Projektiranje pismovnog reza u ClearType standardu

Nađ, Hrvoje

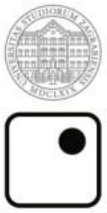
SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

HRVOJE NAĐ

PROJEKTIRANJE PISMOVNOG REZA U CLEARTYPE STANDARDU

DIPLOMSKI RAD

Zagreb, 2012



Sveučilište u Zagrebu
Grafički fakultet

HRVOJE NAĐ

PROJEKTIRANJE PISMOVNOG REZA U CLEARTYPE STANDARDU

DIPLOMSKI RAD

Mentor:
doc. dr. sc., Ivana Žiljak Stanimirović

Student:
Hrvoje Nađ

Zagreb, 2012

SAŽETAK

Diplomski rad temelji se na rješavanju problema prikazivanja tipografije na ekranima u Windows okruženju. ClearType tehnologija ugrađena u novije Windows operacijske sustave omogućava precizno podpikselno iscrtavanje, a u svrhu toga potrebno je napraviti optimizaciju prikazivanja za ekrane.

Cilj diplomskog rada jest napraviti pismovni rez sa efikasnijim radnim tokom za prilagodbu prikaza na ekranima. Radni tok uključuje odabir vrste pisma koje pogoduje čitljivosti, vrstu elektronskog zapisa i krivulja, programiranje u programskom jeziku Python te ručni proces prilagođavanja tipografskih znakova mreži piksela.

Napravljen je pismovni rez i optimiziran prečacem koji prenosi informacije o optimizaciji iz Type 1 tehnologije u TrueType tehnologiju, izbjegavajući direktno optimiziranje u TrueType tehnologiji. Informacije o optimizaciji slovnih znakova prikazu na ekranima automatizirane su u Type 1 tehnologiji putem Python skripte.

Dobiveni pismovni rez dokazuje da postoje novi načini kako unaprijediti računarsku tipografiju prikazivanju na ekranima u procesu koji efikasno smanjuje vrijeme izrade pismovnog reza te dokaz da programiranje donosi jednu sasvim novu dimenziju u izradi pismovnog reza namijenjenom ClearType tehnologiji.

Ključne riječi: Optimizacija, programiranje, Python, ClearType

ABSTRACT

Graduate work is based on solving the problem of displaying typography on screens in a Windows evnironment. ClearType technology built into newer Windows operating systems allows accurate subpixel rendering. In order to do this, there is need to optimize typeface for screen purposes.

The goal is to make a typeface with more efficient workflow during the optimization process for screen purposes. Workflow includes a selection of typeface that favors readability, type of electronic record, curves, programming in Python and manual adjustment of typograhic characters to fit the pixel grid.

Typeface made for this graduate work is optimized with a shortcut that transfers information from the Type 1 optimization technology into TrueType technology, avoiding direct manual optimization in TrueType technology. Information in character optimization is made with automatization of Python script.

The new typeface proves that there are new ways to improve the computational representation of typography on screens in a process that effectively reduces the time making a typeface and evidence that programming brings a totally new dimension in making a typeface for ClearType technology.

Keywords: optimization, programming, Python, ClearType

Sadržaj

1. UVOD

Prikaz tipografije na ekranu tehnički je zahtjevan posao, stoga se u modernoj računalnoj tipografiji razvija kompleksan matematički model kako bi se što više ujednačila kvaliteta prikaza pismovnog reza na različitim platformama i tehnologijama. Svojevrsna različitost operacijskih sustava, digitalnih formata zapisa pismovnog reza, metoda iscrtavanja tipografije, raznovrsnost izlaznih razlučivosti na ekranima i različitost tehnologija proizvodnje ekrana čine prepreku idealnom prikazu tipografije u elektronskim medijima.

Prikaz tipografije na ekranu počinje od operacijskog sustava, u kojem su definirana pravila iscrtavanja tipografije. U najzastupljenijem operacijskom sustavu današnjice (Windows) nalazi se najviše prepreka jednom pismovnom rezu kako bi na ekranima postigao svoj pravilan oblik. Proteklo desetljeće navedeni operacijski sustav dobio je tehnologiju ClearType kako bi se iskoristile prednosti LCD ekrana u vidu poboljšanog prikaza tipografije.

ClearType tehnologija dokazano poboljšava kvalitetu prikazane tipografije na ekranu, ali tek onda ukoliko pismovni rez tehnološki udovoljava zahtjevima ClearType tehnologije. U odnosu na ostale platforme, za navedenu tehnologiju treba napraviti „pametno“ skrivanje informacija na pojedinim zonama svakog tipografskog oblika, gdje operacijski sustav i ClearType tehnologija pomoću različitih algoritama prilagođavaju izgled pismovnog reza trenutnoj veličini tipografskih oblika te sve zajedno usklađuju sa izlaznom rezolucijom ekrana.

Zbog navedene tehnologije ugrađene u Windows operacijski sustav, dizajnerima tipografima je teško prilagoditi pismovni rez jer zahtjeva dodatno računanje, programiranje i planiranje izvedbe tipografije na ekranu. Rezultat cijele situacije jest da na tržištu postoji vrlo mali broj pismovnih rezova koji su prilagođen ClearType tehnologiji. Na globalnoj razini svojim udjelom na tržištu operacijskih sustava, platforma Windows stvara veliki korak unazad sa prikazivanjem tipografije, a tehnologija za poboljšanje postoji - samo je neiskorištena.

Cilj diplomskog rada jest postaviti pravila u izradi ClearType pismovnog reza. Prikazati će se način nošenja sa tehnološkim nedostacima ekrana koji služe kao medij nosioc tipografije u multimedijskom prikazu. U ponudi različith osnovnih vrsta pismovnog reza odrediti će se koja vrsta tipografije je pogodnija

za prikazivanje na ekranima te koje linije na tipografskim znakovima najviše utječu na čitljivost.

Unatoč velikoj ponudi elektroničkih formata zapisa tipografije u ovom radu je odabran najefikasniji zapis čije će se prednosti iskoristiti u projektiranju pismovnog reza. U radu će se prilikom projektiranja pismovnog reza koristiti konvencionalan softverski alat za izradu pismovnih rezova te programski jezik Python u procesu optimiziacije. Prikazani koraci bit će daljnji vodič za pravilnu izradu pismovnog reza koji će na svim platformama biti jednoliko prikazan.

2. TEORIJSKI DIO

2.1. Tipografija

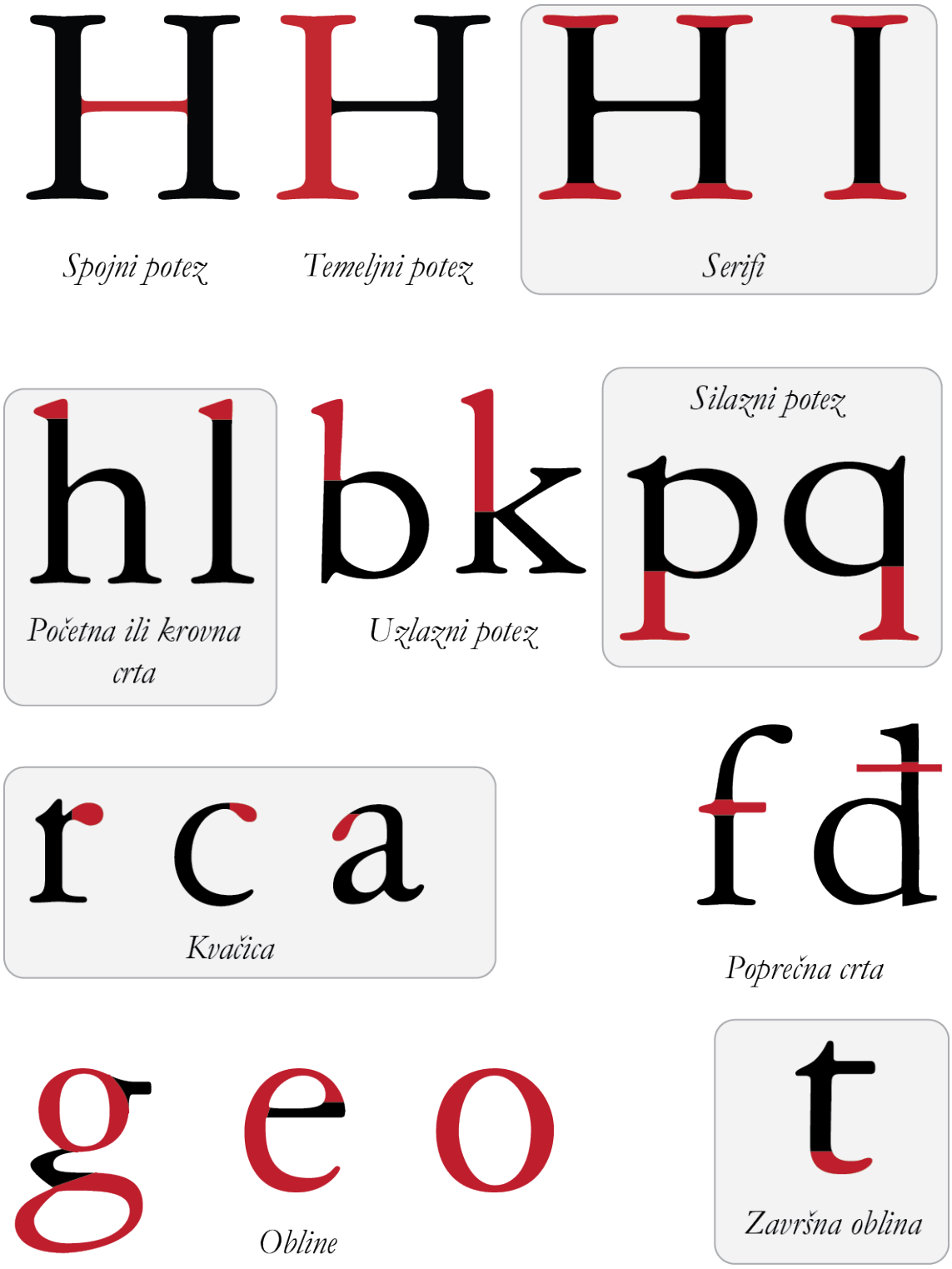
Tipografija je umjetnost koja objedinjuje oblikovanje slovnih znakova i komunikaciju. U različitim područjima i kulturama tipografija ima obilježja specifična samo za navedeno govorno područje. Cilj tipografije jest omogućiti čitljivu i tečnu komunikaciju temeljenu na nizu pravila koja su stvorena upravo radi olakšanja komunikacije. Tipografija se bavi oblicima, međusobnom odnosu slovnih znakova, rasporedu i veličini prostora na kojem se prostire informacija oblikovana nekim pismovnim rezom.

Veličina prostora na kojem se prostire tipografski oblikovan tekst određuje širinu retka i prema tome i tečnost čitanja. Jedno od pravila jest da širina retka ne bude šira od 55-60 znakova na optimalnoj udaljenosti oka promatrača. Duža širina retka jednostavno umara čitatelja pa čitatelj gubi liniju povezivanja između kraja jednog retka i početka novog retka. Prekratak redak isto nije dobar odabir, jer uništava ritam čitanja stalnim cjepkanjem i također umara čitatelja.

Veličina slovnih znakova u pismovnom rezu treba biti optimalno prilagođena namjeni komunikacije. Ukoliko je visina pismovnog reza previsoka, gubi se ritam povezivanja riječi jer premalo riječi stane u vidokrug čitatelja. Premalena visina pismovnog reda dovest će do presitnog teksta i preširokog retka.

Visina razmaka među retcima utječe na ritam čitatelja. Previsok razmak među retcima sječe čitanje i umara čitatelja. Premalen razmak među retcima stvara nered, jer čitatelj gubi orijentaciju i teško povezuje kraj reda sa početkom drugog.

Zadatak tipografije je oblikovanje tipografskih znakova koji kao cjelina čine pismovni rez. Oblik tipografskih slovnih znakova mijenjao se kroz stoljeća, ali nazivi i pravila za osnovne poteze kod tipografskih znakova ostali su do danas nepromijenjeni.



Slika 1 - Osnovni dijelovi slovnih znakova (Izvor: Franjo Mesaroš (1981). *Tipografsko oblikovanje*, Viša grafička škola u Zagrebu, Zagreb)

2.1.1. Klasifikacija pisama

U tipografiji prevladavaju pismovni rezovi različitih oblika i karaktera. Svoju različitost oblika tipografija dobiva evolucijskim razvojem kroz stoljeća. U svojim počecima tipografiju se izgledom često povezivalo sa kaligrafijom, ali kasnijim razvojem oblici su se svojim karakterom počeli odmicati od utjecaja kaligrafije [1]. Obzirom na vrijeme nastanka i oblikovanje tipografiju dijelimo u različite kategorije. Kategoriziranje tipografije nije striktno određeno, već razlikujemo više različitih podjela.

Linotype na svojim web stranicama ističe sljedeće kategorije [2]:

1. Pisma bez serifa
2. Pisma sa serifima
3. Rukopisna pisma kistom
4. Simbolna pisma
5. Pisma za čitanje
6. Arapska pisma
7. Pojednostavljena rukopisna pisma
8. Kaligrafska pisma
9. Gotička pisma

Popularna web stranica dafont.com ima specifičnu i opširnu podjelu. Pošto je podjela velika, ističu se osnovne kategorije [3]:

1. Luksuzna pisma
2. Strana pisma
3. Tehnološka pisma
4. Bitmapna pisma
5. Gotička pisma
6. Osnovna pisma
7. Rukopisna pisma
8. Simbolna pisma
9. Ukrasna pisma

FontShop web stranica ima sličnu podjelu kao i Linotype [4]. Gore navedene podjele spadaju u podjele „prema korisnicima“ koji kupuju pisma u odnosu na namjenu.

Jedna od ozbiljnijih podjela je ona hrvatskog tipografa Franje Mesaroša u njegovoj knjizi „Tipografsko oblikovanje“ koja će bit uzeta u obzir u diplomskom radu [5]:

1. Temeljni oblici
2. Tehnički oblici
3. Individualni oblici

2.1.2. Temeljni oblici

Temeljni oblici razvijali su od otprilike Gutenbergovog vremena pa sve do 19. stoljeća. Uglavnom ih je najlakše opisati kao serifna pisma, ali svojom evolucijom temeljni oblici dobivaju različit karakter tijekom vremena [1]. Podjela temeljnih oblika [5]:

1. Renesansna antikva
2. Prijelazna antikva
3. Klasicistička antikva

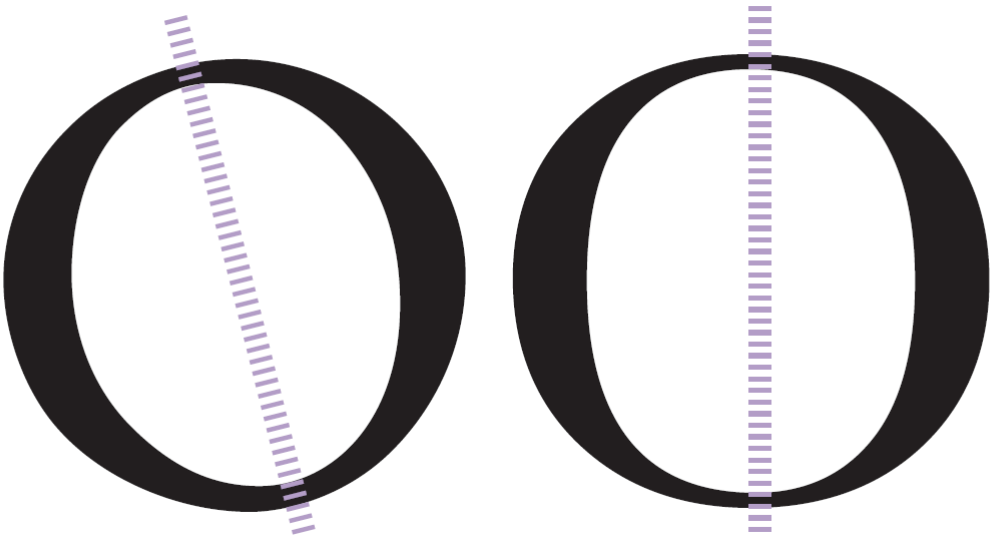
2.1.2.1. Renesansna antikva

Renesansna antikva odiše lijepim potezima i čitljivosti [1]. Pismovni rezovi koji pripadaju ovoj kategoriji jedni su od onih s kojima se susrećemo svakodnevno i prema njihovoj ljepoti uspoređuju se ostali pismovni rezovi [1].

