Dr Selma Rizvić

Kompjuterska grafika i multimedia

U doba sve bržeg razvoja kompjuterske grafike i njene zastupljenosti u mnogim oblastima života, potrebno je na jednom mjestu objediniti znanja koja su potrebna za bavljenje profesijama koja imaju dodira sa ovom tematikom.

Grafički dizajneri, televizijski dizajneri, animatori, dizajneri grafičkih interfejsa, web programeri, inžinjeri grafičkog softvera, sve su to profesije koje imaju u određenom smislu dodira sa kompjuterskom grafikom.

Cilj ove knjige je da pruži osnovna znanja iz nekoliko oblasti koje su nezaobilazne ako se neko želi baviti bilo kojom od ovih profesija. Potrebno je poznavati osnove grafičkog dizajna, filmskog jezika, web dizajna i programiranja, kao i primjenu svih tih znanja u multimedijalnom okruženju.

Ova knjiga bi trebala služiti i kao udžbenik za studente III godine Odsjeka za informatiku ETF-a u Sarajevu za predmet «Računarska grafika»

Sadržaj

Uvod

Kompjuterska grafika danas nije samo oblast koja izučava algoritme za crtanje grafičkih primitiva ili rutine za programiranje grafičkih interfejsa. Ljudska komunikacija je zadnjih godina doživjela ekspanziju pojavom Interneta i način obraćanja sve više postaje vizuelan. Internet je objedinio sve do sada postojeće medije u jedan univerzalni, svima dostupan medij, koji se korisniku obraća jezikom kompjuterske grafike.

Oblasti web dizajna, televizijskog dizajna, kompjuterske animacije i virtuelnih 3D svjetova postaju sastavni dio kompjuterske grafike i oličenje njene multimedijalnosti.

Za proučavanje kompjuterske grafike kao vizuelnog prikaza informacija na kompjuterskom ekranu, neophodno je poznavati principe na kojima rade ti ekrani. U poglavlju Kompjuterski displej sistemi upoznajemo se sa tim principima.

U poglavljima Rasterska grafika i Vektorska grafika upoznajemo se sa osnovnim pojmovima za cijelu oblast kompjuterske grafike kao što su piksel, rezolucija slike, dubina boje, formati digitalnog zapisa slike, koncept Kartezijanskih koordinata, koordinatni sistem kamere.

Četvrto poglavlje prezentira osnove HTML-a, jezika u kojem se pišu web stranice. Ovdje su opisani i koncepti rada Internet stranica na relaciji korisnik-web server.

U sljedećem poglavlju nalaze se osnovi web dizajna, koji će omogućiti čitaocu da na pravi način dizajnira svoje web stranice uspostavljajući ravnotežu između jasnoće prezentirane informacije i dobrog grafičkog izgleda stranice.

Poglavlje koje opisuje principe grafičkog dizajna pruža čitaocu mogućnost da dizajn njegovih web stranica, animacija ili video klipova bude u okviru jasno definisanih principa koje su dosada poznavali samo grafički dizajneri. Poštivanje ovih principa vidljivo će unaprijediti dizajn svake kreacije u okviru kompjuterske grafike.

Autori kompjuterskih animacija svjesni su da se ovaj posao ne može kvalitetno raditi bez poznavanja osnova filmskog jezika i kompozicije kadra jer svaka animacija po svojoj strukturi predstavlja film relativno kratkog trajanja. Sljedeće poglavlje prezentira osnovne pojmove iz filmskog jezika i nekoliko pravila za kompoziciju kadra.

3D modeliranje, kompjuterska animacija i web integrirali su se u oblasti virtuelnih svjetova (ponegdje pod nazivom web 3D). Osmo poglavlje sadrži punu specifikaciju jezika za modeliranje virtuelnih svjetova VRML.

Televizijski dizajn (broadcast design) već odavno čini sastavni dio kompjuterske grafike. U današnje vrijeme nezamisliv je televizijski program bez velikog broja grafičkih elemenata. Snalaženje u ovom tehnički veoma komplikovanom mediju

olakšava poznavanje osnovnih koncepata televizijske produkcije koje nudi poglavlje devet. Nakon ovog poglavlja slijedi pregled formi televizijskog dizajna sa odgovarajućim primjerima.

Poglavlja Načini modeliranja i Kompjuterska animacija sadrže matematički aparat koji je implementiran u softverskim paketima za 3D modeliranje i animaciju i nudi široki spektar tehnika modeliranja različitih tipova objekata i njihovo osvjetljavanje raznim vrstama svjetala. Ovdje su prezentirani i osnovni koncepti kompjuterske animacije.

Poglavlje OpenGL grafičko programiranje daje kratak pregled procesa razvoja kompjuterske igre koristeći biblioteku OpenGL.

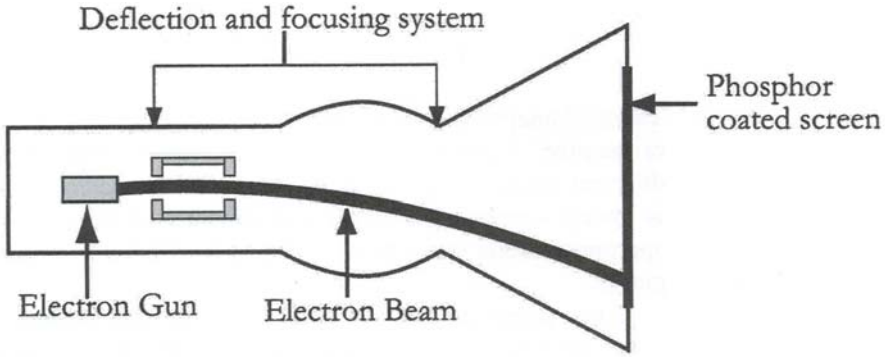
.

.

1. Kompjuterski displej sistemi

1.1. Katodna cijev (CRT)

Kompjuterski displej ili monitor je najvažniji uređaj na kompjuteru. On omogućava vizuelni izlaz iz kompjutera prema korisniku. U kontekstu kompjuterske grafike, sve se nalazi na displeju.
Jedna od najvažnijih tehnologija za gradnju kompjuterskih dipleja je CRT (Cathode Ray Tube) ili katodna cijev.

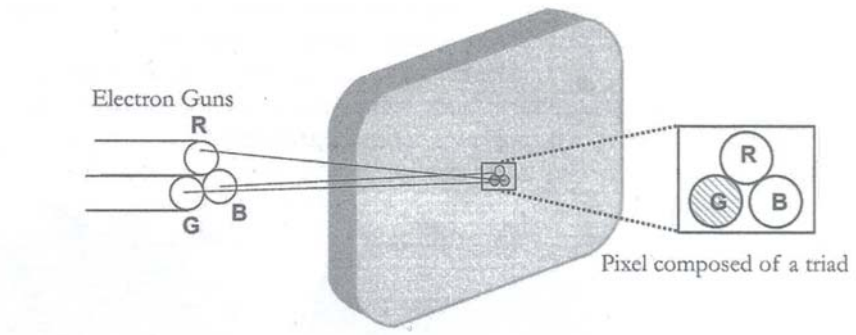


slika 1. katodna cijev

Kao što je prikazano na slici 1. katodna cijev se sastoji od

- elektronskog topa koji emituje snop elektrona (katodne zrake)
- deflection i fokusing sistema koji usmjerava fokusirani snop elektrona prema određenoj poziciji na fosforom premazanom ekranu
- fosforno premazanog ekrana koji emituje malu tačku svjetla proporcionalnu intenzitetu snopa koji ga pogađa

Svjetlo koje emituje ekran je ono koje vidimo na monitoru.
Tačka koja može biti osvijetljena elektronskom snopom naziva se pixel. Intenzitet svjetla koje se emituje može se mijenjati variranjem broja elektrona koji pogađaju ekran. Veći broj elektrona daće svjetliju boju na mjestu odgovarajućeg pixela. Crno-bijeli monitor ima samo jedan fosfor za svaki pixel. Boja pixela može biti postavljena na crnu (nijedan elektron ne pogađa fosfor), bijelu (maksimalni broj elektrona pogađa fosfor) ili na bilo koju vrijednost između.
Kolor CRT monitor ima 3 različita obojena fosfora za svaki pixel. Svaki pixel ima crveni, zeleni i plavi fosfor uređen u trougaonu grupu. Postoje tri elektronska topa i svaki od njih generiše snop elektrona da pobudi jednu od fosfornih tačaka kao što je prikazano na slici 2. Zavisno od proizvođača monitora, pixeli mogu biti okrugle tačke ili mali kvadrati.



slika 2: Kolor CRT koristi crvenu, zelenu i plavu trijadu

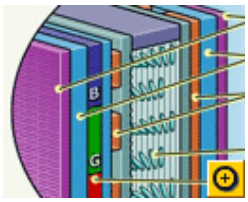
Kako se tačke nalaze međusobno veoma blizu, ljudsko oko ih fuzionira u jednu tačku koja je kolor kombinacija tri osnovne komponente boje (R,G i B).

1.2. LCD displeji (Liquid Crystal Display)

U zadnje vrijeme sve češće se koristi LCD tehnologija u proizvodnji kompjuterskih displeja. LCD ili liquid crystal display je tehnologija koja proizvodi slike na ravnoj površini odsjajem svjetla kroz tečni kristal i kolor filtere

Ovi displeji zauzimaju manje prostora, troše manje energije i proizvode manje topline od klasičnih CRT monitora.

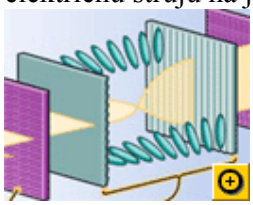
LCD display



Otkriveni 1888, tečni kristali su tečne hemijske supstance čije molekule mogu biti precizno poravnate kada su izložene električnim poljima. Kada su poravnati na odgovarajući način, tečni kristali propuštaju svjetlo.

LCD ekran je višeslojan. Fluorescentni izvor svjetla koji se zove backlight izgleda kao kriška hljeba. Ovo svjetlo prolazi kroz prva dva polarizirajuća filtera. Polarizovano svjetlo zatim prolazi kroz sloj koji sadrži hiljade mjehurića tečnog kristala poredanih u majušne kontejnere koji se nazivaju ćelije. Ćelije su poredane u redove preko ekrana, jedna ili više ćelije čine jedan pixel. Električni vodovi oko ivice LCD ekrana kreiraju električno polje koje uvrće molekule kristala i poravnava svjetlo sa drugim polarizirajućim kristalom i omogućava mu prolaz.

Kod kolor LCD ekrana, svaki pixel je načinjen od tri ćelije tečnog kristala. Svaka od te tri ćelije ima ispred sebe crveni, zeleni ili plavi kolor filter. Svjetlo koje prolazi kroz ćelije sa filterom kreira boje koje vidimo na ekranu. Povremeno mehanizam koji salje električnu struju na jedan ili više pixela nije aktivan, pa se vidi kompletno crn pixel.



Skoro svi moderni kolor LCD-ovi koriste tanki film trazistor, poznat kao aktivna matrica, za aktiviranje svake ćelije.
Kako LCD adresira svaki pixel individualno, oni imaju oštriju sliku od CRT-ova, koji, kad nisu dobro fokusirani, bluriraju različite pixele u slici.

1.3. Frame baferi

Svjetlost koja se na ekranu generiše pomoću snopa elektrona na CRT nestaje brzo – za 10 do 60 mikrosekundi. Da bismo sliku na ekranu zadržali izvjesno vrijeme, ona mora biti ponovo iscrtana prije nego što nestane sa ekrana. Ovo se zove osvježavanje ekrana (refreshing). Većina displej sistema koriste raster scan tehnologiju da bi izvršili proces osvježavanja. U ovoj tehnologiji, snop elektrona se diskretno usmjerava preko ekrana, jedan po jedan red, s desna na lijevo, počevši od gornjeg lijevog ugla ekrana. Kada snop dostigne najdonji red, proces se ponavlja, osvježavajući ekran.
Raster scan sistemi koriste memorijski bafer koji se zove frame bafer (ili refresh bafer) u komese smještaju intenziteti pixela. Osvježavanje ekrana se vrši korištenjem informacije koja je smještena u bafer. Frame bafer možemo zamisliti kao dvodimenzionalni niz. Svaki element niza čuva intenzitet pixela na ekranu koja odgovara njegovoj poziciji.

Za monohromske displeje, frame bafer ima samo jedan bit za svaki pixel. Displej kontroler stalno čita iz frejm bafera i usmjerava elektronski snop samo ako je bit u baferu osvijetljen.

Double buffering

Postoje dva bafera za crtanje po ekranu, Jedan bafer, tzv. foreground bafer se prikazuje na ekranu. Drugi bafer, background bafer se koristi za čuvanje slike. Kada je slika kompletna, dva bafera se zamjenjuju, tako da se onaj koji je prikazivao sada koristi za crtanje i obratno. Zamjena se vrši skoro trenutno. Kako je slika već iscrtana u trenutku prikazivanja na ekranu, rezultujuća animacija izgleda glatko i ne vide se nekompletne slike.

2.Rasterska grafika

Prema osnovnim gradivnim elementima slike, kompjuterska grafika se može podijeliti na

- rastersku grafiku
- vektorsku grafiku

Kod rasterske grafike osnovni gradivni elemnti slike su **pixeli**, a kod vektorske **objekti.**

Prema izvorima u knjizi Rona Brinkmanna "The Art and Science of Digital Compositing", slika se može podijeliti u pravilno raspoređene elemente fiksne veličine koji se nazivaju **pixeli**. Za svaki od tih elemenata određena je boja ili tonalitet (nivo osvijetljenosti). Ljudsko oko ne primjećuje pixele koji su dovoljno mali i gusto postavljeni nego ih veže u kontinualnu sliku. Ova osobina ljudskog oka se naziva **prostorna integracija** i ima veoma značajnu ulogu u digitalizaciji slike, ne samo u računarskoj grafici, nego i u medijima kao što su fotografija i filmska traka koje obično smatramo analognim i kontinualnim.

Parameri pixela koji su potrebni da bi se slika zapisala na digitalnom mediju su njegova **pozicija** i **vrijednost boje** ili tona.

Nadalje, svakoj boji koju ima pixel može se dodijeliti jedinstven broj. Ova informacija se predstavlja, u skladu sa brojem boja koje se žele prikazati, pomoću određenog broja bita. Tako se jednim bitom mogu opisati dvije boje.
Broj bita koji se koristi za opis boje naziva se **dubina boje**. Ako želimo prikazati veći broj boja, to zahtijeva i veći memorijski prostor za prikazivanje. Danas su u upotrebi sljedeći formati rasterskih slika

Dubina boje	Broj boja koje se mogu prikazati
1	2
4	16
8	256
16	64K
24	16M
32	4G

Kada se smještaju u računar, rasterske slike postaju nizovi podataka o boji pixela. Rekonstrukcija takve slike podrazumijeva interpretaciju na isti način kao što je vršeno pohranjivanje. To znači da se treba zabilježiti i način pohranjivanja. Na ovaj načIn se izbjegava pamćenje lokacije svakog pojedinog pixela. Obično se slika sprema red po red, s lijeva nadesno i odozgo prema dolje. Ovi redovi pixela zovu se **raster-scan linije**. Da bi se rekonstruisala ovako spremljena slika potrebno je znati:

- dužinu raster-scan linije (broj pixela u rasteru po horizontali)
- broj raster-scan linija (broj pixela u rasteru po vertikali)
- dubinu boje

Prilikom štampanja rasterske slike važna veličina je i dimenzija rastera u jedinicama dužine koja se zove **rezolucija**. Rezolucija je broj pixela u rasteru po jedinici dužine. Ona se načešće izražava u "dpi" (dots per inch – broj tačaka po inču).

Ako sliku posmatramo primarno kroz karakteristike pixela, a sekundarno kroz način njihovog kombiniranja u kreiranju slike, vidjećemo kolor sliku kao kolekciju nivoa (layer-a) jednostavnijih slika, odnosno **kanala.**

Boja pixela je funkcija od tri komponente – crvene, zelene i plave (Red, Green, Blue). Kombinacijom različitih intenziteta ove tri komponente dobijamo čItav spektar boja za svaki pixel. Ako posmatramo jednu komponentu boje (npr. crvenu), svakog pixela u slici, dobićemo specifični kanal kompletne slike. Tako možemo svaku sliku posmatrati kao kombinaciju crvenog, zelenog i plavog kanala.

Digitalni formati zapisa slike

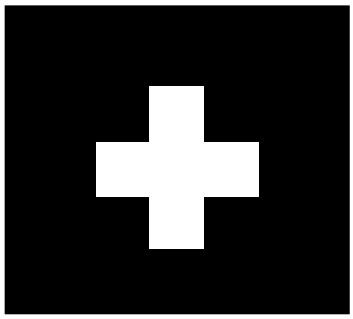
Kada smo digitalizirali sliku u memoriji računara, moramo naći način kako ćemo tu sliku zapisati na disk. Postoji veliki broj različitih formata koje možemo izabrati. Svaki od tih formata ima svoje mane i prednosti. Različiti formati variraju u mogućnostima da podrže pojedine važne osobine pa ćemo u nastavku iznijeti neke karakteristike pojedinih formata.

Kompresija

Kako slike velike rezolucije mogu zauzeti veliki prostor na disku, često je potrebno da se oni kompresuju. Postoji određeni broj tehnika za kompresiju, od kojih neke narušavaju kvalitet slike, a neke ne. Ako šema za kompresiju koja se koristi za smještanje slike može biti potpuno obrnuta , tako da je dekompresovana slika digitalno identična kompresovanoj, kaže se da je šema za kompresiju **bez gubitaka**. Ako je to nemoguće, radi se o kompresiji sa gubicima. Skoro uvijek postoji mogućnost da se zamijeni kvalitet slike za efikasnost u prostoru. Međutim, postoji nekoliko metoda za kompresiju bez gubitaka i jedan od njih je tehnika poznata kao **run-length encoding (RLE)**

Run-length encoding

RLE je tehnika koja omogućava značajan nivo kompresije. Razmotrimo sliku koja ima bijeli plus na crnoj pozadini.



Slika je izuzetno male veličine (12x12), uvećana da bi se vidjeli pojedinačni pixeli. Također, da bismo pojednostavili razmatranje, smatraćemo da je slika crno-bijela, što znači da je potreban samo jedan bit informacije da odredimo boju pixela. U ovom slučaju, 0 će biti crni, a 1 bijeli pixel. Na taj način numerička reprezentacija ove slike izgleda ovako:

0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0
0 0 1 1 1 1 1 1 1 1 0 0
0 0 1 1 1 1 1 1 1 1 0 0
0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0

Kao što vidimo, koristimo 144 karaktera u obliku matrice da predstavimo ovu sliku.

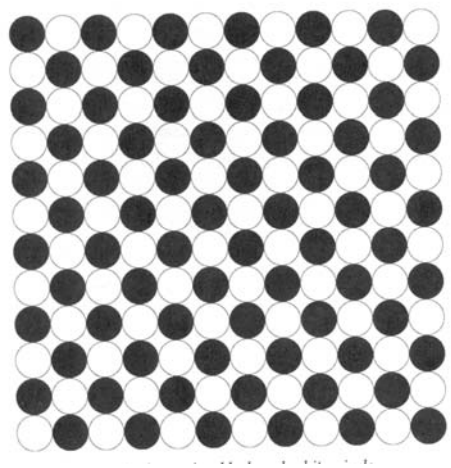
RLE radi na principu analiziranja slike i zamjenjivanja redundantnih informacija sa efikasnijom reprezentacijom. Npr, prva linija naše slike je je string od 12 nula. Ovo se može zamijeniti sa jednostavnom notacijom 12:0. Sada smo sveli opis prve linije sa 12 karaktera na 4. Koristeći ovu konvenciju, možemo zakodirati cijelu sliku na sljedeći način:

12:0
12:0
5:0, 2:1, 5:0
5:0, 2:1, 5:0
5:0, 2:1, 5:0
2:0, 8:1, 2:0
2:0, 8:1, 2:0
5:0, 2:1, 5:0
5:0, 2:1, 5:0
5:0, 2:1, 5:0
12:0
12:0

Totalni broj karaktera koje smo koristili za predstavljanje slike je sada 104. Reducirali smo količinu informacija za smještanje slike skoro za 30%, a originalna slika se može restaurirati bez ikakvih gubitaka.

Odmah se može primijetiti da ovaj metod ne proizvodi uvijek istu količinu kompresije u zavisnosti od slike. Ako je slika potpuno bijela, količina kompresije bi bila

ogromna, ali ako imamo sliku kao što je sljedeća, ova šema za kompresiju bi dala sljedeće rezultate



1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1
1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0
1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1
1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0
1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1
1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0
1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1
1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0
1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1
1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0
1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1
1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0, 1:1, 1:0

Na ovaj način bismo morali koristiti 542 simbola da predstavimo originalna 144.

Poenta ovog primjera nije da dokažemo nešto specifično o RLE kompresiji, nego da istaknemo činjenicu da svi metodi kompresije ne rade jednako dobro na svim vrstama slika. Pokazalo se da je RLE kompresija pogodna za slike koje su generisane pomoću paketa za 3D rendering, jer ova vrsta slika često ima veliki broj pixela identične boje. S druge strane, RLE kompresija nije dobar metod kada su u pitanju slike digitalizirane sa analognog izvora, kao što je film ili video, jer je kod tih slika skoro svaki pixel različite boje.

Kompresija sa gubitkom

Postoji veliki broj formata koji mogu drastično smanjiti prostor potreban za smještanje slike, ako je korisnik spreman da prihvati određeni gubitak u kvalitetu slike. Vjerovatno najpopularniji od ovih formata je onaj koji koristi šemu kompresije koju je definisao Joint Photographic Experts Group. Za ovakve slike se kaže da su smještene u **JPEG** formatu. Jpeg format ima veliki broj prednosti. Najprije, on radi

posebno dobro nad slikama koje potiču sa filma ili videa, dijelom zbog toga što su artefakti koje on proizvodi dizajnirani tako da se manje primijete u područjima koje imaju određenu količinu šuma. Posebno, JPEG kompresija pokušava da održi informaciju o svjetloći i kontrastu (na koje je ljudsko oko posebno osjetljivo), na račun određene definicije boja. Ovaj format također ima prednost da korisnik može odrediti nivo kompresije. Zato se može dobro naći kompromis između kvaliteta slike i količine prostora potrebne za njeno smještanje.

Vrste formata

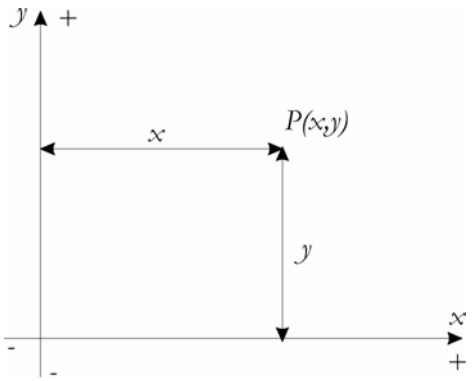
U nastavku su date neke osnovne vrste formata zapisa slike i njihove karakteristike.

- GIF format - dubina boje 8 bita, pogodan za slike koje se prezentiraju na Internetu
- animirani GIF format - isti kao GIF samo što čuva pokret, pogodan za animacije na web stranicama
- JPEG format - univerzalni format, nudi korisniku da izabere kvalitet i kompresiju slike, pogodan za sve primjene
- Targa format - dubina boje do 32 bita, pogodan za štampane materijale, veliki fajlovi
- TIFF format - sličan kao Targa, koristi se u pripremi za štampu, veliki fajlovi
- AVI format - čuva pokret, koristi se za animacije i za zapis digitalizirane slike sa kamere ili video trake
- MPEG format - čuva pokret, koristi se u iste svrhe kao AVI, fajlovi manji od AVI-ja, ali lošijeg kvaliteta
- QuickTime format - čuva pokret, koristi se kod obrade slike na Macintosh račUnarima

3. **Vektorska grafika**

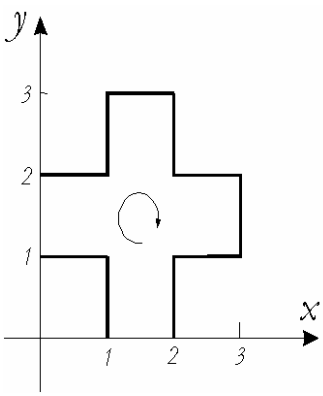
Koncept Kartezijanskih koordinata

Zajednički za cijelu kompjutersku grafiku je koncept Kartezijanskih koordinata, definisan u knjizi "Computer Graphics, Principles nad Practice", od Foley-a i Vam Damm-a. U 2-D slučaju ovo omogućava da se tački na ravnoj površini pristupa uz pomoć horizontalne i vertikalne ose. Tačka se locira mjerenjem dvije udaljenosti paralelne sa osama od njihove presječne tačke koja se zove koordinatni početak. Horizontalne i vertikalne mjere za bilo koju tačku su jedinstvene i zovu se x i y koordinata respektivno.



Slika 1 - x i y koordinate tačke P određene su horizontalnim i vertikalnim udaljenostima tačke P od koordinatnog početka

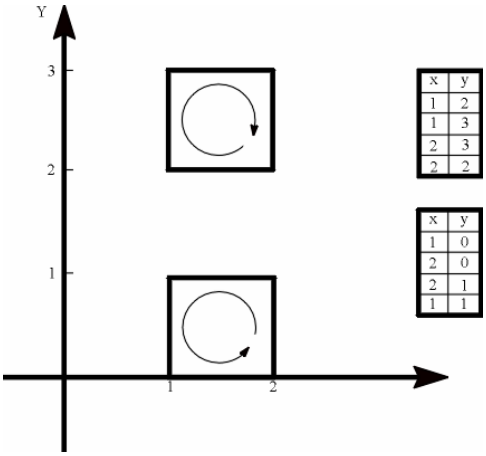
Slika.1. ilustrira ovu šemu i takođe pokazuje konvenciju za pozitivni i negativni pravac.



x	y
0	1
0	2
1	2
1	3
2	3
2	2
3	2
3	1
2	1
2	0
1	0
1	1

Slika 2 - Ilustriran je oblik sa 12 vrhova; njihove koordinate se nalaze u tabeli.

Bilo koji 2-D oblik se može predstaviti pomoću sekvence tačaka ili vrhova kao što je prikazano na slici 2, i kako se se svaki vrh sastoji od uređenog para brojeva (x i y koordinata) oni se mogu lahko smjestiti u kompjuter.



Slika 3

Ova dva kvadrata su digitalizirana u pravcu kazaljke na satu i obrnuto

Međutim, naša predstava o ovom obliku je vizualna i uspostavljena je povlačenjem njegove granice u pravcu kazaljke na satu ili obrnuto. Pravac granice se takođe mora čuvati u kompjuteru uređivanjem koordinata vrhova u jednom od dva niza kao što je prikazano na slici 3

Dvo-dimenzionalne koordinate su grupisane u zagrade kao npr (2.5, 1.5), gdje se 2.5 i 1.5 odnose na x i y koordinate respektivno. Postoji nekoliko korisnih posljedica ove koordinatne notacije: prvo, **skaliranje** oblika se može vršiti na sljedeći način:

$x' = rx$
$y' = ry$

gdje je (x,y) vrh koji je skaliran za r da bi kreirao (x', y'). Drugo, ako se x koordinata poveća za jednu jedinicu, nove koordinate su ekvivalentne istom obliku pomjerenom ili transliranom jednu jedinicu udesno od originalnog oblika. Međutim, oblik se može pomjeriti i u x i u y pravcu, pa je operacija translacije sumirana ovako:

$x' = x + u$
$y' = y + v$

gdje je (x,y) vrh koji je pomjeren za (u,v) na njegovu novu poziciju (x',y').

