] AbiWord [online], Wikimedia Foundation, 2001., dostupno na: https://en.wikipedia.org/wiki/AbiWord [21. 6. 2017.]

[39] Chromium (web browser) [online], Wikimedia Foundation, 2009., dostupno na: https://en.wikipedia.org/wiki/Chromium_(web_browser) [21. 6. 2017.]

[40] GIMP [online], Wikimedia Foundation, 2001., dostupno na: https://en.wikipedia.org/wiki/GIMP [16. 6. 2017.]

[41] Inkscape [online], Wikimedia Foundation, 2003., dostupno na: https://en.wikipedia.org/wiki/Inkscape [21. 6. 2017.]

[42] Pidgin (software) [online], Wikimedia Foundation, 2003., dostupno na: https://en.wikipedia.org/wiki/Pidgin_(software) [21. 6. 2017.]

[43] Desktop environment [online], Wikimedia Foundation, 2002., dostupno na: https://en.wikipedia.org/wiki/Desktop_environment [21. 6. 2017.]

[44] GTK+ Features [online], GNOME Foundation, dostupno na: https://www.gtk.org/features.php [16. 6. 2017.]

[45] S. Hackvän, Where did Spencer Kimball and Peter Mattis go? [online], Internet Archive (Web Publishing), 1999., dostupno na: https://web.archive.org/web/19990417052141/http://www.linuxworld.com/linuxworld/lw-1999-01/lw-01-gimp.html [16. 6. 2017.]

[46] What is the + in GTK+? [online], Internet Archive (GNOME Foundation), dostupno na: https://web.archive.org/web/20120326131857/http://developer.gnome.org/gtk-faq/stable/x90.html [16. 6. 2017.]

[47] Cairo (graphics) [online], Wikimedia Foundation, 2004., dostupno na: https://en.wikipedia.org/wiki/Cairo_(graphics) [16. 6. 2017.]

[48] GTK+ to Use Cairo Vector Engine [online], BizX, 2005., dostupno na: https://developers.slashdot.org/story/05/02/04/2021236/gtk-to-use-cairo-vector-engine [16. 6. 2017.]

[49] GTK+ [online], Wikimedia Foundation, 2001., dostupno na: https://en.wikipedia.org/wiki/GTK%2B [9. 6. 2017.]

[50] E. Loli, GTK+ Goes Cairo; Owen Taylor on X/Cairo/GTK+ Integration [online], OSNews, 2005., dostupno na: http://www.osnews.com/story/9609 [16. 6. 2017.]

[51] List of language bindings for GTK+ [online], Wikimedia Foundation, 2013., dostupno na: https://en.wikipedia.org/wiki/List_of_language_bindings_for_GTK%2B [16. 6. 2017.]

[52] Natural user interface [online], Wikimedia Foundation, 2009., dostupno na: https://en.wikipedia.org/wiki/Natural_user_interface [16. 6. 2017.]

[53] FAQ [online], dostupno na: https://kivy.org/docs/faq.html [16. 6. 2017.]

[54] Raspberry Pi [online], Wikimedia Foundation, 2011., dostupno na: https://en.wikipedia.org/wiki/Raspberry_Pi [16. 6. 2017.]

[55] PyMT [online], Google, dostupno na: https://code.google.com/archive/p/pymt/ [16. 6. 2017.]

[56] kivy/pyjnius: Access Java classes from Python [online], 2012., dostupno na: https://github.com/kivy/pyjnius/ [16. 6. 2017.]

[57] kivy/pyobjus: Access Objective-C classes from Python [online], 2012., dostupno na: https://github.com/kivy/pyobjus/ [16. 6. 2017.]

[58] kivy/kivy-designer: UI designer for Kivy (WIP) [online], 2012., dostupno na: https://github.com/kivy/kivy-designer/ [16. 6. 2017.]

[59] Kivy Garden [online], dostupno na: https://github.com/kivy-garden/ [16. 6. 2017.]

[60] Raspbian [online], 2012., dostupno na: https://www.raspbian.org/ [16. 6. 2017.]

[61] TUIO [online], NUI Foundation, 2008., dostupno na: http://wiki.nuigroup.com/TUIO [16. 6. 2017.]

[62] TUIO [online], Reactable Systems, dostupno na: https://www.tuio.org/ [16. 6. 2017.]

[63] Python on Android? [online], Reddit, 2013., dostupno na: https://www.reddit.com/r/Python/comments/1cgeut/python_on_android/c9gbz2f/ [16. 6. 2017.]

[64] Ž. Mihajlović, 5. Modeliranje i reprezentacija objekata [online], Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, 2016., dostupno na: http://www.zemris.fer.hr/predmeti/irg/predavanja/5_modeliranje.pdf [16. 6. 2017.]