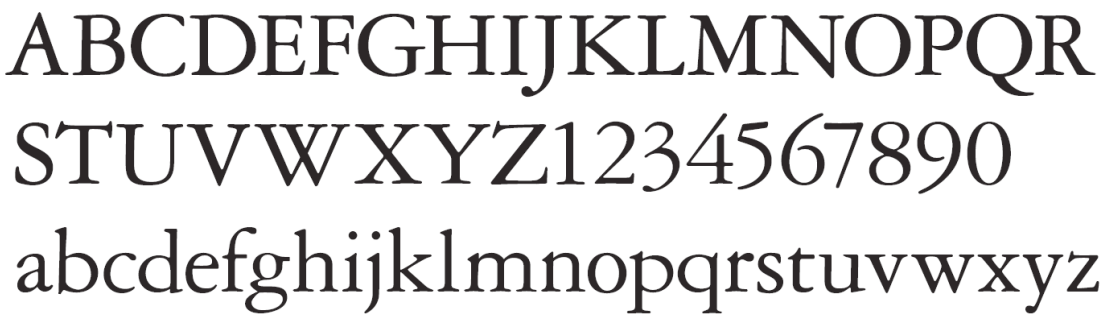
Potezi na tipografskim oblicima su nježni, senzualni i tanki, a koso izvedene obline imaju nagib [1]. Odaju dojam prostora – lijepi su, ugodni i elegantni.

Najpoznatiji i najljepši pismovni rez toga doba je Garamond. Osim svog zanimljivog izgleda, Garamond po prvi puta predstavlja pravi kurzivni oblik koji ne poprima rukopisni oblik [1]. Dotada su kurzivni oblici izgledali kao nakošeno rukopisno pismo, a stil koji je predstavljen Garamond pismom počinje se širiti u krugovima tipografa.

Garamond je i danas preveden u digitalni oblik i spada u najviše korištene pismovne rezove. Njegova primjena na ekranima, općenito renesansne antikve, je vrlo rijetka. Razlozi su niska visina kurentnih znakova i debljinski omjer od 1:1 - 1:4, što je vrlo teško prikazati.



Slika 2 - Usporedba starije nagibne osi (lijevo) i novije nagibne osi (desno)
(Izvor: David Bergsland (2010). *Practical Font Design*, Radiqx press, Mankato, MN, SAD)



Slika 3 - Pismovni rez Garamond
(Izvor: David Bergsland (2010). *Practical Font Design*, Radiqx press, Mankato, MN, SAD)

2.1.2.2. Prijelazna antikva

Prijelazna antikva nastavlja odmak od kaligrafije [1]. Tipografski oblici su izričito uspravni, okrugli oblici poprimaju ravnu vertikalnu os, uključuju više modulacije, manje i ravnije serife, i strogo horizontalno poravnanje [5]. Najpoznatiji predstavnik prijelaznog razdoblja jest pismovni rez Baskerville.

Slika 4 - Baskerville pismovni rez
(Izvor: David Bergsland (2010). *Practical Font Design*, Radiqx press, Mankato, MN, SAD)

2.1.2.3. Klasicistička antikva

Klasicistička antikva evolucijski nasljeđuje prijelaznu antikvu i njene karakteristike. Temeljni potezi su još uspravniji, deblji i stvara se veći optički kontrast u debljinskim vrijednostima. Obline ostaju okomite. Najpoznatiji predstavnici klasicističke antikve su pismovni rezovi Bodoni i Didot. Njihova primjena na ekranima iznimno je rijetka radi svoje specifične debljinske vrijednosti uzlaznih i spojnih poteza. Primjena klasicističke antikve više je okrenuta modernim tiskovinama (Vogue) i logotipovima (Armani, UBS...).

Slika 5 - Pismovni rez Didot (gore) (Izvor: http://helenjtaylordesign.files.wordpress.com/2011/10/screen-shot-2011-10-15-at-15-06-14.png) i Bodoni (dolje)
(Izvor: David Bergsland (2010). *Practical Font Design*, Radiqx press, Mankato, MN, SAD)

2.1.3. Tehnički oblici

Nakon stoljeća evolucije temeljnih oblika, javljaju se jednostavniji oblici pisama. Tehnički oblici gube serife te smanjuju debljinski kontrast između temeljnih i spojnih poteza na minimum [5].

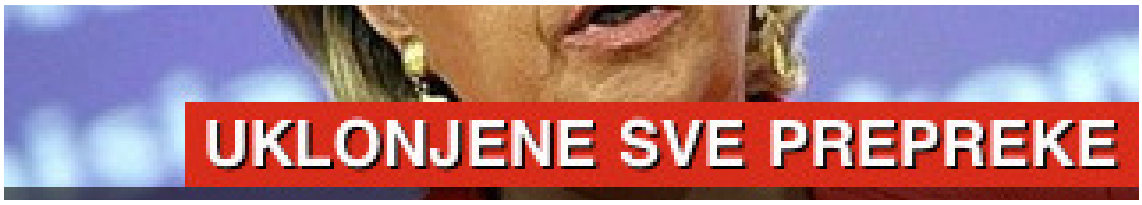
Mijenjali su u početku temeljne oblike u naslovima, a na plakatima su posebno mjesto zauzimali geometrijski oblici, koji su bili vizualno atraktivni, ali nisu bili čitljivi [1].

Tehničke oblike dijelimo u nekoliko kategorija [5]:

1. Grotesk
2. Egyptienne
3. Italienne

2.1.3.1. Grotesk

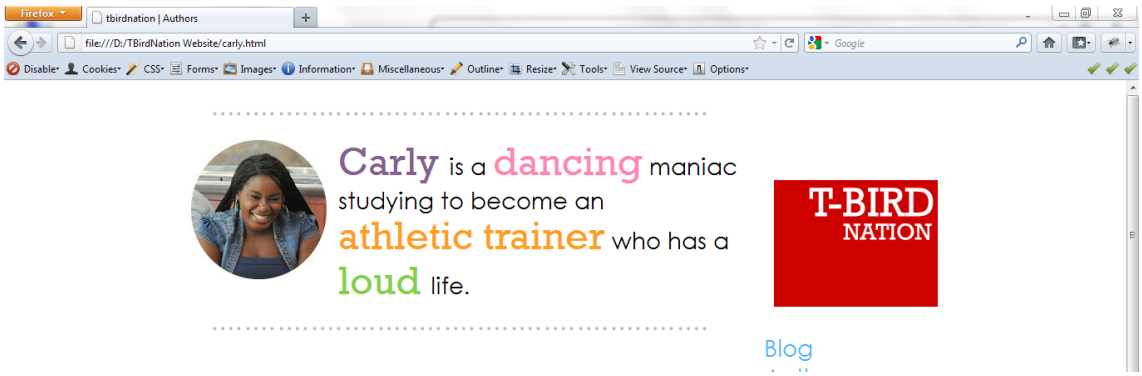
Groteskno pismo potpuna je suprotnost temeljnim oblicima. U odnosu na njih, groteskno pismo gubi serife i izjednačava debljinu temeljnih i prijelaznih poteza [5]. Groteskna pisma su radi toga manje čitka, ali vrlo stilizirana. Najpoznatiji predstavnici ove skupine su Futura, Gill Sans, Univers i Helvetica. Groteskna pisma najzastupljenija su u elektronskim medijima. Njihov oblik lako je iscrtati na ekranima, a pošto gotovo da i nema varijacije debljinske vrijednosti, njihova forma djeluje konstantno i čitko.



Slika 6 - Pismovni rez Helvetica
(Izvor: David Bergsland (2010). *Practical Font Design*, Radiqx press, Mankato, MN, SAD) i njegova primjena u mobilnoj iPhone aplikaciji 24 sata (Izvor: http://appstorehq-production.s3.amazonaws.com/24sata-hr-iphone-782064.320x460.1308825124.09963.jpg)

2.1.3.2. Egyptienne

Pismo egyptienne kombinacija je temeljnih i grotesknih oblika. Ono sadrži serife koji su jednakih debljinskih vrijednosti kao temeljni i spojni potezi. Poznati predstavnici ove skupine su: Rockwell, American Typewriter i drugi [5]. Njihova primjena u elektronskim medijima je u naslovima, dok u tekstu nemaju čestu primjenu radi debelih serifa koji se na manjim veličinama stope sa ostatkom forme slovnog znaka.



Geometric slabs: Rockwell

Slika 7 - Pismovni rez Rockwell
(Izvor: David Bergsland (2010). *Practical Font Design*, Radiqx press, Mankato, MN, SAD)
i njegova primjena na web stranici (Izvor: http://tbirdnation.files.wordpress.com/2011/10/website-4.png)

2.1.3.3. Italienne

Radi se o skupini tehničkih oblika koji poput egyptiennea imaju serife, ali su naglašeniji u odnosu na temeljne i spojne poteze [5]. Poznato pismo koje je predstavnik italienne skupine jest Playbill. Ova skupina pisama zbog svojih estetskih razloga nema čestu primjenu u elektronskim medijima.

PLAYBILL

Slika 8 - Pismovni rez Playbill

2.1.4. Individualni oblici

Individualni oblici nastaju u razdoblju poslije razvoja temeljnih oblika. Kombinacija su modificiranih antikva [5].

Podjela individualnih oblika [5]:

1. Umjetnička antikva
2. Polugrotesk
3. Novinska antikva

2.1.4.1. Umjetnička antikva

Umjetnička antikva ponovo uspostavlja odnos sa kaligrafijom, gdje se tipografski oblici diče slobodnim kaligrafskim potezima. Poznati predstavnik jest pismo Palatino koje predstavlja osvježenu renesansnu antikvu [5]. Primjena ove vrste pisama u elektronskim medijima je rijetka zbog specifičnosti koju dijeli sa temeljnim oblicima.

Slika 9 - Pismovni rez Palatino

2.1.4.2. Polugrotesk

Pismo koje se vremenski smješta na prijelaznom razdoblju između antikve i groteska. Pismo nema serife, nego su krajevi temeljnih poteza naglašeni većom debljinskom vrijednosti koja može varirati [5]. Predstavnik je pismovni rez Optima. Može se ustvrditi da je Optima jedinstvena po svojoj eleganciji i

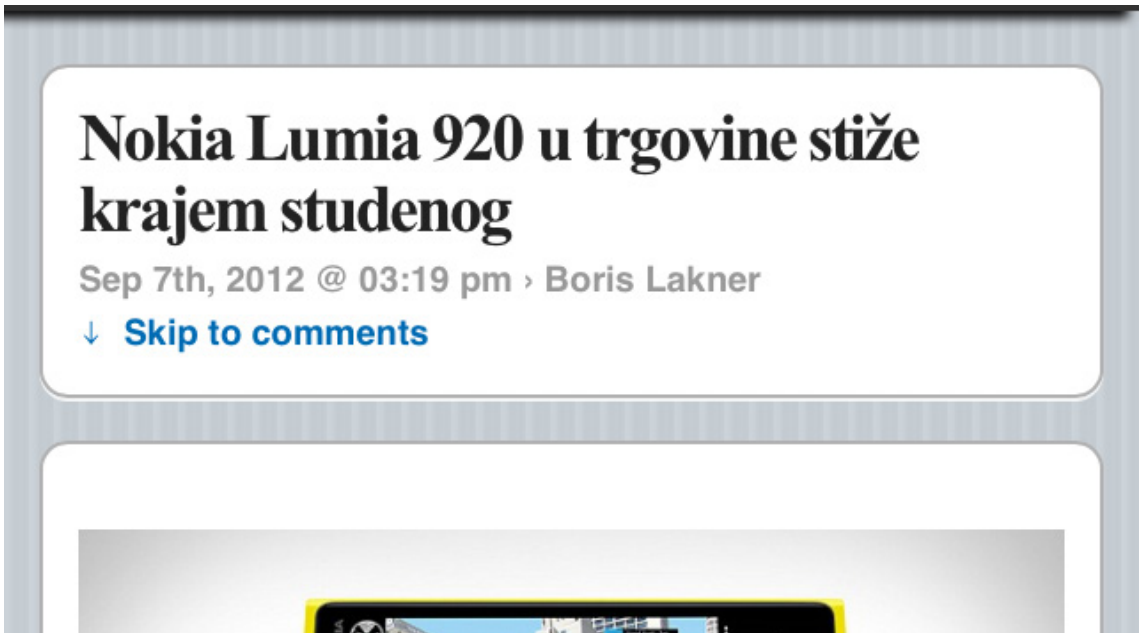
čitljivosti, a po svom obliku predstavlja antikve (debljinske vrijednosti) i groteska. Primjena Optime u elektronskim medijima gotovo je nemoguće naći - nije uvrštena u web pisma, a u elektronskim uređajima ne dolazi predinstalirana.

ABCDEFGHIJKLMNOPQ
RSTUVWXYZ1234567890
abcdefghijklmnopqrstuvwxyz

Slika 10 - Pismovni rez Optima
(Izvor: David Bergsland (2010). *Practical Font Design*, Radiqx press, Mankato, MN, SAD)

2.1.4.3. Novinska antikva

Skupina pisama napravljena u uzoru na temeljne oblike, ali prilagođena čitanju. Karakteristike su manja debljinska razlika između temeljnih i spojnih poteza te naglašeniji serifi radi jasnije izražene linije čitanja [5]. Jedan od glavnih predstavnika jest Times New Roman. Ovaj pismovni rez često je korišten u elektronskim medijima - uvršten u podržana web pisma, jer se nalazi na svim uređajima i operacijskim sustavima.



Slika 11 - Pismovni rez Times New Roman na mobilnoj verziji stranice Mob.hr (izvor: mob.hr/nokia-lumia-920-u-trgovine-stize-krajem-studenog/)

2.2. Računalne zapisne datoteke pisama

Veliki razvoj računarske tipografije obilježio je dolazak PostScript programskog jezika [6]. Sredinom 80-ih godina proteklog stoljeća velik problem bio je ispis dokumenata sa računala. PostScript kao grafički programski jezik dovodi prvu modernu računarsku tipografiju na računala. Do tada je pismovni rez činila skupina bitmapa, od kojih je svaka bitmapa predstavljala jedan slovni znak. Najveći problem sa bitmap tipografijom bio je u tome da se za svaku veličinu morao zasebno generirati skup bitmap datoteka, pa još za različite debljinske vrijednosti, što je dovodilo do potrebe za velikom memorijom pismovnih rezova. Uz to, grafička manipulacija tipografijom bila je ograničena i ozbiljnija grafička priprema na računalima nije bila moguća.

2.2.1. PostScript tipografija

Kao prvi odgovor bitmap tipografiji javlja se PostScript u obliku vektorske tipografije. PostScript je grafički programski jezik koji služi za baratanje i kreiranje grafike te opis stranice za ispis gdje je bilo moguće kombinirati piksel i vektorsku grafiku. Upravo vektorski aspekt PostScripta zaslužan je za novo poglavlje računarske tipografije. Svaki tipografski oblik napravljen unutar PostScripta sačinjen je od ravnih linija i Beziereovih krivulja. S PostScriptom je došlo do naglog razvoja i demokratizacije u izradi tipografije.

2.2.1.1. PostScript Font Type 3

Prva inačica PostScript tipografije nazvana je Type 3 Fonts [6]. Pismovni rez iz prve generacije bio je podosta ograničen. Pismovni rez bio je opisan na način da je u glavi imao kratki opis pismovnog reza, a zatim se nalazio opis tipografskih oblika, svakog zasebno. Svaki tipografski oblik u sebi je imao tri odjeljka koji su ga opisivali [6]:

1. Definirana širina prostora koji zauzima slovni znak
2. Širina i visina zasebnog koordinatnog prostora u kojem se iscrtava (četverac)
3. Točke linija i krivulja koje su tvorile oblik

Prvi problem bio je u tome što jedan tipografski oblik nije mogao ulaziti u dodijeljenu širinu prostora drugog oblika pa prema tome nije bilo definiranog razmaka između dva slovna znaka. Drugi problem predstavljala je nekonzistentnost vertikalnih debljinskih vrijednosti na svim tipografskim oblicima, stoga npr. vertikalni potezi na slovima H i P nisu imali jednake debljine u pikselima na ekranu. Ova nekonzistentnost događala se i pri ispisu dokumenata, a razlika je bila manje vidljiva na izlaznim uređajima visoke razlučivosti. Treći problem je deformacija tipografskih oblika, jer su svi verzalni oblici morali biti iste visine, kao što su i svi kurentni oblici morali za liniju koja je označavala osnovnu visinu kurentnih oblika [6].

Jedno globalno ograničenje za sve pismovne rezove bilo je to da su mogli sadržavati samo 256 znakova, za usporedbu danas pismovni rez koji je izrazito rasprostranjen kao Arial ima preko 10000 znakova. Osim toga, enkodiranje nije bilo riješeno i pismovni rez bio je zapisan u ASCII kodu [6].