Većina kompjutera može izvršavati nekoliko miliona aritmetičkih operacija u sekundi pa je zato sposobna za skaliranje i translaciju velikih skupova koordinata skoro istovremeno. Ta karakteristika ih čini tako snažnim alatkama u manipuliranju oblicima.

Osim skaliranja i translacije, operacija **rotacije** oko koordinatnog početka je takođe važna i postiže se pomoću sljedeće formule:

$x' = x\cos(\vartheta) - y\sin(\vartheta)$
$y' = x\sin(\vartheta) + y\cos(\vartheta)$

gdje je (x,y) tačka koja je rotirana za ugao ϑ u njenu novu poziciju (x',y'). Rotacija je obrnuta od smjera kazaljke na satu kada je ϑ pozitivno i obrnuto. Da bi se rotirao kompletan oblik za ϑ, prvo se izvedu kosinusne i sinusne funkcije da bi se proces reducirao na četiri množenja, jedno sabiranje i jedno oduzimanje za svaki vrh. Sinusne i kosinusne funkcije se izračunaju jednom prije procesiranja koordinata.

Smicanje je također korisna transformacija i računa se na sljedeći način:

$x' = x + y\tan(\vartheta)$
$y' = y$

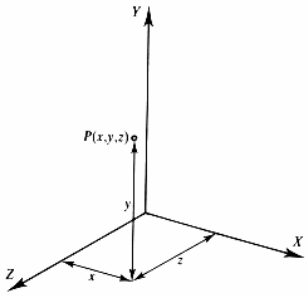
gdje je (x, y) tačka koja je smaknuta za ugao ϑ na njenu novu poziciju (x', y').

Ove četiri operacije - skaliranje, translacija, rotacija i smicanje - formiraju osnovne operacije za manipulaciju objektima u 2-D grafici i imaju kompaktan opis pomoću matrične notacije.

Proširivanje Kartezijanskih koordinata u tri dimenzije zahtijeva da se svakoj tački u 3D prostoru pristupa preko tri koordinate. 3D koordinate se prikazuju kao perspektivna projekcija 2D koordinata.

Definiraju se tri ose čiji presjek se naziva koordinatni početak.

Za opis pozicije objekata koristićemo desno orijentisani koordinatni sistem koji ćemo nazivati **world coordinate system** (WCS). Svaka tačka u 3D prostoru je locirana pomoću tri koordinate (x, y, z) koje predstavljaju paralelne udaljenosti duž tri ose (slika 4) i objekti se mogu konstruisati kao kolekcije poligona čiji se vrhovi definišu pomoću ove koordinatne notacije. Ako je ravni poligon kreiran iz odgovarajućeg lanca ravnih ivica, orijentacija granice poligona zavisi od toga sa koje strane se gleda na tu granicu. Usvojena je konvencija o kojoj ćemo detaljnije govoriti u poglavlju koje opisuje načine modeliranja.

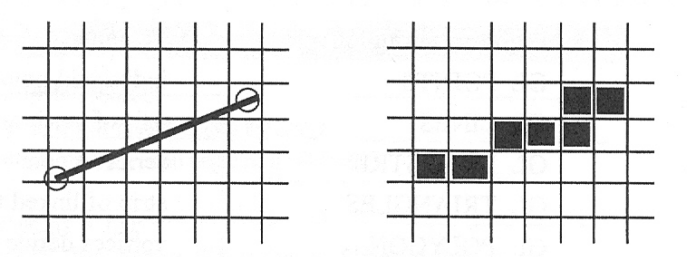


slika 4

Opet su uvedene operacije skaliranja, translacije, rotacije i smicanja koje se baziraju na homogenim koordinatama i opisuju pomoću matrica.

Scan konverzija

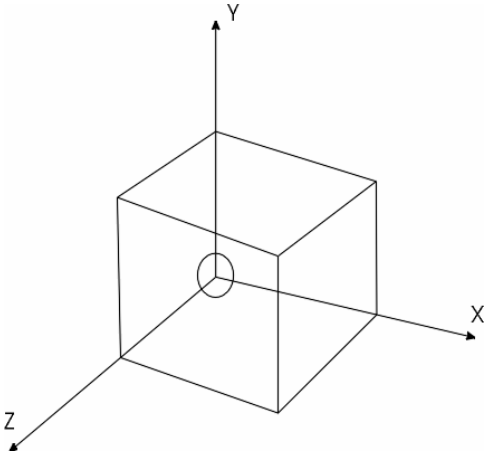
Proces u kojem se idealizirani oblik, kao npr. linija ili krug, iscrtava na ekranu aktiviranjem odgovarajućih pixela, zove se **scan konverzija** ili **rasterizacija**.
Tokom godina razvijeno je nekoliko algoritama za scan konverziju osnovnih geometrijskih oblika u nastojanju da se ovaj proces odvija što jednostavnije i brže.
Najpopularniji algoritam za iscrtavanje linije je **midpoint-lin**e algoritam. Ovaj algoritam uzima x i y koordinate krajeva linije kao ulaz i zatim računa parove x,y koordinata svih pixela između. Algoritam prvo računa fizičke pixele za svaku krajnju tačku. Zatim se povlači idealna linija povezujući krajnje pixele i koristi kao referenca za određivanje koji pixeli se trebaju upaliti tokom njene dužine. Pixeli koji leže manje od 0.5 jedinica od linije se pale, kao što je prikazano na slici.



Osnovni linearni oblici kao što su trouglovi i poligoni mogu se definisati pomoću serije linija. Poligon se definiše kao n vrhova povezanih linijama, kao što je objašnjeno u prethodnom poglavlju.

Koordinatni sistem kamere

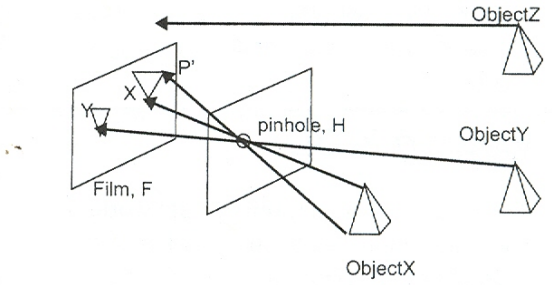
Prema definiciji J. Vince-a u "3D Computer Animation", vizualni izlaz iz sistema za animaciju je ono što se efektivno može vidjeti kroz **kameru**. Kako kamera fizički ne postoji, nego je definirana pomoću nekoliko brojeva, ona se može pozicionirati bilo gdje u 3D prostoru koji opisuje naš virtualni kompjuterski svijet, čak i unutar objekata. U ovom momentu nam odgovara da postavimo kameru u koordinatni početak i usmjerimo je duž pozitivne z-ose kao što je prikazano na slici 5



slika 5

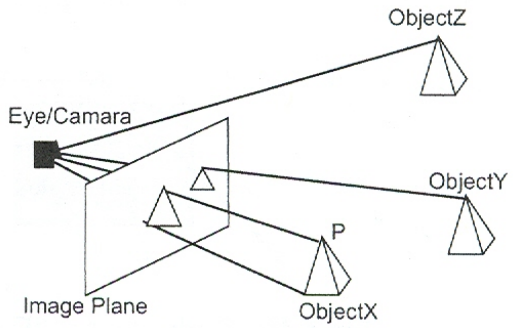
Pinhole kamera

Osnovni cilj u 3D kompjuterskoj grafici je da damo korisniku utisak gledanja fotografije 3D scene, na isti način kako fotografi mapiraju realni 3D svijet na 2D film. Trodimenzionalna kompjuterska grafika simulira princip rada realne kamere u kreiranju 3D slika. Važno je najprije razumjeti kako realna kamera snima slike 3D svijeta koji nas okružuje koristeći jednostavni primjer pinhole kamere. Zatim ćemo proširiti izlaganje na kompjutersku grafiku.



slika 6

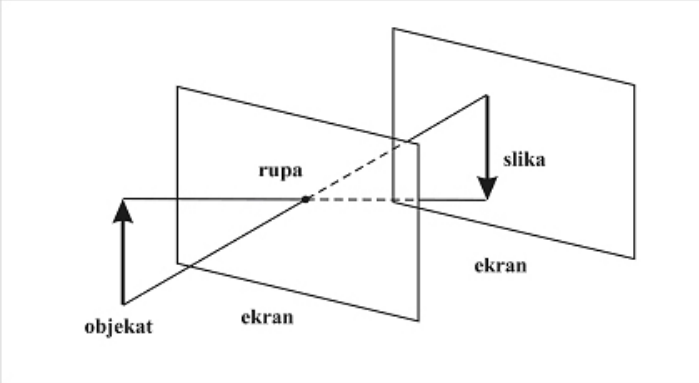
Slika 6 prikazuje pinhole kameru. To je jednostavni box sa fotografskim filmom F pozadi i malom rupom H koja propušta svjetlo kada se otvori.
Razmotrimo šta se dešava kada pravimo fotografiju objekta X. Otvara se rupa H za djelić sekunde. Zraka svjetla pogađa X u tački P, prolazi kroz H i pogađa film u tački P`, uzrokujući eksponiranje filma u toj tački. Tačka P` formira sliku tačke P na negativu. Sve tačke objekta se mapiraju na film na ovaj način. Drugi objekat Y, iste veličine ali dalje po poziciji, mapira se na Y` i izgleda manji po veličini. Objekti kao što je Z, čije zrake svjetla ne prolaze kroz H, nije fotografisan. Primijetimo da je formirana slika invertovana. U kompjuterskoj grafici film se stavlja ispred pinhola. Ovakva postavka osigurava da je slika okrenuta na pravu stranu. Pinhole možemo nazvati eye, viewpoint ili pozicija kamere, a film image plane, kao što je prikazano na slici 7.



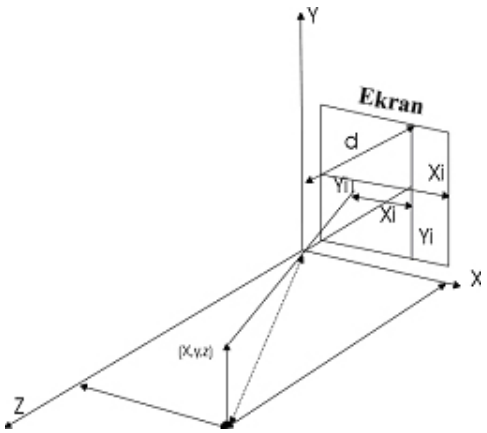
slika 7

Proces slikanja scene ostaje manje-više isti. Zrake (zvane projectors) se generišu od viewpointa prema objektu. Objekti za koje te zrake presijecaju image plane biće procesirani i prikazani na ekranu. Ostali, kao što je Z biće odsječeni iz slike.
Image plane se može posmatrati kao da se sastoji od dvodimenzionalne mreže koja sadrži kolor informaciju mapiranu na pixele ekrana pomoću viewing transformacija.

Najjednostavniji model kamere je tzv. pinhole kamera. Ona se sastoji od dva ekrana. Na prvom ekranu postoji mala rupa. Kroz tu rupu prolaze neke od zraka svjetla koje se emituje ili je odbijeno od objekat. Te zrake formiraju invertovanu sliku objekta na drugom ekranu.

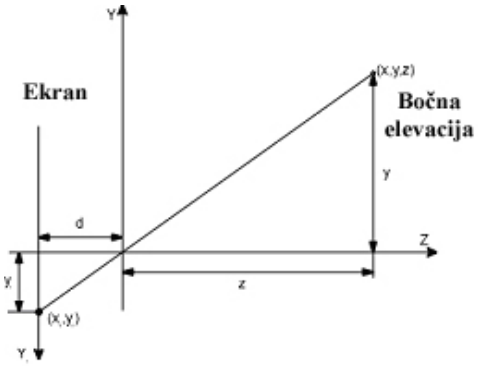


Ako pretpostavimo da postoji skup vrhova koji se nalazi u WCS i vidljiv je kameri, perspektivna projekcija tih vrhova se može izračunati pretpostavljajući da kamera ima pinhole konstrukciju i da joj je otvor smješten u koordinatni početak. Imaginarni ekran za gledanje je postavljen na udaljenost d iza pinhola da uhvati invertovanu perspektivu scene. Ovaj optički fenomen obrtanja slike neće uzrokovati problem jer se to unutar virtualnog svijeta može popraviti okretanjem ekrana za gledanje kao što je prikazano na slici 9

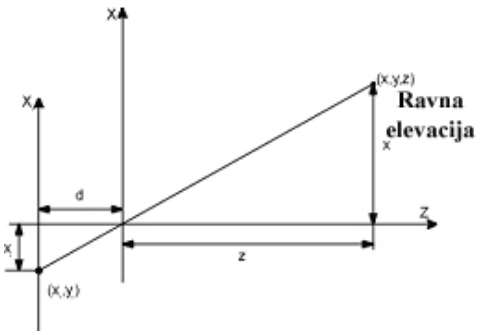


slika 9

Koristeći geometriju sličnih trouglova, slika 10 pokazuje bočnu i ravnu elevaciju WCS aksijalnog sistema i aksijalnog sistema kamere. Pozicija tačke (x, y, z) je locirana u (x_i, y_i) na ekranu za gledanje pomoću sljedećih relacija:



slika 10



$y/z = y_i/d$
i
$x/z=-x_i/d$

pa je zato

$x_i = -dx/z$
$y_i = dy/z$

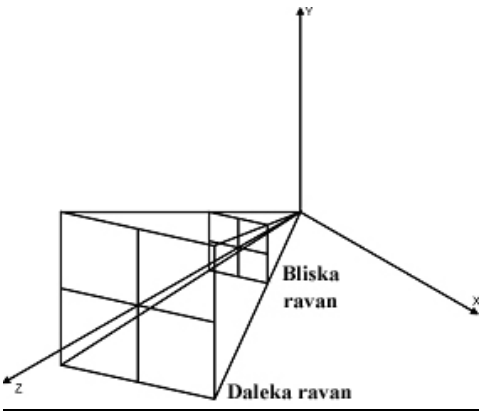
Ove dvije jednakosti su sve što je potrebno za perspektivnu projekciju.

Sve operacije koje se mogu izvesti sa kamerom simuliraju se pomoću kompjutera uz korištenje odgovarajućih izračunavanja.

Odrezivanje pogleda kamere

Konvencionalna kamera može zabilježiti samo sliku scene koja se fizički nalazi ispred nje i unutar polja pogleda objektiva. Međutim, matematička kamera koja se koristi u kompjuterskoj grafici efektivno "vidi" sve. Razmotrimo samo tačku u koordinatnom sistemu kamere sa negativnom z koordinatom. Kada se ova vrijednost zamijeni u jednakosti za perspektivnu projekciju, vrijednosti x_i i y_i će imati suprotne znakove u odnosu na one koje bi imali kada bi z koordinata bila pozitivna. Ako je ovo dozvoljeno, krajnja slika bi bila veoma zbunjujuća jer bi objekti iza kamere bili okrenuti naglavačke i naopako i zaklanjali bi one ispred kamere.

Očigledno, vrhovi iza kamere se moraju ukloniti prije transformiranja u perspektivnu projekciju, ali jednostavno uklanjanje tačaka sa negativnom z koordinatom neće biti dovoljno. Ivice koje povezuju dva vrha moraju biti odrezane da bi ostao njihov vidljivi dio. Mogu postojati i vrhovi koji su toliko udaljeni da su virtualno nevidljivi i također trebaju biti odrezani. Zapravo postoje tri klase vrhova koji zahtijevaju odrezivanje sa scene: vrhovi koji su preblizu kameri, vrhovi koji su predaleko od kamere i vrhovi ispred kamere, ali izvan njenog polja pogleda. Ako ograničimo zapreminu prostora koji sadrži vrhove vidljive kameri imaćemo šest ravni: dvije koje ograničavaju "bliske" i "daleke" ravni i četiri koje uokviruju format slike kamere.

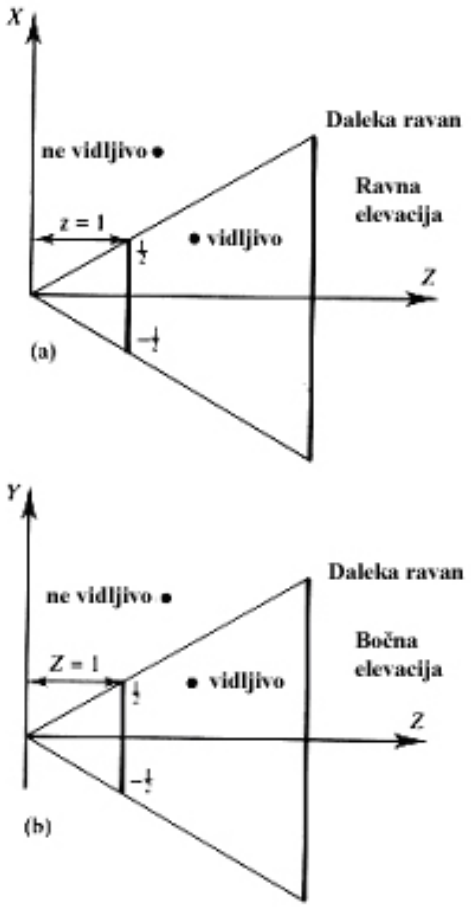


slika 11

Slika 11 ilustrira ovu konfiguraciju koja se zove **frustum pogleda**. Za početak razmotrimo problem identificiranja objekata koji su potpuno nevidljivi kameri.

Pretpostavimo da je W=1 (W - širina slike) i da je ravan projekcije, gdje je z=1 blizu ravni odrezivanja.

Tada objekat čije su z koordinate manje od jedan može sav biti odrezan. U praksi se dešava da se u bazi podataka koja sadrži geometriju postavlja parametar koji indicira da je objekat nevidljiv i da ga ne treba rendati. Slično, objekti čije z koordinate sve prelaze daljinu udaljene ravni odrezivanja su također nevidljivi. Sada dolazimo do lijeve, desne, gornje i donje ravni odrezivanja.



slika 12

U skladu sa slikom 12a vidimo ravni pogled na frustum i možemo primijetiti da je tačka (x_i, y_i, z_i) nevidljiva ako je $x_i/z_i>=1/2$. Slično je kada je $x_i/z_i<=-1/2$. Ovo možemo pisati kao:

$2x_i>=z_i$
$2x_i<=-z_i$

Na slici 12b možemo primijetiti da je bilo koja tačka (x_i, y_i, z_i) nevidljiva ako je $y_i/z_i>=1/2$ odnosno $y_i/z_i<=-1/2$, što možemo pisati kao

$2y_i>=z_i$
$2y_i<=-z_i$

Dakle, objekat može biti potpuno odrezan ako svi njegovi vrhovi zadovoljavaju gornje relacije. Ako koordinate nisu skalirane na veličinu prozora W, onda su uslovi nevidljivosti sljedeći:

$2x_i/W >= z_i$
$2x_i/W <= -z_i$
$2y_i/W >= z_i$
$2y_i/W <= -z_i$

Objekti koji prođu gornje testove su ili parcijalno ili potpuno vidljivi. Za određivanje njihovih vidljivih dijelova se koriste rigorozniji algoritmi kao što su Sutherland - Hodgman-ov ili Cyrus-Beck-ov.

Sada smo u stanju da lociramo kameru u WCS i usmjerimo je na skup objekata i sračunamo perspektivni pogled. Ovaj proces se sastoji iz pet koraka:

(1) locirati i pozicionirati kameru
(2) sračunati matricu za konverziju vrhova iz WCS u CCS (camera coordinate system)
(3) konvertovati vrhove pomoću te matrice
(4) odrezati neželjene dijelove objekata
(5) izvršiti perspektivnu transformaciju
Ovi koraci se moraju izvršiti prilikom svake promjene kamere ili scene.

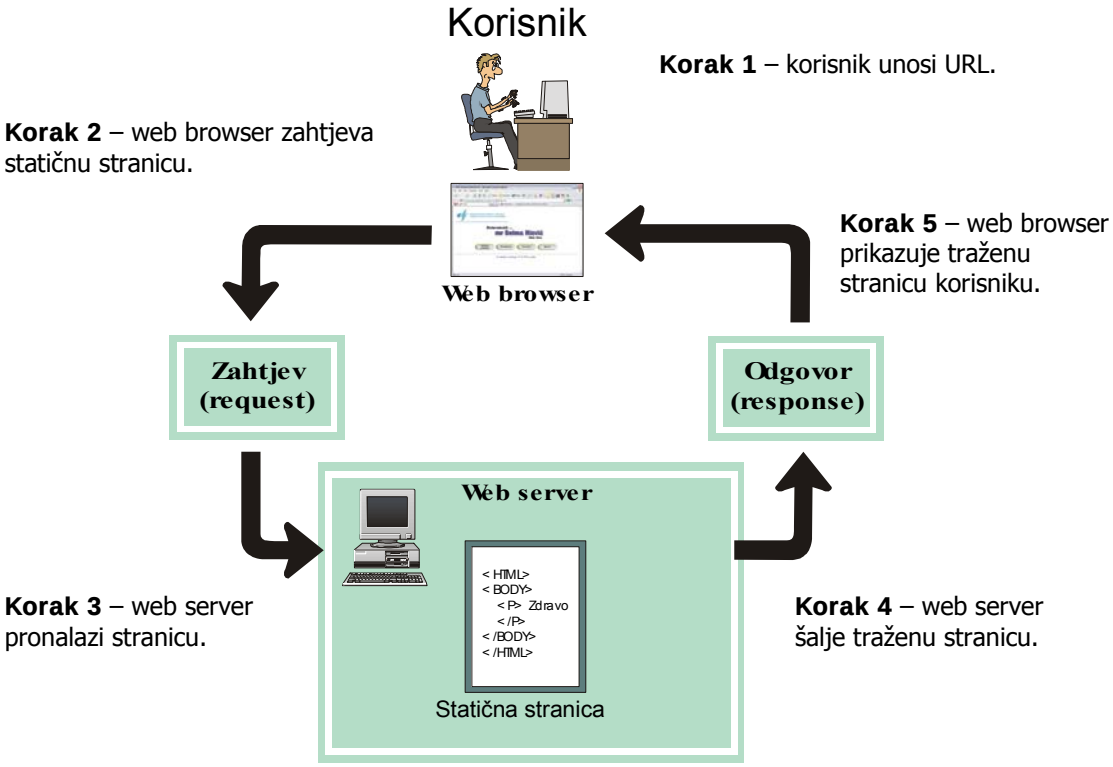
4. Osnovi HTML-a

Procesiranje statičnih web stranica

Regularan web sajt se sastoji od HTML stranica i fajlova kojima upravlja kompjuter na kojem se nalazi web server.

Web server je softver koji srevira web stranice kao odgovor na zahtjev klijenata putem web browsera. Zahtjev za stranicom (page request), se generiše kada korisnik klikne na link unutar web stranice, izabere bookmark u web browseru ili unese URL (uniform resource locator) u web browserovo adresno tekstualno polje (text box).

Konačan sadržaj regularne web stranice je određen od strane web dizajnera i ne mijenja se pozivom stranice.



Svaka linija HTML koda je napisana od strane dizajnera prije nego što se stranica postavi na server. Pošto se HTML kod ne mijenja jednom kada se stranica postavi na server, ovakve stranice se nazivaju statičnim stranicama.
Kada web server dobije zahtjev za statičnom stranicom, on pročita zahtjev, nađe stranicu i pošalje je browseru, upravo kako je prikazano na gornjoj slici.

Šta je HTML?

Hypertext markap language (HTML) predstavalja standardiziranu specifikaciju koju koristi World Wide Web za kreiranje i prepoznavanje hipermedijskih dokumenata. Sam HTML dokumenat je sastavljen od markap koda nazvanog *tagovi*. Dakle, izgled, struktura i formatiranje HTML dokumenta zavisi od tagova. Tag je zatvoren unutar zagrada i na izvjestan način predstavalja jednu instrukciju HTML jezika. Na primjer:

<TAG>

Tagovi obično odlaze u parovima, na primjer:

<TAG> </TAG>

pri čemu drugi tag praktično znači «kraj instrukcije», ali nije rijetkost da se kao instrukcija pojavljuje samo jedan tag. Tag završetka uvijek ima znak «/» ispred imena taga unutar zagrada. Jedini tag koji potpuno odstupa od ove specifikacije, a koji je nama od značaja je tak komentara koji izgleda ovako:

<!-- Ovo je komentar -->

Kao i kod drugih programa, preporučuje se «obilna» upotreba komentara unutar HTML dokumenata.

HTML nije osjetljiv na veličinu slova koja čine ime taga tako da

<body>, <Body>, <BoDy>, <BODY>

predstavljaju jednu istu instrukciju. Međutim, postoje tri nepisana pravila «lijepog» pisanja HTML koda:

1. Tagove treba uvijek pisati velikim slovima
2. Tagovi uvijek treba da počinju u novom redu
3. HTML treba struktuirati prilikom pisanja (mada HTML nije struktuiran jezik – ustvari uopšte ne sadrži bilo kakve strukture)

Ova tri pravila su sastavljena u cilju lakšeg «čitanja» i održavanja koda jer sasvim jednu stvar predstavlja čitanje ovakvog koda:

```
<HTML>
        <HEAD>
                <meta http-equiv=Content-Type content="text/html;
        charset=iso-8859-2">
                <TITLE>Web stranice Selme Rizvić</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFFF">
                <p>
```

```
                <A HREF=IDI NA STRANICU GRAFIKE></A>
                <AHREF="obavjestenja/index.htm"></A>
                <A HREF=MOJ CV></A>
                <img src="images/logo.jpg" width="377"
        height="106"><br>
                <img src="naslov.jpg"></p>
        <p>
        <center>

                <OBJECT classid="clsid:D27CDB6E-AE6D-444553540000"
                 codebase= «http://download.macromedia.com/»
                WIDTH="550"  HEIGHT="45" id="grafika1" ALIGN="">
                        <PARAM NAME=movie VALUE="grafika1.swf">
                        <PARAM NAME=quality VALUE=high>
                        <PARAM NAME=bgcolor VALUE=#FFFFFF>
                        <EMBED src="grafika1.swf" quality=high
                        WIDTH="550" HEIGHT="45" NAME="grafika1"
                        ALIGN=""TYPE="application/x-shockwave- flash"
                        PLUGINSPAGE="http://www.macromedia.com/ ">
                        </EMBED>
        </OBJECT>
    </center>
    </p>
    <hr>
    <center>
        Posljednje osvjeľenje: 21.10.2003. godine
    </center>
</BODY>
</HTML>
```

dok je čitanje ovakvog koda

```
<HTML><HEAD><meta http-equiv=Content-Type content="text/html;
charset=iso-8859-2"><TITLE>Web stranice Selme
Rizvić</TITLE></HEAD><BODY bgcolor="#FFFFFF">
<p><A HREF=IDI NA STRANICU GRAFIKE></A> <A
HREF="obavjestenja/index.htm"> </A>   <A HREF=MOJ CV></A><img
src="images/logo.jpg" width="377" height="106"> <br> <img
src="naslov.jpg"></p> <p> <center> <OBJECT classid="clsid:D27CDB6E-
AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/"
 WIDTH="550" HEIGHT="45" id="grafika1" ALIGN=""> <PARAM
NAME=movie VALUE="grafika1.swf"> <PARAM NAME=quality VALUE=high>
<PARAM NAME=bgcolor VALUE=#FFFFFF> <EMBED src="grafika1.swf"
quality=high bgcolor=#FFFFFF  WIDTH="550" HEIGHT="45"
NAME="grafika1" ALIGN=""
 TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/ "></EMBED> </OBJECT>
```

```
</center> </p> <hr> <center> Posljednje osvjeľenje: 21.10.2003. godine
</center> </BODY></HTML>
```

gotovo nemoguće.

