Osvrt na odgledano predavanje

Prof.dr.sc. Klaudio Pap započeo je predavanje spomenuvši kako studenti na grafičkom fakultetu i u kojem programu izrađuju font. Kao što je to i spomenuo već i na uvodnom predavanju, a i u odrađenoj prvoj vježbi, koristili smo se aplikacijom *Fontographer* te je spomenuo kako se može isto to raditi i u *FontLab-u*. U prvih 30 minuta videa profesor objašnjava samo sučelje Fontographera i najvažnije postavke i alate kojima se mi studenti počinjemo koristiti, također tu dolaze i novi pojmovi i definicije koje se trebaju naučiti za daljnje napredovanje. ASCII standard je u decimalnom obliku. Spomenulo se kako je font uređena skupina kodnih pozicija te kako se svaki kodni znak tj. slovo nalazi u svome digitalnom četvercu koji je omeđen od 5 pravaca. Ti pravci koji izrađuju digitalni četverac su se prije nosili u rukama dok se to nije sve digitaliziralo i znatno olakšalo radom na računalu. Povećanjem veličine fonta skaliramo i digitalni četverac, po zadanom je između 2 linije u Fontographeru. Desna linija je dizajnerska naredba i samo se ona može pomicati te se slova mogu preklapati ovisno gdje je ona pomaknuta. To nas dovodi do situacije kada imamo slovo „V“ i slovo „A“ jedno do drugog. Najčešće se javlja situacija da kada neka riječ, npr. Avangarde, u ovom slućaju je to čitljivo, bude nečitljiva zbog velike praznine između slova „A“ i „V“, zato se koristimo desnom linijom kako bi se ona poklopila sa lijevom linijom novog slova, kao što je profesor konkretno pokazao u Fontographeru. Nazvali smo ih parovi podrezivanja. U fontu ne postoji mjerna jedinica kao inč, metar, centimetar već relativna jedinica.

TrueType fontovi u sebi sadrže parove podrezivanja koji će funkcionirati bez obzira koristimo li se mi tim fontom u Photoshopu, Illustratoru, InDesignu, Wordu ili nekom drugom programu koji ima mogućnost unosa teksta. Ako iz svog fonta koji, ne sadrži znakove sa kvačicama, radimo nove znakove, konkretno „Č“ ili „Ć“, uzeti ćemo već postojano „C“ te ćemo ga kopirati i pronaći kodnu poziciju za „Č“ ili „Ć“ i tamo nacrtati tu kvačicu. Isto tako kada radimo „iz ničega“ nove znakove, uvijek je dobro napraviti prvo onaj znak iz kojeg možemo laganim dodavanjem ili brisanjem napraviti novi bez puno posla, konkrentije: ako imamo slovo „O“ puno ćemo lakše već napraviti slovo „Q“ ili iz slova „D“ u „Đ“ ili kurent slova „a“ i „d“. Spominje se laboratorij koji je prije postojao na Grafičkom fakultetu u kojem su se sa olovnim slogom izlijala slova za tiskanje knjigotiskom. Spominje se važnost Bezierove krivulje i njezine jednadžbe sa pomoćnim linijama (plusićima) koje sam i u osobnom iskustvu vidio i u Illustratoru.

Profesor zatim prelazi na softver PSConvert koji je napisan u C++ te on, u ovom konkretnom slućaju, dizajnira linije u obliku lepeze koje se šire od 0 do 180 stupnjeva. Samo objašnjenje rada toga softvera nije uopće kompliciranо koliko su zapravo ti kodovi koje treba znati i koji kod što točno predstavlja, tj. koja je njegova funkcija u svemu tome. Prikazano je kako se u UEStudio programu zapravo pišu ti kodovi ali je i prije toga pokazano kako se jednostavnim interface softverom možemo koristiti ako nam kodovi nisu najjača strana. Jednostavno upišemo potrebne podatke, u ovom slućaju su to bili inicijali našeg imena i prezimena te gustoća linija koje se šire u obliku lepeze i zadnji pritiskom na „Generate“ mi bismo trebali u otvorenom programu (Illustrator) dobiti rezultat koji je ovisio o našim unesenim podatcima. Da bismo nešto promijenili jednostavno bismo sve isto ispočetka