2.2.1.2. PostScript Font Type 1

Iako u svom nazivu ima manji broj, radi se o novijoj, drugoj generaciji PostScript tipografije. I danas je vrlo popularan ovaj oblik zapisa, a prema nekim informacijama, radi se o vrlo vjerojatno najpopularnijem pismovnom zapisu [6].

U odnosu na prijašnji Type 3, nova inačica unosi sasvim različit pristup. Prva razlika je u zapisu, gdje su pismovni oblici mogli biti zapisani binarno i u ASCII kodu. Lansiranjem PDF datoteka i svojeg PDF čitača, Acrobat, Adobe je proradio na enkodiranju i indeksiranju polja znakova, kako bi omogućio pretragu unutar dokumenta. Indeksiranje dokumenata riješeno je dodijeljivanjem jedinstvenih imena slovnih znakova unutar tablice znakova.

U prvoj inačici svi verzalni znakovi i svi kurentni znakovi morali su imati iste visine. To je riješeno na način da je dodijeljena varijacija visine slovnih znakova. Na gornjoj granici definirano je maksimalno do 6 različitih varijacija, a na donjoj do maksimalno 5 različitih varijacija. Donešene su i mogućnosti varijacije debljinskih poteza isto u ograničenim količinama. Ovo nije riješilo problem, već ga je donekle prilagodilo pa je ovaj dio i u najdorečenijoj verziji PostScript tipografije ostao neriješen. Velika novost su sugestije rasterizacijskim pogonima u operacijskim sustavima. Sugestije u tipografiji su informacije koje posjeduje svaki znak, a pomažu rasterizaciji i optimizaciji iscrtavanja, odnosno prikazivanja oblika (linija i krivulja).

2.2.2. TrueType, OpenType i Apple Advanced Typography

TrueType tehnologija napravljena je zajedničkom suradnjom Apple-a i Microsofta, jer ih tadašnja PostScript tehnologija nije zadovoljavala [6]. Nakon izdavanja prve inačice svaka TrueType tehnologije svaki proizvođač krenuo je svojim putem u izradi novih rješenja za ekranski prikaz.

TrueType tipografija zapisana u binarnom kodu, za razliku od PostScripta gdje je dominirao ASCII kod. U odnosu na PostScript, TrueType tipografija temelji se većem broju tablica kod koji svaka tablica sadrži posebnu informaciju o pismovnom rezu i znakovima. Prilikom dizajniranja tablica od početka je riješen problem enkodiranja znakova, što je ovoj tehnologiji bila velika prednost u odnosu na Adobeovu. Uglavnom bili su riješeni svi veći problemi koji su mučili PostScript tipografiju - od varijacija debljine debljinskih poteza, visine i širine znaka, do podrezivanja koje prije nije uopće postojalo [6].

TrueType pismovni rez nije ovisan o definiranom rasterizacijskom pogonu operacijskog sustava, već svaki operacijski sustav može svojim metodama iscrtavati tipografske oblike.

Razvoj TrueType tehnologije omogućio je razvoj OpenType standarda (koji je razvijao Microsoft zajedno sa Adobeom) i AAT standarda (Apple Advanced Typography - razvijen od Applea). Novo razvijeni standardi su u svojoj osnovi TrueType standardi sa proširenjima u vidu novih tablica koje donose nove i različite mogućnosti [6].

2.3. Prikaz računarske tipografije na ekranu

U ovom radu rješava se prikaz tipografije na ekranu, s obzirom na izuzetno važnu ulogu u prijenosu informacija koju oni danas imaju. Za razliku od papira na kojem je tipografija nosioc informacija već stoljećima, na ekranima dolazi do deformacije izvornog oblika pismovnog reza radi ograničenja kao što su razlučivost i različitost metoda optimizacije.

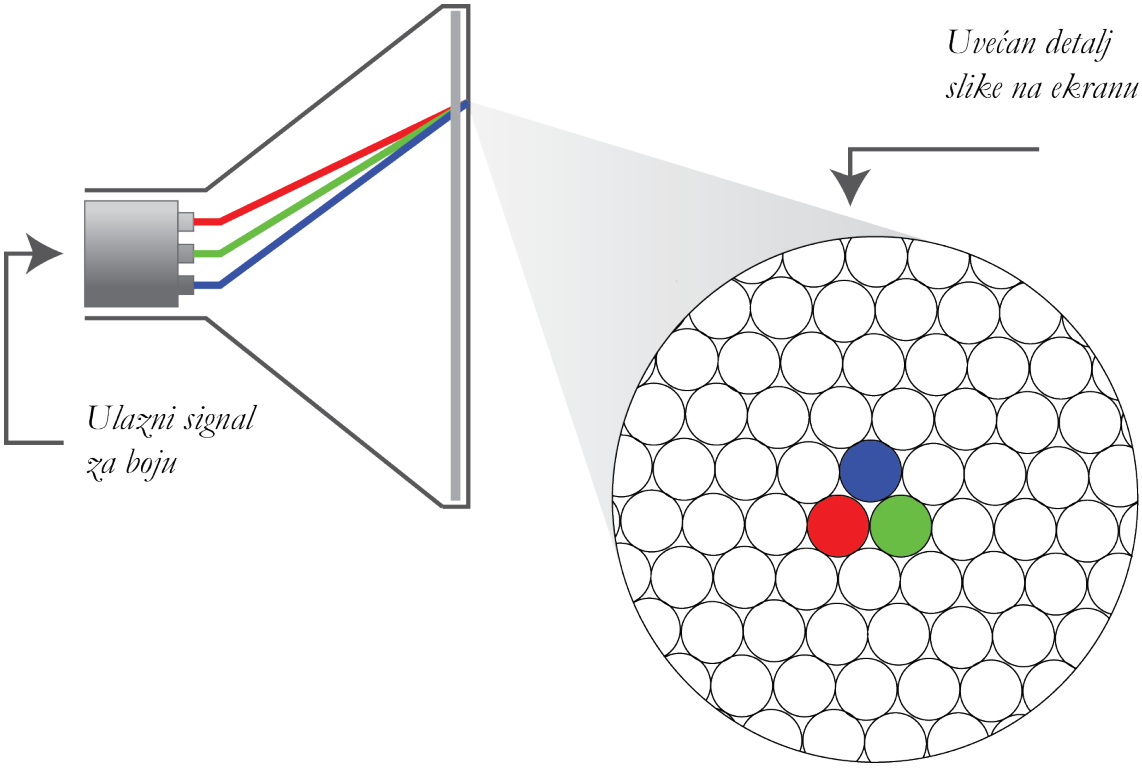
Glavni razlog što tipografija na ekranima ne poprima svoj idealni oblik jest razlučivost ekrana. Razlučivost se može najbolje opisati kao mrežu koja sadrži određen broj redaka i stupaca. Unutar polja koja nastaju križanjima redaka i stupaca smještaju se pikseli – po jedan u svako polje. Usporedbe radi, razlučivost ispisa puno je veća od razlučivosti ekrana, stoga se još uvijek ne može usporediti estetska kvaliteta izvedbe tipografije na ta dva medija.

Tijekom godina razvijanja i ekrani su evoluirali. Danas poznajemo tri glavne tehnologije prikaza ekrana: CRT, LCD i e-Ink. Sa stajališta tipografije najvažnije je kakve karakteristike imaju čelije ekrana. Postoje čelije dvije vrste čelija:

1. Čelije kod koji se ne može upravljati podpikselima
2. Čelije kod kojih se može upravljati podpikselima.

2.3.1.CRT tehnologija

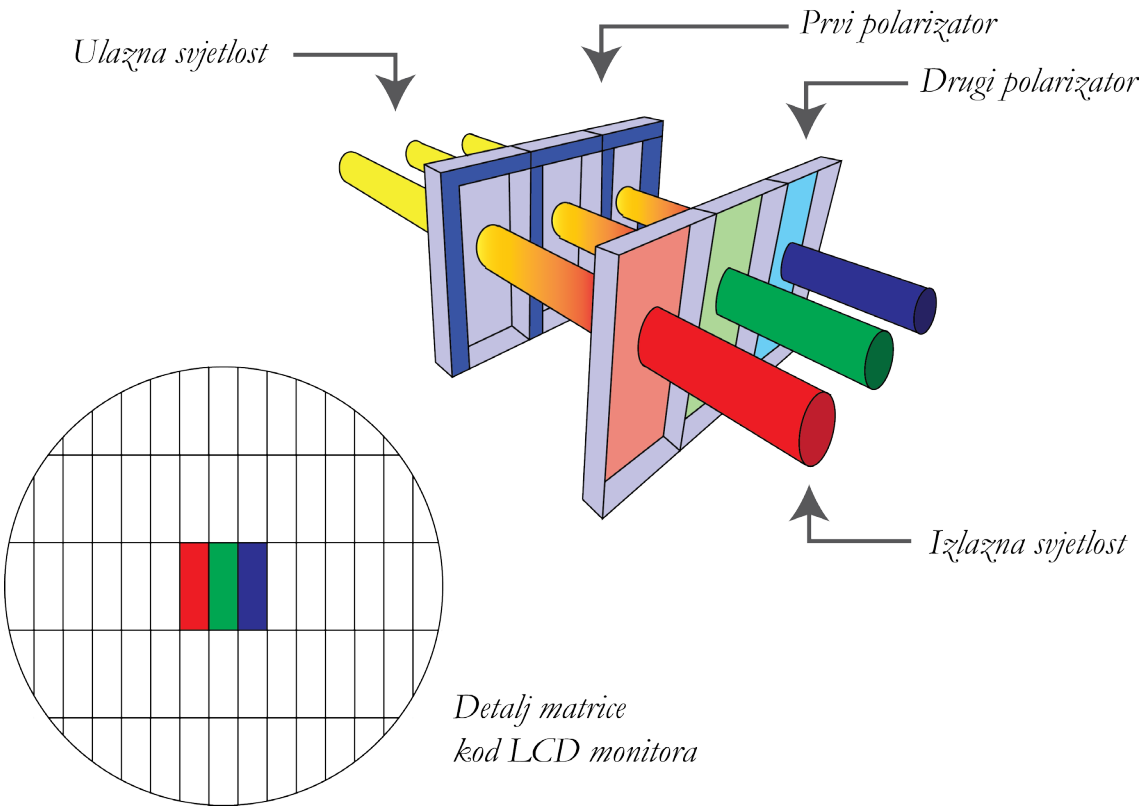
Ekrani koji koriste CRT tehnologiju najstariji su oblik ekranske tehnologije na računalima, koju su preuzeli od televizora. Ispod ekrana nalazi se sloj fosfornih čestica koji su osjetljivi na elektrone. Elektrone „ispaljuje“ katoda prema anodi i taj smjer elektronskih čestica daje sliku na ekranima. Piksel se stvara od tri točkice na matrici koje tvore „trokut“. Pošto se boja stvara pomoću crvenih, zelenih i plavih impulsa, moguće je dobiti i 127 nijansi sivih tonova. Nije moguće selektivno gašenje podpiksela jer svjetlo dolazi iz jednog izvora (katode).



Slika 12 - CRT tehnologija
(Izvor: http://4.bp.blogspot.com/-rcLCrJ5tbRo/TW1ZldxbCII/AAAAAAAAGc4/DrtVLvf8fAI/s1600/crt.gif)

2.3.2. LCD tehnologija

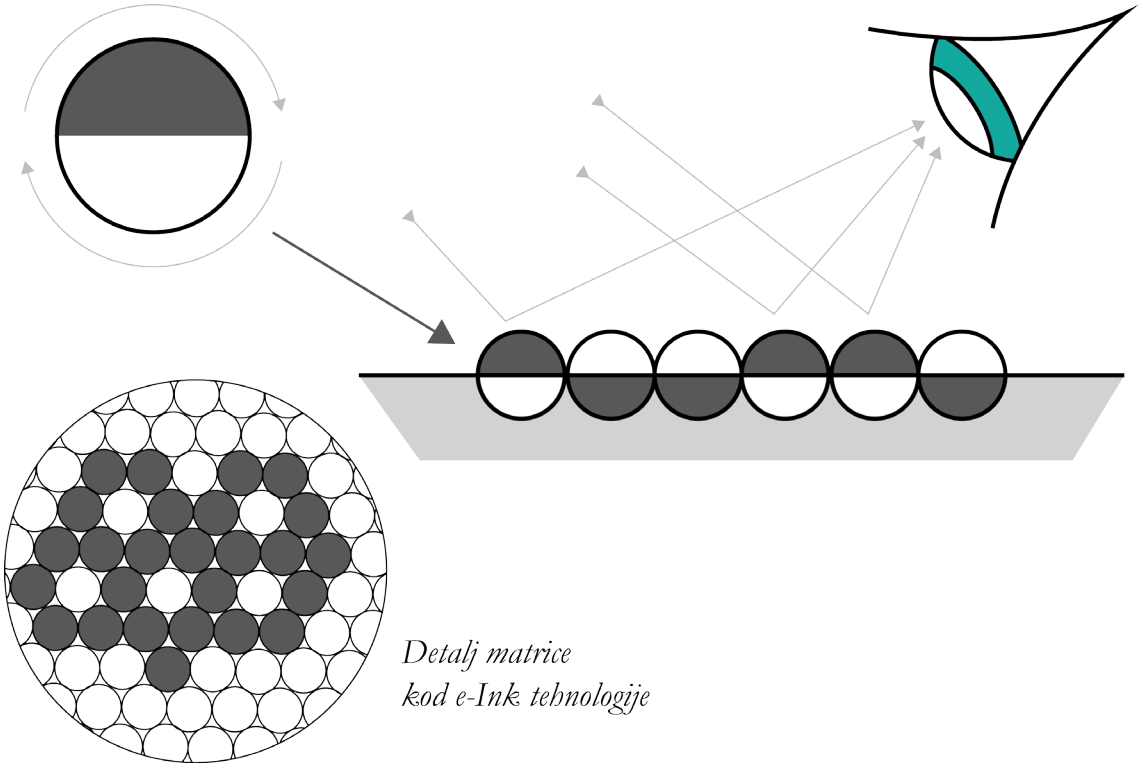
Za razliku od CRT tehnologije gdje na zaslon dolazi svjetlost iz jednog izvora, kod LCD tehnologije svaki piksel predstavlja jedan izvor svjetlosti i ponaša se individualno u odnosu na druge piksele. Čelija jednog piksela sastoji se od tri podpiksela: crvenog, zelenog i plavog. Svaki od tih podpiksela moguće je individualno isključiti i uključiti. Stoga cijela čelija piksela ne poprima jednu boju, već se boja selektivno nalazi u odjelima/podpikselima.



Slika 13 - Prikaz LCD tehnologije i matrice
(Izvor: http://www.hk-phy.org/energy/commercial/office_phy/images/display_unit.jpg)

2.3.3. E-Ink tehnologija

Ova tehnologija njeguje drugačiji princip. Za razliku od CRT i LCD tehnologije koje sliku stvaraju na principu izvora svjetlosti, ova tehnologija radi na principu pasivnog izvora svjetlosti. Izvor svjetla (sunce, UV lampa, ...) u nekoj prostoriji reflektira se od površine e-Ink ekrana i na taj način se dobiva slika u oku promatrača – svojstvo svake materije jest da reflektira svjetlo dobiveno od izvora u određenoj količini. Na taj način dobivaju se svojstva papira, što rezultira prirodnijim ugođajem za čitanje. Cilj ove tehnologije i jest zamijeniti papir u ekološkom smislu. Tehnologija koja pomaže stvoriti sliku radi tako da se iza površine ekrana nalazi skup okruglih čestica čije su polovice obojene u različitim bojama – jedna crna, druga bijela. Bijela boja okrenuta je prema promatraču ukoliko ne prikazuje sadržaj, dok je crna okrenuta ukoliko zahvaćena čestica prikazuje neki sadržaj. Prve verzije e-Ink ekrana prikazivale su isključivo crno-bijelu sliku, bez sivih tonova, ali novije generacije pomoću polarizacijskih filtera na površini ekrana mogu prikazati boje iz cijelog vidljivog spektra. Jedna točkica predstavlja „piksel“ i prema tome ona preuzima cijelom svojom površinom jedan ton boje.



Slika 14 - Prikaz e-Ink tehnologije

2.4. Tehnologije iscrtavanja tipografije na ekranima i optimizacija

Već dugo vremena stvara se kompromis u borbi sa ograničenim mogućnostima ekrana na kojima se prikazuje tipografija. Prva *bitmapna* tipografija nije imala problem sa prikazom koji ima današnja linijska tipografija – napravljena je bila određena veličina pismovnog reza koja se upotrebljava i napravljeno je točno toliko slika (*bitmapa*) koliko je bilo slovnih znakova i veličina u kojima se nudio pismovni rez. Kvaliteta pismovnog reza je bila odmjerena točno za te veličine i nije pogodovala nekim rasterizacijskim procesima operacijskog sustava – izgled tipografije izgledao je jednako na svim platformama.