Neki tagovi imaju skup atributa unutar zagrada, koji su ili opcioni ili obavezni, zavisno od taga. Na primjer tag <IMG>:

```
<IMG SRC="slika.jpg" BORDER="0" ALIGN="middle" ALT="Moja slika">
```

definiše uključivanje slike unutar HTML dokumenta. Njegovi atributi prikazani ovdje (a ima ih još) su:

SRC – izvorni (source) fajl fotografije, sa vrijednosti "slika.jpg"
BORDER – okvir oko fotografije u pikselima
ALIGN – poravnanje teksta sa fotografijom, sa argumentom "middle" što znači da će se teks, ili drugi objekti, poravnavati u odnosu na sredinu (horizontalno) fotografije
ALT – ispisuje zadati opis slike na mjestu slike, ukoliko se slika iz nekih razloga ne učita, ili klijent ne može iscrtati sliku.

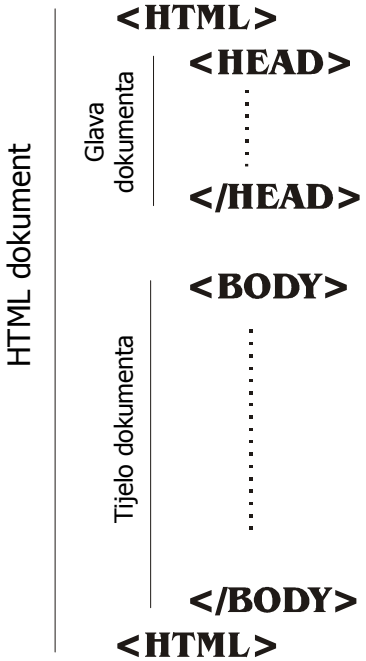
Prilikom kreiranja dokumenta, treba imati na umu da je formatiranje teksta uglavnom igorisano od strane browsera. Tj., bilo koju tehniku formatiranja da koristite browser će je interpretirati kao jedan prazan znak (single space). Generalno, bilo šta od slijedećih navedenih tehnika formatiranja da koristite, browser će prikazati jedan prazan znak:

- Jedan Tab/Više tabova
- Jedan prazan znak/Više praznih znakova
- Jednostruke oznake pragrafa/Višestruke oznake paragrafa
- Jedan prelaz na novu stranu/Više puta ponovljen prelaz na slijedeću stranu

Dakle, u cilju formatiranja stranice moramo se koristiti drugi metodama ugrađenim u sam HTML.

Struktura HTML dokumenta

Svaki HTML dokumenat treba da započne sa <HTML> tagom i završi sa </HTML> tagom. Startni <HTML> tag kaže browseru da je u pitanju HTML-formatirani dokumenat i označava početak dokumenta. Završni tag </HTML> označava kraj dokumenta i uvjek je zadnji tag u HTML dokumentu. Mada postoje HTML dokumenti bez ova dva taga, «lijepo» programiranje nalaže njihovu upotrebu.



Sam dokumenat treba da ima glavu i tijelo. Glava se nalazi odmah iza <HTML> taga i koristi se za specifikaciju ključnih aspekata dokumenta, kao što je na primjer ime dokumenta. Početak i kraj glave dokumenta označeni su sa <HEAD> i </HEAD>.

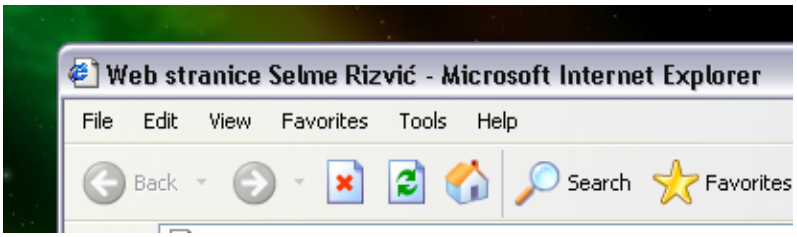
Odmah poslije glave dokumenta nalazi se tijelo dokumenta. Tijelo sadrži tekst i objekte koji se prikazuju unutar klijentovog browsera. Početak i kraj tijela su označeni sa <BODY> i </BODY>.

Sadržaj glave

Glava, ili zaglavlje kako je još neki zovu, se primarno koristi u cilju obezbjeđivanja informacija Web serveru o samom dokumentu. Od velikog broja tagova koji se koriste unutar glave, nama su interesantni slijedeći:

<TITLE></TITLE> Koji definiše naslov dokumenta. Naslov je prikazan na naslovnoj traci browsera. Na primjer:

<TITLE>Web stranice Selme Rizvić</TITLE>
Rezultira sa:



<LINK> Identificira relacije dokumenta prema drugim dokumentima
<META> Omogućava identifikaciju generalnih ili meta-informacija o dokumentu

Sadržaj tijela

<BODY> Tag

<BODY> tag je mjesto gdje se mogu definisati svojstva koja treba primjeniti na čitav dokumenat, kao što je boja pozadine dokumenta, pozadinska slika unutar dokumenta i sl. Najvažniji atributi ovog taga su:

BGCOLOR – definiše pozadinsku boju dokumenta. U heksadecimalnom je obliku i definiše RGB boju, npr: #000000 kao crnu boju ili #FFFFFF kao bijelu boju.
Primjer: <BODY BGCOLOR="#CFFD27">

BACKGROUND – pozadinska slika dokumenta.
Primjer: <BODY BACKGROUND="pozadina.jpg">

ALINK – definiše boju hipertekst linka dok je aktivan. Primjer ALINK="#34FC64"
LINK – definiše boju hipertekst linka koji nije posjećen.
VLINK – definiše boju linka nakon što ga je korisnik posjetio
TEXT – definiše boju teksta, koji ne predstavlja link, unutar tijela dokumenta.
Primjer: text="#000099"

Sada jedan BODY tag može izgledati naprimjer ovako:

```
<BODY BGCOLOR="#FFFFFF" LINK="#00CF43"  VLINK="#98FC54"
TEXT="#000099">
```

Umetanje grafičkih fajlova

Grafički fajlovi se u HTML dokument unose pomoći <IMG> taga. Najbolje je koristiti grafičke fajlove tipa .JPEG i .GIF zbog brzine prenosa. Standardni atributi ovog taga su:

SRC="URL" – gdje url predstavlja put do željenog fajla
BORDER=pixels – veličina okvira oko slike u pikselima
ALIGN="gdje" – poravnanje slike u odnosu na sadržaj oko nje
"gdje" - može biti MIDDLE, LEFT,TOP,RIGHT,BOTTOM itd.
HEIGHT=pixels
WIDTH=pixels – ova dva atributa rezervišu navadeni prostor u pixelima na stranici bez obzira na stvarnu veličinu slike. Ako je slika manja biće skupljena, dok ako je veća biće raširena, da popuni kompletan rezervisani prostor. Ako ovi atributi nisu navedeni, slika će zauzeti onoliko prostora na stranici kolika je njena stvarna veličina.
ALT – ispisuje zadati opis slike na mjestu slike, ukoliko se slika iz nekih razloga ne učita, ili klijent ne može iscrtati sliku.

Primjer taga:
<IMG SRC="http://www.etf.unsa.ba/~srizvic/naslov.jpg" ALIGN="middle" BORDER=0>

Rad sa tekstom

Definisati ćemo neke korisne tagove za rad sa tekstom, mada oni nisu jedini:

<P></P> - Paragraf tag

U HTMLu vizuelni način razbijanja teksta u paragrafe se postiže korištenjem taga <P>. Kada browser vidi paragraf tag, on završava liniju u kojoj se trenutno nalazi i unosi prazno mjesto prije unošenja novog teksta (u novoj liniji) ili objekta koji slijedi iza paragraf taga. Završni tag paragraf taga nije obavezan, ali je poželjan kako bi se u kodu vizuelno mogli paragrafi odmah primjetiti.
Korisni atributi:

ALIGN="gdje" – poravnava tekst paragrafa unutar dokumenta
"gdje" – može biti CENTER,LEFT,RIGHT.

Primjer:
<P ALIGN="LEFT"> Tekst paragrafa </P>

<PRE></PRE> Unaprijed formatiran tekst

Definisanje sekcije kao unaprijed formatiranog teksta je veoma korisno. Omogućava upotrebu stanardnog ASCII tekst formatiranja u cilju formatiranja teksta u dokumentu. U sekciji teksta deklarisanog kao predformatiranog, možete koristiti bilo koji od svojih ASCII trikova, uključujući tabove, višestruke tabove i prazne linije. U ovom slučaju, browser neće ignorisati vaše formatiranje, nego će ispisati sve onako kako ste naveli u dokumentu.

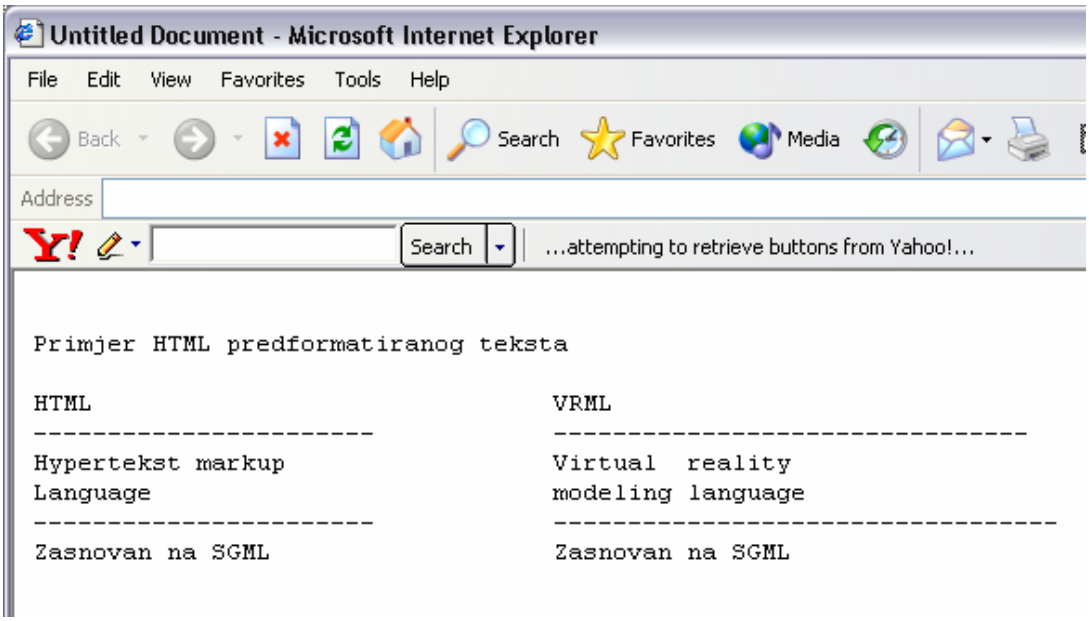
Primjer:

<PRE>

Primjer HTML predformatiranog teksta

HTML VRML
---------- ----------
Hypertekst markup Virtual reality
Language modeling language
---------- ----------
Zasnovan na SGML Zasnovan na SGML
</PRE>

za rezultat daje:

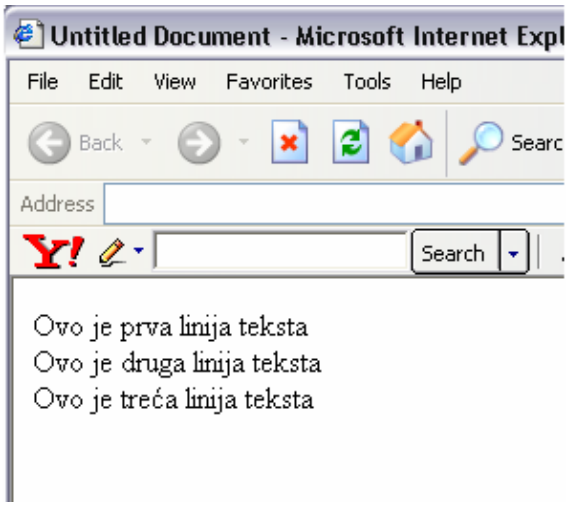


 Prelaz u novu liniju

Tag prelaza u novu liniju <BR> vam omogućava da pređete u novu liniju bez dodavanja razmaka između linija (kako to radi <P>).
 predstavlja jedan od tagova koji nemaju završni tag. Primjer:

Ovo je prva linija teksta
Ovo je druga linija teksta
Ovo je treća linija teksta

Daje za rezultat:

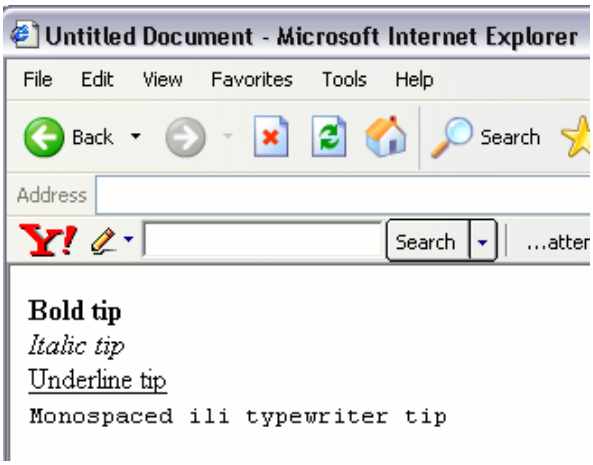


Fizički stilovi

Fizički stilovi govore browseru tačan format u kojem da ispisuje tekst. U HTMLu četiri fizička stila su bold, italic, underline, monospaced. Primjer:

<B> Bold tip </B>
<I> Italic tip </I>
<U>Underline tip </U>
<TT>Monospaced ili typewriter tip</TT>

za rezultat daje:



<FONT></FONT> Fontovi

Tip fonta koji se koristi za prikaz teksta je obično određen u korisnikovom browseru. Ovo daje korisniku kontrolu nad izgledom fonta. Međutim, ponekad mi kao dizajneri želimo promjeniti izgled fonta koji se prikazuje korisniku. Ovo nam omogućava <FONT> tag. Njegovi najznačajniji atributi su:

FACE="Ime fonta" – postavlja font face za tekst unutar taga. "Ime fonta" predstavlja ustvari sistemsko ime kao što je "Time New Roman" ili "Arial". Prilikom postavljanja font facea treba voditi računa o tome da li korisnik ima taj font face na svom računaru. U slučaju da ga nema, browser koristi defaultni font face definisan od strane browsera. Ovo može dovesti do pojave "čudnih" znakova na ekranu, zato oprez sa font faceovima.

COLOR="boja" – postavlja boju fonta unutar taga. Boja je u heksadecimalnom formatu #rrggbb ili može biti, u nekim slučajevima, ime kao "red" ili "blue".

SIZE=veličina – Određuje veličinu fonta. Postoje dva načina određivanja veličine fonta: relativan (u odnosu na veličinu baznog fonta) ili direktno upotrebom vrijednosti od 1 do 7. Da bi se definisao font u odnosu na bazni font ispred broja se postavlja znak – ili + kako bi se naznačila veličina promjene u odnosu na bazni font. Bazni font se postavlja tagom npr.:

```
<BASEFONT SIZE=5>
```

Primjer:

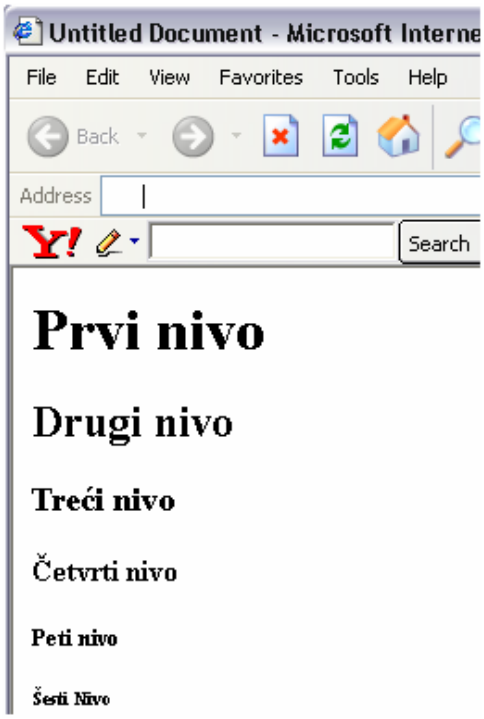
```
<FONT SIZE=3 COLOR=#147108 FACE="Tahoma"> Tekst </FONT>
```

<H></H> Naslovi

Naslovi (Headings) se često koriste u cilju organizovanja stranice. HTML omogućava do 6 nivoa kreiranja naslova počevši od <H1> do <H6>. Naslovi imaju završni tag koji je obavezan. HTML naslovi se prikazuju boldirani i veličina naslova zavisi od nivoa naslova. Tako je <H1> najveći dok je <H6> najmanji. Browseri obično ubacuju razmak ispred i iza naslova. Primjer:

```
<H1> Prvi nivo </H1>
<H2> Drugi nivo </H2>
<H3> Treći nivo </H3>
<H4> Četvrti nivo </H4>
<H5> Peti nivo </H5>
<H6> Šesti Nivo </H6>
```

za rezultat daje:



Primijetite da poslije <H> taga ne slijedi tag za prelaz u novi red, mada browser pređe u novi red.

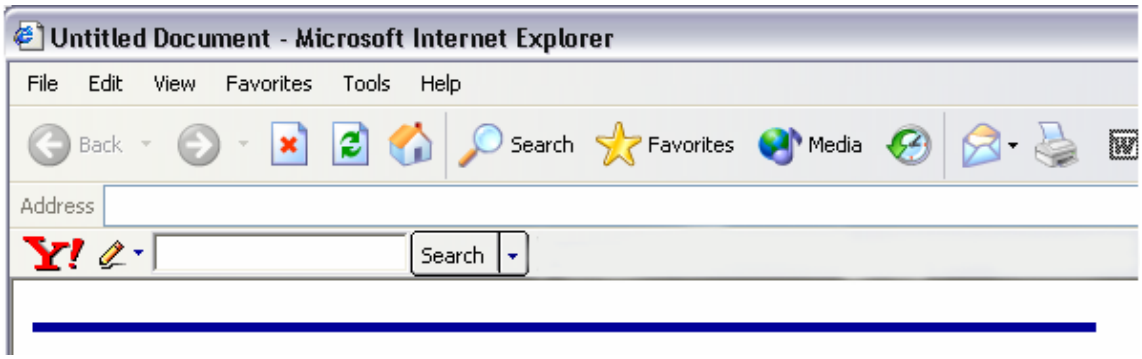
<HR> Horizontalna linija

Ovaj tag je izuzetno koristan za vizuelno odvajanje cjelina i često se koristi. Nema završnog taga. Korisni atributi:

ALIGN="gdje" – označava gdje unutar dokumenta da se izvrši poravnanje linije
COLOR="boja" – boja linije. Daje se u heksadecimalnom obliku #rrggbb.
SIZE="pixel" – određuje debljinu linije u pixelima
WIDTH="procenat" – određuje dužinu linije u procentima, relativno u odnosu na dokument.
Primjer:

```
<HR COLOR="#000099" ALIGN="left" SIZE="5" WIDTH="68%">
```

za rezultat daje



<DIV></DIV> Sekcije dokumenta

HTML omogućava dijeljenje dokumenta u sekcije (divisions) upotrebom taga <DIV>. Upotrebom ALIGN atributa sa <DIV> tagom moguće je specificirati poravnanje paragrafa unutar sekcije dokumenta. Primjer:

```
<DIV ALIGN="right">Tekst za poravnanje</DIV>
```

<A></A> Linkovi

Web bez linkova nebi bio interaktivan. Većina Web dokumenata sadrži linkove. Linkovi djeluju kao pointeri na druge resurse ili fajlove na Webu.Koristeći linkove, možete povezati textove, grafiku, i multimedijske objekte na svom dokumentu. Dobra stvar sa povezinjem teksta, grafike i drugih objekata putem hipertekst linkova je da se oni mogu nalaziti bilo gdje na Webu. Tag hiperlinka <A> (Anchor) ima obavezan atribut HREF, koji sadrži URL lokaciju na koju link pokazuje. Između startnog taga i završnog taga, upisuje se tekst koji postaje linkom. Klikom na ovaj tekst korisnik dolazi do željenog linka. Primjer:

```
<A HREF="http://www.etf.unsa.ba">Link na ETF</A>
```

Također se umjesto teksta može ubaciti i slika. Primjer:

```
<A HREF="http://www.etf.unsa.ba"><IMG SRC="etflogo.jpg"></A>
```

Pomoću linkova također se mogu vezati fajlovi za download. Primjer:

```
<A HREF="vrml97-en.zip">VRML97</a>
```

Postoje tri tipa linkova koji se mogu koristiti:

- Linkovi sa relativnim putevima do fajlova
- Linkovi sa direktnim putevima do fajlova
- Linkovi unutar dokumenta

Relativni putevi

Možemo pristupati lokalnim fajlovima, fajlovima na našem lokalnom Web serveru, koristeći se relativnim putevima. URL-ovi sa relativnim putevima generalno ne specificiraju protokol ili Web server u linku. Kada koristimo relativne puteve da bismo došli do fajla, lociramo fajl u odnosu na poziciju trenutnog fajla. Pristup fajlu u odnosu na trenutni fajl podrazumjeva da je trenutnom fajlu već pristupljeno na određenom Web serveru pomoću određenog protokola. Relativni putevi se mogu koristiti na tri načina:

1. Fajl u trenutnom direktoriju
   ```
   <A HREF="odsjeci.html">Spisak odsjeka na ETF-u</A>
   ```

2. Fajl u roditeljskom direktoriju trenutnog direktorija
   Ovaj fajl je lociran u direktoriju iznad trenutnog direktorija

<A HREF="../akademskiProgram.html">Akademski program ETF-a</A>

Ovaj fajl je lociran dva direktorija iznad trenutnog direktorija
<A HREF="../../index.html">Home Page ETF-a</A>

3. Fajl u poddirektoriju trenutnog direktorija
   Ovaj fajl je u poddirektoriju nazvanom informatika
   <A HREF="informatika/ri.html">Odjsek za računarstvo i informatiku</A>

Direktni putevi

Drugi način pristupa fajlovima je pomoću direktnih puteva. Ovo se postiže navođenjem kompletnog puta do fajla kojem želimo pristupiti. Za pristup fajlovima koji se nalaze na lokalnom Web serveru ne mora se navesti protokol pomoću kojeg se pristupa, dok za pristup ne lokalnom Web serveru protokol se obavezno mora navesti. Ovo dakle znači da postoje dva ključna načina pristupa fajlovima direktno:

1. Navođenjem punog puta uključujući protokol za transfer
   Slijedeći fajl se može nalaziti na ne lokalnom serveru
   <A HREF="http://www.etf.unsa.ba/akademski/odsjeci.htm">Odsjeci</A>
2. Navođenjem punog puta bez navođenja protokola
   Slijedeći fajl se mora nalaziti na lokalnom serveru
   <A HREF="/akademski/odsjeci.htm">Odsjeci</A>

Interni linkovi (linkovi unutar dokumenta)

Linkovi unutar dokumenta omogućavaju moćan način navigacije kroz dokument, pogotovu kroz dugačke dokumente. Koristeći interne linkove, možete obezbjediti način da se lako skoči iz jedne sekcije dokumenta u drugu. Ovakvi linkovi se sastoje iz dva dijela. Prvi dio specificira link sa ključnom riječju koja se koristi unutar taga u formi sličnoj drugim linkovima koje smo vidjeli. Jedini izuzetak je da ključna riječ u linku ima ispred sebe znak #. Na primjer:

<A HREF="#Naslov"> Nazad na naslov</A>

Drugi dio je labela, koja predstavlja lokaciju na koju treba skočiti nakon klika na link. Labela se kreira uporebom taga <A NAME></A> gdje je tekst na koji treba skočiti smješten između tagova.Na primjer:

<A NAME="Naslov"><H1>Odjeci na ETF-u</H1></A>

Kod upisa URLova treba obratiti pažnju na velika i mala slova budući da su neki operativni sistemi osjetljivi na velika i mala slova pa www.Etf.Unsa.Ba/INDEX.html ne predstavlja isto što i www.etf.unsa.ba/index.html.

Upotreba listi

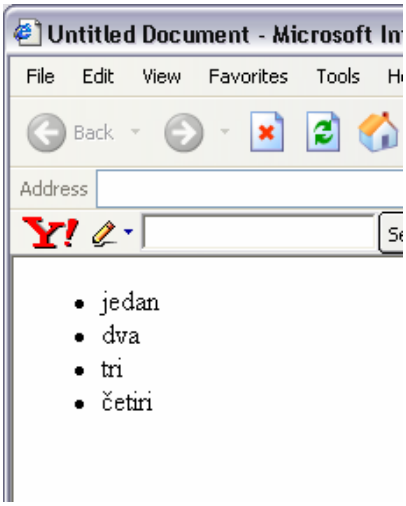
Liste su jedne od najkorištenijih alata pri pisanju web stranica. Liste mogu dati jasan pregled i vizuelan utisak. Dva najkorištenija tipa listi su tzv., neuređena lista i uređena lista.

<UL></UL>Neuređena lista

Kod elemenata neuređene liste se ispred svakog elementa nalazi tačka (bullet). Ova lista se definiše <UL></UL> parom tagova. Svaki element liste započinje <LI> i završava sa </LI>. Na primjer:

```
<UL>
        <LI>jedan</LI>
        <LI>dva</LI>
        <LI>tri</LI>
        <LI>četiri</LI>
</UL>
```

za rezultat daje



<OL></OL> Uređena lista

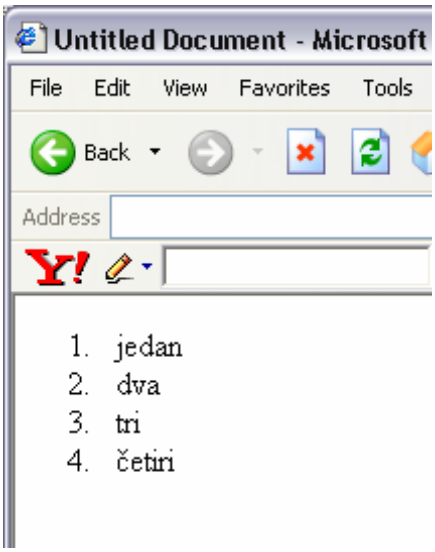
Za razliku od elemenata neuređene liste, kod koje se ispred svakog elementa nalazi tačka, kod elemenata uređene liste ispred svakog elementa nalazi redni broj elementa. Ova lista se definiše <OL></OL> parom tagova. Svaki element liste započinje <LI> i završava sa </LI>.

Na primjer:

```
<OL>
```

```
        <LI>jedan</LI>
        <LI>dva</LI>
        <LI>tri</LI>
        <LI>četiri</LI>
</OL>
```

za rezultat daje



Upotreba tabela

Tabele su također još jedan veoma često upotrebljavan alat pri pisanju Web stranica. Tabele se sastoje od redova u kojima se nalaze ćelije. Svaka tabela počinje statrnim tagom <TABLE> i završava sa tagom </TABLE>. Najvažniji atributi su:

BORDER=pixels – definiše veličinu ruba tabele u pixelima
BGCOLOR="boja" – definiše pozadinsku boju tabele u #rrggbb formatu
BACKGROUND="URL" – specificira pozadinsku sliku
WIDTH="procenat" – specificira širinu koju tabela zauzima unutar dokumenta. Navodi se u procentima.