[65] Vertex Buffer Object [online], Wikimedia Foundation, 2008., dostupno na: https://en.wikipedia.org/wiki/Vertex_Buffer_Object [16. 6. 2017.]

[66] Kivy (framework) [online], Wikimedia Foundation, 2012., dostupno na: https://en.wikipedia.org/wiki/Kivy_(framework) [10. 6. 2017.]

[67] Kv Design Language [online], dostupno na: https://kivy.org/docs/gettingstarted/rules.html [16. 6. 2017.]

[68] Separation of concerns [online], Wikimedia Foundation, 2003., dostupno na: https://en.wikipedia.org/wiki/Separation_of_concerns [16. 6. 2017.]

[69] wxWidgets [online], Wikimedia Foundation, 2003., dostupno na: https://en.wikipedia.org/wiki/WxWidgets [14. 6. 2017.]

[70] MetaCASE tool [online], Wikimedia Foundation, 2009., dostupno na: https://en.wikipedia.org/wiki/MetaCASE_tool [16. 6. 2017.]

[71] Computer-aided software engineering [online], Wikimedia Foundation, 2004., dostupno na: https://en.wikipedia.org/wiki/Computer-aided_software_engineering [16. 6. 2017.]

[72] History [online], dostupno na: https://www.wxwidgets.org/about/history/ [14. 6. 2017.]

[73] Center for Naval Analyses [online], Wikimedia Foundation, 2004., dostupno na: https://en.wikipedia.org/wiki/Center_for_Naval_Analyses [16. 6. 2017.]

[74] 0 A.D. (video game) [online], Wikimedia Foundation, 2005., dostupno na: https://en.wikipedia.org/wiki/0_A.D._(video_game) [16. 6. 2017.]

[75] J. Smart, K. Hock, S. Csomor, Cross-Platform GUI Programming with wxWidgets, Prentice Hall, Upper Saddle River, 2005.

[76] Code::Blocks [online], Wikimedia Foundation, 2005., dostupno na: https://en.wikipedia.org/wiki/Code::Blocks [16. 6. 2017.]

[77] CodeLite [online], Wikimedia Foundation, 2008., dostupno na: https://en.wikipedia.org/wiki/CodeLite [16. 6. 2017.]

[78] FileZilla [online], Wikimedia Foundation, 2004., dostupno na: https://en.wikipedia.org/wiki/FileZilla [16. 6. 2017.]

[79] S. Quarton, The 8 Best FTP Clients for WordPress Users in 2016 [online], Elegant Themes, 2016., dostupno na: https://www.elegantthemes.com/blog/resources/best-ftp-clients-for-wordpress-users [16. 6. 2017.]

[80] Cocoa (API) [online], Wikimedia Foundation, 2002., dostupno na: https://en.wikipedia.org/wiki/Cocoa_(API) [16. 6. 2017.]

[81] wxEmbedded [online], Koan, dostupno na: http://www.wxembedded.org/ [16. 6. 2017.]

[82] Platform Details [online], dostupno na: http://docs.wxwidgets.org/trunk/page_port.html [14. 6. 2017.]

[83] Overview [online], dostupno na: https://www.wxwidgets.org/about/ [16. 6. 2017.]

[84] WxWidgets Compared To Other Toolkits [online], 2006., dostupno na: https://wiki.wxwidgets.org/WxWidgets_Compared_To_Other_Toolkits [14. 6. 2017.]

[85] List of Integrated Development Environments [online], 2006., dostupno na: https://wiki.wxwidgets.org/List_of_Integrated_Development_Environments [14. 6. 2017.]

[86] List of language bindings for wxWidgets [online], Wikimedia Foundation, 2013., dostupno na: https://en.wikipedia.org/wiki/List_of_language_bindings_for_wxWidgets [16. 6. 2017.]

[87] start-up tvrtka > razvojna tvrtka [online], Institut za hrvatski jezik i jezikoslovlje, dostupno na: http://bolje.hr/rijec/start-up-tvrtka-gt-gt-razvojna-tvrtka/2/ [12. 8. 2017.]

[88] About [online], dostupno na: http://opencv.org/about.html [12. 8. 2017.]

[89] 2. Lexical analysis [online], Python Software Foundation, dostupno na: https://docs.python.org/2/reference/lexical_analysis.html [12. 8. 2017.]

[90] In Pyside, why does emiting an integer > 0x7FFFFFFF result in "OverflowError" after the signal is processed? [online], Stack Exchange, 2012., dostupno na: https://stackoverflow.com/questions/10762809/in-pyside-why-does-emiting-an-integer-0x7fffffff-result-in-overflowerror-af [20. 8. 2017.]

[91] gimpzoommodel.c [online], GNOME Foundation, dostupno na: https://git.gnome.org/browse/gimp/tree/libgimpwidgets/gimpzoommodel.c [18. 8. 2017.]