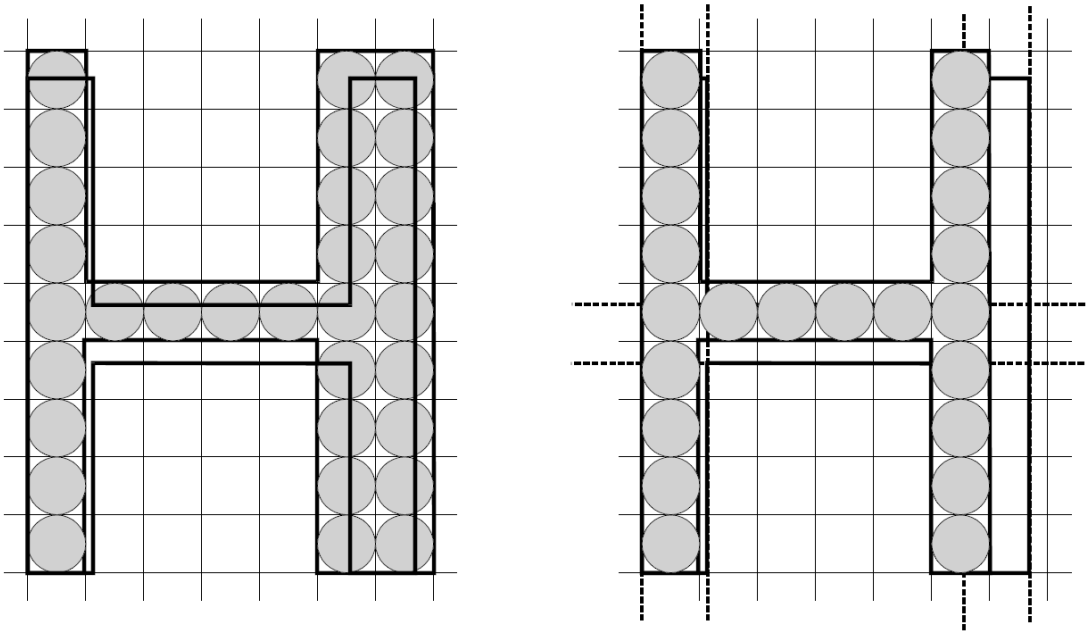
Pojavom linijske tipografije dolazi uglavnom do velikih problema – linijska tipografije jest skalabilna, ali ona nikada ne naliježe na mrežu piksela kako bi trebala. Tadašnje vrijeme CRT tehnologije je još dodatno ograničavalo prikaz jer nije bilo podpiksela. Prvi PostScript Type 3 pismovni rezovi prikazivali su se na ekranima vrlo loše – debljinski potezi varirali su, nije postojalo podrezivanje u slovnih znakovima i skalabilnost nije valjala.

Veliku evoluciju donijeli su setovi informacija koji su pomagali u optimiziranju izgleda tipografije na ekranima. Zapisu pismovnog reza dodana je informacija na razini slovnih znakova kako se koji slovni znak prikazuje na određenoj gustoći piksela.

2.4.1. PostScript Type 1 optimizacija

Kako bi se ispravio loš prikaz kao kod Type 3 verzije, Type 1 donosi mogućnost programiranja seta informacija koje pomažu pri prikazu tipografije.

Type 1 optimizacija donosi set informacija o svakom pojedinom znaku da bi se što ispravnije prikazao. Informacije sadrže koordinate točaka na potezima i linijama te debljinske vrijednosti horizontalnih i vertikalnih poteza kako bi se pri povećavanju izgubilo što manje informacija [7].



Slika 15 - Prikaz Type 1 optimizacije za mrežu ekrana. Slika lijevo prikazuje slovo H bez optimizacije, a desna sa Type 1 optimizacijom
(Izvor: FontLab Ltd. (2006). *FontLab Studio 5 - User's Manual for Windows*, FontLab Ltd.)

U svoje vrijeme negativna strana Type 1 optimizacije bila je ta što se ona oslanjala na optimizacijske mogućnosti operacijskog sustava. U to vrijeme operacijski sustavi radili su sa bitmap tipografijom i nisu imali u sebi rasterizacijsku logiku za linijsku tipografiju.

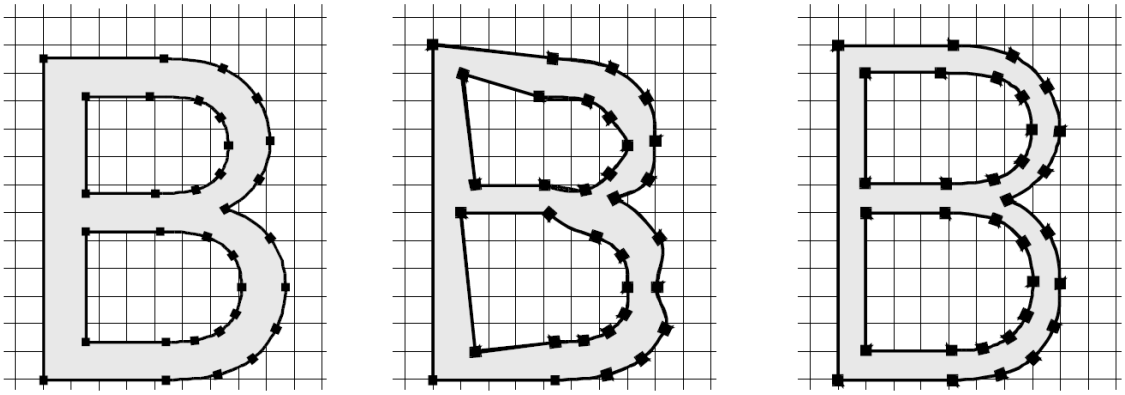
Tek kasnije, u 21. stoljeću, prednosti ovakvog pristupa nalaze prednosti, jer operacijski sustavi sada imaju sposobne rasterizacijske operatore koji mogu upravljati skalabilnošću.

2.4.2. TrueType optimizacija

Pošto princip Type 1 tipografije nije odgovarao tadašnjoj tehnologiji, svojim dolaskom TrueType tehnologija donijela je revoluciju u prikazu na ekranima. Njezin mehanizam odnosi se prema linijskoj fotografiji kao prema bitmap zapisu – svaki individualni piksel može se kontrolirati, tj. može se isključiti/uključiti prema potrebi [7]. Na tadašnjim ekranima TrueType pismovni rez imao je veliku prednost u čitljivosti u odnosu na Type 1 rješenja.

Osnova TrueType mehanizma za optimizaciju iscrtavanja tipografije jest TrueType programski jezik u kojem su se pisale naredbe kako iscrtavati znakove na pojedinim gustoćama razlučivosti [7]. U osnovi operacijski sustav

je samo trebao imati interpreter koji će izvršavati redoslijedom TrueType instrukcije. Instrukcije za optimiziranje iscrtavanja izvodile su se u odnosu na mogućnosti TrueType interpretera, ali ukoliko je u kodu napisana funkcija koju interpreter ne sadrži, ona nije kočila iscrtavanje pismovnog reza.

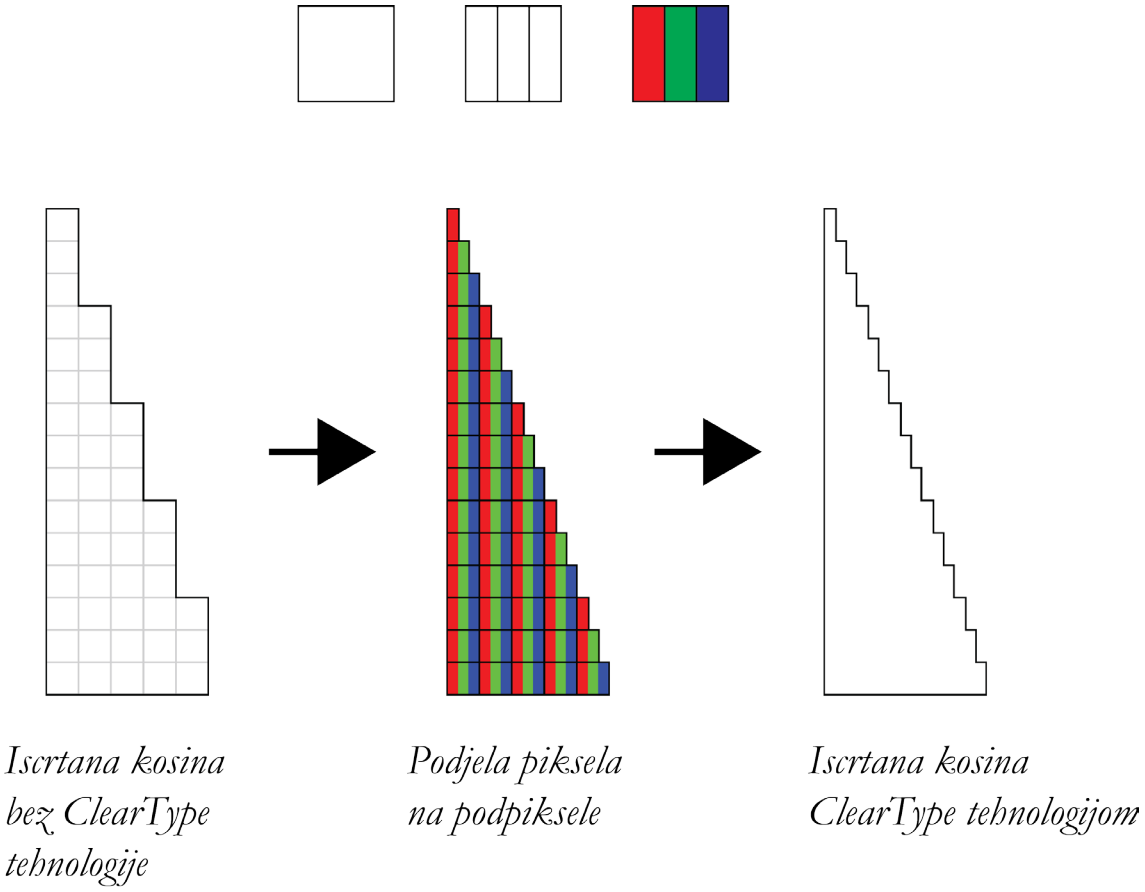


Slika 16 - TrueType optimizacija. Prva slika prikazuje slovo B koje se nalazi umreži piksela i ne nasjeda točno. Druga slika prikazuje tzv. delta 1 točke koje se povlače ručno na mrežu piksela. Treća slika prikazuje delta 2 algoritam koji vrši tranziciju točaka
(Izvor: FontLab Ltd. (2006). *FontLab Studio 5 - User's Manual for Windows*, FontLab Ltd.)

2.4.3. ClearType optimizacija

Probitkom LCD tehnologije stvoren je niz novih tehnika zaglađivanja tipografskih oblika. Zahvaljujući tome što je piksel u LCD tehnologiji podijeljen na tri podpiksela omogućeno je još kvalitetnije zaglađivanje rubova slovnih znakova.

ClearType pojavio se na izmaku snaga TrueType tehnologije. Do tada evoluirana TrueType tehnologija dobila je mogućnost Anti-aliasinga u osam različitih sivih tonova – radilo se o poboljšanjima, ali TrueType interpreter nije mogao ulaziti u podpiksele i tu je razvoj TrueType interpretera zastao. U odnosu na prijašnje interpretere, ClearType radi sa svim vrstama tipografskih datoteka (PostScript, TrueType i OpenType). Iako su TrueType pismovni rezovi i dalje imali dobru kvalitetu iscrtavanja na ekranima, Delta funkcije u TrueType programskom jeziku rade na principu piksela, a ne podpiksela pa ClearType tehnologija ignorira postojeće Delta instrukcije. TrueType formati iscrtavaju se djelomično precizno kako su zamišljeni [7].



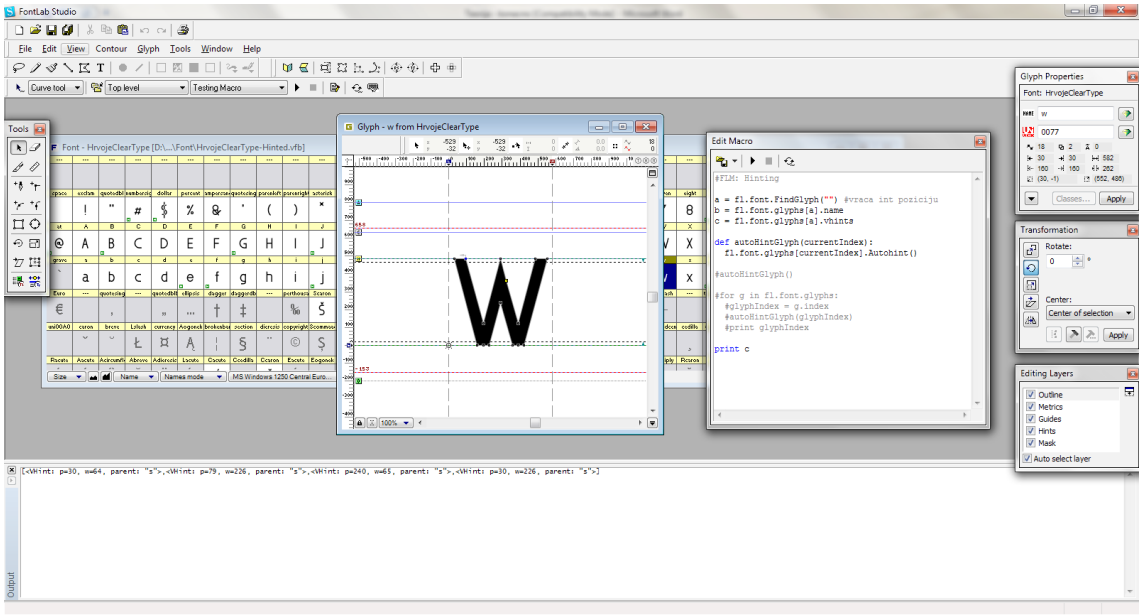
Slika 17 - Prikazuje način na koji ClearType tehnologija upravlja podpikselima (Izvor: http://www.grc.com/ctwhat.htm)

Za razliku od Type 1 tehnologije, optimiziranje za TrueType radilo se ručno i iznova za svaku novu veličinu. U slučaju promjene mogućnosti interpretera, trebalo je opet iznova uzeti izvorni kod i modificirati ga ukoliko je autor želio iskoristiti nove prednosti TrueType interpretera. Za Type 1 tehnologiju to nije bilo potrebno, samo su se definirale točke za povećavanje, a cijeli posao preuzima rasterizacijski set instrukcija u operacijskokm sustavu. U konačnici TrueType je uzimao puno vremena tipografima i programerima, a njegove glavne prednosti – Delta funkcije nisu radile u ClearType standardu. Ionako teško programiranje za optimiziranje prikaza pismovnog reza u TrueType tehnologiji više nije bilo dovoljno za napredak. ClearType tehnologija najednom daje do tada zanemarenom standardu za ekrane (barem po pitanju kvalitete iscrtavanja), PostScriptovom Type 1, priliku da se približi TrueType tehnologiji.

2.5. Alati za izradu diplomskog rada

2.5.1. FontLab Studio 5

Najrazvijeniji softverski alat za izradu pismovnog reza je FontLab Studio. Tvrtka koja je autor FontLab Studio softvera kupila je konkurentnu tvrtku i njen proizvod Fontographer. U usporedbi sa popularnim Fontographerom, FontLab Studio puno je kompleksniji i detaljniji, stoga nailazi na široku primjenu u ozbiljnim tvrtkama koje se bave tipografijom. Osim crtanja, omogućeno je i optimiziranje izgleda pismovnog reza za prikaz na ekranima. Ono je omogućeno ručno ili programski – u dva odjeljka: TrueType optimizacija i Type 1 optimizacija. Izvoz uratka moguće je u vrlo velikoj paleti izlaznih formata.



Slika 18 -Prikaz sučelja FontLab Studio softvera

Najveća prednost FontLab softverskog alata u odnosu na ostale alate iste kategorije jest njegova podrška za Python programski jezik. Pomoću Python programskog jezika mogu se proširiti mogućnosti FontLaba, automatizirati i ubrzati proces izrade pismovnog reza.

Slovni znakovi izrađuju se u četvercu. Četverac je prostor dodijeljen svim slovnim znakovima unutar pismovnog reza. Upravljanje četvercem moguće je i kroz Python skripte.

2.5.2. Python programski jezik

Python je objektno-orijentirani programski jezik, koji se svrstava u rang sa C baziranim programskim jezicima. Njegova sintaksa je potpuno drugačija od drugih programskih jezika. Usporedbe radi, na kraju svake linije koda koja se izvršava ne stavlja se točka-zarez, već prijelazom u novi red odvajaju se naredbe. Blokovi koda ne izoliraju se klasičnim vitičastim zagradama, nego uvlakom koja mora biti ujednačena kroz cijelu datoteku u kojoj se nalazi Python skripta.