Redovi tabele se definišu tagovima <TR></TR>. Najvažniji atributi su:

BORDERCOLOR="boja" – definiše boju okvira reda u #rrggbb formatu
BGCOLOR="boja" – definiše pozadinsku boju reda u #rrggbb formatu
VALIGN="pozicija" – definiše vertikalnu poziciju objekata u redu koji obuhvata ćelije

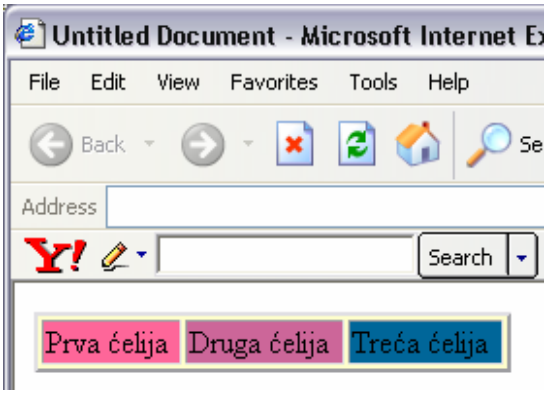
Ćelije se definišu <TD></TD> tagovima. Najvažniji atributi su:

BORDERCOLOR="boja" – definiše boju okvira ćelije u #rrggbb formatu
BGCOLOR="boja" – definiše pozadinsku boju ćelije u #rrggbb formatu
VALIGN="pozicija" – definiše vertikalnu poziciju objekata u ćeliji

Primjer:

```
<TABLE BORDER=2 BGCOLOR=#FFFFCC WIDTH="30%">
        <TR>
                <TD BGCOLOR="#FF6699">Prva ćelija</TD>
                <TD BGCOLOR="#CC6699">Druga ćelija</TD>
                <TD BGCOLOR="#006699">Treća ćelija</TD>
        </TR>
</TABLE>
```

za rezultat daje



5. **Web dizajn**

Web dizajn možemo definisati kao proces dizajna i projektovanja web sajtova.

Ovaj složeni proces možemo podijeliti u dvije faze:

- Proces planiranja i razvoja sajta
- Dizajn sajta

Planiranje i razvoj sajta

Prvi zadatak koji web dizajn tim ima prilikom kreiranja novog sajta je definisanje skupa ciljeva koji se žele postići tim web sajtom. Poznato nam je da postoje sajtovi koji služe raznim svrhama, od onih reklamnih, preko sajtova za edukaciju, sajtova pojedinih komercijalnih firmi, obrazovnih organizacija, te sajtova za zabavu.

Prije početka projektovanja sajta potrebno je uraditi sljedeće:

- identificirati ciljnu publiku sajta
- napraviti izjavu o svrhi sajta
- definisati glavne ciljeve
- ustanoviti okvire informacija koje će sajt sadržati
- identificirati podatke i grafičke resurse koje ćemo koristiti u razvoju sajta

Publika

Publika jednog sajta može biti raznorodna. Bilo koja osoba koja surfa po Internetu može se zakačiti na vaš sajt. Međutim, kada projektujemo sajt moramo se odlučiti kojoj se publici obraćamo.

Osnovne grupe publike mogu biti:

- web surferi
- početnici i povremeni korisnici
- eksperti i česti korisnici
- internacionalni korsinici

Ako želimo da se obratimo web surferima kao glavnoj ciljnoj grupi, stranice sajta treba da izgledom podsjećaju na stranice nekog magazina. One treba da imaju snažnu grafiku i naglašen opis sadržaja stranice.

Početnici i povremeni korisnici se privlače jasnom strukturom sajta i lahkim pristupom informacijama. Jakob Nielsen iz Sun Microsistema je ustanovio da manje od 10% korisnika ikada skrolira stranicu.

Eksperti i česti korisnici zavise od našeg sajta da dobiju informaciju brzo i na vrijeme. Oni ne vole mnogo grafike na sajtu i preferiraju padajuće tekst menije koji se brzo učitavaju.

Internacionalni korisnici zahtijevaju prevod barem ključnih stranica sa menijima. U sastavu sajta treba izbjegavati lokalni žargon ili tehničke akronime. Posebno treba paziti u prikazu datuma.

Izjava o svrhi

Kratka izjava o svrhi i zadacima može formirati osnovicu za projekovanje sajta. Ovo može biti i alatka pomoću koje analiziramo uspjeh našeg sajta.
Primjer jedne ovakve izjave je sljedeći:

«Očekujemo da web sajt naše institucije postigne sljedeće ciljeve u narednih 12 mjeseci:
Web sajt će reducirati zahtjeve upućene centralnom uredu o aktivnostima institucije, rokovima, obavezama i cijenama i informacijama o sastancima. Očekujemo da će web stranica također omogućiti da uštedimo na pošiljkama i procesiranju rutinske korespondencije članova. Web sajt će sadržati sav materijal koji se trenutno objavljuje u našoj kvartalnoj novini, ali davati i ažurnije informacije o događajima. Nakon godinu dana ćemo možda i ukinuti papirno izdanje naših novina»

Važno je imati na umu da je izgradnja web sajta dinamički proces, a ne jednokratan projekat sa statičnim podacima. U našim planovima treba da bude pokriveno dugotrajno uređivanje i tehničko održavanje sajta. Bez ove duže perspektive vaša web prezentacija će doživjeti istu sudbinu kao mnogi entuzijastički započeti, ali nezavršeni projekti.

Dizajn interfejsa

Postoji nekoliko osnovnih pravila za dizajn interfejsa.

Web stranice se razlikuju od knjiga i drugih dokumenata u jednoj krucijalnoj stvari: hipertekst linkovi omogućavaju korisniku da pristupi jednoj web strani bez prethodne. Zato web stranice treba da budu više nezavisne nego stranice u običnoj knjizi. Ovo obično znači da hederi i futeri web stranica treba da budu informativniji od onih kod štampanih stranica. Bilo bi apsurdno ponavljati copyright, autora i datum izdavanja knjige na dnu svake stranice, ali individualne web stranice često trebaju takvu informaciju jer jedna web stranica može biti jedini dio vašeg sajta koji će korisnik ikada vidjeti.

Zato u okviru svake stranice treba da se nalaze odgovori na pitanja:

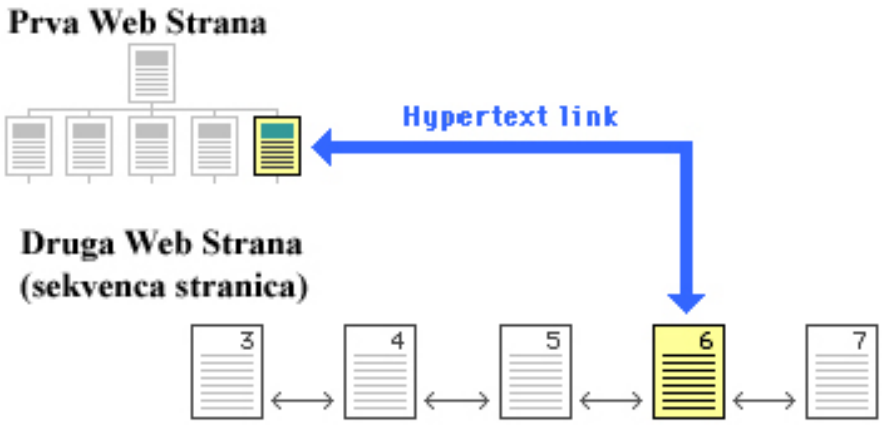
- ko - ko se obraća publici
- šta - naslov dokumenta mora biti jasan jer se on prvi pojavi kod učitavanja stranice
- kada - informacija o zadnjem update-u stranice
- gdje - staviti home URL barem na glavne stranice kako se informacija o porijeklu stranice ne bi izgubila prilikom štampanja dijela stranice

Dizajn interfejsa treba da bude orijentisan ka korisniku. Svojim dizajnom treba maksimalno olakšati korisniku upotrebu naše stranice.

Navigacija

Navigacija je posebno važna u dizajnu interfejsa.

Trenutnim stanjem web-tehnologije, veza korisnik-web strana je omogućena putem navigacije hyperteksta izmedju dokumenata. Bez lokalne organizacije informacija, glavni problem na web-strani je nedostatak informacija o tome gdje se trenutno nalazimo. Jasne, konzistentne ikone, grafičke šeme i grafički, odnosno tekst prikaz i kratak sažetak na ekranu može uliti korisniku povjerenje, da pronadje ono što traži bez gubitka vremena.



Korisnici bi uvijek trebali biti u mogućnosti da se vrate na home-page, a takodjer i na ostale važne navigacijske tačke unutar lokalne stranice. Glavni linkovi, koji bi trebali biti prisutni na svakoj strani sajta su najčešće grafička dugmad koja omogućavaju osnovne navigacijske linkove i pomažu pri kreiranju grafičkog identiteta koji obavještava korisnika da je još uvijek unutar sajta.

Bar dugmadi je veoma koristan (omogućava mnogo izbora na malom prostoru), predvidiv (uvijek je tu, na dnu strane) i omogućava konzistentan grafički identitet svakoj strani Netscape site-a.

Posebno je važno kod navigacije da izbjegavamo tzv. dead end stranice. Svaka web stranica treba da sadrži barem jedan link. Najniže stranice u hijerarhiji treba da sadrže link na Home stranicu ili neku drugu lokalnu stranicu kako korisnik ne bi napustio naš sajt.

Još jedna važna kategorija kojoj treba težiti u dizajnu interfejsa je brz pristup informacijama. Brzina pristupa informacijama zavisi od hijerarhijske organizacije informacija u našem sajtu.

Dizajn sajta

Prilikom dizajna web sajta moramo voditi računa o tri aspekta:

- organizovanje informacija

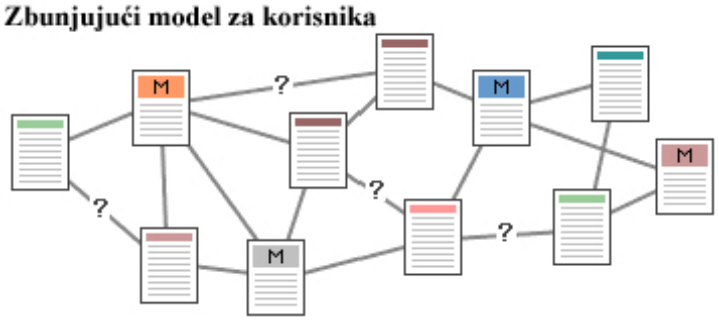
- struktura stranice
- elementi sajta

Organizovanje informacija

Preglednost stranice je značajan faktor njenog kvaliteta. Dobra preglednost se postiže dobrom organizacijom informacija koje prezentiramo na stranici.

Svaka organizacija se sastoji od uspostavljanja hijerarhije važnosti pojedinih informacija.

Uočimo razliku između sljedeće dvije šeme hijerahije



Prva šema predstavlja logički organizovanu hijerarhiju između Home stranice, stranica podmenija i stranica sadržaja. Druga šema predstavlja model veoma zbunjujući za korisnika, jer su stranice međusobno nelogično i zapleteno povezane, pa je moguće da korisnik izgubi kontekst stranice.

Praksa je pokazala da meniji ne treba da sadrže ni previše ni premalo linkova. Optimalan broj linkova za meni strukturu je 6 linkova po meniju.

Postoje 4 načina organiziranja informacija u sklopu web sajta: niz, mreža, hijerarhija informacija i web mreža.

1. Niz

Predstavlja linearni slijed informacija, npr. hronološki, abecedno sortiran itd.

2. Mreža

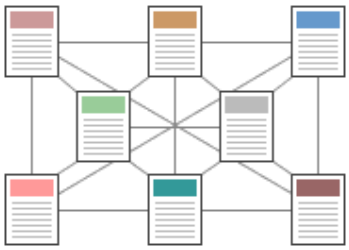
Uspostavlja relacije između pojedinih kategorija informacija po vertikali i po horizontali.

3. Hijerarhija informacija

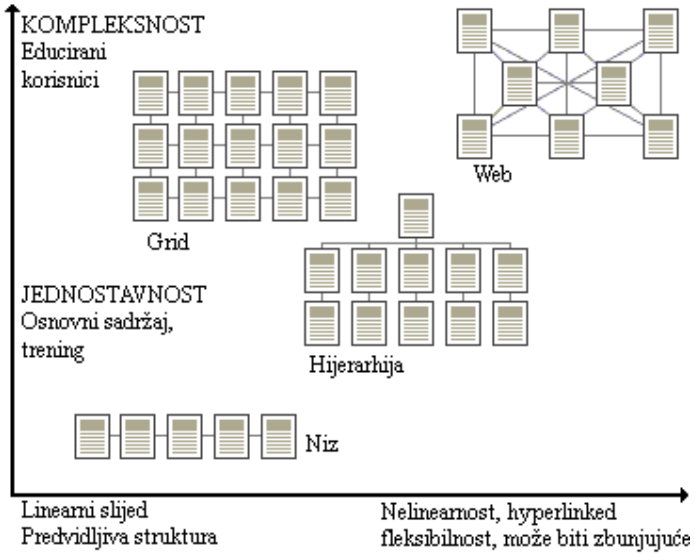
Informacija su organizovane kao idejni slijed početne stranice. Ovo je najbolji način organizacije za web

4. Web mreža

Povezivanje sa informacijama sa eksternih web stranica. Ovaj način dobro koristi mogućnosti web linkovanja, ali je veoma nepraktičan za korisnika.



Na sljedećem dijagramu prikazan je odnos izmedju kompleksnosti korištenja stranice i linearnosti prikaza informacija za prethodne 4 kategorije organizacije.



Elementi sajta

Web sajt se obično sastoji od sljedećih elemenata:

- početna (home) stranica
- podstranice sa menijima
- stranice sadržaja

Sve web stranice su organizirane oko početne stranice, koja funkcionira kao ulaz u sadržaj web stranica na cjelokupnom sajtu. U hijerarhijskoj organizaciji, home page se nalazi na vrhu dijagrama, a sve stranice na web sajtu bi trebale da imaju direktni link natrag na početnu stranicu. World Wide Web URL početne stranice predstavlja adresu koja se koristi da bi korisnici došli na vašu web stranicu, i adresa početne stranice bi u godinama koje dolaze mogla postati podjednako važna kao adresa vaše ulice ili firme. Na vrhu home page-a trebalo bi se nalaziti ono što web korisnici trebaju prvo vidjeti kada pristupe vašoj stranici (ili stranici vaše kompanije, u slučaju da se radi o koorporacijskim stranicama), pa je stoga pravilan dizajn ključan za uspjeh vaše stranice.

Važna odluka koju treba donijeti prilikom kreiranja home stranice je odnos između količine grafike i teksta koji ćemo upotrijebiti.

Grafika čini stranicu atraktivnom za korisnika, ali i sporijom za učitavanje.

U donošenju ove odluke pomoći će nam razmatranje kojoj ciljnoj publici se obraćamo našim sajtom, odnosno šta je svrha našeg sajta.

Postoje i kompromisi u rješavanju ove dileme. Možemo ponuditi korisniku alternativni prikaz stranice, tj. da se odluči između tekst prikaza i prikaza sa grafikom.

Drugi kompromis koji je jednostavniji i bolji jeste upotreba grafičkog banera na vrhu stranice, sa tekstualnim linkovima u ostalom dijelu stranice.

Kako se web stranice često ažuriraju, potrebno je nekako istaknuti prisustvo novih informacija na stranici. Postoje dva načina za rješenje ovog problema:

- ikone New pored nove informacije
- uvođenje What's new stranice gdje ćemo grupisati linkove na nove informacije

Welcome ekrani mogu biti atraktvan način za ulazak u sajt, ali također mogu postati dosadni korisniku koji često ulazi u sajt, posebno ako sadrže animaciju koja se dugo puni. Kompromisno rješenje bi moglo biti da se korisniku ostavi mogućnost da klikom prekine punjenje ili odvijanje animacije i odmah uđe u sajt.

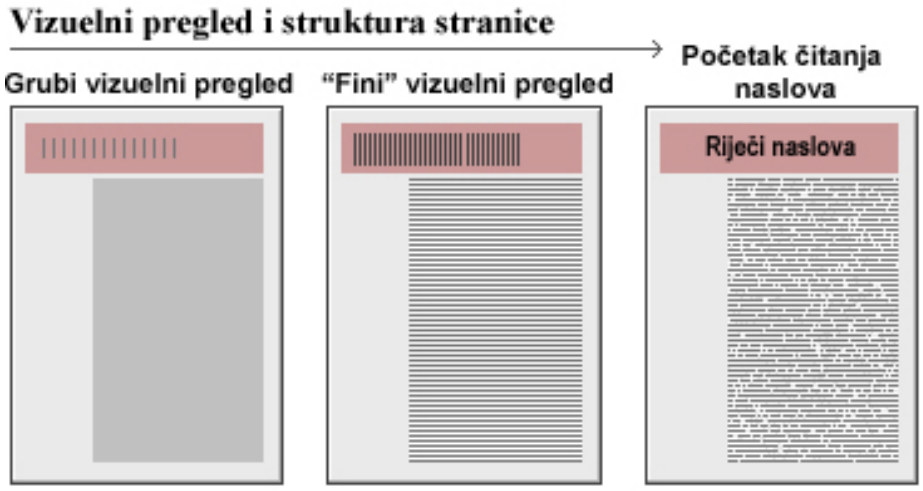
Dizajn stranice

Grafički dizajn kreira vizuelnu logiku, optimalni balans između vizuelnog doživljaja i grafike ili tekstualne informacije. Bez vizuelnog uticaja oblika, boje i kontrasta, stranice su obično dosadne i neće motivirati posjetioca da ispita njihov sadržaj. Dokumenti sa gustim tekstom, bez kontrasta i vizuelnog olakšanja koje nudi grafika i pažljivi prelom strane i tipografija, su teži za čitanje, naročito na ekranima sa niskom rezolucijom. Kakogod, bez dubine i kompleksnosti teksta, visoko grafičke stranice rizikuju razočarenje kod korisnika, jer nude slab balans vizuelnog doživljaja, tekstualne informacije, i interaktivnih hypermedia linkova.

Kada smo uspostavili kompromis između količine grafike i teksta koji ćemo upotrijebiti u dizajnu stranice, potrebno je pažljivo uspostaviti vizuelnu hijerarhiju.

Naglasićemo važne elemente i organizovati sadržaj logično i predvidljivo.

Posjetilac stranicu vidi prvo kao veliku količinu oblika i boja (slika ispod), sa elementima iz prednjeg plana koji su u kontrastu sa pozadinom. Tek kasnije počinje da izdvaja određene informacije, i to prvo iz slika, ukoliko su prisutne, a tek onda počinje raščlanjivati ″teži″ sadržaj teksta i počinje čitati pojedine riječi i fraze:



Zato su ukupni grafički balans i organizacija stranice presudni da biste uveli čitaoca u vaš sadržaj. Dosadna stranica ispunjena tekstom će odbiti oko kao masa monotonog sivila, ali stranica osiromašena dizajnom ili ispunjena suviše velikim slikama će

također natjerati iskusnog korisnika da potraži drugi sadržaj. Ono što želite je prikladan balans koji privlači oko vizuelnim kontrastom:

Proporcija i "prikladnost" su ključ uspješnih dizajnerskih rješenja, ali ove stvari mogu biti određene samo kontekstom vašeg glavnog cilja u razvoju Web stranice, samim sadržajem, i najvažnije očekivanjima vaših korisnika.

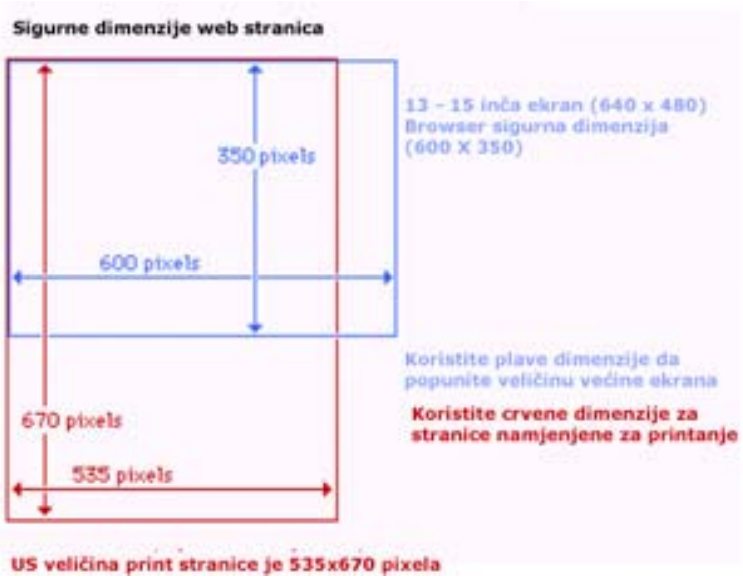
Veoma je važno odgovarajućim dizajnom usmjeriti pogled korisnika.

Na Zapadu engleski čitaoci čitaju slijeva na desno, i od vrha stranice prema dnu. Ova fundamentalna vizuelna osa dominira u mnogim dizajnerskim rješenjima i osnova je za najpogodniji grafički dizajn publikacija. Vrh stranice je uvijek najdominantnija lokacija, ali na Web stranicama gornji dio stranice je posebno važan jer su gornja četiri inča stranice sve što je vidljivo na 14 do 16 inčnim kancelarijskim monitorima. Umjerene pastelne sjene boja koje se tipično mogu naći u prirodi su najbolji izbor za pozadinu ili sporedne elemente, pogotovo ako ste novi u grafičkom dizajnu i izboru boja. Izbjegavajte upadljive, visoko zasićene osnovne boje osim u područjima maksimalnog značaja, ali ih čak i tada upotrebljavajte oprezno. Tekst uvijek mora biti u snažnom kontrastu sa pozadinom.

Čuvajte se grafičkog dotjerivanja. Horizontalne linije, grafičke crtice, sličice i druge vizuelne oznake imaju svoje povremene korisnike, ali primjenite svaku umjereno da biste izbjegli šaren i konfuzan raspored. Isto se odnosi i na tekst veće veličine na Web stranicama.

U dizajnu podstranica veoma je važna dosljednost. Ako smo izabrali jednu grafičku šemu na glavnoj stranici, treba je koristiti na čitavom sajtu. Možemo ponavljati isti baner ili bar sa linkovima na svakoj stranici sajta, čime smo riješili pitanje dosljednosti u dizajnu i poboljšali navigaciju.

Također treba imati na umu da ekran ne bi trebao biti manji od stranice. Većina korisnika nema običaj da skrolira stranice, tako da se može desiti da neka važna informacija ostane izvan okvira ekrana i ne bude nikad pročitana.



Na sljedećoj slici vidimo primjer lošeg i dobrog izgleda stranice

Tipografija

Dobra tipografija ovisi o vizuelnom kontrastu između jednog i drugog fonta, i kontrasta između blokova teksta i njihovog okruženja. Prosječnom čovjeku pažnju najviše privlači jaki kontrast i različitost uzoraka pa je na ove atribute potrebno obratiti posebnu pažnju pri dizajniranju stranica. Premda treba paziti i na neka pravila. Na primjer, ako se svi fontovi na stranici stave na Bold to će na kraju izgledati kao da se "derete" na čitaoce. S druge strane, ako je font jednoličan, posjetioci će ga teško primjećivati zbog manjka kontrasta između teksta i okoline.

Tabele

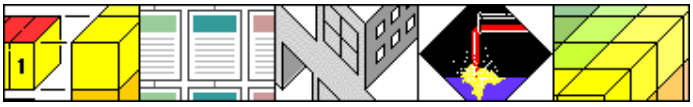
Nevidljive tabele su trenutno jedina HTML mogućnost za izgled stranice. Ako jednostavno stavite komad teksta na stranicu, dužina linija je određena dimenzijama korisnikovog prozora browsera. Kada korisnik promjeni veličinu prozora, tekst se pregrupiše, ispunjavajući drugačiji prostor.
Pri pretjeranom ustrajanju kod ovih standarda, nekad se možete dovesti u neki vid neprilike i izgubiti svoje posjetioce.
Da izbjegnete ovo koristite nevidljive tabele da definišete područje svojih stranica. Koristite ćelije tabela da kreirate margine, stavite vaš tekst u ćelije da ograničite dužinu linije (idealno od 10 do 12 riječi u liniji), i koristite ćelije da pozicionirate elemente na stranici.

Web grafika

Upotreba GIF i JPEG file-ova

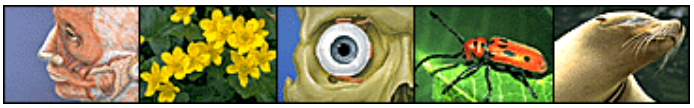
Pošto Netscape i drugi browseri podržavaju GIF i JPEG grafiku u web stranicama za slike se mogu koristiti oba grafička formata. Većina web dizajnera koristi GIF format za kreiranje crtanih elemenata i izabrati će JPEG format uglavnom za fotografije kompleksnih fotografskih ilustracija, medicinske slike i ostale tipove slika gdje kompresioni učinak JPEG formata neće znatno uticati na kvalitet slike.

Prednosti GIF file-ova



- Veoma široka primjena ovog grafičkog formata na Webu
- Svi browseri Weba podržavaju GIF format za inlined slike.
- Slike dijagrama bolje izgledaju u GIF formatu nego u JPEG-u.
- GIF podržava providnost i isprepletenost.

Prednosti JPEG slika



- Velika kompresija proporcija je moguća za brže downloadovanje.
- Daje odličan rezultat u većini fotografija i medicinskih slika.
- Podržava “full-color” slike (24 bita “true-color” slike).

Ako stavite HEIGHT (visina) i WIDTH (širina) tagove slike u vaše izvorne HTML tagove za opis slika (image source tags), ta informacija govori browseru koliko prostora na stranici da rezerviše za sliku. To omogućava browseru da formatira stranicu čak i prije nego se grafički fajlovi počnu downloadirati. Ovo ne ubrzava učitavanje grafike (naravno da ništa osim brže konekcije to ne može uraditi) ali to omogućava korisniku da brzo vidi osnovni format stranice. Ako upišete HEIGHT i WIDTH tagove grafičkih dijelova vaše stranice, browser će najčešće prvo učitati i ispisati blokove teksta, a onda će "polako" učitati grafičke fajlove na mjesta predviđena za to. Tako korisnik može početi sa čitanjem vaše stranice i prije nego se grafički fajlovi učitaju.

HEIGHT i WIDTH tagovi su dodaci osnovnom tagovima za opis slike:

```
<IMG SRC="picture.gif" HEIGHT="30" WIDTH="475">
```

Za najbolje performanse, uvijek, ali uvijek u osnovne tagove za opis slike dodajte informacije o visini i širini (čak i za male slike, kao npr. dugmiće isl.)

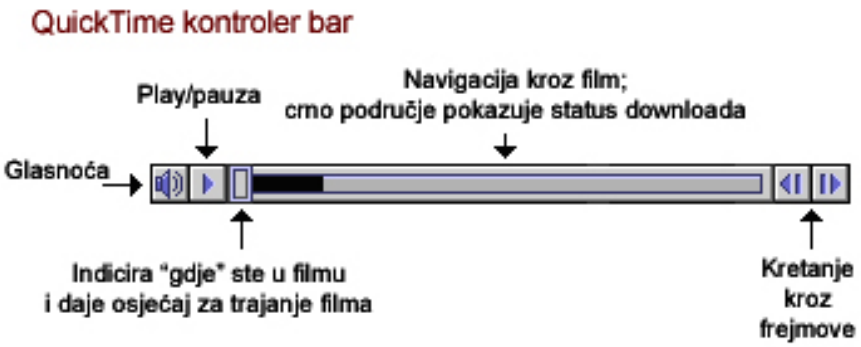
Web multimedia i animacije

Možda je najmoćnija osobina računarske tehnologije mogućnost da kombinuje tekst, grafiku, zvuk, i pokretne slike. Multimedia je imala spor napredak ka web-u zbog ograničenja bandwidth-a, ali svakim danom se pojavljuju nova rješenja.

U korištenju animacija na web stranicama treba biti oprezan, jer predugo čekanje na učitavanje animacije može natrjerati korisnika da odustane.

Animacije treba koristiti samo ako je njihova upotreba opravdana u odnosu na sadržaj stranice.

Osigurajte korisnicima statusnu informaciju i kontrole kada prezentirate multimedijalne elemente. Npr. QuickTime kontroler bar, iako možda ne izgleda lijepo, vrlo je važan i efikasan element interfejsa koji omogućava i kontrolu i statusnu informaciju. On omogućava korisnicima da pojačaju ton, plejaju i stanu i kreću se kroz film, i također nudi informaciju o statusu downloada filma.



