Važnost informacijsko-komunikacijskih tehnologija u metodici nastave na grafičkom fakultetu

Predavanje započinjemo fontom. Font (pismo, tekst) je resurs koji se na grafičkom fakultetu samostalno izrađuje. Čitanje se nije promijenilo, čitali smo prije kao i danas, ali se dizajn slova, slovnog znaka mijenjao. Otvorimo li Fontographer, vidjet ćemo program u kojem izrađujemo fontove i editiramo postojeće. Zapravo što je font? Font je lijepa uređena nakupina kodnih pozicija, a na svakoj postoji slika (eng. glif). Ako pogledamo tablicu, po američkom kodnom standardu ASCII su kodirana slova, danas se mi sami brinemo kako će izgledati ta slova. Ako uzmemo slovo A, verzal slova i stavimo ga u digitalni četverac, stavili smo ga u pravokutnik omeđen pravcima. U prošlosti smo ga mogli osjetiti na dlanu, u olovu. Zapravo digitalni četverac je jedan poligon zatvoren beskonačnim pravcima. Tako da možemo dizajnirati bilo kakva slova, npr. s kvačicama. Uprogramima za slaganje teksta (Word, Photoshop, Indesign), možemo ih poslagati na pisnu liniju. Otvorimo li metriku fonta, možemo vidjeti kako će se ti znakovi slagati zajedno. Ako nam simulacija ne paše, mi ne možemo pomicati univerzalnu desnu liniju, pa radimo iznimke i možemo slova podvući jedno pod drugo. Možemo slovo pomicati za m jedinica, relativna jedinica, jer u programu nemamo centimetre ili inche. Ako želimo napraviti više istih slova, no s različitim kvačicama, kopirat ćemo slovo c i prenijeti na kodnu poziciju gdje želimo dizajnirati novo slovo. Prvo radimo slova koja su slična jedan drugome, ili njihov izgled proizlazi iz sličnog slova. Ako pak radimo rukopisna slova, onda nastojimo što više slova rukom napisati i izaberemo što je tipično za naš rukopis, zatim skeniramo, stavimo na kodno mjesto i vektoriziramo tu našu sliku znaka. Za razliku od prošlosti, danas imamo program u kojem možemo dizajnirati slova bez primjene tonera i boja. U softverima možemo kreirati slova, te ih exportati te vidjeti kako se ona nalaze u koordinatnom sustavu. Konkretno, softver je programiran u C++ programu po parametrima, rezoluciji, širini i visini. U PostScript programu učimo naredbe kako postavljati slova da bi nam kasnije korištenje u Adobe Illustratoru bilo lakše i ravnije. PostScript uređaji su današnji printeri jer imaju mogućnost prenošenja slike na papir, samo što mi sliku simuliramo na ekranu. Nadalje postoji više softvera za neke postavke u grafici koje PostScript stvara i zna. Imamo simulator koji ima puno više parametara. Npr. tekst je postavljen u krug i mi njemu želimo promijeniti rezoluciju ovisno za koji uređaj se radi, nisko ili visokorezolucijski. Ako otvorimo „source“ naučit ćemo koji je koji parametar. Reguliramo kako će ispis izgledati na konačnom digitalnom platnu. Danas postoji puno konstruktora slike, no najčešće koristimo Adobe Photoshop. Kako pokazati nešto što je u svakom vektorskom programu uobičajeno? To su Bezierove krivulje. U Fontographeru vidimo znakove, i uzmemo li slovo o, odnosno okruglo slovo, vidimo točke, to su točke Bezierovih krivulja. Tangente točke se označavaju s plusevima u vektorskoj grafici, da nemamo dileme. Postoji matematički izvod Beziera, koji se sastoji od točaka, prve, zatim natezne i druge točke između kojih nastane krivulja, u drugim softverima je to povezano, pa govorimo o spoju. S obzirom kako pomičemo točke i kako crtamo krivulje, tako se i pomiče jednadžba. Ako točku prebacimo u neki drugi mode, ona nema više jednadžbe, to je nezavisno. Ako točku