napisali i u Illustratoru bi nas pitalo kako smo napravili promjenu te dobivamo dvije opcije; da odustanemo (cancel) ili da generiramo nove podatke koje smo unijeli. U UEStudio programu profesor nam je pokazao kako se to isto može mijenjati mijenjajući brojeve u određenim stupcima, naravno izgleda zbunjujuće s obzirom na onaj lakši interface koji je s razlogom napravljen kako bi svatko mogao to isprobati. Zapravo i nije nešto kompliciranо kada se jednom shvati jer i sam taj princip promjene nekakvih brojeva i dobivanje različitih rezultata možemo i primjeniti na igrice ili dizajniranje aplikacija.

Idući primjer je program koji će ispisati tekst u spiralu, sami olakšani interface ima ovaj put malo više opcija nego ovaj prije njega. Dok se u Illustratoru tekst u spiralu može još lakše koristiti i uređivati, no u ovom primjeru smo to uređivali mijenjajući varijable gustoće same spirale ili slova. Također postoji mogućnost promjene parametra rezolucije gdje možemo odrediti koliki će nama biti papir na kojem mi izrađujemo tu spiralu zato što se nekada zna dogoditi da sama spirala ode izvan papira te ju ne vidimo u potpunosti, onda visina fonta i faktor snage spirale, drugim riječima gustoća spirale. Opet u UEStudio programu možemo vidjeti razne kodove i brojeve koje na prvi pogled i nemaju baš najjasniju smisao kao sami interface ali provedevši malo više vremena sve ima svoju glavu i rep. Kako je profesor rekao, na početku tog programa se nalaze kodovi za izradu samog interfaceagdje su se izdvojile varijable koji program koristi dok se pri dnu nalaze kodovi koji rade grafiku. Dakle ide na isti princip kao i prvi primjer samo je u ovom slućaju spirala sa tekstom dok je u prvom slućaju slova sa linijama unutar njih koji se šire.

Bezierova krivulja, jako važna parametarska krivulja 3. stupnja koja se prvi put upotrijebila u tvornici automobila. Izuzetno je važna ne samo u Fontographeru već i u ostalim programima gdje se koristimo krivuljama. Konkretno u Fontographeru, tangente te krivulje se označavaju plusićima i njima možemo manipulirati kako će naša krivulja izgledati. Prebacimo li pomoćnu točku u *Corner point* onda ona postaje nezavisna i na njoj se ne koristi jednadžba, dakle pomaknemo li samo nju gore ili dolje, druga točka se neće uopće pomaknuti dok se u Bezierovoj krivulji pomaknuvši jednu točku automatski pomiče i druga točka jer se koristi jednadžba. U tangentnom modu, linije tj. pravac line tangentu na neku drugu točku i tako nastaje krivulja, te ju ne možemo pomicati lijevo ili desno dok smo u tangentnom modu. Postoji opcija *Predictable curves* koja nam i sama govori što znači, predvidljive krivulje, gdje tijelo te krivulje mora ići baš onako kako je za to predviđeno.

Naučili smo što je PostScript, programski jezik namjenjen grafici tj. vektorski opis stranice koji simulira ispis, u ovom slućaju se njime može provjeriti koliko su studenti naučili što je i kako se koristi Bezierova krivulja. Upoznajemo *Curve tool* koji sadrži 4 točke po X Y koordinati. Prva točka je tekuća ravno točka prije samog alata *Curve tool.* Naredbom „move to“ stvaramo prvi točku curve tool naredbe. Konkretan primjer je bio vektorski napravljen auto koji je imao malo povišenu haubu te ju je trebalo smanjiti. U Illustratoru bismo to vrlo lagano mogli kada bismo samo selektirali krivulju i igrali se pomoćnom linijom dok ne smanjimo haubu na zadovoljavajuću poziciju. No, u ovom slućaju smo to trebali smanjiti tako da se prvo nađe redak u kojem se nalazi ta krivulja hauba, profesor je dodatno olakšao taj posao napisavši „hauba“ pored tog parametra koji nam određuje poziciju tih malih plusića, smanjenjem iz 200 u 150 smo smanjili tu krivulju bez ikakvih problema.