Najbolji format za video animaciju je animirani GIF, a za audio i video MPEG.

6. **Osnovi grafičkog dizajna**

Grafički dizajn je kreativna djelatnost koja ima za cilj da putem likovno-grafičkog izražavanja i uz odgovarajuća tehnička sredstva ostvari kvalitet savremene vizuelne komunikacije.

6.1 Uvod u dizajn

a) dizajn je funkcionalan

Dizajn objašnjava "kako" stvari: kako naručiti poklon, kako se kretati po web stranici, kako služiti interesima klijenta, kao komunicirati sa gledaocima, kako saopćiti informacije.
Još važnije, dizajn saopćava ideje, koncepte i funkcije određenoj publici. Publika se sastoji iz ciljnih grupa kao što su starosne grupe, polovi, grupe sa određenim materijalnim mogućnostima.

b) dizajn je informativan

Dizajn je sredstvo kojim se informacija saopćava njenom konzumentu. To podrazumijeva isticanje prioriteta u čitanju informacije, tako da gledalac u što kraćem roku percipira željenu informaciju. Pogrešno je dati prioritet dizajnu nad informacijom, tako da gledalac ne pročita poruku. To je posebno prepoznatljivo kod dizajna plakata, jer je svrha da poruka bude uočena u vrlo kratkom periodu posmatranja (npr. billboard plakati pored ulice se uočavaju prilikom veoma brzog prolaza pored njih u automobilu)

c) dizajn koristi vizuelni jezik

Vizuelni jezik ima određena sredstva izražavanja kao što su negativ/pozitiv, padanje/dizanje, stabilno/nestabilno, naprijed/nazad, debelo/tanko, veliko/malo, glatko/grubo, providno/neprovidno, dolazi/odlazi.

d) dizajn je proces

Proces dizajna počinje kreiranjem od generalne informacije do specifičnih detalja. U današnje vrijeme možemo primijetiti kako su koncipirane reklamne kampanje u medijima. Kampanja obično počinje uvođenjem gledaoca u predmet sa nekom kratkom naznakom, a onda se se vremenom dograđuje i razvija do postizanja potpunog cilja.

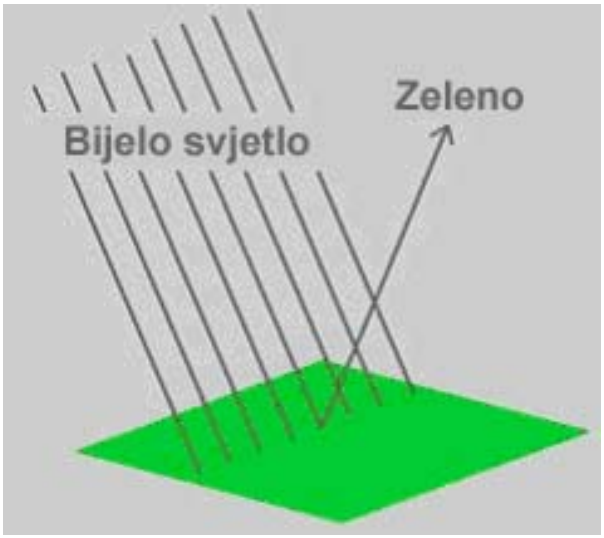
6.2 Teorija boje

Kada govorimo o boji mislimo na osjećaj povezan sa slikama koje kreira ljudski sistem vida koji se sastoji od očiju, optičkih nerava i centra za vid u mozgu. Riječ kao što je "zeleno", "plavo", "crveno", "svijetlo" i "tamno" se koriste da opišu taj osjećaj i ako ih budemo dalje analizirali naći ćemo se u domenu filozofije.

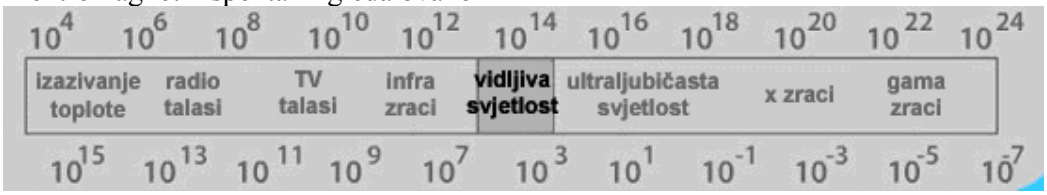
Iz iskustva znamo da Sunce isijava nešto što mi zovemo svjetlost, što igra centralnu ulogu u procesu vida. Naučnici koji su proučavali prirodu svjetla u prošlim stoljećima akumulirali su određenu količinu informacija o njegovom ponašanju. Npr. u praznom prostoru svjetlost putuje u pravim linijama brzinom oko 300 000 000 m/sec. Njena transmisija se može opisati modelom oscilirajućih električnih i magnetnih talasa koji se talasaju sinusoidalno i imaju određenu frekvenciju i talasnu dužinu.
Njutn je demonstrirao da kada sunčeva zraka prođe kroz prizmu ona se disperzira u spektar boja od crvene, narančaste, žute, zelene, plave, indigo i ljubičaste; plavi kraj spektra je disperziran više nego crveni. Dalji eksperimenti su pokazali da se ove diperzirane boje razlikuju samo u talasnoj dužini koja je kod crvene 700 nm, kod zelene 520 nm i kod plave 480 nm.
Prihvaćeni model vidjenja podrazumijeva da kada zrake svjetlosti - sa inicijalnom distribucijom talasnih dužina - osvijetle površinu, neke se talasne dužine apsorbiraju a neke reflektiraju. Ako posmatrač uočava ovo reflektovano svjetlo, onda osjećaj za boju zavisi od prirode izvora svjetlosti i apsorptivnih karakteristika površine.
Međutim, iako ovaj jednostavni model omogućava zadovoljavajući mehanizam za opisivanje jednostavnih optičkih fenomena, to ne objašnjava zašto osjećaj za boju također zavisi od površina koje su susjedne površini koja se posmatra

a) kvaliteti boje

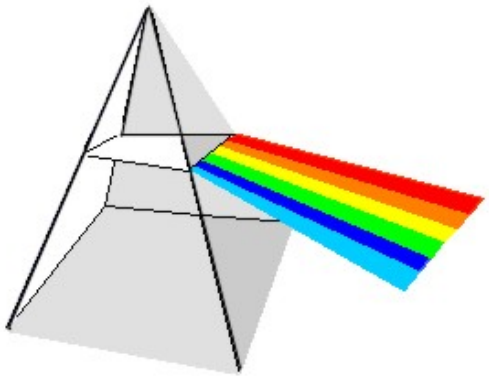
Boje koje vidimo u prirodi su refleksija svjetla na površinu oko nas.



Zelena površina apsorbuje svu vidljivu svjetlost osim zelene.
Elektromagnetni spektar izgleda ovako



Vidljivi spektar se pomoću prizme dijeli na sljedeći način



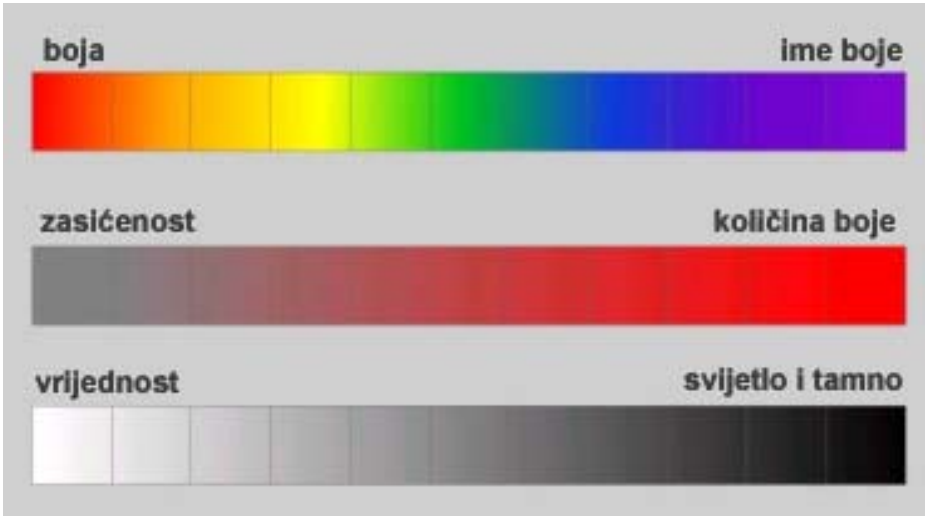
Dimenzije boje

Boja se definiše kroz svoje tri dimenzije:

- hue – boja
- saturation - zasićenost
- value – vrijednost

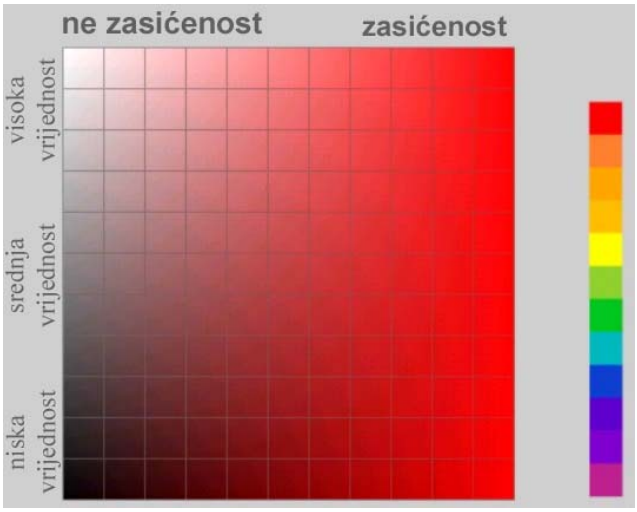
Termin hue može imati vrijednosti crvene, žute, zelene, cyan, plave i magenta. Saturation definiše koliko bijele svjetlosti je sadržano u boji (sjetimo se da jednaki iznosi crvene, zelene i plave stvaraju bijelu boju). Vrijednost se koristi za razlikovanje svijetle boje od tamne.

Konvencija za hue jeste da 0 predstavlja crvenu, 1/6 žutu, 1/3 zelenu, 1/2 cyan, 2/3 plavu, 5/6 magentu i 1 također crvenu. Boja bez ikakve bijele komponente je potpuno zasićena sa nivoom 1, dok totalno nezasićena boja kao npr siva ima nivo 0. Vrijednost koja predstavlja svjetloću boje, varira od 0 za crnu do 1 za bijelu

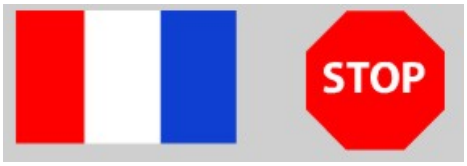


Inherentni kvaliteti boje

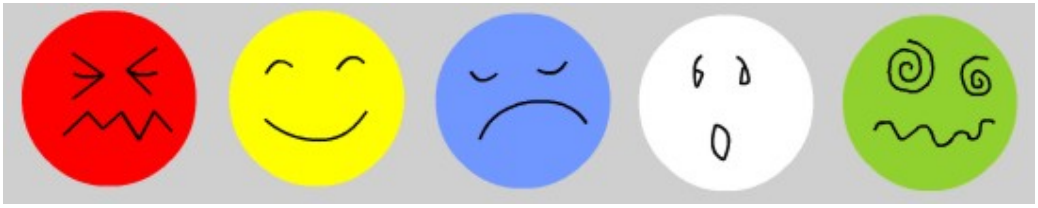
Svaka boja ima svoje inherentne kvalitete koji variraju kako je prikazano na sljedećoj slici.



Emotivni kvaliteti boje mogu biti simbolički:



i emocionalni



U grafičkom dizajnu boje se koriste kao sredstvo za saopćavanje određene poruke, tako da se pri izboru boja uvijek vodi računa na emociju koju pojedine boje bude u posmatraču.

b) aditivni i subtraktivni kolor model

Aditivni kolor model (RGB) koriste ekranski displeji. Ovaj model miješa boje pomoću svjetla. Boja se sastoji iz crvene (Red), zelene (Green) i plave (Blue) komponente.
Subtraktivni model (CMYK) se koristi u štampi. U ovom modelu boje se miješaju pomoću inka, a osnovne komponente boje su Cyan, Magenta, Yellow i blacK.

c) osnovne harmonije boja

Krug boja

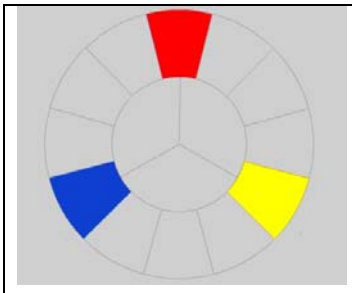
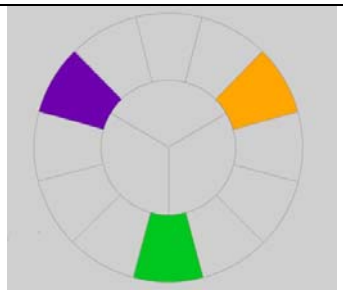
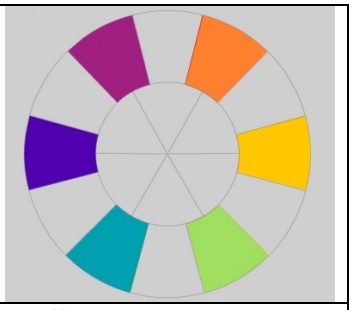
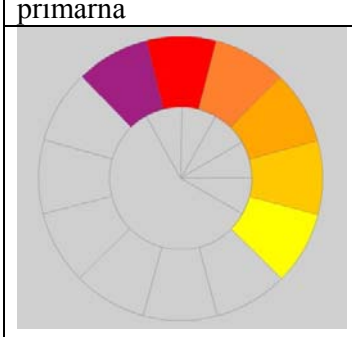
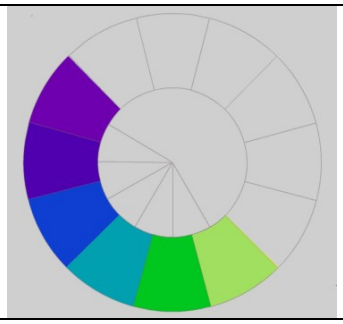
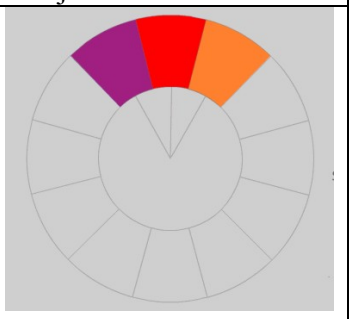
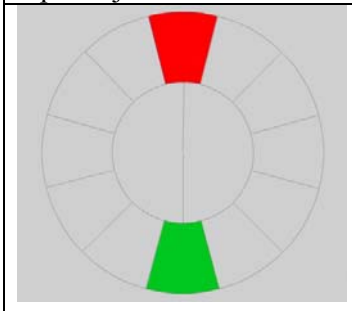
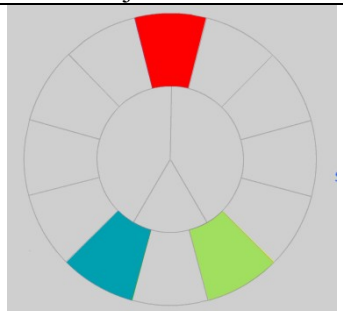
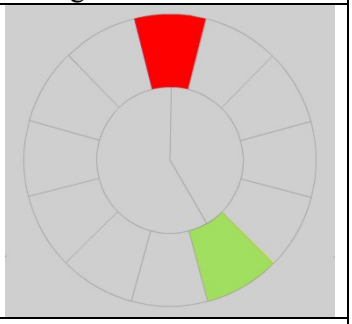
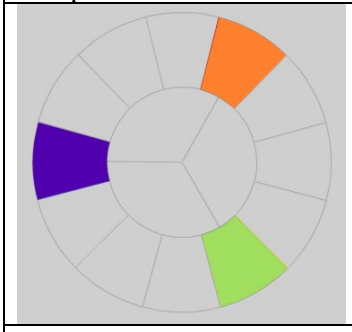
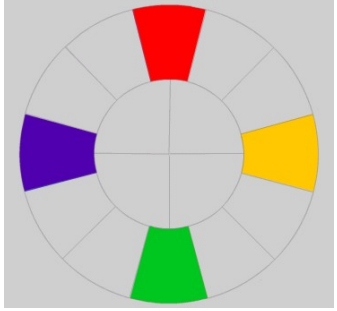
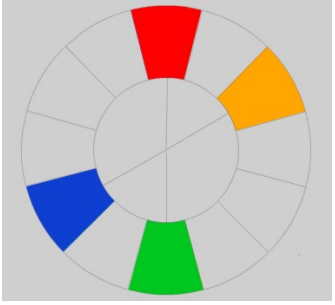
Odnosi između boja mogu se vizuelno predstaviti pomoću kruga boja. To je spektar boja omotan oko kruga.

Prvi krug boja je izumio Sir Isaac Newton. On je razdvojio bijelu sunčevu svjetlost u crvene, narančaste, žute, zelene, cyan i plave zrake i zatim spojio dva kraja spektra da pokaže prirodnu progresiju boja.

Osnove današnje teorije boja postavio je Johannes Itten, švicarski teoretičar boje i umjetnosti, koji je predavao na školi za primijenjenu umjetnost u Weimar-u, Njemačka. Itten-ov krug boja se bazira na crvenoj, žutoj i plavoj boji kao primarnoj trijadi boja i uključuje 12 boja.

Prema teoriji boje, harmonične kombinacije boja koriste bilo koje dvije boje koje stoje nasuprot jedna drugoj na krugu boja, bilo koje tri boje jednako razdvojene na krugu boja formirajući trougao ili bilo koje 4 boje tj. dva suprotna para boja.

Harmonične kombinacije boja zovu se kolor šeme ili harmonije boja.

		
primarna	sekundarna	tercijarna
		
tople boje	hladne boje	analogna
		
komplement	podijeljeni komplement	nepogodan
		
trijadna harmonija	tetradni komplement	tetradni podijeljeni komplement

Postoje sljedeće klasične harmonije boja:

- **Monohromatska kolor šema** koristi varijacije u svjetloći i zasićenosti jedne boje. Ova šema izgleda čisto i elegantno. Možemo je koristiti za kreiranje cjelokupnog raspoloženja. Primarna boja može se integrirati sa neutralnim bojama kao što je crna, siva ili bijela. Međutim, pomoću ove šeme teško je istaći najvažnije elemente.

- **Analogna kolor šema** koristi boje koje su susjedne jedna drugoj na krugu boja. Jedna boja se koristi kao dominanatna, dok se druge koriste da obogate šemu. Ova šema je slična monohromatskoj, ali ima više nijansi

- **Komplementarna kolor šema** pravi se od dvije boje koje su suprotne jedna drugoj na krugu boja. Ona izgleda najbolje kada se stavi topla boja nasuprot hladne, npr. crvena nasuprot zeleno-plave. Ova šema ima jak kontrast. Kada koristimo ovu šemu važno je izabrati dominanatnu boju, a njenu komplementarnu boju koristiti za akcente.

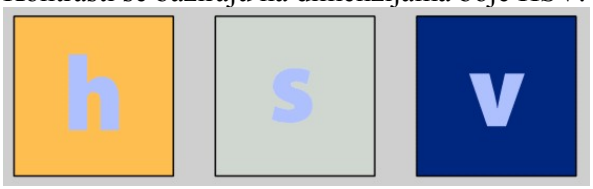
- **Podijeljeni komplement** je varijacija komplementarne kolor šeme. Ona koristi boju i dva susjeda njenog komplementa. Ova šema omogućava jak kontrast bez napetosti komplementarne kolor šeme.

- **Trijadna kolor šema** koristi tri boje koje su jednako razmaknute na krugu boja. Ova šema je popularna među umjetnicima jer nudi jak vizuelni kontrast, a održava balans i bogatost boje. Trijadna šema nije tako kontrastna kao komplementarna, ali izgleda harmoničnije.

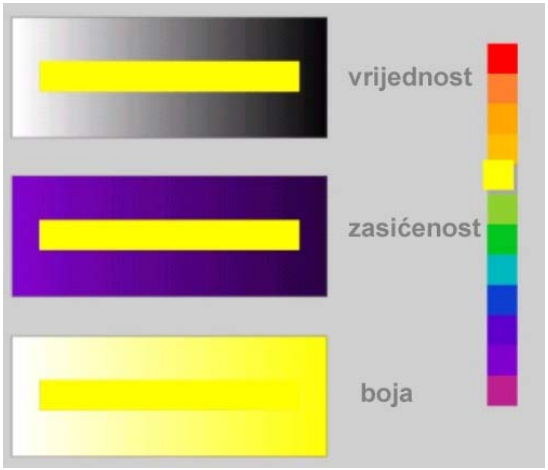
- **Tetradna ili dvostruko komplementarna** kolor šema je najbogatija od svih šema jer koristi 4 boje koje su raspoređene u dva komplementarna para. Ova šema je teška za harmonizaciju ako se sve 4 boje koriste u istim količinama. Zato treba izabrati boju koja će biti dominantna.

d) kontrasti boja

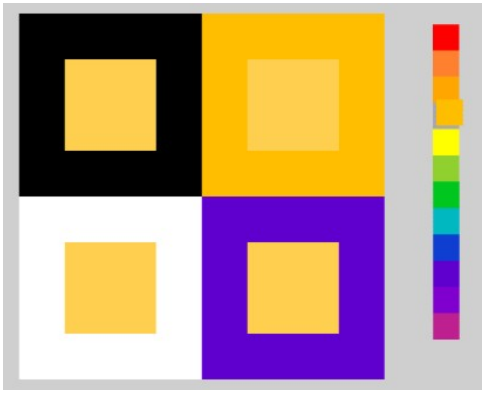
Kontrasti se baziraju na dimenzijama boje HSV.



Simultani kontrast se odnosi na vizuelni uticaj jedne boje na drugu kada se postave jedna pored druge ili jedna ispod druge.



HSV kontrast

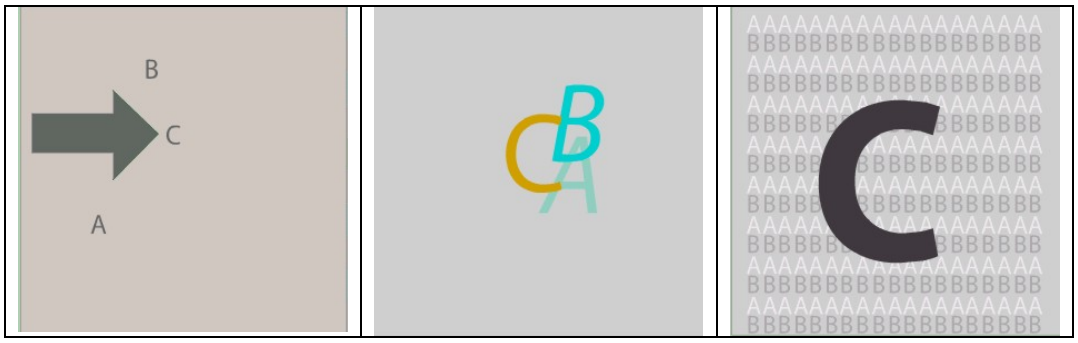


Boje svih centralnih boxova su identične. Vidljiva je razlika u njihovoj percepciji zavisno od boje koja je ispod njih.

6.3 Kompozicija i layout

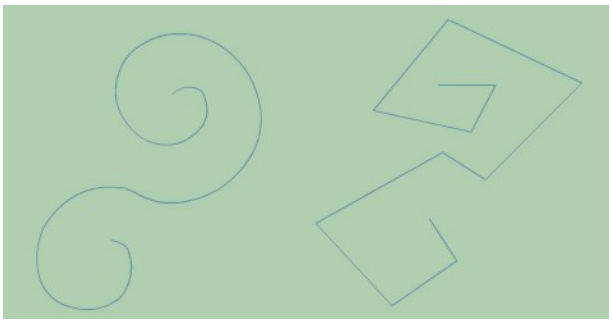
a) akcenat

Naglašavanje dijela slike koji treba da priviče pažnju gledaoca. To se postiže jačim tonovima boje ili primjenom kontrasta, debljinom linije itd. Neki od ovih načina biće opisani u nastavku.



b) elementi dizajna

LINIJA je prostorni pojam za putanju kretanja tačke ili povezani niz tačaka u prostoru, odnosno na određenoj površini. Svaka linija postavljena na čistoj ograničenoj površini dekomponuje tu površinu i na njoj ostavlja svoj statičan ili dinamičan trag, zavisno od toga da li je prava ili kriva.



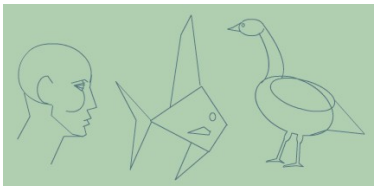
OBLIK, KONTURA predstavlja nekoliko linija različite usmjerenosti koje ograničavaju dio prostora i formiraju površine različitih izgleda.

Slijede osnovne grupe kontura:

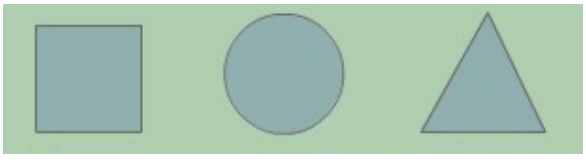
-karakteri

- simboli

- forme

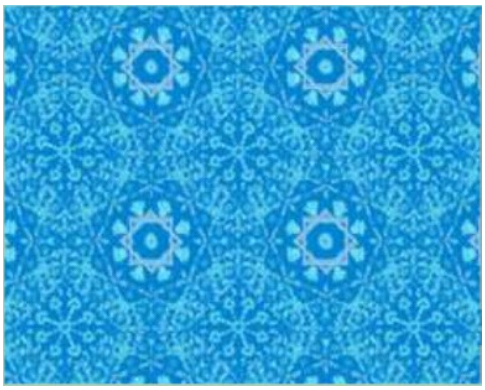


- tri osnovna geometrijska oblika



četvorougao krug trougao

TEKSTURA – izgled i utisak površine



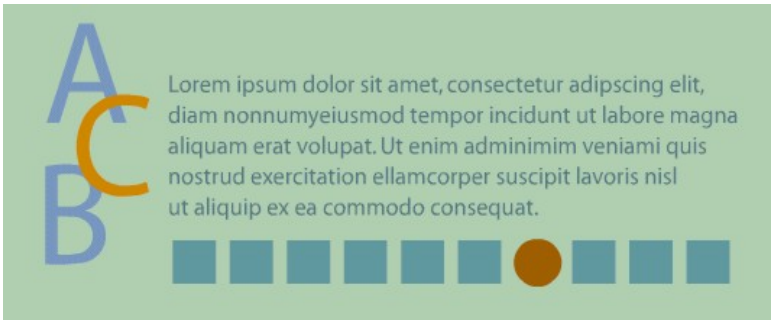
PROSTOR – udaljenost područja između stvari ili oko njih

KONTRAST – veoma je važno u dizajnu razlikovati oblik i pozadinu. Najvažnija informacija ili objekat koji je predstavlja treba da dominira u kompletnom dizajnu. Nasuprot tome, pozadina treba da bude što diskretnija kako bi se ta dominacija mogla ostvariti. Zato se uvijek koriste kontrasti između tamnih i svijetlih površina (ako je objekat u prvom planu taman, pozadina treba da bude svijetla i obrnuto). Ako je pozadina sastavljena iz više boja (npr. slika uslikana kamerom) onda se na nju primjenjuje neki blur efekat ili joj se ukida kolor.