[92] Find all files in a directory with extension .txt in Python [online], Stack Exchange, 2010., dostupno na: https://stackoverflow.com/questions/3964681/find-all-files-in-a-directory-with-extension-txt-in-python [18. 8. 2017.]

[93] How to import a module given the full path? [online], Stack Exchange, 2008., dostupno na: https://stackoverflow.com/questions/67631/how-to-import-a-module-given-the-full-path [18. 8. 2017.]

[94] 31.1. imp [online], Python Software Foundation, dostupno na: https://docs.python.org/2/library/imp.html [19. 8. 2017.]

[95] Efficient String Concatenation in Python [online], 2004., dostupno na: https://waymoot.org/home/python_string/ [19. 8. 2017.]

[96] Is it possible to list all functions in a module? [online], Stack Exchange, 2010., dostupno na: https://stackoverflow.com/questions/4040620/is-it-possible-to-list-all-functions-in-a-module [19. 8. 2017.]

[97] Labels [online], KDE, 2008., dostupno na: https://community.kde.org/KDE_Visual_Design_Group/HIG/Labels [21. 8. 2017.]

[98] Differentiating between signal sources in PySide [online], Stack Exchange, 2012., dostupno na: https://stackoverflow.com/questions/9144936/differentiating-between-signal-sources-in-pyside [19. 8. 2017.]

[99] getting value from QSpinBox created in a QFormLayout [online], Stack Exchange, 2013., dostupno na: https://stackoverflow.com/questions/15892723/getting-value-from-qspinbox-created-in-a-qformlayout [19. 8. 2017.]

[100] QDialog without tab in task bar [online], Stack Exchange, 2013., dostupno na: https://stackoverflow.com/questions/14716386/qdialog-without-tab-in-task-bar [19. 8. 2017.]

SAŽETAK

U ovome radu opisana je biblioteka Qt, višeplatformski radni okvir otvorenog koda za izradu grafičkih korisničkih sučelja računalnih programa. Pruženi su uvid u povijest nastanka Qt-a, njegova svojstva, tri osnovne značajke (apstrakcija od izrade grafičkog sučelja, signali i slotovi i metaobjektni prevoditelj) i jezična poveznica koja omogućuje korištenje Qt biblioteke u programskom jeziku Python. Primjenom jezičnih poveznica programer ne mora poznavati programski jezik u kojemu je biblioteka izvorno napisana jer poveznica sadrži objekte i metode ili funkcije biblioteke napisane u programeru poznatom jeziku. Predstavljene su tri druge široko rasprostranjene višeplatformske biblioteke otvorenog koda za izradu sučelja računalnih programa: GTK+, Kivy i wxWidgets. Pritom su istaknuta svojstva po kojima se te biblioteke razlikuju od Qt-a i ona koja su im zajednička. Na kraju je predstavljen primjer programa napisanog u Pythonu pomoću Qt biblioteke koji omogućuje učitavanje i prikaz slika, njihovu obradu koristeći skriptirane filtere te pregled i pohranu izmijenjene slike. Filteri obrađuju slike primjenom OpenCV biblioteke.

Ključne riječi: Qt, Python, grafičko korisničko sučelje, PySide, jezična poveznica, obrada slika, OpenCV

ABSTRACT

Graphical user interface in Python using Qt

This paper describes the Qt library, an open-source, cross-platform framework used for developing graphical user interfaces of computer programs. An insight is given into the history of creation, features, three principal properties of Qt (abstraction of the graphical user interface, signals and slots and the metaobject compiler), and a language binding which allows the usage of Qt in Python programming language. Language bindings relieve the programmer of the need to learn the language in which a library is originally written as the binding contains objects and methods or functions of the library written in a language known to the programmer. Three other widespread open-source cross-platform libraries for graphical user interface development are presented: GTK+, Kivy, and wxWidgets. Features which differ Qt from these libraries are emphasized as well as features which Qt has in common with them. Finally, a program example written in Python using Qt is demonstrated, which allows loading and viewing images, their manipulation using scripted filters, previewing and saving the altered image. Filters perform image processing using the OpenCV library.

Keywords: Qt, Python, graphical user interface, PySide, language binding, image processing, OpenCV

ŽIVOTOPIS

Siniša Komaromi rođen je 26. siječnja 1996. godine u Vukovaru. 2010. godine završava Osnovnu školu Nikole Andrića u Vukovaru i iste godine upisuje se u Tehničku školu Nikole Tesle, smjer tehničar za računalstvo. 2014. godine završava srednju školu i upisuje se na sveučilišni preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

Siniša Komaromi

PRILOZI

U prilogu završnog rada se na CD-u uz elektroničku verziju završnog rada u obliku Word i PDF datoteka nalazi i cjelokupan izvorni kod programa predstavljenog u četvrtom poglavlju.