Spominje se SVG jezik, *Scalable Vector Graphics* koji poznaju svi browseri (Mozilla, Chrome, Opera itd.) te je on iz područja XML jezika za web. Profesor je u ovom primjeru konkretno pokazao gibanje jednog predmeta po krivulji u plavoj boji. Naravno to je sve napravljeno u UEStudiju. Naravno, tu krivulju mi možemo i sakriti tako da kao rezultat dobijemo predmet kako ide po nevidljivoj krivulji. Također možemo promijeniti vrijeme putanje te krivulje, hoće li ona ići 5 sekundi ili 10 sekundi. Vezano je samo za moment ispisa kada nešto prikazujemo. Sami predmet je također napravljen od posebnih kodova koji se zajedno nalaze sa tom krivuljom i vremenom putanje. Imamo 6 brojeva za *Curve tool* i u SVG. I dalje sve to ide na isti princip kao i prošla 2 primjera samo što je u ovom slućaju vektorska animacija u web pregledniku koju mi možemo zumirati koliko hoćemo ali ona i dalje ne postaje mutna ili pikselična kao što to primjetimo u Photoshopu.

U idućem primjeru vidimo kružnicu od manjih kružnica i lopatice koje proizlaze iz centra kružnice, no najvažnija stvar kod toga je kako se boja kružnica i lopatica mijenja te profesor detaljnije objašnjava korištenje boja u digitalnom i tiskovnom načinu. Spominje se HSB color sustav gdje H stoji za HUE, S za *Saturation* te B za *Brightness*. HUE je spektar napravljen u kružnici od 0 do 360 koji može biti od 400 do 700 nanometara. Ako stavimo od 0 do 1 to je isto kao i od 0 do 360, ali ako stavimo od 0 do 0.5 onda je to od 0 do 180, jer je logički pola od 360 jednako 180 (isto kao i 0.5 od 1). Ako želimo regulirati tako da svaka kružnica ima posebnu boju onda trebamo raditi HUE kroz petlju gdje će se HUE mijenjati ili jednostavno napišemo random number (rn) gdje će se to automatski napraviti u domeni rada HUE.

Profesor prelazi na objašnjavanje pojma rastriranje, varanje s istom bojom dodavanjem vode ili bijele boje da onda dobijemo druge nijanse iste te boje koja iz daleka izgleda kao da smo koristili par sličnih boja. Pomoću programa GSView profesor nam objašnjava kako se rasterski pokaže manipuliranje jednom bojom različitih nijansi, u ovom slućaju je to jako grubo pokazano da mi vidimo o čemu se tu radi, u praksi je to jako malo, skoro neprimjetno prikazano jer mi to ne vidimo golim okom sve dok se to ne približi jako blizu. Tako se na primjer te velike reklame na autocestama i rade, dođemo li jako blizu vidjeti ćemo točkice koje mi naravno nikada nećemo vidjeti gledamo li mi tu reklamu iz neke udaljenosti iz auta jer nama ta slika izgleda sasvim normalno bez ikakvih točkica i bilokakvih nepravilnosti, s time se i znatno uštedi na tiskanju boja jer nema potrebe da takvo nešto veliko bude gusto kao da printamo nešto za A4 format. Pokazavši te rasterske elemente u GSView programu, opet u UEStudio programu možemo mijenjati gustoću i oblik tih rasterskih elemenata promijenivši brojeve posebnih varijabli.

Profesor napominje kako se u HTML obliku koristi samo RGB sustav boja te pokazuje kako se matematičkim jednadžbama ovisno o zadanoj sivoći koliko ta točkica mora napasti domenu od -1 do 1 da bi simulirala zacrnjenje. Također, ako stavimo RGB sliku na nešto što ćemo kasnije otisnuti, ta slika u RGB sustavu zapravo prolazi kroz konverziju gdje se ona pretvara u CMYK sustav i dobiva se nešto drukčija slika. To su dva različita svijeta.

Vedran Novosel