VELIČINA obuhvata mjere, dimenzije i proporcije i u analizi i doživljaju forme ima veliki značaj. Odnosi i upoređivanje ovih vrijednosti neposredno se doživljavaju, a veličina je jedan od bitnih uslova da se izrazi forma i definiše u prostoru.

c) principi kompozicije i aranžmana prostora layout-a

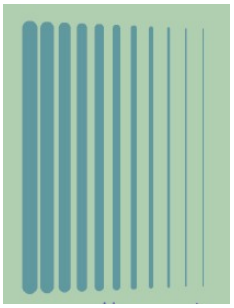
- akcenat – koji se elementi prvi primijete prilikom prvog čitanja



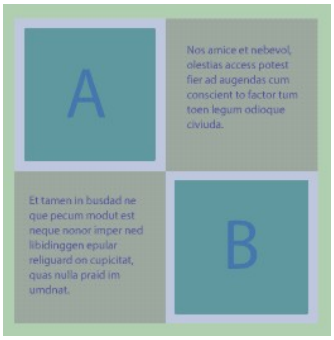
- ravnoteža – ravnomjerna raspodjela težine

simetrična asimetrična

- ritam – patern koji se dobije ponavljanjem elemenata koji se variraju
  - smirujući – elementi postavljeni u jednakim intervalima (glatki ritam) proizvode smirujući efekat



  - živi – nagle promjene u veličini i poziciji oblika proizvode uzbuđeno raspoloženje
- jedinstvo – da elementi izgledaju kao da idu jedan uz drugi. Grupisanjem, ponavljanjem ili postavljanjem elemenata na jedinice mreže može se postići uniformnost.



d) formalni sistemi kompozicije

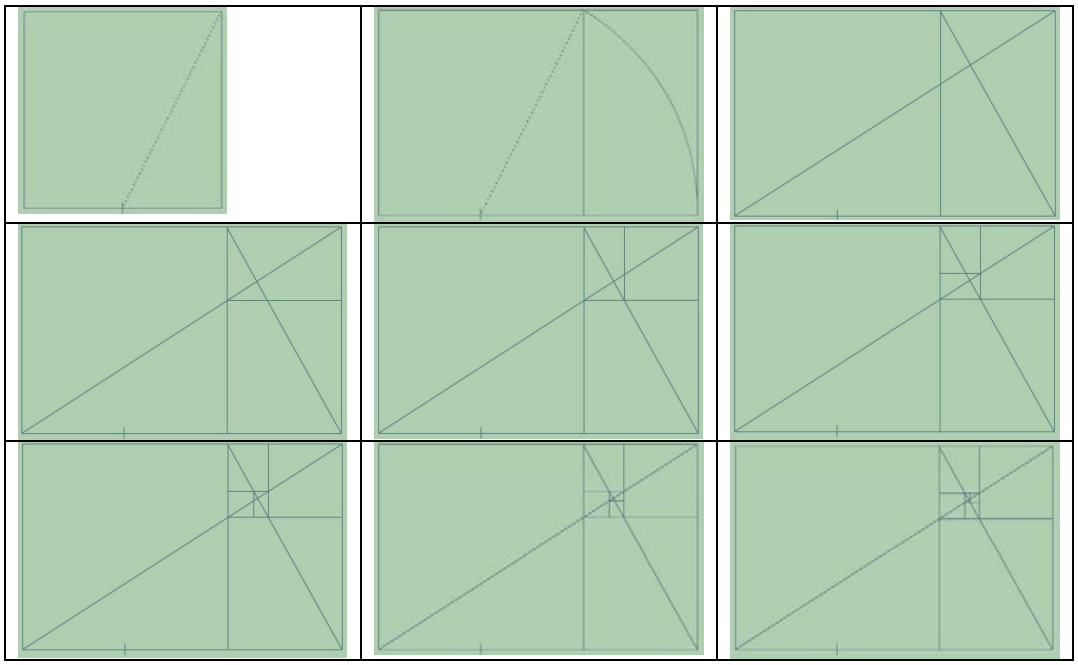
- korištenje mreže
- konstrukcija osnovnog trougla

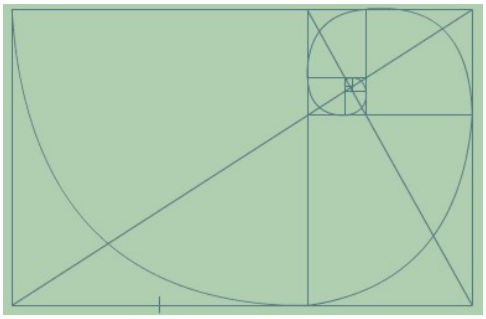
- zlatni rez

Sam pojam “zlatni rez” ili “zlatni presjek” nastao je još u starom Egiptu i zasniva se na utvrđenom redu i proporciji u prirodi. U aritmetičkom smislu, ovaj princip može se izraziti u formuli:

a+b/a = a/b

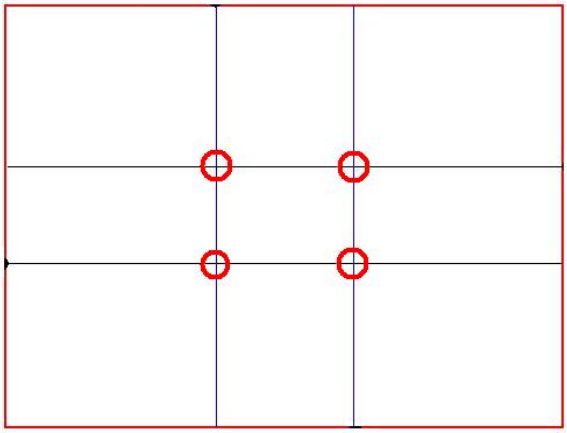
Ovo znači da se dimanzija u cjelini (a+b) odnosi prema većem dijelu (a) kao veći dio (a) prema ostatku (b).





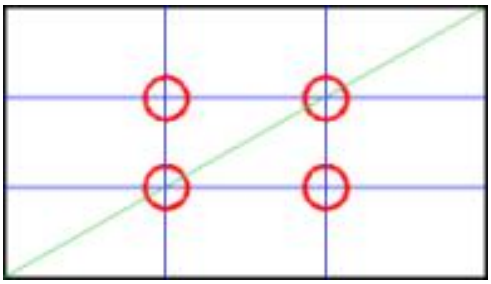
Proporcionalno smanjivanje četvorouglova može se kreirati koristeći zlatni rez.

Ako podijelimo naš ekran ili fotografiju po principu zlatnog reza dobićemo 4 tačke gdje treba da budu smješteni objekti koje želimo akcentirati.



- pravilo trećina

Ako ekran ili okvir kadra podijelimo na trećine po vertikali i horizontali, dobićemo 4 presječne tačke gdje se smještaju važni objekti

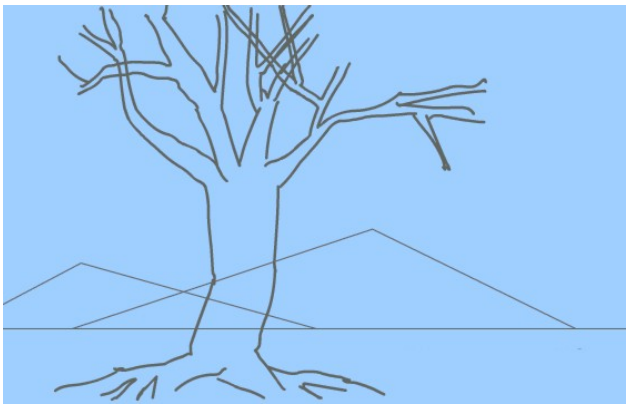


Ovaj sistem kompozicije se vrlo često koristi u fotografiji i biće detaljnije objašnjen u poglavlju koje razmatra kompoziciju kadra.

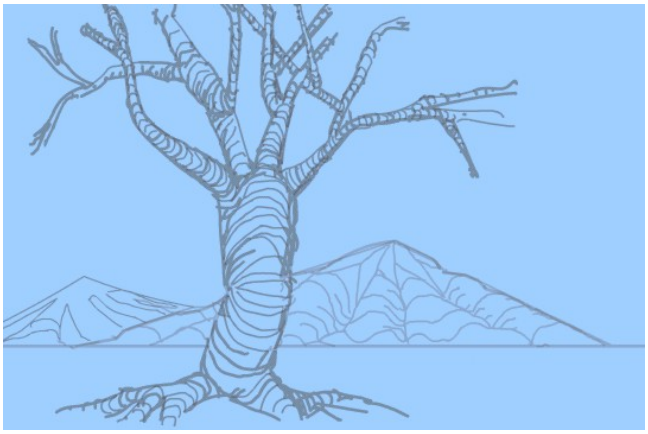
6.4 Perspektiva

a) Perspektiva je iluzija trodimenzionalnosti
- atmosferska perspektiva

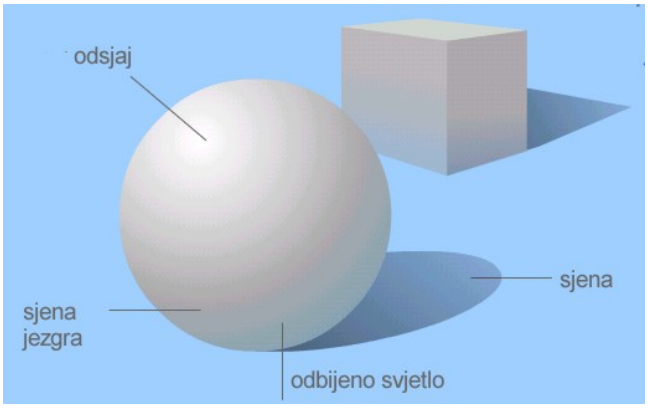
- indikatori pravca
  - linije kontura



- preklapajuće forme

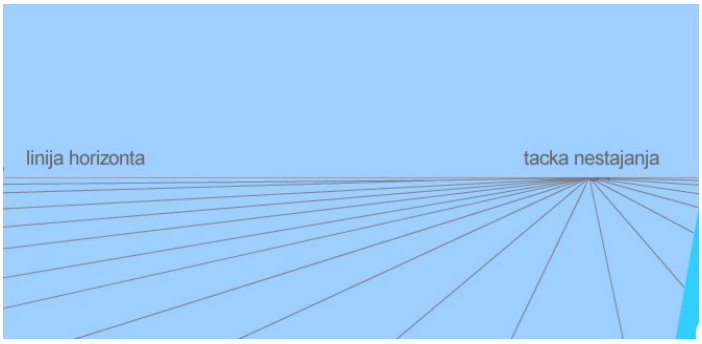


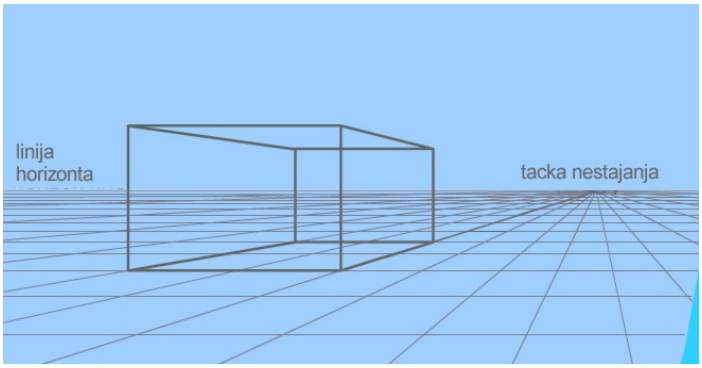
- osobine svjetla

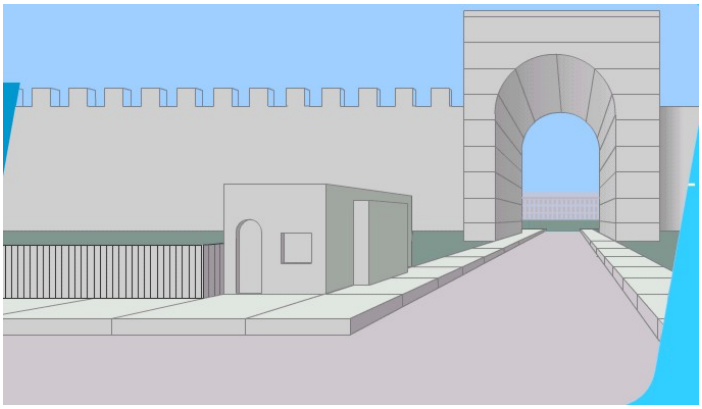


b) perspektiva jedne tačke

Linije konvergiraju prema jednoj tački nestajanja na liniji horizonta i tako kreiraju iluziju dubine.

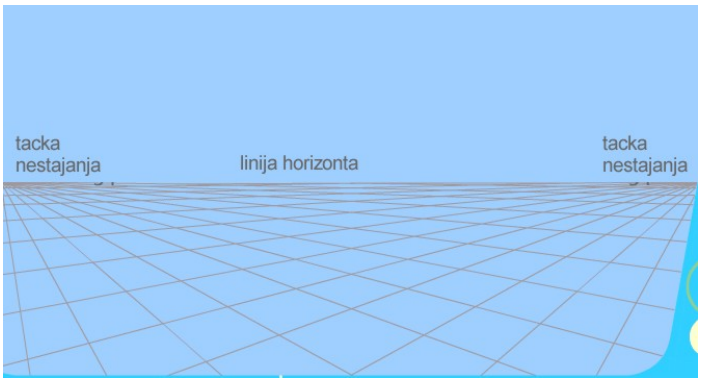


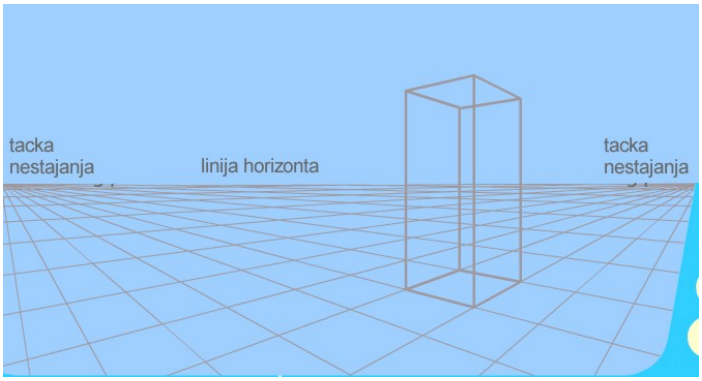


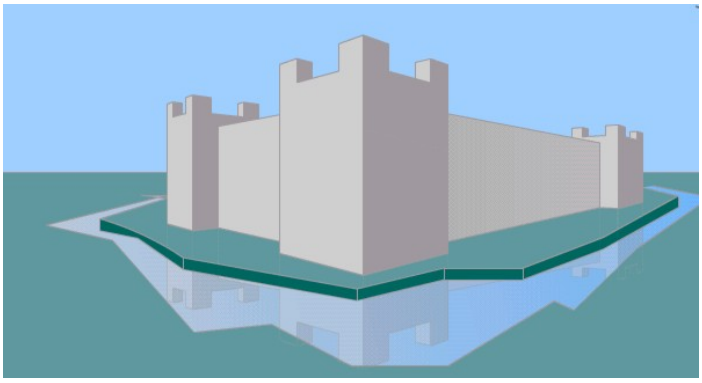


c) perspektiva dvije tačke

Linije konvergiraju prema dvije tačke nestajanja na liniji horizonta stvarajući iluziju dubine.



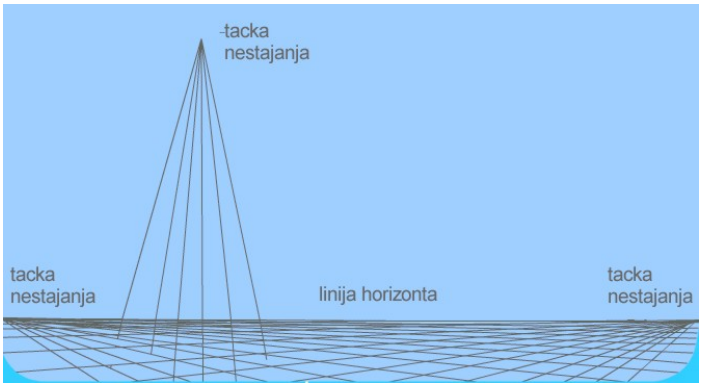


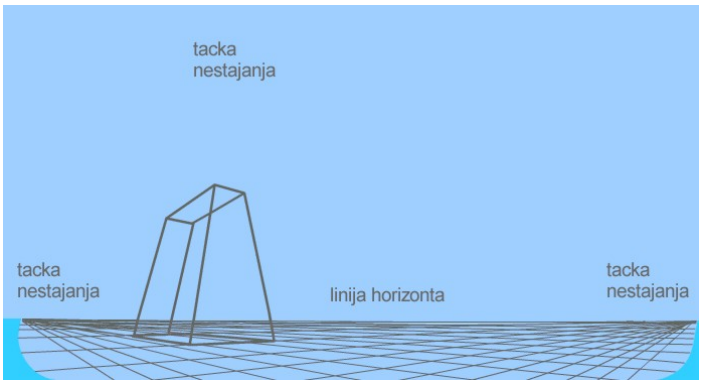


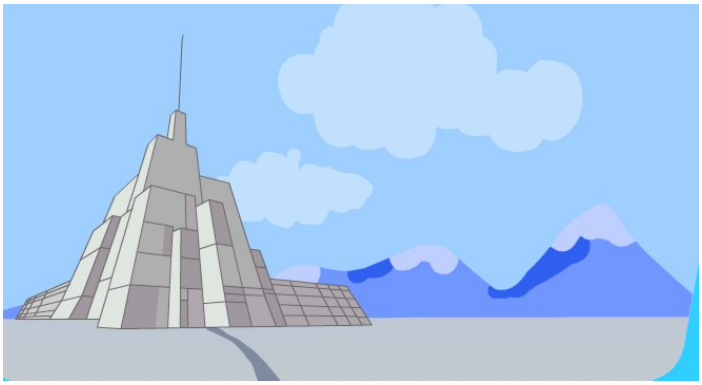
Vertikalne linije su paralelne, dok horizontalne linije konvergiraju prema tačkama nestajanja.

d) perspektiva 3 tačke

Linije konvergiraju prema tri tačke nestajanja, dvije na liniji horizonta i jednoj okomitoj na liniju horizonta, stvarajući iluziju dubine.







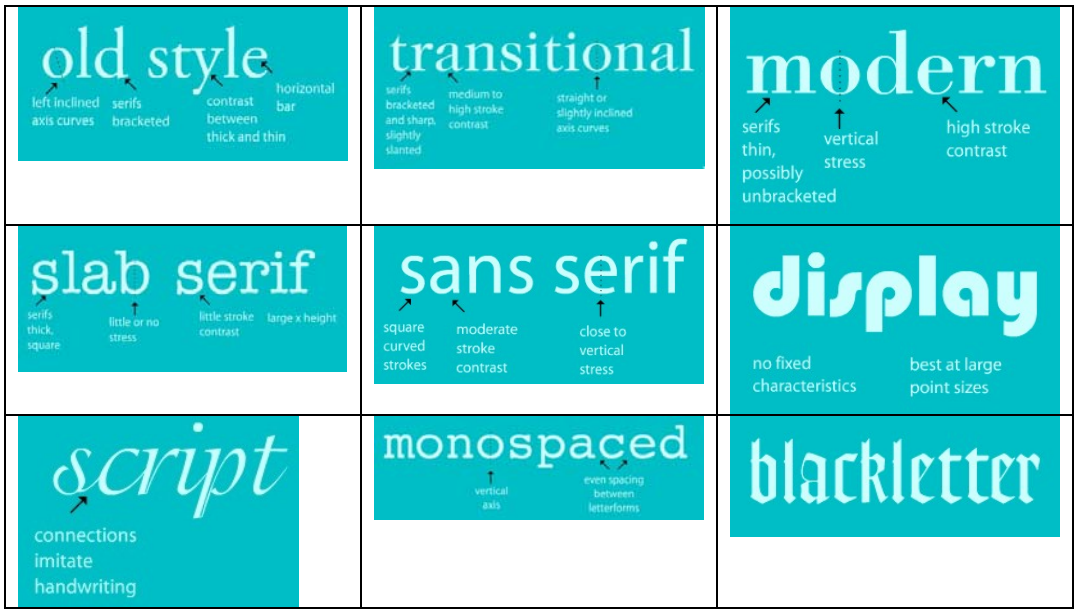
Sve linije konvergiraju prema tački nestajanja i nijedna linija nije paralelna niti okomita na horizont.

6.5 Tipografija

a) anatomija tipografije

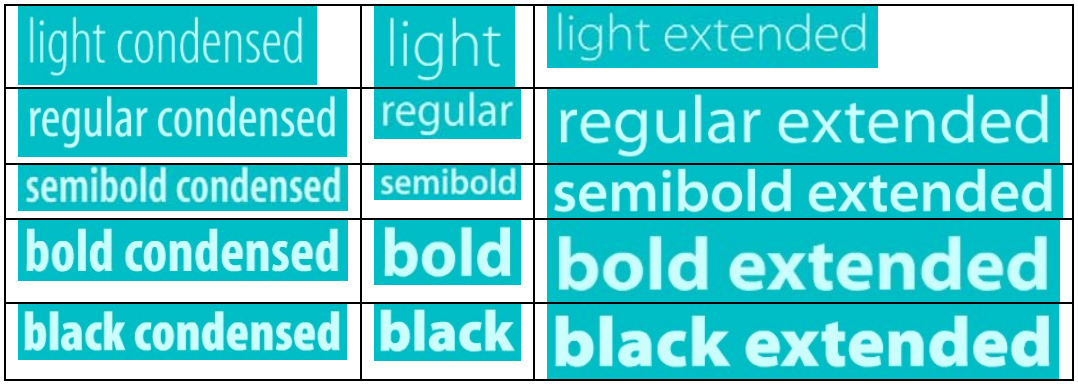
Izrazi typeface (lik ili tip slova) i font često se koriste kao sinonimi. Međutim, typeface je dizajn karaktera unificiran prema konzistentnim vizuelnim osobinama, dok je font kompletan skup karaktera u bilo kakvom dizajnu, veličini, obliku, stilu ili tipu.

b) Klasifikacija tipova slova



c) familije tipova slova

Familija tipova slova je grupa tipova koje povezuju slične vizuelne karakteristike. Familije tipova se sastoje od različitih težina i širina. Osnovni font može biti light, regular, semibold, bold i black, a svaki od tih podtipova može biti condensed i extended.

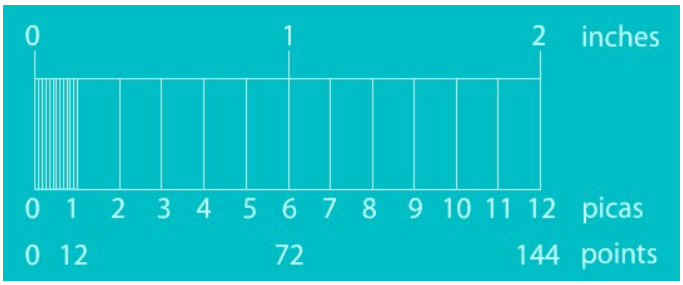


d) mjere

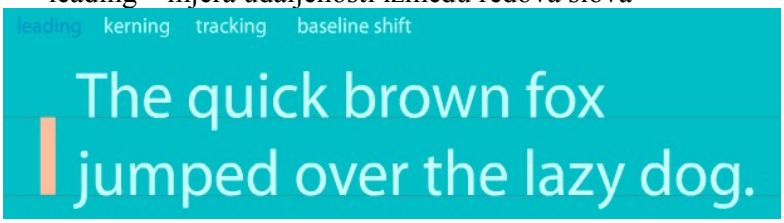
- Relativne mjere su pica i point.

6 picas = 1 inč
12 points = 1 pica
72 points = 1 inč

Monitori računara imaju ekransku rezoluciju 72 dpi što približno odgovara veličini pointa.



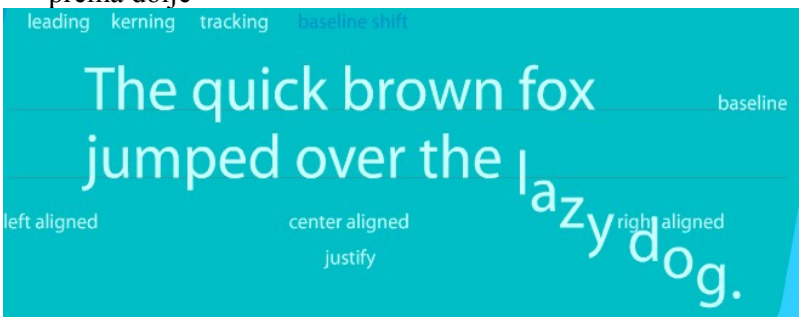
- prostorne mjere
  - leading – mjera udaljenosti između redova slova



  - kerning – mjera udaljenosti između pojedinih slova

  - tracking – mjera udaljenosti između slova unutar riječi

  - baseline shift – mjera udaljenosti između slova i bazne linije prema gore i prema dolje



Poravnanja paragrafa teksta:

- left (lijevo)
- right (desno)
- center (centralno)
- justify (ravnomjerno)

7. **Osnovi filmskog jezika i kompozicije kadra**

Poznato je da su film i televizija mediji koji imaju svoj specifični način izražavanja. Taj način izražavanja se zove filmski jezik. Kao i svaki drugi jezik, on ima svoju gramatiku, sintaksu i stil. Da bismo se bavili televizijskim dizajnom moramo poznavati osnovna izražajna sredstva filmskog jezika, kao i njegovu terminologiju.

7.1 Gramatika filmskog jezika

U filmskoj "rečenici" riječi su zamijenjene slikama. Gramatička analiza proučava funkciju svake slike koja je u sastavu filmske rečenice.

7.1.1. Plan

Plan je predstavljanje jedne misli slikom. On daje manje ili više široku sliku o nekoj sceni prema ideji koju treba da izrazi ili prema predmetu ili osobi na koju treba da privuče pažnju.

Planovi se nadovezuju jedan na drugi, da bi izazvali niz misli koje stavaraju film.

Plan ima jednu prednost nad riječi. Riječ je apstraktna i apeluje na maštu koja mora sebi da predstavi misao koju riječ sadrži. Plan je konkretan jer nam direktno daje sliku ideje.

Planovi mogu da ispune sve funkcije riječi u jednoj rečenici. Oni se mogu uporediti sa prilozima, koji određuju prirodu misli, odgovarajući na čitav niz pitanja koja bi se mogla postaviti.

Npr.
Prilog za mjesto: Gdje? Smješta radnju u prostoru. Npr. jedan salon prikazan iz svih uglova pokretima kamere.

7.1.2 Vrste planova

Slika predstavljena planom je manje ili više široka prema ideji koju izražava. Osim toga, ona prema potrebi može biti komponovana tako da privuče pažnju gledaoca na jedan predmet ili osobu koji su važni za razvoj radnje.

Kadriranje se sastoji u isticanju glavnog predmeta u okviru koji sačinjavaju ivice slike. Važna stvar ili osoba postavlja se jasno u odnosu na ivice slike, čuvajući harmoničnu ravnotežu svih elemenata koji se istovremeno pojavljuju ne odvraćajući pažnju gledaoca.

Veličina plana ustanovaljava se prema veličini glavnog predmeta u odnosu na ivice slike.

a) Opći plan

Odgovara na pitanje : Gdje?

Kao prilog za mjesto i ponekad za vrijeme, opći plan daje opći okvir radnje. To je čas geografska karta, čas ogromna panorama ili pogled iz aviona ili pejzaž iz planine ili iz ravnice. On može da smjesti radnju u ljeto, u zimu, u dan ili noć.

Opći plan iz filma "Put jednog gauča"

b) Srednje-opći plan

Odgovara na pitanje: Gdje? Kada? Kako?