prebacimo u treći način, tangetni način, onda je ta točka točno tangenta na tu krivulju. U bilo kojem softveru budućnosti, moramo znati koristiti režime točaka spajanja Bezierovih krivulja, a ne bavit se s čistom upotrebom. Kada radimo neko slovo, mi zapravo radimo predviđanje ponašanja tih točaka u četvercu. Ako radimo neke krivuljne dizajne, bilo da se radi o tipografiji ili klasičnoj izradi rozeta, moramo koristiti beziera. Bezierove krivulje su krivulje trećeg stupnja iz skupine predvidljivih krivulja. To su krivulje gdje se s položajem kontrolnih točaka, odmah se stvara predikcija čovjeku gdje bi te krivulje mogle ići. Dakle bezier je pobjednik svih jednadžbi koje su htjele ući u vektorsku grafiku. Oblik slova kontroliramo koeficijentima u jednadžbi, također i ono što je pod korijenom da ne bi došlo do imaginarnog broja. Bezierova krivulja se počela koristiti kad je Pier Bezier počeo primjenjivati za dizajn haube u Renaultovoj tvornici automobila. Danas je to standard u Illustratoru, u Fontographeru i svim vektorskim programima da bi grafičarima olakšali samo dizajniranje znakova. Kada to upotrijebimo u PostScriptu, te alatke su zamjena za naredbe. Kada imamo Illustrator i kliknemo „save as“ ps ili eps mi smo dali neku naredbu tomu i te naše naredbe su generirane. Programi i PostScript ne razumiju isti jezik pa se za to koristi PostScript Driver. U softveru imamo nacrtan auto, grafiku. U njemu možemo mijenjati načine prikaza kodova i ako nam je neki dio preizbočen, preko koordinatnog sustava i Bezierove krivulje njega možemo popraviti. Otvorimo li tu grafiku u PostScriptu vidjet ćemo neke naredbe. Naredba za PostScript je curveto. Kliknemo na dio koji želimo promijeniti, te vidimo šest točaka, odnosno četiri točke koje se nalaze u Bezierovoj krivulji imaju još i x i y os, no prva točka se smatra kako tekuća točka koju moramo napraviti prvu prije stvaranja prve toče curveto naredbe. Možemo promijeniti koordinate te kada spremimo u PostScript ispisu ćemo vidjeti rezultat. Ako tako krenemo raditi grafike, bolje ćemo shvatiti naredbe i zašto je to tako nacrtano. Ako naučimo raditi preko naredbi u PostScriptu, znat ćemo raditi u ostalim vektorskim programskim jezicima. SVG je jezik za prikazivanje dvodimenzionalne vektorske grafike, bilo nepomične, bilo animirane. SVG je jezik iz porodice XML jezika. Svojstvo vektorske grafike je da nije vezano za rezoluciju. Vezana je samo za moment ispisa kad nešto prikazujemo. Kad renderiramo, animaciju možemo povećavati koliko želimo, ali se neće uništiti prikaz. U piksel grafici npr. Photoshopu, rezolucija je zadana i možemo ju samo resemplirat ali to zamućuje sliku i gubimo informacije. Ako uđemo u text editor vidimo kodove, i ti kodovi su jako dobra ulaznica za učenje animacije općenito. Kako ju savladati i iskodirati. Sljedeća tema je Bezier u animaciji. Na slici vidimo animaciju staze po kojoj putuje objekt. Kako se definira ta staza, kad želimo da je vidljiva. Na taj način možemo raditi razne animacije fizikalnih pokusa, npr. kako klizi tijelo niz kosinu, kako ubrzava ili usporava, jer u softveru imamo sve parametre za trajanje ove animacije. Imamo sve parametre gdje možemo sve fizikalne pojave i pokuse animirati, uostalom tako se i oni rade. Da bi približili to studentu, uvijek počinjemo od jednostavnih stvari, da prvo naučimo što je Bezierova krivulja, uvedemo ju u PostScript i uđemo u softver, tehnologije, čije se slične naredbe koriste, zato je to kompatibilno. Zato je važno kad se bavimo informacijskim tehnologijama, ne skočit na najtežu tehnologiju, nego ić bazno, da kasnije možemo na bilo koju tehnologiju skočiti kad god želimo. Danas je moderno znati skriptne jezike, ali skriptni jezik bez ovog predznanja za