Python može usko biti povezan s bazama podataka, a u FontLabu logičku bazu podataka predstavlja pismovni rez sa slovnim znakovima. Python je tu da proširi mogućnosti FontLab softvera - pisanje i izrada novih alata, automatizacija repetitivnih poteza i način za izbjegavanje navigacije kroz sučelje. Sve opcije koje postoje u FontLab sučelju mogu se dohvatiti putem Python programskog jezika. Python skripte pisane u FontLab softveru nazivaju se još i Macro programi [7].

```
def logistic_map(x,r):
    return r*x*(1-x)

# time axis is 1 second sampled at 10KHz
t = N.linspace(0,1,10000,endpoint=0)
# driving force
dforce = N.sin(10*N.pi*t) + N.sin(22*N.pi*t) + N.sin(26*N.pi*t)

# resulting time series
series = N.zeros((10000,1),'d')
series[0] = 0.6 # initial condition
for i in range(1,10000):
    series[i] = logistic_map(series[i-1],3.6+0.13*dforce[i])

# define the flow
sequence = [mdp.nodes.EtaComputerNode(), mdp.nodes.TimeFramesNode(10),
            mdp.nodes.PolynomialExpansionNode(3), mdp.nodes.SFANode(output_dim=1),
            mdp.nodes.EtaComputerNode()]

flow = mdp.Flow(sequence, verbose=1)
# train the flow
flow.train(series)

# execute the flow to get the SFA estimate of the driving force
slow = flow.execute(series)

# rescale driving force to compare with SFA estimate
resc_dforce = (dforce - N.mean(dforce,0))/N.std(dforce,0)
```

Slika 19 - Primjer Python programskog jezika
(Izvor: http://c431376.r76.cf2.rackcdn.com/338/fninf-02-008/image_m/fninf-02-008-g004.jpg)

3. EKSPERIMENTALNI DIO

3.1. Izrada pismovnog reza

Pošto se diplomski rad temelji na izradi vlastitog pismovnog reza namijenjenog prikazu na ekranu, važno je odabrati vrstu pisma koja će bit čitljiva na ekranu i na manjim razlučivostima. Odabir je izrada tipografije bez serifa jer se serifi teže prikazuju na manjim razlučivostima. Kad je i moguće prikazati oblik serifa, njegov oblik često narušava formu i oblik slovnog znaka. Kod serifnih pisama problem je i varijacija debljinskih poteza, čiji se blagi prijelaz iz tanjeg u deblji potez i obratno ne može ispravno prikazati na ekranima sa nižim razlučivostima.

3.1.1. Odabir formata zapisa pismovnog reza

Odabir formata zapisa najosnovniji je i najvažniji korak. O njemu ovisi podrška za različite platforme, vrsta enkodiranja i način optimizacije za prikazivanje na ekranima. U ovom radu izabran je format OpenType. OpenType je raširen format koji podržavaju sve platforme. Unutar tog formata izrada će se temeljiti na PostScript i TrueType krivuljama radi usporedbe i analize u kvaliteti iscrtavanja.

Zbog OpenType formata moguće je uključiti i veći broj slovnih znakova radi pokrivanja više govornih područja. Bez obzira na vrstu krivulja koje oblikuju slovne znakove u OpenType formatu, moguće je obuhvatiti i hrvatske slovne znakove. Pismovni rez prvo će biti napravljen sa PostScript krivuljama, a zatim sa TrueType krivuljama transformacijom iz PostScript krivulja.

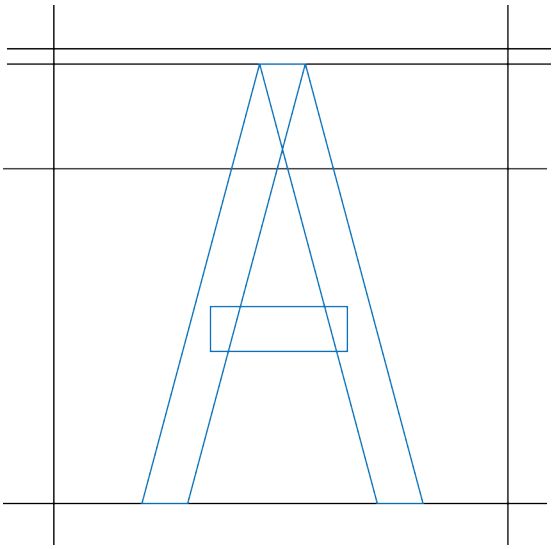
3.1.2. Dizajn pismovnog reza

Kako je za pismovni rez odabrana tipografija bez serifa, potrebno je udovoljiti još nekim zahtjevima ekrana. Velik broj pismovnih rezova dizajnirano je u tiskarske/ispisne svrhe, a onda se eventualno pomoću optimiziranja pokušava povećati kvaliteta prikaza na ekranu. Za potrebe diplomskog rada dizajniranje ide obrnutim procesom – prvo namjena ekranima, zatim namjena ispisu. Svrha je stvoriti lijepo, moderno pismo sa dodatnom vrijednosti optimizacije. Uz nove zahtjeve dolazi i do nove estetike tipografije za prikaz na

ekranima. Naravno uz uvjet da je izuzetno čitljiva. Kako bi se povećala čitljivost na ekranima, potrebno je povisiti visinu kurentnih slova. Ovo znači da će kurent na manjim veličinama biti većih dimenzija od dimenzije kurenata kod normalnih pisama.

Slika 20 - Prikaz pismovnog reza sa nižom visinom kurenata (Arial, lijevo) i višom visinom kurenata (Verdana, desno)

Slijedit će se princip pismovnog reza Verdana koji je još davne 1996 godine bio dizajniran prikazivanju na ekranima, a i danas slovi za jedan od najboljih pismovnih rezova pogodnih čitanju na ekranima. Verdana se odlikuje visokim i širokim oblicima, a posebno se ističu visoki i široki kurentni oblici u odnosu na verzale, koji svojom čitljivošću nadilaze popularne pismovne rezove kao što su Helvetica, Arial, Gill Sans, Franklin Gothic i drugi.



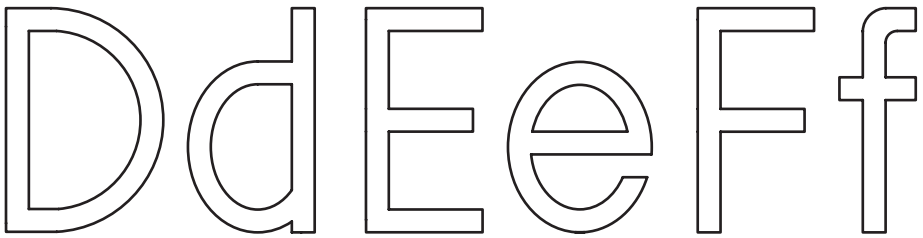
Slika 21 - Prikaz definiranog četverca u Illustratoru

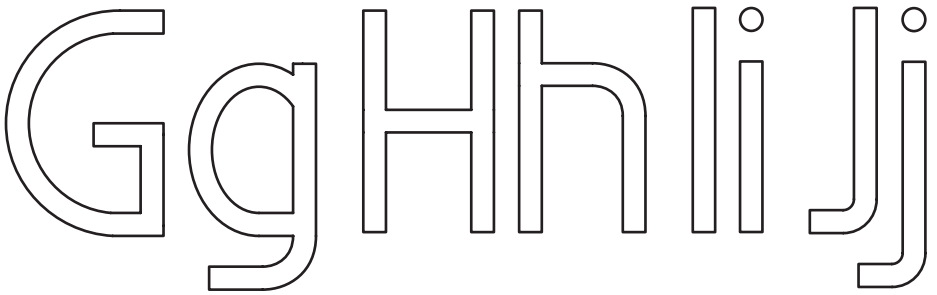
Kod svih pisama koji se prikazuju na ekranu uočava se devijacija debljinskih vrijednosti. Tanki potezi se posebno teško prikazuju na ekranima, pa prijelazi iz debljih u tanje poteze i obratno izgledaju poprilično loše. U diplomskom radu cilj je izbjeći te stvari pa pismovni rez ne smije sadržavati varijacije debljinskih poteza – svi debljinski potezi, uključujući vertikalne, horizontalne i oble poteze imaju istu debljinu radi što boljeg prikaza na ekranima.

Mreža i slovni znakovi dizajnirani su u softveru Adobe Illustrator radi većih i lakših mogućnosti vektorskog crtanja. Oblici su crtani alatom Pen Tool, bez ispune, i zatim prebačeni u FontLab Studio 5 softver.

3.1.3. Slovni znakovi

Pismovni rez sadrži i hrvatske slovne znakove, a oni su postavljeni na kraju jer su varijacija slovnih znakova c, s, z i d.

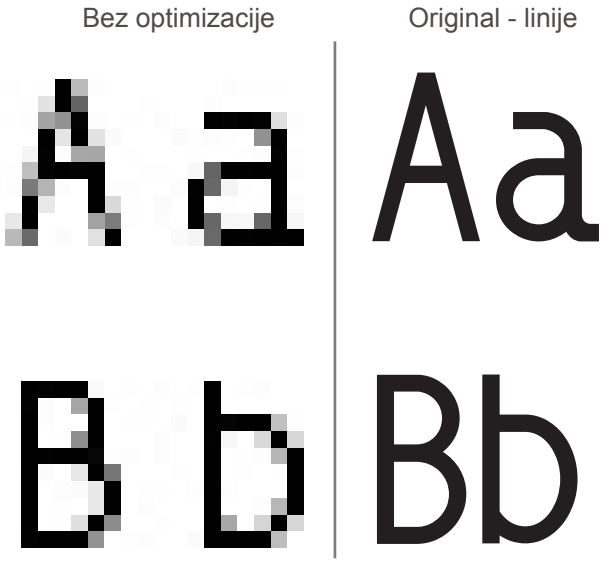




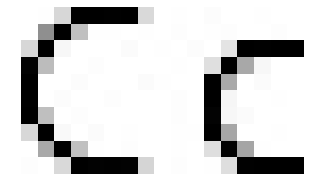

Kk Ll Mm
Nn Oo Pp
Qq Rr Ss
Tt Uu Vv

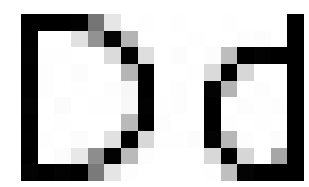
3.1.4. Pismovni rez bez optimizacije

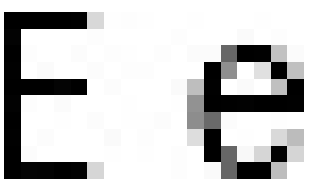
Sljedeći primjeri pokazuju uvećano kako izgleda pismovni rez u Type 1 tehnologiji bez optimizacije za prikaz na ekranu.