Njime se ističe dekor radnje. On nam daje okvir radnje sa njenim ambijentom. On precizira mjesto u općem planu. Iz njega mogu da se izvedu različite ideje: bogatstvo ili siromaštvo, ljeto ili zima itd. bez obzira na glumce koji se tu kreću.
To može biti neka ulica, kuća ili čak dio kuće.
Ambijent će se stvoriti npr.vojnim defileom, noćnim slavljem, banketom.

Srednje-opći plan iz Flemingovog filma "Dr Džekil i g. Hajd"

c) Srednji plan

Odgovara na pitanje: Gdje? Ko?

Srednji plan sakuplja ličnosti radnje u dekoru gdje se raspoznaju detalji. On nam predstavlja glumca aktivno, nezavisno od dekora koji nam je poznat iz općeg plana. On stvara ljudsku atmosferu.

Srednji plan iz filma "Sombrero"

d) Srednje-krupni plan (ameriken)

Odgovara na pitanje: Šta rade? Zašto?

Srednje-krupni plan napušta grupu glumaca da bi nam prikazao jedan dio (najviše četiri). On ih izdvaja da bi ih uveo u radnju, dozvoljava nam da ih bolje upoznamo, dozvoljava nam da vidimo opću liniju njihove igre i da procijenimo njihove stavove. On će se npr. pokazati savršeno opravdanim kada nam pokazuje bijes glumca koji se izražava u naglim pokretima ruku i gornjeg dijela tijela.

Srednje-krupni plan (Džon Kicmiler u filmu "Živjeti u miru")

e) Krupni plan

U krupnom planu glava glumca pokriva cijelo platno. Njegovo lice nam je potpuno izloženo i po njegovim crtama i očima možemo da pročitamo šta njegov mozak misli, intimne i duboke reakcije njegove duše.

Krupni plan ima neospornu prenosnu snagu i hiljadu puta jasnije objašnjava od svih pokreta i riječi. On može također da se primijeni i na predmete, koji, zahvaljujući njemu, dobiju ulogu dramske ličnosti.

Krupni plan užasnutog ljudskog lica iz filma "Ajvanho"

f) Detalj

Vrlo krupan plan koji nam pokazuje detalj koji naše oko obično ne vidi – oko, usta, dio lica ili glave.

On ima za cilj da izoluje jedan izraz lica ili da specijalno privuče pažnju na neki detalj natijelu (npr. ožiljak) ili na neki predmet koji bi nam bilo nemoguće otkriti bez njega.

Detalj i krupni plan moraju biti povezani da bi mogli ispuniti svoju funkciju.

2. Planovi slika

Na jednoj slici se razlikuje više planova, koji se određuju prema vezi sa glavnim subjektom.

a) Zadnji plan

On grupiše sve ono što se nalazi daleko od glavnog subjekta. On se nalazi u pozadini slike, kao "rikvand".
Zadnji plan koji se vidi kroz otvorena vrata ili prozor ili zarez dekora zadnjeg plana, zove se "pozadina". Da bi mu se ulio život i da bi izgledao stvarniji, često se oživljava licima ili vozilima koja prolaze kroz njega. Zadatak mu je da da dubinu dekoru.

b) Drugi plan

Nalazi se neposredno iza glavnog subjekta.

c) Prednji plan

Sve što se nalazi ispred glavnog subjekta.
On vrlo često služi kao znak raspoznavanja, da bi oko moglo da ocijeni veličinu različitih ličnosti ili predmeta ne sceni, dubinu slike, razdaljinu koja odvaja elemente. To može da bude drvo, zid ili bilo koji predmet koji gledalac dobro poznaje.

7.1. 3. Pokreti kamere

Kamera se može pomjerati na različite načine: naprijed, nazad, u krug, tako da nam pokaže ono što moramo da upoznamo da bismo osjetili atmosferu radnje ili pratili njen razvoj. Kamera nam čas pokazuje šta vidi glumac, čas zauzima naše mjesto i navodi nas da učestvujemo u radnji ili se trudi da nam da različite utiske.

a) Panorama

Panorama je kruženje kamerom koja može da se obrće u svim pravcima, oko svoje pokretne ose. Proizvedeni efekat je sličan glavi koja gleda, okreće se lijevo, desno, diže se i spušta, dok tijelo ostaje nepomično.
Ovaj postupak se upotrebljava u narativnom smislu. Kamera ispituje cijeli predio ili prati neku osobu ili predmet koji se kreće.

b) Far

Kamera se za vrijeme snimanja pokreće zajedno sa svojim stativom.

U faru unaprijed kamera se približava jednom predmetu, jednoj ličnosti, da bi pokazala njihove detalje. Upotrebljava se u analitičkom smislu. Polazeći od jedne grupe, kamera će se približiti specijalno jednoj ličnosti, počinjući općim planom da bi stigla na krupni plan. Far omogućava "analizu" jednog detalja.

U faru unazad, ona će se udaljiti od krupnog plana glumca da bi ga stavila nazad u grupu. Njen smisao je sada sintetički, ujedinjujući u jednom prostoru elemente analizirane izbliza.

U bočnom horizontalnom faru kamera će pratiti glumca koji se kreće da bi ostao u kadru.

U vertikalnom faru kamera prati penjanje ili silaženje niz neke stepenice, radnju ustajanja ili sjedanja, lift.

7.1.4. Uglovi snimanja

Kamera je normalno postavljena na nivou gledaoca u stojećem stavu. Ugao snimanja (rakurs) se određuje drugim pozicijama.

a) Gornji rakurs

Kamera je smještena na nivou višem od predmeta koji snima. Ona je u "potpunom gornjem rakursu" ako se nalazi vertikalno iznad predmeta.

Gornji rakurs smanjuje, spljoštava ličnosti ili predmete jače ili slabije, zavisno od njegove naglašenosti. Moralno, on znači psihičku ili fizičku klonulost, slom, smrvljenost ili opsesiju. Ličnost snimljena odozgo nadolje izgleda nam mala i jadna.

Lice u krupnom planu, snimljeno odozgo nadolje, daje utisak intelektualne slabosti, ludila.

Gornji rakurs pravi grotesku: čovjek snimljen od glave ka nogama izgleda smiješno.

Gornji rakurs se može upotrijebiti i da nam pokaže šta glumac vidi odozgo sa sprata, stepenica.

Gornji rakurs iz filma "Sami u svijetu"

b) Donji rakurs

Kamera je smještena niže od predmeta koji se snima. Vertikalno ispod predmeta biće u "potpunom donjem rakursu".

Donji rakurs povećava značaj ličnosti ili stvari, uvećava ih manje ili više prema njegovom naglašavanju. Moralno, on znači moć, snagu, prevlast, moralnu veličinu, autoritet, ponos, nadmenost.
^ovjek snimljen odozdo nagore daje utisak fizičke snage koja ruši.
Lice snimljeno odozdo nagore izgleda životinjski.
Donji rakurs pravi izdužen, povećan i elegantan stas ličnosti.
Također se može upotrijebiti da nam pokaže šta glumac vidi odozdo.

Donji rakurs iz filma "Julije Cezar"

7.1.5. Kadar

Kadar se sastoji od snimljenog materijala koji kamera može da registruje u jednom svom neprekidnom radu.

Promjenljivost kadra

a) nepokretni kadar: kamera snima cijelu radnju ne pomjerajući se, pod istim uglom
b) pokretni kadar: za vrijeme snimanja kamera ide naprijed i nazad (far), okreće se oko ose (panorama), ide nadole i nagore (donji i gornji rakurs), da bi nam prikazala različite vidove radnje, da bi pratila ličnosti u njihovom kretanju i da bi nam saopćila različite utiske svojstvene svakom od njenih pokreta.

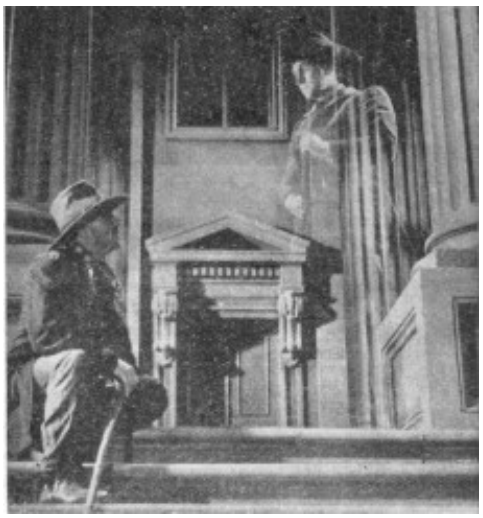
Plan i kontraplan

a) plan: kamera, okrenuta u jednom pravcu, registruje radnju, predmet i jednu ili više osoba.
b) kontraplan: kamera, okrenuta u suprotnom pravcu, registruje istu radnju, isti predmet i iste ličnosti, koje su date s lica, dok su u kadru bile viđene s leđa.

Tu se premješta kamera, a ne glumci.

Dvostruka ekspozicija

Na glavnoj slici je utisnuta druga, providna slika.



Dvostruka ekspozicija

U filmu nam dvostruka ekspozicija na konkretan način prikazuje misao i snove glumca. U zadnje vrijeme vrlo često se koristi compositing, što je neka vrsta višestruke ekspozicije, gdje broj slika koje se vide jedna preko druge nije ograničen. O compositingu će biti riječi u posebnom poglavlju.

7.1.6 Interpunkcija

Slike (planovi) koje se nižu jedna za drugom uobličuju jednu misao, stvaraju filmsku rečenicu.
Rečenice stvaraju pasuse, poglavlja, dijelove, da bi na kraju formirale film. Kao u književnosti, pojedini dijelovi se odvajaju znacima interpunkcije. Filmska interpunkcija je skup pravila koja regulišu pauze u radnji.

a) Odtamnjenje

Slika postepeno izlazi iz potpune tame. Upotrebljava se na početku jednog dijela ili cjeline.

b) Zatamnjenje

Slika postepeno nestaje u crnom. Upotrebljava se za završetak cjeline.

c) Zatamjnjenje i odtamnjenje

Kombinacija koja se upotrebljava za odvajanje dvije cjeline.

d) Pretapanje

Slike scene koja se završava postepeno nestaju, a istovremeno se postepeno pojavljuju slike sljedeće scene. Obilježava prelaz u vremenu i prostoru.

7.1.7 Sekvenca

Sekvenca je niz planova koji stvaraju jednu cjelinu i potpuno izražavaju jednu misao. Zbog toga sekvenca može imati širi smisao nego rečenica u literaturi. Ona može uzeti proporcije pasusa ili čak cijelog poglavlja.

Vrste sekvenci

a) Nezavisna

Ova sekvenca ne zavisi ni od jedne druge i nijedna druga ne zavisi od nje. Nema, znači, nikakve veze između prethodne i sljedeće sekvence. Ona izražava misao koja zavisi od radnje i u sebi sadrži jednu cjelinu.

b) Glavna

Ova sekvenca ne zavisi ni od jedne druge, ali jedna ili više drugih sekvenci zavise od nje. Ona izražava potpuno jednu misao na koju se logično nadovezuje druga. Potrebno

je da se ona odgovarajućim postupkom interpunkcije priključi onima koje od nje zavise.

c) Potčinjena

Ova sekvenca zavisi od neke druge i upotpunjuje je; s druge strane može biti i njoj potrebno da se upotpuni jednom ili sa više sekvenci koje iz nje proističu. Ona može biti direktna, indirektna, u vezi sa okolnostima itd.

Uloge sekvence su da stvara djelo i da stvara ritam filma.

Unutar cjeline filma sekvence mogu biti

- u uvodu – sekvenca ekspozicije
- u glavnom dijelu
  - opisne sekvence
  - sekvence akcije
  - retrospektivne sekvence
  - eksplikativne sekvence
- u raspletu – finalna sekvenca

7.1.8 Ritam

Ritam se stvara pomoću načina nizanja slika, po tempu i nizu planova. On se upotpunjuje zvucima i muzikom koji prate radnju.

Elementi ritma

a) Slika

Ritam u slikama se može izraziti brzinom kretanja predmeta ili ličnosti, kao i brzinom pokreta kamere.

b) Sekvenca

Veličinom svojih planova: sve većim i većim (od općeg do krupnog plana) ili sve manjim i manjim (od krupnog plana do općeg), ona povećava važnost predmeta ili je umanuje.

c) Sekvence međusobno

Svojim sve dužim ili sve kraćim trajanjem one usporavaju radnju ili je ubrzavaju. Izostavljanjem nepotrebnih elemenata radnja se zgušnjava i pažnja gledaoca ne rasipa.

d) Šumovi

Šumovi bilo koje prirode doprinose realnosti slika. Njihovim ubrzanim ili usporenim ponavljanjem oni daju svoj doprinos ritmu filma i pojačavaju emocije koje postoje kod gledaoca. Važno je da njihov ritam odgovara ritmu sekvence koju prate.

e) Riječi

Dobro prilagođene, malobrojne riječi idu u dubinu i direktno pogađaju gledaoca. Njihov manje ili više brz tok utiče na ritam radnje, stvara različIte emocije, prema njihovoj sadržini, vrsti snazi i brzini govora.

f) Muzika

Muzika se upotrebljava

- da se stvori realni ambijent scene,
- da bi se stvorio zvučni fon za radnju i atmosfera mjesta gdje se ona događa
- da bude imitacija pokreta, krikova, riječi, šumova i tada ima komičan efekat
- prati osjećanja da bi ih istakla

g) Tišina

Tišina je ponekad rječitija od šumova, riječi i muzike. Umjetnost je znati stvoriti tišinu. Ona često prethodi trenucima zebnje, očekivanju značajnih događaja.

Sporo izvlačenje davljenika za noge treba da pojača napetost kod gledalaca koji ne znaju čije će se lice postepeno pojaviti iz vode.

7.2 Kompozicija kadra

Pravila fotografske i filmske kompozicije su pravila, izvedena iz raznih područja ljudske djelatnosti koja su se razvila prije pojave fotografije kao npr. slikarstva, dizajna i arhitekture, i dobro ih je poznavati, kako bi se objekti na fotografiji ili u kadru pravilno i tečno rasporedili i činili skladnu cjelinu.

Pravila koja su u nastavku objašnjena za fotografiju, primjenjuju se analogno i na kadar filma.

Kao što piše Đulijano Belić u svojoj “Maloj školi fotografije”, jedno od osnovnih pravila kaže da na fotografiji ne bi smjelo biti previše objekata, odnosno da "gužva" nije poželjna. Prevelik broj objekata na fotografiji dovodi do toga da naše oko ne zna koji bi prije pogledalo, odnosno glavni objekat kojeg smo fotografirali gubi na značenju i jasnoći dodavanjem prevelikog broja dodatnih objekata.

Prilikom gledanja fotografije, naše oko mora lako zapaziti koji je glavni objekat na njoj.

Lijeva fotografija prikazuje 3 barke i jedrilicu. Oko ne zna koji je objekat najvažniji, je li to jedrilica kao najveća, ili barke jer ih je 3, ili crvena barka jer se jarka boja ističe.

Desna fotografija ne ostavlja nikakvu dilemu.

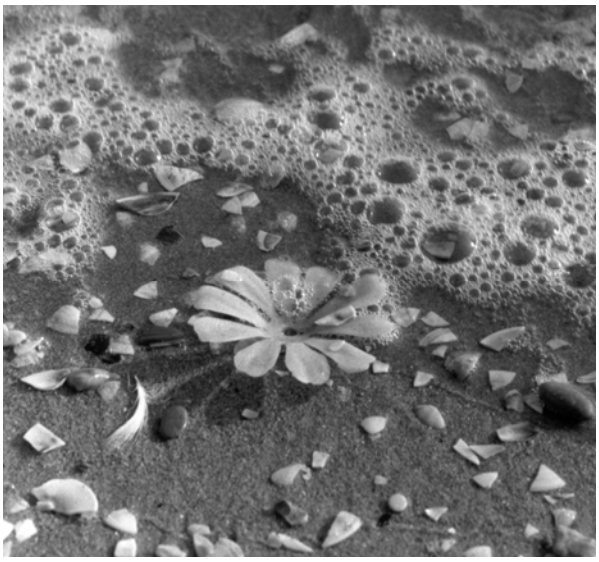
Lijeva fotografija prikazuje prekrasan tulipan, ali pozadina je previše uočljiva. Zelena boja i stabljika previše se ističu i ometaju pogled u uživanju tulipana.

Desna fotografija je mnogo bolja. Pozadina ne ometa pogled na tulipan, a daje mu ugodan kontrast.

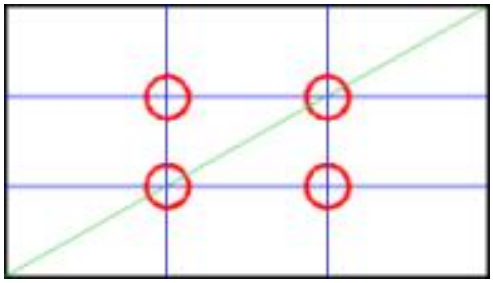
Zbog toga se u fotografisanju portreta često koristi tzv. široki objektiv koji zamagljuje pozadinu, a ostavlja oštar objekat u prvom planu.

Pravilo trećina

Objekat koji snimamo ili fotografiramo postavljen u sredinu kadra djelovaće statično i nezanimljivo. Poželjno je lagano odmaknuti objekat od centra kadra.



Zamislite da je kadar podijeljen na tri dijela po horizontali i vertikali (plave linije). Crveni kružići kao mjesta presijecanja linija su mjesta na koje možete postaviti objekt koji snimate. Koji ćete od četiri centra odabrati ovisi od toga šta snimate. Ponekad ćete moći birati gdje postaviti objekt, a ponekad će situacija sama odrediti mjesto objekta u jedan od 4 presjeka.



Ukoliko je objekat veći, npr. čovjek u uspravnom položaju, onda ga možete smjestiti u dva kružića, i to lijeva dva ili desna dva. Isto tako ako je neki objekt horizontalan (npr. horizont) smjestite ga na gornja ili donja dva kružića po horizontali.

Zelena linija predstavlja dijagonalu koja povezuje dva najbolja sjecišta. Pogled onoga koji gleda kadar ulazi u kadar dolje lijevo, prolazi kroz prvo sjecište, kroz centar kadra, i dolazi do sjecišta gore desno, gdje prestaje gledati u središte interesa u kadru. Oko obično "ulazi" u kadar s lijeve strane jer je naučeno da čita tekstove na taj način, pa se to preslikava i na fotografiju, odnosno film, dok je uzlazni tok zanimljiviji od silaznog (koji bi išao od gornjeg lijevog do donjeg desnog ugla). Ova je jedno od najvažnijih pravila kompozicije i zovemo ga "pravilo trećina".

Prvu fotografiju s cvijetom smo malo prekadrirali, i postavili cvijet u jedan od četiri centra (crveni kružić). Sad je cijela fotografija mnogo zanimljivija.

Smjer kretanja objekta

Ukoliko na fotografiji imate motiv koji je neutralan po pitanju gledanja / kretanja u nekom smjeru (statičan je), onda ostavite mjesta s lijeve strane, jer je ljudsko oko naučeno da čita s lijeve strane na desnu, pa tako "čita" i fotografiju. Zato je dobro ostaviti malo mjesta da oko "uđe" u fotografiju baš s lijeve strane.

Na prvoj fotografiji desno vidimo kako je prekršeno pravilo trećina, i po vertikali i po horizontali. Osim toga, nije dovoljno naglašen smjer kretanja djevojčice tako da se ostavi više prostora na strani u smjeru kretanja.

Fotografija lijevo je korigirana, i djevojčica je i po vertikali i po horizontali tačno po pravilu trećina, a osim toga je naglašen smjer kretanja.

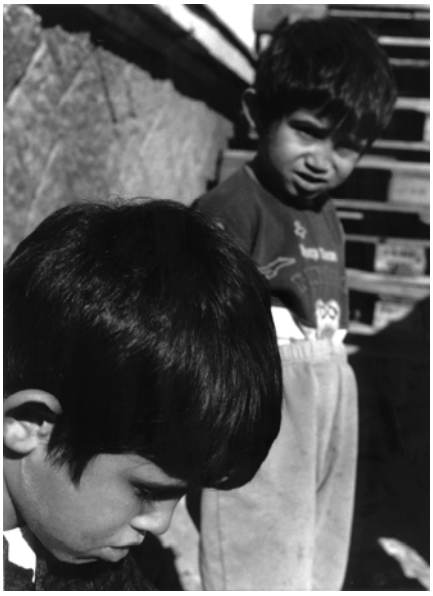
Simetrična ravnoteža

Na fotografiji desno, ravnoteža je postignuta podjednakom količinom prostora koje zauzimaju nebo i građevina od kamena. U ovom slučaju imamo apsolutnu ravnotežu po principu 50% - 50% prostora na fotografiji.

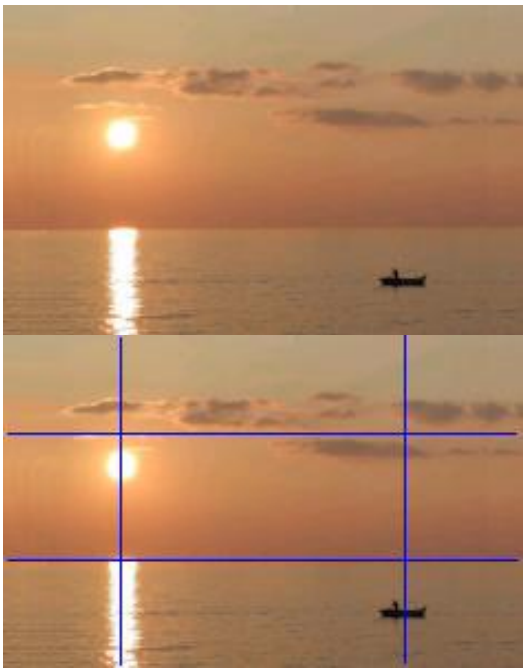
Sličan se efekat može dobiti npr. ravnotežom tamnih i svjetlih dijelova fotografije.

Nesimetrična ravnoteža

Dva dječaka na fotografiji su u savršenoj ravnoteži. Pritom nije bitno što nisu iste veličine, dapače, kad god možete nastojte napraviti nesimetričnu ravnotežu



Ostale ravnoteže



Pogledajmo fotografiju gore lijevo. More i nebo s oblacima na prvi pogled nisu u ravnoteži, jer more zauzima manje mjesta nego nebo i oblaci. No sjetite se pravila trećina, i napomene da podjela na dva jednaka dijela nije poželjna. Ovdje je fotograf procijenio da je nebo važnije od mora, i dao nebu 2/3 a moru 1/3 prostora na fotografiji. I to je savršena ravnoteža. Oko odmah primiječuje taj balans, ali tačno osjeća da je nebo važnije, pa će pažljivije pogledati nebo, dok će more više biti kao dodatni ukras.

Pogledajmo sad sunce na nebu i barku na moru. Vidimo da su ta dva objekta postavljena u suprotnim krajevima, a osim toga su ta dva objekta podjednako udaljeni od horizonta i time su oni u ravnoteži. Dodatnu ravnotežu postigli smo jakim sjajem sunca i crnom (dakle suprotnom) bojom barke.

Na desnoj fotografiji smo dodatno ucrtali linije po pravilu trećina, i vidimo da one vertikalne nisu baš savršene (ne dijele po vertikali fotografiju na tri jednaka dijela), ali su i one u ravnoteži! Lijeva vertikalna je udaljena od lijevog ruba tačno kao i desna vertikalna od desnog ruba.

Fotografija lijevo nije u ravnoteži, jer izgleda kao da će završetak dizalice svaki tren pasti na zemlju. Desna fotografija pak prikazuje i kran dizalice na kojem visi kraj dizalice, i fotografija je u ravnoteži

Linije i njihova primjena

Uspravne linije

Fotografije prikazuju Augustov hram u Puli. Na njemu su jako izražene uspravne linije. Fotografija lijevo prikazuje pogled s određene udaljenosti. No ukoliko želimo dodatno istaknuti grandioznost spomenika, možemo se približiti objektu fotografiranja i snimiti ga iz podnožja, kako prikazuje desna fotografija. Uspravne linije potenciraju visinu, veličanstvenost. Kako ponekad ne znamo kolika je visina objekata, dobro je u fotografiju uključiti i neki objekat poznate visine, npr. čovjeka, kako bi se imao osjećaj koliko je šta visoko.

Horizontalne linije

Horizontalne linije naglašavaju spokoj, odmor, mirnoću. Kao kad se lagano penjete uz stepenice, bez imalo žurbe. U njima nema užurbanosti i uzbuđenja.

Dijagonalne linije

Na dvije gornje fotografije jako su izražene dijagonalne linije. Ograda i linije na fasadi na fotografiji lijevo, te način na koji su poredani automobili na fotografiji desno jako naglašavaju prikaz naše osnovne namjere na fotografijama, a to je kretanje u dubinu po dijagonali. Dijagonalne linije snažno naglašavaju kretanje.

Linije koje se sužavaju s jedne strane

Korištenje linija koje se s jedne strane sužavaju i sakupljaju prema drugoj strani naglašavaju na fotografiji lijevo da je naš osnovni motiv kraj fotografije, odnosno naglašavanje kretanja u tom smjeru.

"S" linija (linija ljepote) - je najzanimljiviji oblik linija. Ima zanimljiv i oku vrlo ugodan oblik. Daje fotografiji jaku dozu šarma i privlačnosti, pa je zato i nazvana linija ljepote.

8. Osnovi VRML-a

Prije nego što počnemo objašnjavati osnove jezika za modeliranje virtuelne realnosti, moramo napomenuti da se virtuelni svjetovi nastali kao proizvod programiranja u ovome jeziku prikazuju na kompjuteru pomoću posebnog VRML browser-a.

Jedan od najrasprostranjenijih browsre-a za VRML je Cosmo Player koji je shareware i može se downloadovati sa stranice www.cai.com/cosmo/.

Ovaj browser se instalira kao plug-in na Internet Explorer i aktivira se prilikom pristupa fajlu sa ekstenzijom .wrl

Struktura, sintaksa i koncepti

Prva formalna specifikacija jezika za modeliranje virtualne realnosti (VRML - Virtual Reality Modeling Language) je obznanjena 2. novembra 1994. godine. Od toga datuma pa do danas ovaj jezik se svakodnevno razvija.

Prva specifikacija ovog jezika je bila VRML 1.0 kao prvi pokušaj kreiranja Internet 3D jezika.

VRML 2.0 unio je promjene u strukturi fajla i dodata je mogućnost animacije.

Standardizovani VRML 2.0 sada se koristi pod imenom VRML 97.

VRML zauzima posebno mjesto u spektru programskih jezika, negdje između jezika opće namjene kao što su C++ ili Lisp i jezika za opis i formatiranje stranice kao što su PostScript ili HTML. Ovo proizlazi iz činjenice da iako VRML ima veoma specifičnu namjenu - opisivanje trodimenzionalnih scena i objekata - njegov zadatak je tako kompleksan da su mu potrebne mnoge osobine generaliziranih programskih jezika. To omogućava, bez obzira na inicijalnu kompleksnost za učenje, ne samo uvođenje treće dimenzije u Web, nego i neograničene mogućnosti daljeg razvoja.

Da bismo napravili prve korake u treću dimenziju, moramo razumjeti strukturu i sintaksu VRML-a. Najprije ćemo se upoznati sa osnovnim karakteristikama jezika, opisujući VRML-ov file format, objasniti nodove sa njihovim osobinama i ponašanjem, a zatim se posvetiti nekim važnim tipovima variabli i primitivama ugrađenim u VRML.

8.1 Osnovne karakteristike VRML-a

VRML je u osnovi način grupisanja objekata. Kako je izveden iz SGI-ove (SGI-Silicon Graphics) Open Inventor grafičke biblioteke i file formata, VRML - ov poseban fokus su 3D grafički objekti. Ovi objekti mogu sadržati bilo koji tip podataka i, kao što ćemo vidjeti, od toga zavisti mogućnost proširivanja VRML-a.