ovu grafiku nema nikakvog smisla. Uvijek ćemo učiti iznova, zato je najbolje učiti na ovaj način kakvu metodologiju pruža grafički fakultet. Kod naredbi, kad se nauči da treba imati šest brojki za curve to, tada je u svg-u slično jer je Adobe napravio taj sustav da Bezier uvijek kreće od tekuće točke pa napravimo ove dodatne. Kad dizajniramo stazu, onda radimo objekt, taj žuti trokutić i onda radimo animaciju po toj stazi. Ili ako ne želimo onda si ovaj vidljivi dio ogradimo, da ga ne vidimo, onda ponovo startamo i nećemo vidjeti taj vidljivi dio. Ako želimo ubrzati animaciju, to je samo jedan atribut svg-a od xml elementa, animate motion. Možemo ubrzati da animacija prijeđe put za 5 sekundi. Zatim vidimo jednu kružnicu koja u sebi ima mnogo kružnica, zapravo jednu smo vrtili po petlji od 0° do 360°, ali mijenjamo boju. I prvo se moramo upoznati s kodom koji to generira. Prvi dio ide u interface, a iz ostalog iščitavamo dijelove, mala kružnica, velika kružnica itd. Cijela stvar se odigrava u petlji, cijeli broj koraka. Kako mi mijenjamo naredbe, mi naređujemo oblicima kako da se mijenjaju i vidimo trenutne promjene. Da bi znali promijeniti boju, moramo znati kolarne sustave. Da bi učili kolarne sustave, moramo položiti nekoliko kolegija, najpraktičnije je kad taj kolarni sustav upotrijebimo na nečem konkretnom. Ovdje smo koristili sethsbcolor sustav. Moramo znati koje domene zahtjevaju određene boje. Ako želimo posebnu boju za svaku kružnicu moramo raditi petlju. Npr. ovo koloriranje možemo koristiti za umjetno obojenje crno-bijelih filmova. Za umjetno koloriranje, moramo znati koji dio kruga je zelena i dobit ćemo taj zeleni dio. Zatim prelazimo na rastriranje. Rastriranje je čovjek izmislio samo zato da bi s jednom bojom mogao napraviti n nijansi. Na ekranu vidimo nekolko nijansi sive boje, u tisku je to malo teže dobiti pa gledamo rastere. Rasteri nam prikazuju istu boju samo tamnije oko doživljava ono gdje su točkice gušće, a svjetlije je ono gdje su točkice rjeđe. Kod amplitudno modularnih rastera imamo jednake razmake između točkica, odnosno elemenata, a razlika u tome je što su elementi veći i manji. Što se ta točkica smanjuje to je naše ljudsko oko bolje prevareno. Na velikim plakatima su točke velike, no mi to gledamo na velikoj udaljenosti i zato nam se čini kao savršena slika. Za ovakvo stvaranje nam je potrebna matematika, odnosno formule. Kada znamo određene matematičke formule onda ih trebamo samo prebaciti u određeni jezik, konkretno u PostScriptu. Na fakultetu sami ponude profesori neke formule i kad studenti shvate određeni mehanizam onda ih oni sami mogu mijenjati. Sinusni raster koji je na kunama je napravljen na grafičkom fakultetu. Oni se moraju nalaziti u domeni od -1 do 1. Da li je nešto krivotvorina ili ne, stavimo u rasterski element. Kada govorimo o digitalnoj boji, boji koja je simulirana na ekranu, boji koju želimo ostvariti u tiskarskoj tehnologiji, ceradi automobila. Dok smo na ekranu mi smo u rgb tehnologiji, u tiskovnom dijelu ona ne postoji, dakle nema crvenu, zelenu i plavu. Na grafičkom fakultetu ne nastaju samo klasični grafički inžinjeri, nego inžinjeri multimedije i vizualnih komunikacija, netko ko zna čitati jezik u PostScriptu. U HTML-u nema niti hsb ni cmyk sustava. Moramo se staviti u rgb sustav. Usmjereni smo na ekran. Nacrtamo sliku u Illustratoru, spremimo ga kao .svg i stavimo ga u html kod da se prikaže na određenom dijelu. Word poznaje samo rgb color sustav. PDF je postao standard za tisak, za komunikaciju, standard čitanja. HTML i PDF oba prikazuju tekst, slike, no PDF može cmyk color sutav i hsb, PDF poznaje pojam stranice, a HTML ne.