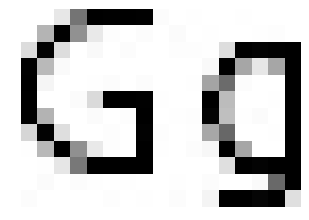


Bez optimizacije





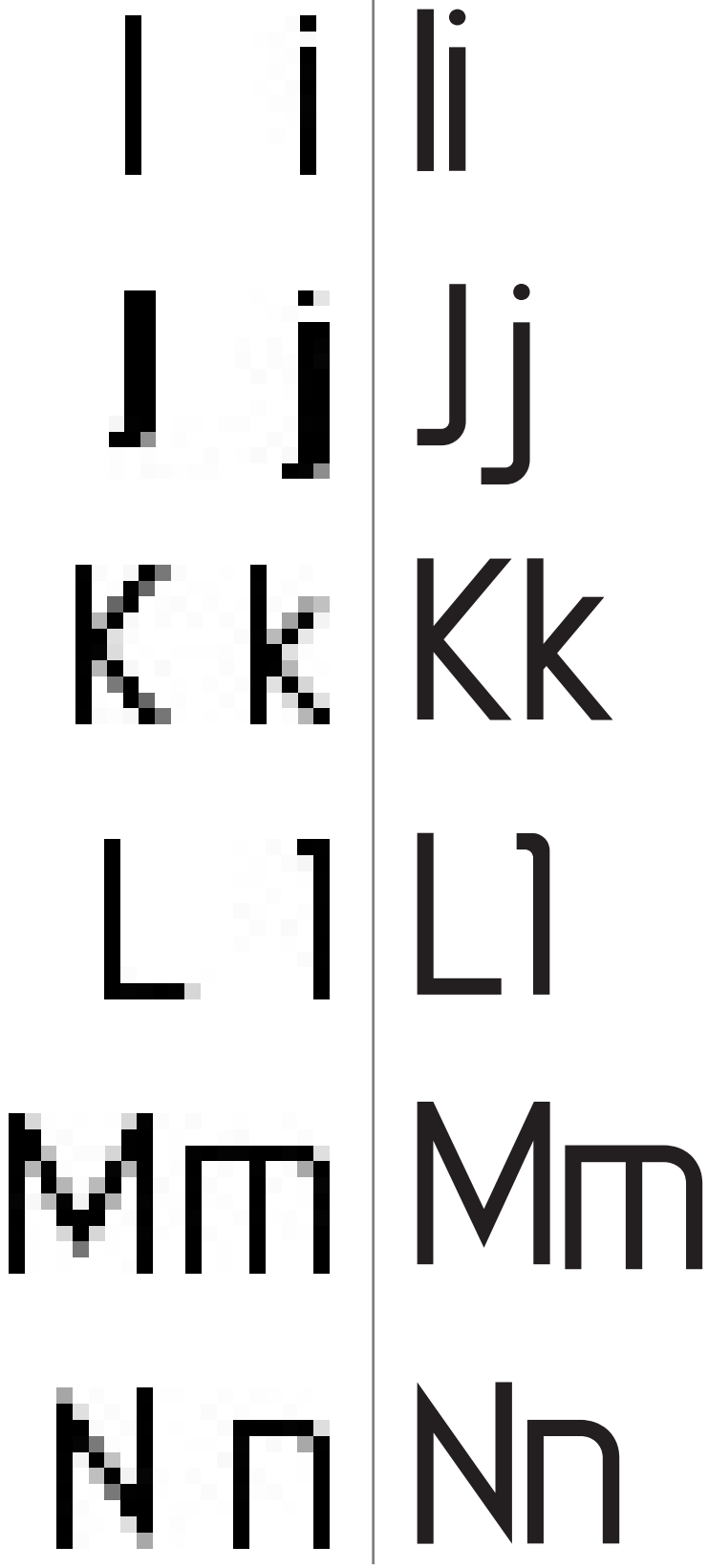




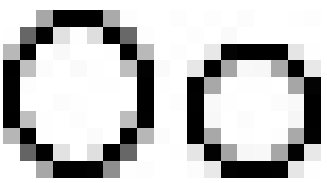
Original - linije

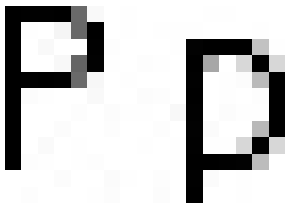
Bez optimizacije

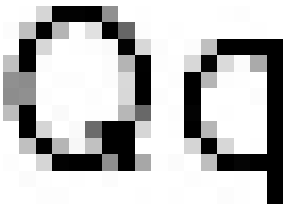
Original - linije

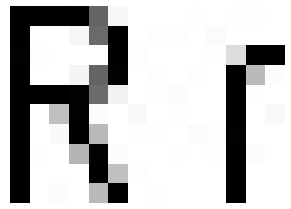


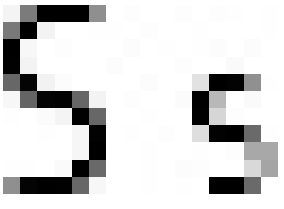
Bez optimizacije

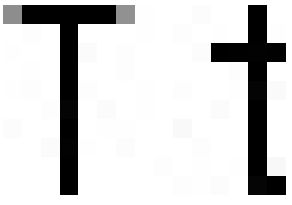






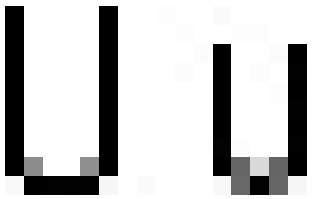


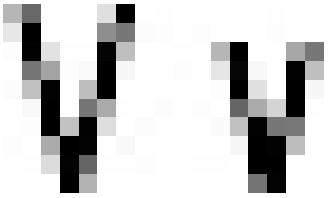


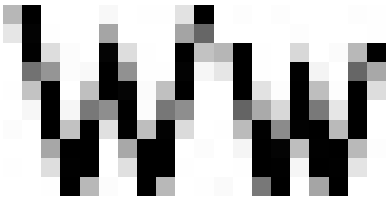


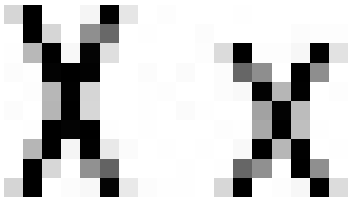
Original - linije

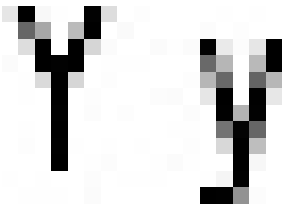
Bez optimizacije

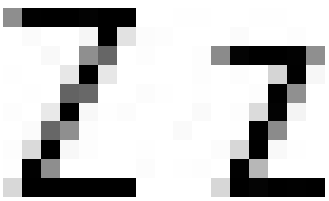












Original - linije

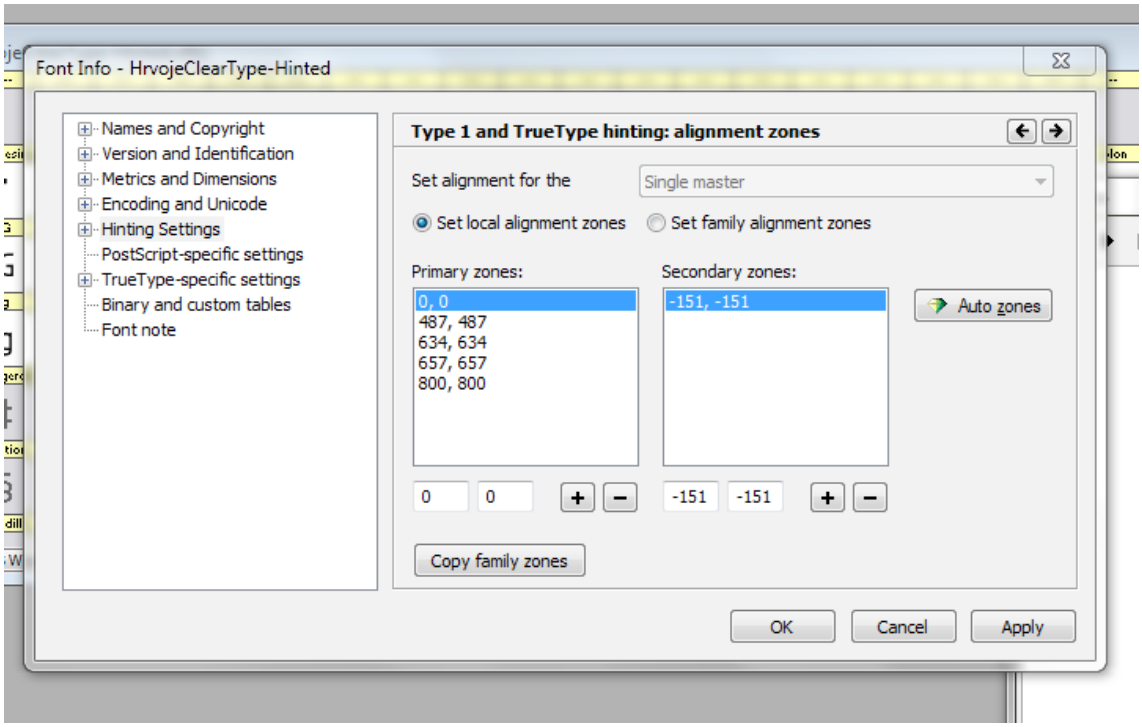
Povećavanje pismovnog reza prikazano je sljedećim primjerom. Vidljive su nepravilnosti, a posebno do izražaja dolazi slovni znak “j”:

Slika 22 - Prikazuje skalabilnost prismovnog reza bez optimizacije

3.2. Optimizacija

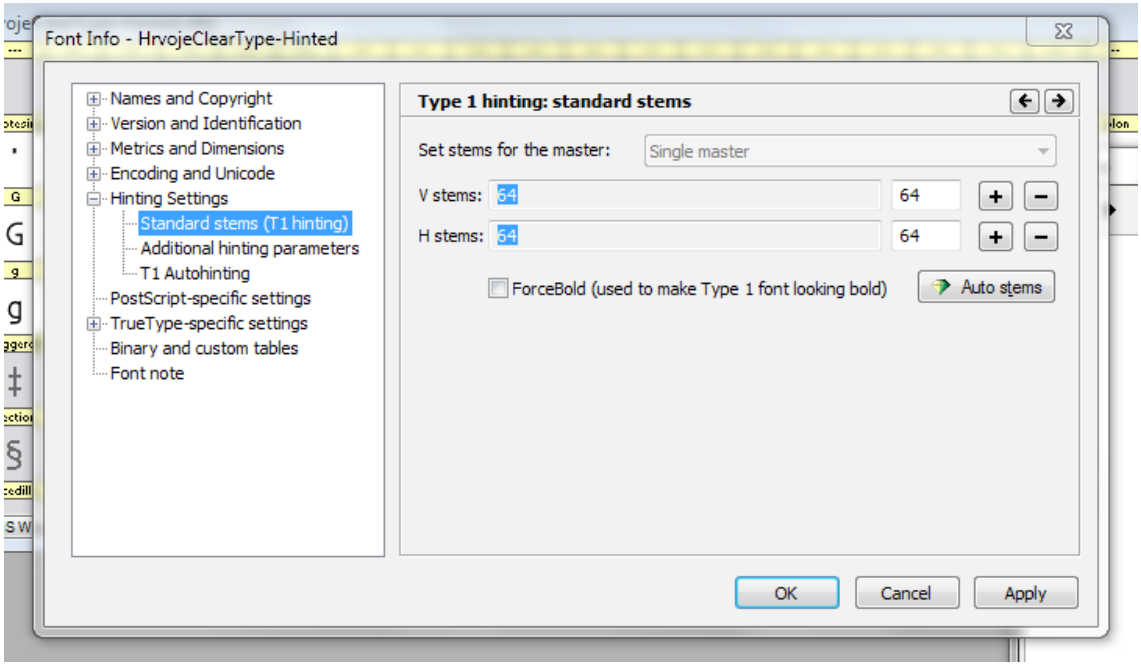
3.2.1. Type 1 optimizacija pomoću Python programskog jezika

Radi se o pojednostavljenju postupku optimiziranja Type 1 tipografije. Kod Type 1 pristupa optimiziranju se pristupa jednostavno – prvo se odrede linije poravnanja teksta, debljinske vrijednosti horizontalnih i vertikalnih poteza sa mogućom varijacijom.



Slika 23 - Prikaz podešavanja zona poravnanja za Type 1 optimizaciju

Nakon toga se ručno povlače smjernice za horizontalne i vertikalne poteze za svaki slovni znak posebno. Ovaj proces uzima puno vremena, s time da se smjernice povlače prema prvim parametrima koji su definirani – linije poravnjanja i debljinske vrijednosti. Kad već postoje definirane te vrijednosti, dobro bi bilo iskoristiti ih, da se ne gubi puno vremena na postavljanje smjernica. Upravo ovdje uskače programski jezik Python. Zahvaljujući Pythonu moguće je automatizirati radnje i preskočiti logički vrlo jednostavne poslove, a fizički vrlo zamorne poput apliciranja jednakih parametara na više slovnih znakova i slične repetitivne zadatake.



Slika 24 - Prikaz ručnog unosa debljinskih vrijednosti za horizontalne i vertikalne poteze

Ukoliko se prebacuju krivulje nekog slovnog znaka iz grafičkog programa u FontLab, može se dogoditi da krivulje nisu zatvorene. Provjera takvih stvari uzima vrijeme, jer se mora ulaziti u svaki pojedini slovni znak i označavati i provjeravati je li objekt zatvoren te u kojem smjeru. Za usporedbu, to se može napraviti pomoću programskog jezika Python u samo nekoliko linija koda gdje se zadaje izlistavanje svih slovnih znakova pismovnog reza. Pronalaze se točke koje se ne dodiruju i ne zatvaraju oblik slovnog znaka. Na taj način saznaje se o kojim slovnim znakovima se radi bez pretjerano utrošenog vremena oko traženja.

3.2.2. Python podrška u Fontlabu

Pisanje Python koda u FontLab softveru odvija se u zasebnom prozoru, a napisane naredbe zovu se Macro programi.

Nije sve tako sjajno oko podrške za Python u navedenom softveru. Prvi problem je što je podržana samo jedna inačica Python programskog jezika - mora biti točno 2.4 - iako se Python razvio i danas postoje puno novije inačice i naprednije inačice. Drugi problem je u tome što metode, funkcije i atributi nisu dovoljno dobro dokumentirani. Da se osoba snađe u tome treba imati puno vremena za testiranje i iskustvo pisanja u Python programskom jeziku.

Neslužbena dokumentacija nalazi se na web adresi *http://www.e-font.de/flpydoc/* i u njoj se nalazi popis funkcija, metoda i atributa na koje su naišli korisnici. Jedan od kontributora tome je zaposlenik tvrtke FontLab, ali njegovo zadnje sudjelovanje u tom projektu je 2004. godine. Radi toga vrlo malo ljudi programira i automatizira radnje u FontLab softveru.

Neslužbeni popis klasa, a od toga se u diplomskom radu koriste klase koje su označene [8]:

1. Anchor
2. AuditRecord
3. Canvas
4. Component
5. Dialog
6. Encoding
7. EncodingRecord
8. Feature
9. Font
10. FontLab
11. Glyph
12. Guide
13. Hint
14. Image
15. KerningPair
16. Link
17. Matrix
18. NameRecord
19. Node
20. Options
21. Point
22. Rect
23. Replace
24. TrueTypeTable
25. TTH
26. TTHCommand
27. TTHPoint
28. TTInfo
29. TTPoint
30. TTStem

3.2.3. Klasa FontLab()

Radi se o hijerarhijski osnovnoj klasi, a predstavlja programsko sučelje. U njoj se nalaze sljedeće metode i atributi [8]:

» Atributi: *font, ifont, ifontslist, glyph, iglyph, iobject, mainwindow, path, filename, version, productnumber, serialnumber, username, count, count_selected, window, delta, scale, tablet_active, tablet_pressure, preview*
» Metode: *Close, Open, Save, GenerateFont, Add, UpdateFont, SetFontWindow, UpdateGlyph, EditGlyph, CallCommand, Selected, Select, Unselect, Message, ScreenToGlyph, GlyphToScreen, UpdateRect, HitContour, GetCanvas, GetConvert, BeginProgress, TickProgress, EndProgress, Random, TransformGlyph, ForSelected, SetUndo*

U ovoj klasi dohvaća se svaka akcija koja se tiče programskog sučelja. Pomoću ove klase mogu se saznati glavne informacije o pismovnom rezu te njima upravljati. Moguće je spremati promjene, izvoziti pismovni rez u raznim formatima i generirati razne promjene.

3.2.4. Klasa Font()

Po hijerarhiji je odmah ispod klase FontLab(). Predstavlja slovne znakove sa sljedećim metodama i atributima [8]:

» Atributi: *file_name, family_name, full_name, font_name, font_style, menu_name, apple_name, font_id, pref_family_name, pref_style_name, mac_compatible, default_character, weight, weight_code, designer, designer_url, classes, glyphs, vguides, hguides...*
» Metode: *New, Open, Save, OpenAFM, SaveAFM, Reecode, FindGlyph, DefineAxis, GenerateUnicode, GenerateNames, GenerateGlyph, has_key, GenerateFont*

Ova klasa predstavlja cijeli pismovni rez sa pripadajućim parametrima koji su svojstveni samo za taj pismovni rez. Moguće je upravljati enkodiranjem, generirati ime pismovnog reza i slovnih znakova.

3.2.5. Klasa Glyph()

Hijerarhijski ispod klase Font(). Uključuje sve opcije vezane za uređivanje slovnog znaka. Metode i atributi klase Glyph() [8]:

» Atributi: *parent, nodes, mark, note, hhints, vhints, vlinks, hlinks, hguides, vguides, components, replace_table, kerning, name, image, index...*
» Metode: *Selection-Methods, Metrics-Methods, Overlap-Methods, Hints-Methods, Anchor-Methods, Transformation-Methods*

Klasa Glyph() predstavlja četverac slovnog znaka. U ovoj klasi sve se radi na razini samo jednog objekta (slovnog znaka). Za pozivanje ove metode na drugim slovnim znakovima treba se koristiti klasom Font().

3.2.6. Optimizacija iscrtavanja pomoću zadanih klasa

Uvjet koji FontLab stavlja pred Macro naredbe jest da svaki program počinje sa komentarom:

```
#FLM: ImePrograma
```

Ovdje po ime programa mora pisati naziv skripte.

Pošto već postoje unesene vrijednosti poravnanja i debljinskih poteza, trebalo bi aplicirati optimizaciju na svakom slovnom znaku.

Prvi zadatak jest dohvatiti slovni znak te provjeriti je li dohvaćen traženi slovni znak. Kod izgleda sljedeće:

```
#FLM: Hinting

a = fl.font.FindGlyph(„a“)
b = fl.font.glyphs[a].name

print b
```

Ove tri linije koda vratit će ime znaka koji je dohvaćen. Prva linija koda predstavlja varijablu koja traži slovni znak a. Slovni znak je pronađen i kao rezultat se vraća broj koji predstavlja taj znak u kodnom sustavu. U drugoj liniji koda nova varijabla traži ime slovnog znaka po broju u kodnom sustavu koji je vratila prva varijabla. Ispisom se u ispisnom prozoru ispisuje slovo „a“, što potvrđuje da je moguće dohvatiti individualni slovni znak.

Sada je moguće provjeriti sadrži li slovni znak smjernice koje služe za optimizaciju iscrtavanja. Upravo to sljedeća skripta provjerava:

```
#FLM: Hinting

a = fl.font.FindGlyph(„a“)
b = fl.font.glyphs[a].vhints
c = fl.font.glyphs[b].hhints

print b
print c
```

U drugoj i trećoj liniji koda program potražuje horizontalne i vertikalne smjernice. Na oba upita u ispisnom prozoru dobivaju se prazne uglate zagrade, što znači da nema apliciranih smjernica, što je i točno.

Sljedećom skriptom apliciraju se horizontalne i vertikalne smjernice, koje će biti provjerene ispisom:

```
#FLM: Hinting

a = fl.font.FindGlyph(„a“)
b = fl.font.glyphs[a].vhints
c = fl.font.glyphs[b].hhints

def autoHintGlyph(currentIndex):
    fl.font.glyphs[currentIndex].Autohint()
autoHintGlyph(a)

print b
print c
```

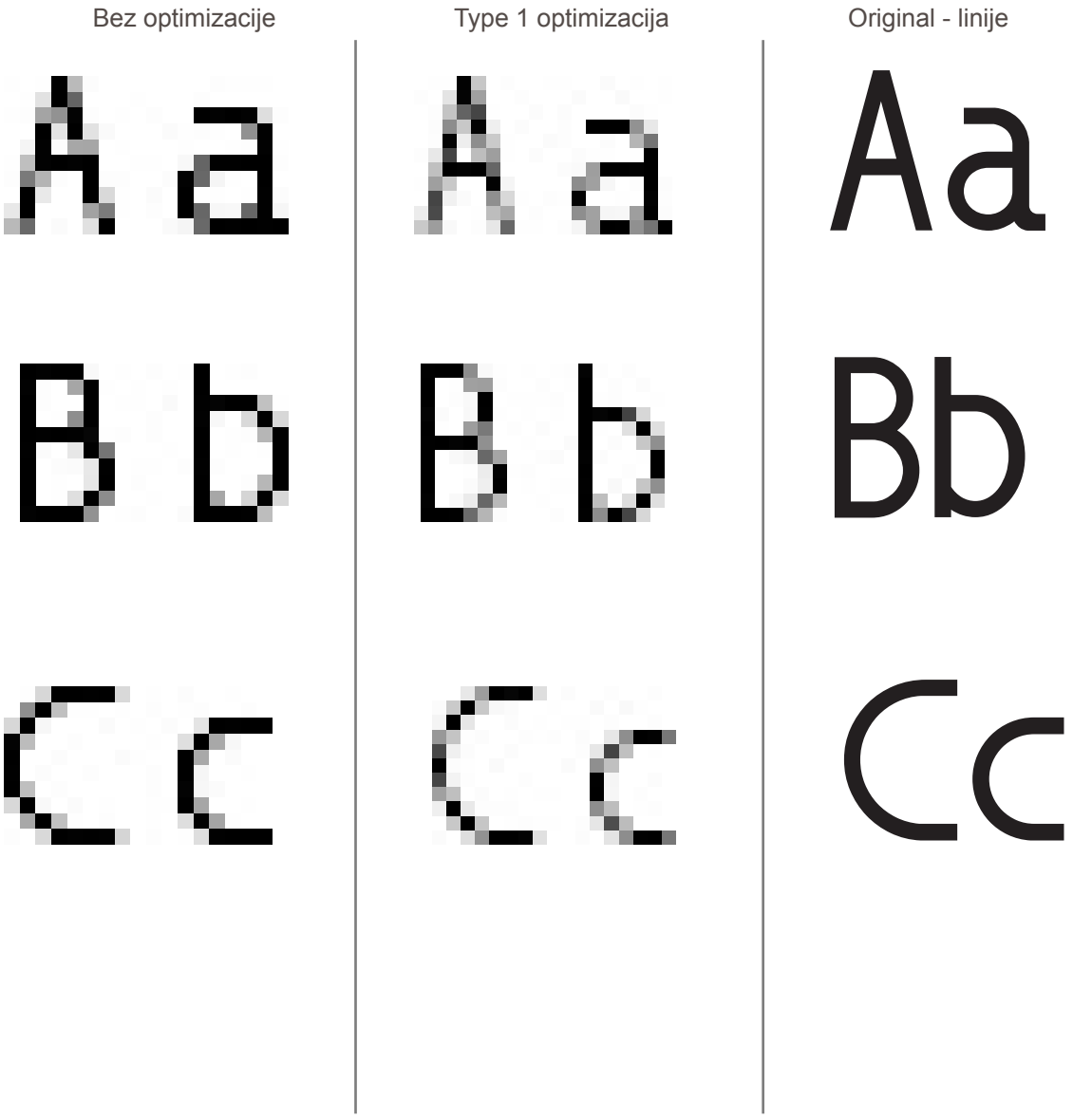
U funkciji autoHintGlyph zovemo metodu Autohint() koja postavlja horizontalne i vertikalne smjernice. Pri ispisu dobiva se:

```
[<VHint: p=30, w=64, parent: “a”>,<VHint: p=286, w=66, parent: “a”>]
[<HHint: p=0, w=64, parent: “a”>,<HHint: p=248, w=64, parent: “a”>,<HHint:
p=421, w=64, parent: “a”>]
```

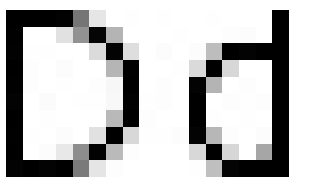
Aplikacija smjernica je uspjela. U gornjem redu nalazi se i popis vertikalnih smjernica, a u donjem redu popis horizontalnih smjernica. Ova metoda je vremenski puno uspješnija od ručne metode za Type 1 tipografiju. Pošto je aplikacija uspjela na jednom slovnom znaku, treba je aplicirati na sve slovne znakove sljedećom skriptom:

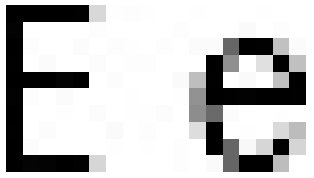
```
#FLM: Hinting

a = fl.font.FindGlyph(„a“)

def autoHintGlyph(currentIndex):
    fl.font.glyphs[currentIndex].Autohint()
for g in fl.font.glyphs:
    glyphIndex = g.index
    autoHintGlyph(glyphIndex)
```

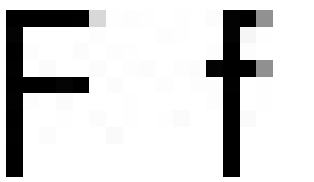
Ovime su aplicirane smjernice na sve znakove u pismovnom rezu. S ovime je završila Type 1 optimizacija za ekrane. Usporedba neoptimiziranih i optimiziranih slovnih znakova:

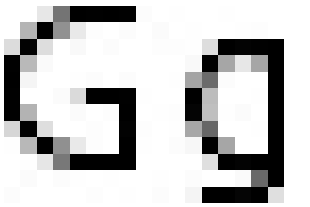


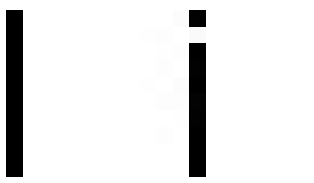
Bez optimizacije



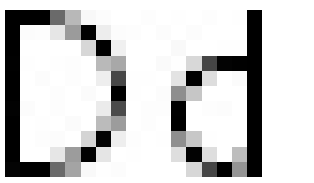


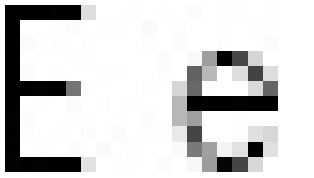


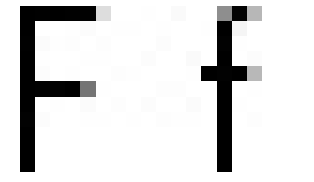


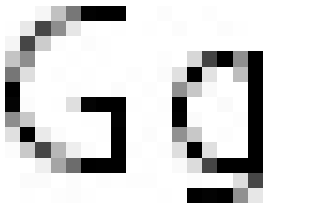


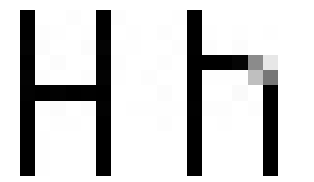
Type 1 optimizacija

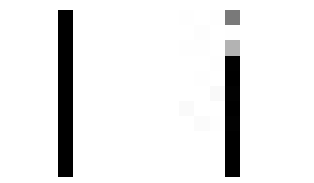






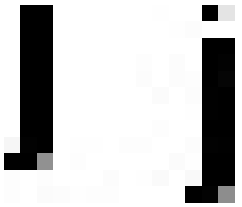


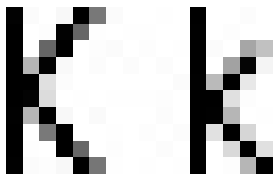


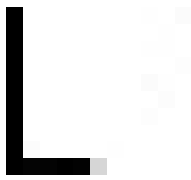


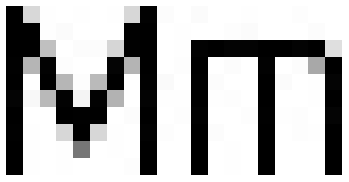
Original - linije

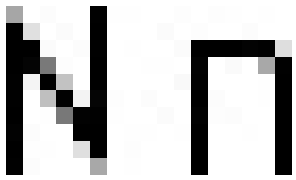
Bez optimizacije

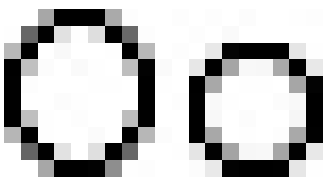




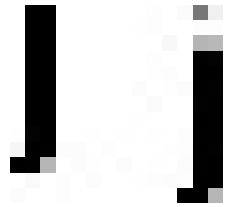


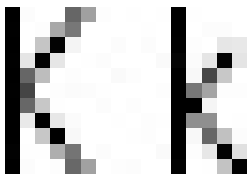


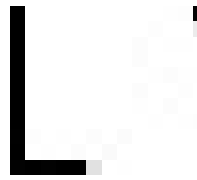


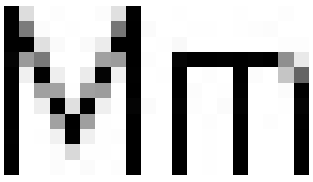


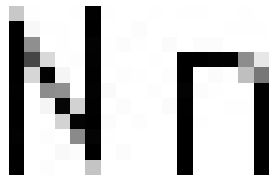
Type 1 optimizacija

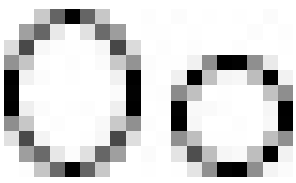




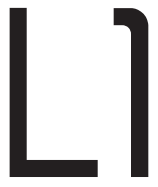




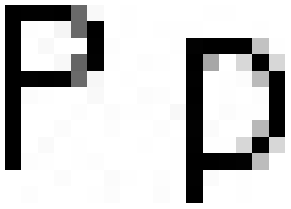


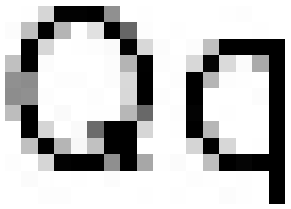


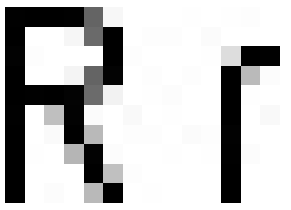
Original - linije

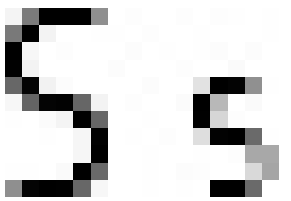


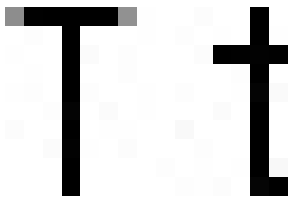
Bez optimizacije

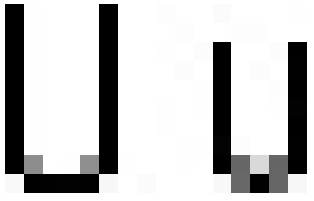




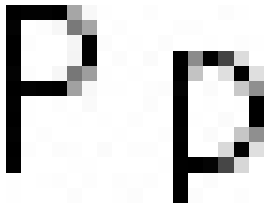


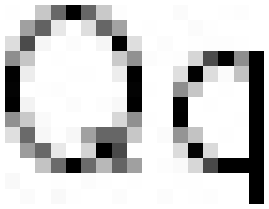


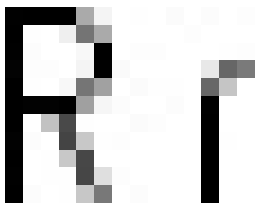


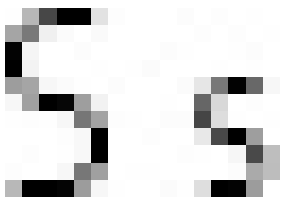


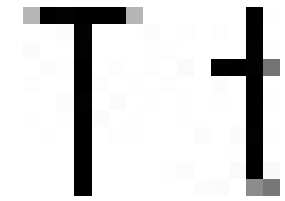
Type 1 optimizacija

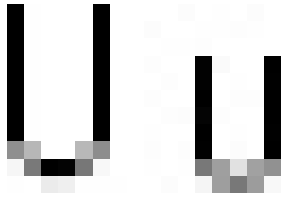




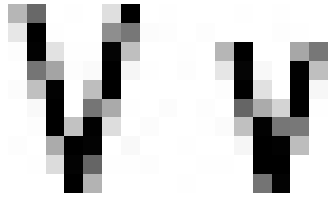
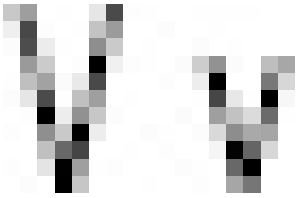
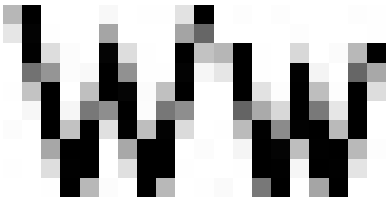
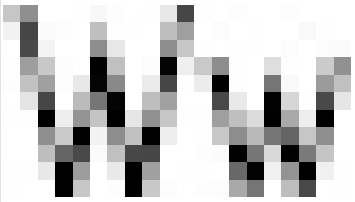
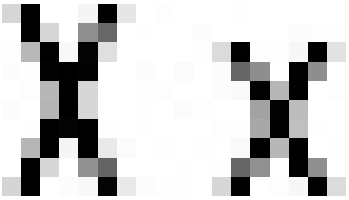
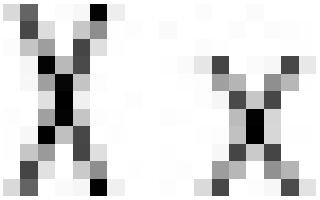
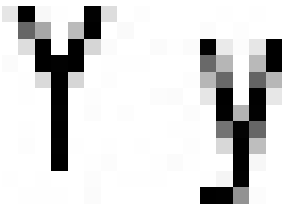
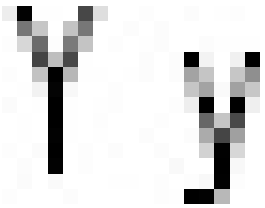
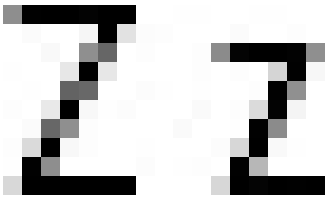
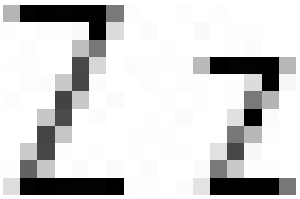




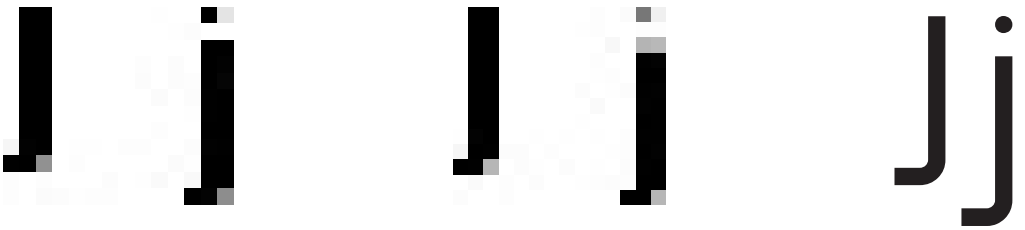




Original - linije

Bez optimizacije	Type 1 optimizacija	Original - linije
		Vv
		Ww
		Xx
		Yy
		Zz

Slika 25 - Prikazuje povećavanje pismovnog reza nakon Type 1 optimizacije



Nakon Type 1 optimizacije nije vidljiv velik korak naprijed. I dalje problematično slovo “j“ ne prikazuje se jednoliko na svim veličinama. Na gotovo svim slovnim znakovima vidljiva je varijacija pri povećavanjima.

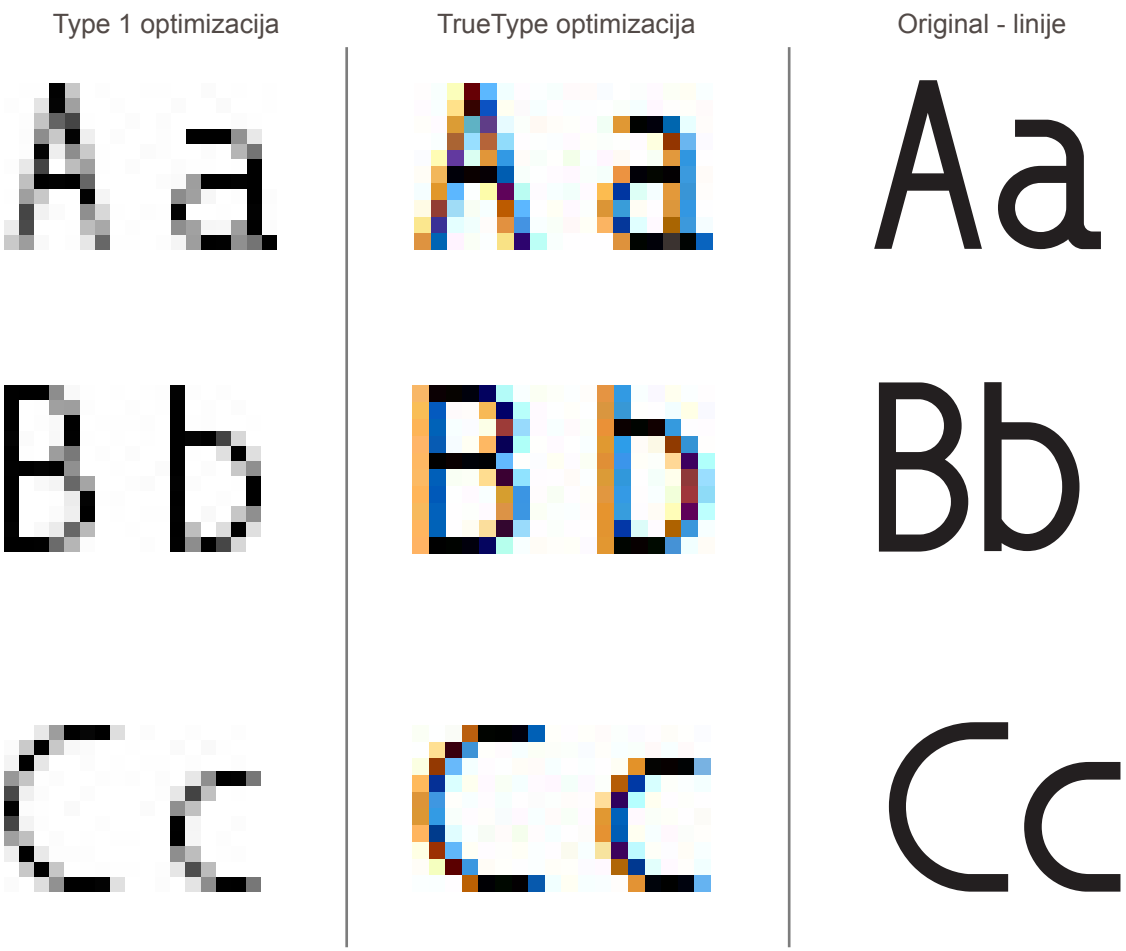
Vidljivo je da se Type 1 optimizacijom još uvijek ne mogu dobiti kvalitetni rezultati u Windows okružju. Pošto se Type 1 optimizacija svodi na opisivanje oblika svakog slovnog znaka, u idućem koraku moguće je transformirati te informacije radi TrueType optimizacije za ClearType tehnologiju.

3.2.7. TrueType optimizacija

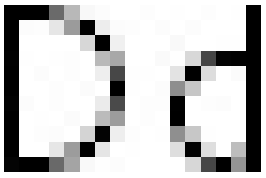
Pošto se Type 1 optimizacija nije pokazala učinkovita, prelazak na TrueType optimizaciju sljedeći je korak u optimizaciji pismovnog reza.

Zahvaljujući FontLab sotveru moguće je podatke o smjernicama iskoristiti za TrueType optimizaciju. Konverzija se ne vrši više pomoću programskog jezika Python. U izborniku pri generiranju pismovnog reza, odabire se format OpenType TTF i zatim se jednostavno pismovni rez sprema u OpenType format sa TTF krivuljama čije su informacije za krivulje i optimiziranje već preuzete iz Type 1 optimizacije.

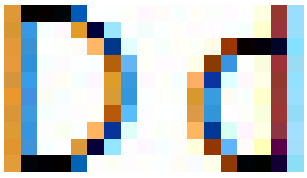
Primjer usporedbe sa Type 1 optimizacijom:



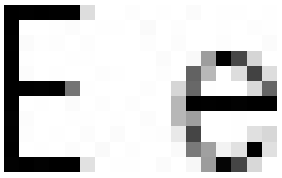
Type 1 optimizacija

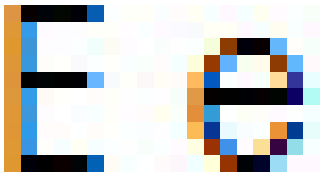


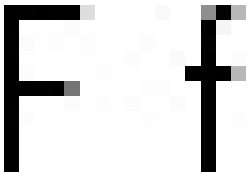
TrueType optimizacija

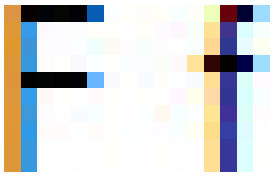


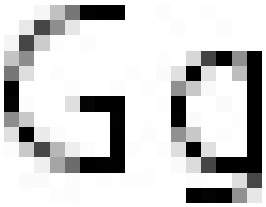
Original - linije

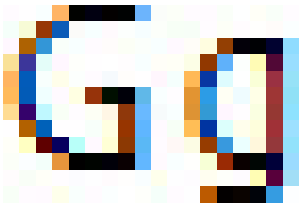


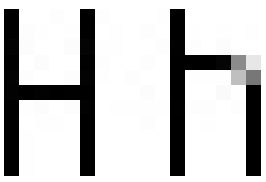


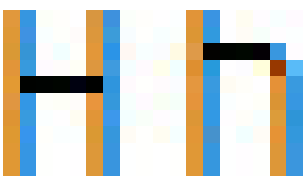


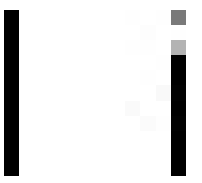


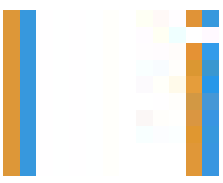






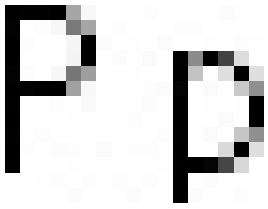


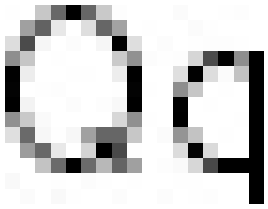


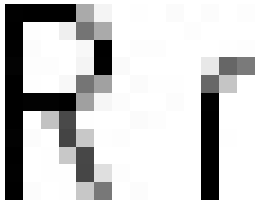


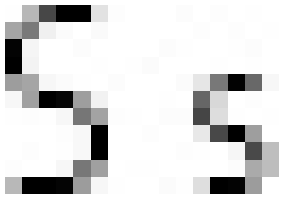
Type 1 optimizacija | TrueType optimizacija | Original - linije

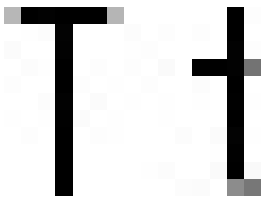
Type 1 optimizacija

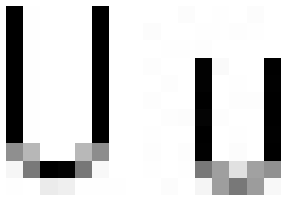




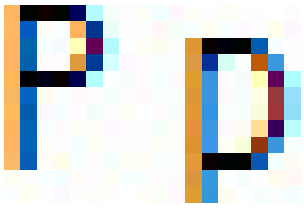


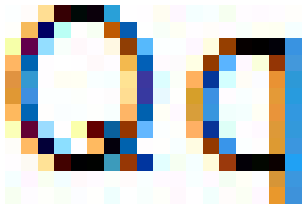


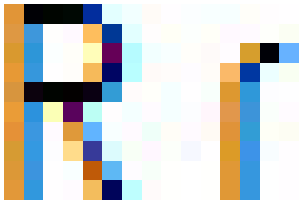


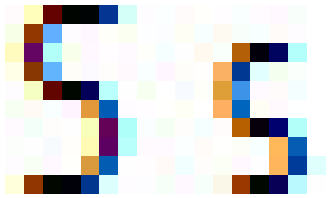


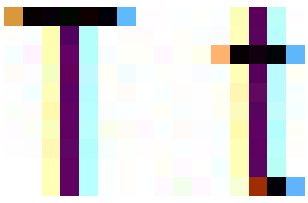
TrueType optimizacija

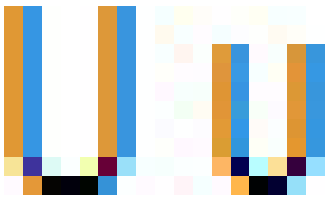




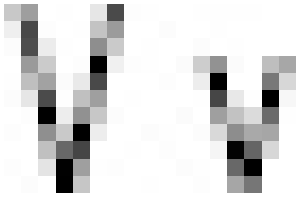
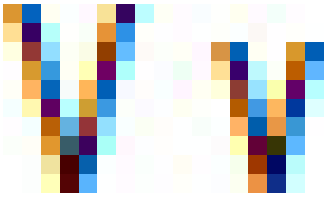
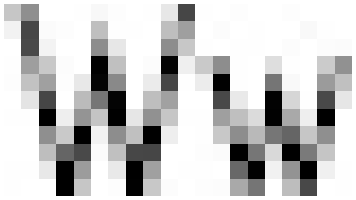
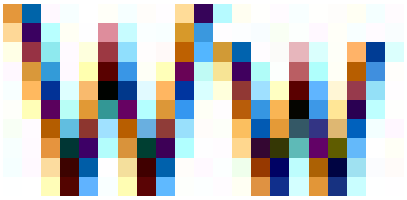
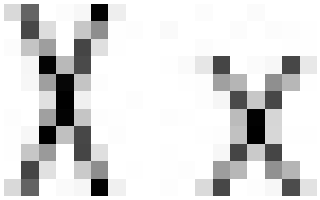
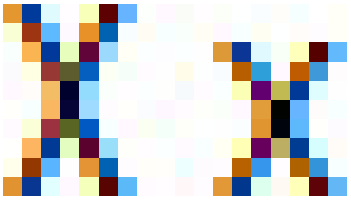
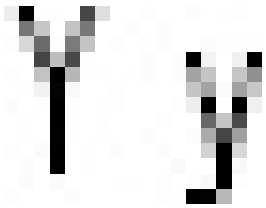
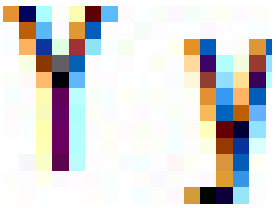
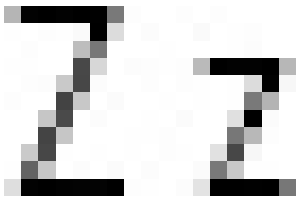
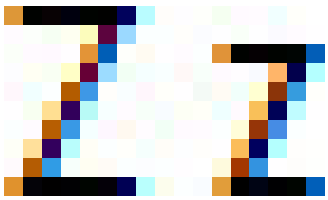








Original - linije

Type 1 optimizacija	TrueType optimizacija	Original - linije
		Vv
		Ww
		Xx
		Yy
		Zz

Slika 26 - Prikazuje povećavanje TrueType linija. U odnosu na Type 1, povećavanje kod TrueTypea ne dolazi do velikog gubitka u prikazu

Na ovom primjeru ClearType tenhologija iskorištena je u dobrim razmjerima. U odnosu na Type 1 optimizaciju provedena je podpikselna kontrola, zato su rubovi slovnih znakova u različitim bojama. Ovim efektom stvara se bolja optička varka u oku promatrača, što poteze na slovima čini zaglađenijima. Na primjerima slovnih znakova ne vidi se točno podpikselno iscrtavanje, jer se radi o logičkom uvećanju piksela, a ne fizičkom kao kod slike. U realnoj primjeni slovni znakovi izgledaju još zaglađenije.

Pri povećavanju vidljivo je da se minimalno gubi na kvaliteti i obliku slovnih znakova. Nije moguće izbjeći posljedice, jer ipak je prikaz ograničen obzirom na razlučivost ekrana.

4. REZULTATI I RASPRAVA

U diplomskom radu prikazana je metoda optimizacije pismovnog reza za prikaz na ekranima. Prikazan je efikasan način gdje se uštedom vremena uz jednostavno znanje o programiranju mogu napraviti vrhunski rezultati.

Raspravljene tehnologije optimizacije će procvatom novih standarda na internetu dobiti veliki značaj u borbi protiv neoptimiziranih pisama. Nova verzija kaskadnog jezika (CSS3) omogućuje korištenje bilo kojeg pismovnog reza na internetu, a početi će stvarati probleme sa neoptimiziranim pismima kod korisnika sa Windows operacijskim sustavom. Kako Microsoft ne odustaje od svoje filozofije iscrtavanja tipografije, jedino rješenje koje u potpunosti može iskorijeniti ovaj problem su ekrani sa vrlo visokom razlučivosti. Trenutno takvih ekrana nema komercijalno dostupnih pa optimiziranje tipografije ostaje jedino rješenje.

Zasad nema pravih uputstava kako dobiti savršene rezultate u ClearType tehnologiji. Čini se da takvo što niti ne postoji, jer predstavljanjem nove DirectWrite tehnologije – osim što je učinjen korak naprijed za tipografiju na internetu u Windows okruženju, Microsoft kao da je priznao poraz za ClearType i GDI procesor za optimizaciju tipografije. Ostali korisnici koji su još na Windows XP operacijskom dodatno otežavaju stvari tipografima, jer na tom operacijskom sustavu nije uključen ClearType optimizator, iako postoji. Windows XP korisnici najveća su baza korisnika i posebno treba paziti na tipografiju za taj specifičan operacijski sustav.

Osim dizajnera koji „konzumiraju tipografiju“, postoji i scena koja živi od tipografije. Tipografima je zbilja teško danas napraviti pismovni rez koji će na svim platformama identično i zadovoljavajuće izgledati. Većina tipografa radi na Mac računalima, a za Mac okruženje postoji najviše softverskih alata za izradu tipografije. Stoga većina tipografa nije svjesna s čime se bore Windows korisnici, njima je optimizacija pismovnog reza za prikaz na ekranima sasvim stran pojam.

U primjerima koji su navedeni kod optimizacije vidljivo jest da ne može sve ostati na osnovnoj razini optimizacije. Često je potrebno ući u TrueType sustav i napraviti neke preinake. U ovom radu je pronađeno programsko rješenje za jednostavnije i brže rješavanje problema optimizacije.

Tipografija u Windows okruženju nailazi na velike prepreke zbog toga što Microsoft ne održava nikakve seminare, ne drže se predavanja o mogućnostima izrade tipografije, o unaprijeđenju rasterizacijskih algoritama, općenito situacija ne pogoduje tipografima. A sa druge strane, najviše su zavisni o tehnički potkovanim tipografima, jer bez ikakve optimizacije tipografija u Windows operacijskom sustavu nema previše smisla. Tipografa koji optimiziraju pismovne rezove za prikaz na ekranima ima vrlo malo, toliko malo da je velikim tvrtkama poput MonoType Imaging-a često neisplativo optimizirati nove i već postojeće pismovne rezove.

Rješenje doneseno u ovom diplomskom radu zasada je jedno od rijetkih koje poboljšava i ubrzava razvoj pisama namijenjenih prikazu na ekranima. U radu je uspješno izbjegnut proces ručne optimizacije pomoću automatizacije proces. Nije samo stvar kvalitete optimizacije, pitanje je i automatizacije procesa optimiziranja, gdje dizajneri tipografi mogu lako primijeniti optimizaciju na svom pismovnom rezu.

5. ZAKLJUČAK

Unutar zadanih tehnoloških okvira pronađeno je rješenje kako kako što brže i efikasnije optimizirati tipografiju za prikaz na ekranima. Razvijen je moderan, čitljiv i sustavno optimiziran pismovni rez. Rezultat je potaknut dobrim informiranjem i ispitivanjem područja optimiziranja. Obzirom na kvalitetu i lakoću implementacije, svaki tipograf može iskoristiti ove korake kako bi uljepšao doživljaj i poboljšao čitljivost svog pismovnog reza u Windows okruženju.

Koraci za bolju optimizaciju podrazumijevaju analizu ekranske tehnologije, operacijskih sustava, formata zapisa, vrstu pisama i analizu debljinskih vrijednosti. Slijedom koraka koji su objašnjeni u diplomskom radu može se napraviti kvalitetan pismovni rez koji pogoduje modernim tehnologijama, a ujedno zadovoljava kriterije tehnologija koje nisu aktualne.

Rješenje doneseno u diplomskom radu namijenjeno je široj publici i implementiranju na postojećim pismovnim rezovima koji ne sadrže optimizaciju. U konačnici, najvažnije se obračunati sa kompliciranom Windows tehnologijom radi koje treba optimizirati pismovne rezove. Na ovaj način moguće je boriti se i sa niskim razlučivostima ekrana koje su najveća prepreka prikazu tipografije.

Radni tok razvijen u diplomskom radu okrenut je prema tehničkoj strani tipografije koju mnogi zanemaruju. Programiranje je stepenica koja ubrzava razvoj i treba biti što prije prihvaćeno u krugovima tipografa dizajnera. Dizajneru koji se može barem malo okrenuti tehničkoj strani priče, savjeti i koraci iz ovog diplomskog rada predstavljeni konkretnim rješenjem ukazuju na ubrzanje projektiranja i optimizacije pismovnog reza.

6. LITERATURA

1. David Bergsland (2010). *Practical Font Design*, Radiqx press, Mankato, MN, SAD
2. http://www.linotype.com/catalog/categories.html, *Linotype - FONTS BY CATEGORIES*, 2.9.2012.
3. http://www.dafont.com, *DaFont.com*, 2.9.2012.
4. http://www.fontshop.com/fonts/category/, *FontShop.com -Typefaces by Category*, 2.9.2012.
5. Franjo Mesaroš (1981). *Tipografsko oblikovanje*, Viša grafička škola u Zagrebu, Zagreb
6. Yannis Haralambous (2007). Fonts & Encodings, O'Reilly Media Inc., Sebastopol, SAD
7. FontLab Ltd. (2006). *FontLab Studio 5 - User's Manual for Windows*, FontLab Ltd.
8. http://www.e-font.de/flpydoc/, *Unofficial FontLab/Python API Reference*, 10.8.2012.

6.1. Popis slika