Kako jezik radi sa objektima, možemo ga ubrojati u objektno orijentisane programske jezike. VRML-ovi objekti, koji se zovu **nodovi**, mogu virtualno sadržati bilo koji tip

informacije i mogu imati djecu. Oni se mogu također ponovo koristiti kroz ‘instanciranje'. Da bi nodovima omogućio jednostavan način međusobnog uticaja, VRML istupa iz tipične tendencije objektno orijentisanih jezika da su im individualne komponente nelinearne ili slučajno organizovane. Naprotiv, VRML source file je graf scene, hijerarhijski fajl koji određuje sekvencijalni redoslijed parsiranja nodova. Ovo omogućava programeru da kontroliše redoslijed renderovanja scene i bolju kontrolu nad njenim izgledom.

Zbog toga, koncizan opis VRML-a može biti formuliran na sljedeći način: **VRML je objektno orijentisani, 3D grafički jezik za modeliranje čiji se izvorni skript interpretira linearno, od početka do kraja fajla, da bi se izrenderovala scena.**

Koncepti kontrolne strukture - stanje i separatori

Linearnost je krucijalna da bi jedan nod mogao uticati na stanje drugoga. Npr, nod koji opisuje izvor svjetla će uticati na sve nodove koji slijede iza njega. Ovaj uticaj na stanje noda može biti ograničen upotrebom separatora - jedne od dvije kontrolne strukture u VRML-u. Ovo izvršava push i pop operacije nad tekućim stanjem, odvajajući separator i njegov sadržaj od ostatka scene.

Nakon nailaska na separator, push spašava tekuće stanje grafa scene u memoriju. Objekti djeca separatora se interpretiraju crtanjem nodova ili mijenjanjem stanja sljedećih nodova. Kada se naiđe na kraj separatora, pop restorira spašeno stanje za nastavak izvršenja grafa scene.

Druga kontrolna struktura je prosta grupa, koja se upotrebljava kad god nod ima djecu nodove. Ona ne izvršava nikakvu operaciju nad tekućim stanjem, omogućavajući mu da prođe sve objekte i ostane nepromijenjeno.

Nakon predstavljanja nekih VRML-ovih definiranih nodova, vidjećemo primjere obje ove kontrolne strukture.

Nodovi

Kako je sve u VRML-ovom grafu scene nod objekat, specifikacija noda mora omogućiti izuzetnu fleksibilnost. Nod ima veliki skup karakteristika koje mora imati, i u nekim slučajevima, uputstva kako će te karakteristike biti izražene. Osobine noda su sljedeće:

Tip objekta. Verzija 1.0 VRML-a definira 36 nodova i mehanizme kako nodovi mogu biti definirani u budućnosti. Zasad nodovi mogu biti oblici ili načini crtanja oblika, kao što je tekstura, izvor svjetla, ugao rotacije ili nekla vrsta grafičke transformacije.

Varijable sa parametrima koji opisuju veličinu, boju, rotaciju ili bilo koju drugu karakteristiku sa opsegom mogućih vrijednosti. Ove varijable se u VRML-u zovu **polja** i nod može imati nula ili više polja.

Ime noda. Ono je opcionalno i nod može funkcionirati bez njega. Međutim, postojanje imena povećava snagu i fleksibilnost noda i cijele scene. S jedne strane to

omogućava ponovno korištenje noda, a s druge strane pruža način manipuliranja objektima unutar VRML izvornog koda ili izvana. Ako damo nodu ime, ono ne mora biti jedinstveno, ali to mora biti njegovo jedino ime.

Neki nodovi mogu imati djecu nodove, ili druge objekte koje nod potpuno sadržava. Ovaj tip noda se zove **grupni nod** i njegova djeca se prolaze linearno kada se scena čita i renderuje. Grupni nod može imati nula ili više djece. Kao što je ranije spomenuto, ovakvo grupisanje ne vrši push i pop stanja scene kada se prolazi. Ono se prvenstveno koristi za opisivanje kompleksnih objekata koji se sastoje iz mnogo primitiva.

Nodovi koje definiše VRML 1.0 mogu se podijeliti u tri klase: oblik, osobina i grupa. Nodovi oblika su jedini objekti u VRML-u koji zapravo prouzrokuju pojavu objekta na sceni. Nodovi osobina utiču na način kako se ti objekti crtaju na sceni i uključuju elemente kao što je osvjetljenje i materijali. Grupni nodovi organizuju i pridružuju nodove tako što ih sadrže. Grupni nod i sva njegova djeca se tretiraju kao jedinstven objekat. Neki tipovi grupnih nodova mogu vršiti kontrolu da li će se dijete crtati ili ne.

Koordinatni sistem

Koristeći tipičnu matematičku reprezentaciju 3D prostora, VRML izražava svoj svijet pomoću x, y i z osa u Kartezijanskoj projekciji. Pozitivna z-osa projektuje se u ravan pogleda, a negativna van, prema posmatraču. X-osa leži na dnu ravni pogleda, i njen pozitivni dio je usmjeren nadesno. Pozitivna y-osa počinje na dnu ravni pogleda na lijevoj strani i njen pozitivni dio je usmjeren prema gore. Koordinatni početak (0 0 0) zato leži u donjem lijevom uglu ravni pogleda.

Standardna jedinica mjere za skaliranje je metar, sa dužinama i udaljenostima izraženim na ovaj način. Uglovi su izraženi u radijanima.

8.2 Sintaksa i struktura fajla

VRML file format

Svaki fajl koji sadrži VRML izvorni kod mora početi linijom:

```
#VRML V2.0 utf8
```

Ovo upozorava browser o sadržaju u formatu fajla koji slijedi.

Napomenimo da je VRML case senzitivan, tj. da nije svjedeno da li se ime nekog noda ili polja piše malim ili velikim slovima.

Bilo koja linija koja počinje sa oznakom # će se tretirati kao komentar. Ove komentare server koji dostavlja dokument browseru ne mora čuvati, a i browser može bit konfigurisan da ignoriše komentare tokom čitanja.

Prazan prostor u VRML kodu koji nije potreban za parsiranje također se može ukloniti iz fajla prije prenosa kroz mrežu.

Nakon prve linije koja služi za identifikaciju, fajl sadrži jedinstven VRML nod. Ovaj nod može biti grupni nod koji sadrži veoma veliki broj djece.

Ekstenzija u imenu fajla za VRML fajlove je ‘.wrl', skraćenica od world (svijet) uspostavljena kao konvencija u Verziji 1.0 VRML specifikacije. Najmanje dva druga izvora zastupaju primjenu ekstenzije ‘.vrml', što je logično jer i HTML fajlovi imaju ekstenziju ‘.html'.

Nodovi

Sada kad znamo šta je nod i koje osobine on posjeduje, razmotrimo specifikaciju njegove sintakse. Nod je predstavljen pomoću naredbe:

[DEF objectname] objecttype š [fields] [children]ć

Jedini neophodni dijelovi definicije su deklaracija tipa i vitičaste zagrade. Ime, polja i djeca su opcionalni. Ključna riječ ‘DEF' se stavlja ako nodu dajemo ime.

Ime noda ne smije početi cifrom i ne smije sadržavati kontrolne karaktere, prazaan prostor, jednostruke ili dvostruke navodnike, backslahove, uglate zagrade, plus znakove ili tačke.

Sintaksa polja noda je ime polja koje slijedi vrijednost. Polja su odvojena jedno od drugog i od vrijednosti pomoću spacea. Ako polje može imati višestruke vrijednosti, one trebaju biti zatvorene u uglate zagrade i odvojene zarezima.

Jednostavni primjer je Cube nod koji ima tri polja: width, depth i height. Ovo se izražava sa:

```
DEF MyCube { width 3.3 depth 1.67 height 3.14 }
```

Poredak polja nije bitan. Ako je polje izostavljeno, treba biti omogućena odgovarajuća default vrijednost u nodu koji definira taj tip objekta i korišten taj default.

Djeca noda imaju istu sintaksu kao i njihov roditelj. Zapravo nod koji je deklarisan bilo gdje u VRML dokumentu će slijediti ovu istu sintaksu.

Varijable, osobine, grupe i oblici

Tipovi polja

Da bi se objekti renderovali u tri dimenzije potreban je veliki broj matematičkih operacija, koje zahtijevaju veliki broj tipova podataka. VRML je posudio najneophodnije od tih tipova podataka od Open Inventor-a.
Šesnaest tipova podataka definiranih u VRML-u mogu se podijeliti u dvije klase:

- polja sa jednostrukom
- polja sa višestrukom vrijednošću

Jednostruka vrijednost može biti broj (realni ili cijeli), Boolean, vektor ili slika. Konvencija za imena polja je je da svaka jednostruka vrijednost ima prefiks ‘SF', a višestruka ‘MF'.

Tipovi polja jednostruke vrijednosti

SFBitMask

Ovo polje sadrži bit flag koji može biti prebacivan u različita stanja kroz upotrebu mnemoničkih imena. Imena se definišu kada se tek kreira polje. Kao primjer može poslužiti Cone nod (kupa), koja ima SFBitMask pod imenom ‘parts' koje ima bitmaske pod imenima ‘ALL', ‘SIDES' i ‘BOTTOM'. Vrijednost se može postaviti kada korisnik kreira dijete Cone i onda će se odrediti kako ostali nodovi utiču na njega.

SFBool

Prosta Boolean varijabla koja može biti zapisana ili numerički ili kao string tj. 0 i ‘FALSE' su ekvivalentni, kao i 1 i ‘TRUE'.

SFColor

Prosti metod sa jednostrukom vrijednošću za opis boje. Polje sadrži ‘trojku' - tri broja za jednu vrijednost. Ova tri broja se kreću od 0.0 do 1.0, i predstavljaju crvenu, zelenu i plavu boju respektivno.

SFEnum

Polje koje sadrži vrijednosti za enumeracioni tip. Ono obično definira mnemoničke labele za vrijednosti.

SFFloat

Floating point broj sa jednostrukom preciznošću zapisan u naučnoj notaciji.

SFImage

Ovo se polje koristi za smještanje bitmapirane slike. Format polja je zapisivanje tri integera koji predstavljaju širinu, visinu i broj dijelova koje slika posjeduje. Zatim slijede heksadecimalne vrijednosti odvojene spaceom koje prestavljaju bit-mapu.

Slika je nekompresovana bitmapa, gdje je svaka komponenta (obično piksel) predstavljena intenzitetima boje. Monohromatska slika će biti prestavljena sa po jednim bajtom za piksel, gdje je 0x00 bez intenziteta, a 0xFF najjači intenzitet (‘0x' označava heksadecimalnu notaciju).

Kolor pikseli će imati po jedan bajt za crvenu, zelenu i plavu komponentu. Može se dodati četvrti bajt koji predstavlja transparenciju, sa vrijednošću 1.0 za potpuno transparentno i 0.0 za potpuno neprozirno. Podaci moraju počinjati iz donjeg lijevog ugla i završiti se gronjim desnim uglom slike.

Slijedi primjer:

```
2 4 3 0xFF0000 0xFF00 0 0 0 0 0xFFFFFF 0xFFFF00
```

Ovo predstavlja sliku sa širinom 2 piksela i visnom 4 piksela. Donji lijevi piksel je crven, gornji desni zelen, dva srednja reda piksela crna, gornji lijevi piksel je bijel, a gornji desni piksel je žut.

SFLong

Ovo polje sadrži jedan 32-bitni integer. On može biti zapisan decimalno, heksadecimalno (0x) i oktalno (počinje sa 0).

SFMatrix

Ovaj tip polja se koristi da predstavi matricu transformacija. On je zapisan kao 16 realnih brojeva odvojenih spaceom koji se čitaju po redovima.

Npr. matrica koja predstavlja translaciju za 7.3 jedinice po x-osi se zapisuje sa:

1 0 0 0 0 1 0 0 0 0 1 0 7.3 0 0 1

SFRotation

Koristi se za predstavljanje proizvoljne rotacije i predstavljen je sa četiri floating pointa odvojena spaceom. Prve tri vrijednosti predstavljaju osu rotacije, a zadnja vrijednost je iznos desno-orijentisane rotacije oko te ose u radijanima.

Npr. rotacija od 180 stepeni oko x-ose se piše kao

1 0 0 3.14159265

SFString

Sekvenca ASCII karaktera poznata kao string. Ako string sadrži prazan prostor, on mora biti započet i završen sa dvostrukim navodnicima. Bilo koji karakter, uključujući novu liniju, može se pojaviti u stringu. Dvostruki navodnik se može uključiti u string pomoću backslash-a, npr:

‘Jonah \'Guts\' Bloomberg'

SFVec2f

Sadrži 2D vektor zapisan kao par floating point vrijednosti. Vrijednosti predstavljaju pravac i udaljenost koju vektor prolazi od tekuće tačke.

SFVec3f

Sadrži 3D vektor predstavljen sa tri floating point vrijednosti odvajene spaceom. Dodatna vrijednost izražava smjer vektora u 3D prostoru.

Tipovi polja višestruke vrijednosti

Vidjeli smo da možda karakterizacija ‘jednostruka vrijednost' nije ispravno ime, jer su mnogi od prethodnih tipova sastavljeni od više pojedinačnih parametara. Ono što zapravo razlikuje jednostruke od višestrukih tipova polja u VRML-u je sposobnost da opisuju ili djeluju nad jednim ili više osobina ili objekata. Dok SFColor može opisati samo jednu boju, njena odgovarajuća višestruka vrijednost MFColor može definisati bilo koji broj vrijednosti boje. Ove vrijednosti onda mogu biti pridružene objektima prema njihovoj poziciji u listi vrijednosti.

Razlike između jednostrukih i višestrukih tipova polja mogu se povezati sa razlikom između jednog sloga i niza slogova. Jedan slog može sadržati bilo koji broj vrijednosti i tpova podataka, kao što SFImage sadrži podatke o bitmapi i nju samu. Međutim, niz slogova može sadržati bilo koji broj slogova koji se mogu referencirati po indeksu.

Kasnije ćemo analizirati kada se koriste polja sa višestrukom vrijednošću i koji uticaj ona imaju na opis scene. Napravimo sada pregled koje vrste višestrukih tipova polja postoje u VRML-u.

MFColor

Ovaj tip polja sadrži bilo koji broj RGB boja. Boje se pišu kao tri floating point broja koji predstavljaju crvenu, zelenu i plavu komponentu, odvojene spaceom. Kada postoji više od jedne kolor vrijednosti, one se odvajaju zarezima, a cijelo polje se zatvara u uglate zagrade. Npr. polje

[1 0 0 , 0.0 1.0 0.0, 0 0 1]

predstavlja crvenu, zelenu i plavu boju respektivno.

MFLong

U poljima ovog tipa može se nalaziti bilo koji long ili 32-bitni integer. Integeri se mogu pisati u decimalnom, heksadecimalnom ili oktalnom formatu. Kao i kod MFColor i svih ostalih tipova višestruke vrijednosti, za odvajanje se koriste zarezi, a cijelo polje se piše u uglatim zagradama.

MFVec2f

Ovo polje sadrži bilo koji broj 2D vektora. Vektori se predstavljaju na isti način kao i kod jednostrukih tipova.

MFVec3f

Trodimenzionalni vektori imaju također višestruki tip, koji odgovara sintaksi i pravilima za pisanje jednostrukih trodimenzionalnih vektora.

8.3 Vrste nodova u VRML-u

Tipovi polja koje smo upravo opisali čine osnovu VRML-a. Pomoću njih može se virtualno opisati bilo koji tip 3D objekta. Samo renderovanje scene zavisi od definicije i pojavljivanja nodova i određenih pretpostavki i ponašanja ugrađenih u VRML. Ove pretpostavke uključuju koncepte kao što je povezivanje materijala ili teksture, ograničavanje boxova i način gledanja na zapremine.

Autori VRML 1.0 specifikacije su uključili određeni broj najkorisnijih nodova za kreiranje virtualnog svijeta. Postoji 36 ovih objekata koji se mogu podijeliti u 4 grupe: geometrija, osobine, grupni nodovi i WWW nodovi. Istražićemo neke od tih nodova da bismo stvorili mogućnost razmatranja pretpostavki i ponašanja koje smo spomenuli.

Prosti geometrijski nodovi

Drugo ime za ove nodove je nodovi oblika i oni uključuju AsciiText, Cone, Cube, Cylinder, IndexedFaceSet, IndexedLineSet, PointSet i Sphere.

AsciiText

Format ovog noda je:

```
AsciiText {
      string      ''    # MFString - tekst koji se prikazuje
      spacing     1     # SFFloat - pomnožen sa visinom teksta da bi
se
                        #pronašao iznos za koji će se povećavati y-
                        #koordinata za jednu liniju teksta
      justification     LEFT  # SFEnum - LEFT, CENTER ili RIGHT
                        #poravnanje
      width       0     # MFFloat - bilo koja vrijednost koja nije 0.
                        # Nula indicira da bi tekst trebao zauzeti
                        #svoju pravu širinu
}
```

U ovoj definiciji default vrijednosti za svako polje su napisane desno od imena polja. U VRML-u, ako pojava noda ne specificira vrijednosti svih polja u definiciji noda, on nasljeđuje default vrijdnosti iz te definicije.

Pojava ovog tipa noda izgleda ovako:

```
DEF Title AsciiText { string 'Primjer \'AsciiText\' noda'
                  height 1
                  justified CENTER
}
```

Kako nismo specificirali vrijednosti za 'width', koristi se default vrijednost 0.

Na tekst u nodu utiču bilo koje transformacije tekućeg stanja. Bilo koji nodovi koji prethode ovom nodu ili drugim nodovima će uticati na način na koji se crta nod. Za AsciiText najvažniji nod koji može uticati na njegov izgled je FontStyle nod, ali i bilo

koji nod koji mijenja osvjetljenje ili tekući materijal može mijenjati izgled ASciiText noda.

Ako je definirana tekuća tekstura, ona će biti primijenjena na tekst od početka prve linije prema gore duž svakog karaktera.

Cone (kupa)

Cone nod je po defaultu definisan da bude centriran u (0, 0, 0) i proteže se jednu jedinicu mjere u svim pravcima. To znači da je njegova visina od baze do vrha dvije jedinice, a radius jedna jedinica. Vrh konusa je na (0, 1, 0) i njegova baza je centrirana na (0, -1, 0). On je definisan kao da ima dva dijela - stranice i bazu odnosno dno.

Definicija dijelova postaje veoma važna kad počnemo definisati osnovne oblike na koje želimo omotati teksture ili materijale. Definisanje dijelova oblika omogućava nam veću kontrolu postavljanja teksture.

Različita postavljanja materijala u VRML-u biće razmotrena kasnije. Za sada prepostavimo da je jedan ili više materijala postavljeno na objekat prema tekućem načinu postavljanja. Generalno, oni se primjenjuju ili na površinu ili na cijeli objekat. Isto važi za teksture i normale. U slučaju kupe, ako se tekući način povezivanja odnosi na dijelove, materijal će prvo biti primijenjen na strane, a zatim na dno.

Objekat je, kao i sve ostalo, podložan uticaju tekućeg kumulativnog stanja transformacije i crta se sa tekućom teksturom i materijalom.

Ako postoji tekstura, ona će biti primijenjena na kupu na sljedeći način: strane će biti omotane u smjeru kazaljke na satu (gledano odozgo) počevši od zadnje strane kupe ili dijela koji je najdalji od početka z-ose. Na dno se primijenjuje krug teksture sa gornjim uglom teksture najbližim početku z-ose. Početak z-ose je dvodimenzionalna ravan na koju se renderuje scena - obično monitor kompjutera.

Cone nod je definisan sa

```
Cone {
      parts ALL    # SFBitMask - koristi se da bi se odredilo da li
VRML
            # treba crtati SIDES, BOTTOM ili ALL dijelove kupe
      bottomRadius # SFFloat - veličina baze
      height 2     # SFFloat - udaljenost baze od vrha
}
```

Cube (kocka)

Ovaj nod može predstavljati šestostrani objekat sa stranama koje se sastaju pod pravim uglovima. ‘Kocka' možda nije pravi izraz jer dimenzije strana ne moraju biti jednake. Default lokacija objekta je takva da je on centriran na (0, 0, 0) i širi se po dvij jedinice u svim pravcima, od -1 do +1 po svim osama. Objekat se renderuje u skladu sa tekućom kumulativnom transformacijom koristeći tekuću teksturu i materijal.

Redoslijed stavljanja materijala na kocku je prednja strana, zadnja strana, lijeva, desna, vrh i dno.

Cube nod se definiše sa:

```
Cube {
            width 2     #SFFloat
            height      2     #SFFloat
            depth 2     #SFFloat
}
```

Cilindar

Ovaj objekat je također centriran na (0, 0, 0) i širi se u svim pravcima po jednu jedinicu. Cilindrični dio se omotava vertikalno oko y-ose. Objekat ima tri dijela: vrh, dno i stranice. Renderuje se u skladu sa tekućom kumulativnom transformacijom koristeći tekuću teksturu u materijal.

Poredak kojim se materijal primjenjuje na dijelove objekta je stranice, vrh i zatim dno. Tekstura se primjenjuje na cilindar na sličan način kao na kupu.

Cylinder nod se definiše sa:

```
Cylinder {
            parts       ALL   #SFBitMask - određuje koje dijelove
                        #cilindra browser prikazuje
      radius      1     #SFFloat
      height      2     #SFFloat

}
```

Sphere

Sfere se prikazuju pomoću noda koji posjeduje samo jedno polje: dužinu radiusa. Default dužina je jedna jedinica i sfera je centrirana u koordinatni početak.

Kako nema dijelove ili stranice, sfera ignoriše tekuća povezivanja za materijale i teksture. Umjesto toga ona koristi prvi raspoloživi materijal i postavlja ga prema posebnim vlastitim normalama. Ako se na sferu treba primijeniti tekstura, ona će biti omotana oko cijelog objekta, počevši od pozadine sfere u smjeru kazaljke na satu, na isti način kao kod cilindra.

Sphere nod se definiše sa:

```
Sphere {
      radius      1     #SFFloat
}
```

Kompleksniji nodovi oblika

Sada kad smo se upoznali sa osnovnim oblicima, već možemo naslutiti kako se tipovi polja koriste u definisanju parametara neophodnih za matematičko opisivanje objekta. Međutim, 3D scena koja teži fotorealizmu neće biti sastavljena samo od kocki, konusa

i sfera. Sljeddeći pologonalni objekti imaju veći nivo kompleksnosti. Poligonalno modeliranje je najefikasniji i najpopularniji metod za crtanje realističnih objekata. Sve od ljudskog lica do planinskih terena renderuje se pomoću poligona, pa su oni prirodan izbor za grafički vokabular VRML-a.

IndexedFaceSet

To je najčešće primjenjivana geometrija u sceneam generisanim pomoću inventora. On predstavlja 3D objekat konstruisan od poligonalnih strana koje su definisane pomoću serija vrhova ili koordinatnih tačaka. Uzevši tekuću lokaciju za njegov koordinatni početak, vrhovi se postavljaju jedan po jedan, sa poljem coordIndex koje specificira indeks svake koordinate, sa indeksom -1 koji označava kraj strane.

Indeksi koordinata upućuju na trojke koordinata. Kao što je bio slučaj sa manje kompleksnim nodovima oblika, vrhovi se renderuju u skladu sa tekućom kumulativnom transformacijom koristeći tekuću teksturu i materijal.

Materijal se po defaultu primjenjuje na sve strane, ali je to moguće promijeniti tako da se materijal primjenjuje posebno na svaku stranicu prema poretku koji je naveden u polju materialIndex ili normalIndex. Ako nije definisan takav index, strane će se obilaziti prema poretku u kome su definisane.

Slično, normale za IndexedFaceSet nod se mogu specificirati ili generisati automatski.

IndexedFaceSet nod se definiše sa:

```
IndexedFaceSet {
      coordIndex        0     #MFLong - lista koordinata poligonalnih
      #stranica. Index -1 označava kraj jedne stranice i početak
      #druge
      materialIndex   -1      #MFLong - lista materijala koji se #koriste na vrhovima
poligona
      normalIndex       -1    #MFLong - indeksi normala koje se
#primjenjuju na objekat
      textureCoordIndex -1    #MFLong određuje način primjenjivanja
#teksture također pomoću indeksa

}
```

Objasnimo prethodnu definiciju pomoću primjera

```
Separator { # početak odvojenog objekta

      Coordinate3 {
      point [ -1 1 1, -1 -1 1, 1 -1 1, 1 1 1,
            -1 1 -1, -1 -1 -1, 1 -1 -1, 1 1 -1]
}
```

sada imamo indeksiranu listu koordinata koje definiću osam tačaka kocke centrirane u (0, 0, 0) koja se širi po jednu jedinicu u svim pravcima

```
Material {
      diffuseColor [1 0 0, 0 1 0, 0 0 1, 1 1 0]
}
```

definišemo indekse 0, 1, 2 i 3 tekućih boja. Materijali su crveni, zeleni, plavi i žuti respektivno.

```
NormalBinding {
      value PER_FACE_INDEXED
}
```

ovo postavlja povezivanje za normale

```
MaterialBinding {
      value PER_VERTEX_INDEXED
}
```

postavlja povezivanje za materijale

```
IndexedFaceSet {
```

primijetimo da nismo imenovali nijedan od ovih nodova, pa ključna riječ ‘DEF' nije neophodna

```
CoordIndex [0, 1, 2, 3, -1, 3, 2, 6, 7, -1,
# prednja i zadnja strana kocke
      7, 6, 5, 4, -1, 4, 5, 1, 0, -1, # dno i zadnja strana
      0, 3, 7, 4, -1, 1, 5, 6, 2, -1] # vrh i desna strana
# primijetimo da se ove vrijednosti odnose na koordinate iz Coord3 noda, koji definiše tačke sa indeksima od 0 do 7.

materialIndex [ 0, 0, 1, 1, -1,
# ovaj indeks kontroliše koji se od obojenih materijala koji su ranije definisani postavlja na koje vrhove. Materijali se postavljaju po broju indeksa u redoslijedu kako su definisani u polju CoordIndex

      2, 2, 3, 3, -1,
      0, 0, 1, 1, -1,
      2, 2, 3, 3, -1,
      0, 0, 0, 0, -1,
      2, 2, 2, 2, -1]
}

# kraj IndexedFaceSet-a
}
# kraj separatora
```

Nodovi osobina

Da bismo objasnili neke detalje iz prethodnog primjera, dolazimo do druge kategorije nodova u VRML-u, nodova koji defnišu osobine.

Coordinate3

Ovaj nod definiše skup 3D tačaka po njihovim koordinatama. Na ovaj skup se može referencirati po indeksu svake koordinate iz nodova IndexedFaceSet, IndexedLineSet ili PointSet.

Definisanje ovog noda ne crta ništa tokom renderovanja, on zapravo mijenja tekuće koordinate u sceni za korištenje od strane sljedećih nodova.

Coordinate3 nod se definiše sa

```
Coordinate3 {
      point 0 0 0 #MFVec3f
}
```

Material

Ovaj nod definiše osobine materijala koji je pristupačan svim nodovima koji slijede ovaj nod. Pojedinačnim materijalima se može pristupiti po broju indeksa.

Koncept materijala je definisan kao način na koji se svjetlo reflektuje od površinu objekta i Material nod definiše nekoliko parametara koji omogućavaju implementaciju različitih ‘izgleda' objekta. Ovi parametri su ambijentna, difuzna, spekularna i emisivna boja, sjajnost i transparencija.

Postavljanje materijala na objekat zahtijeva upotrebu MaterialBinding noda čije se postavljanje može različito interpretirati za različite nodove, zavisno od njihove građe.

Material nod se definiše sa

```
Material {
      ambientColor            0.2 0.2 0.2 #MFColor
      diffuseColor            0.8 0.8 0.8 #MFcolor
      specularColor     0 0 0  #MFColor
      emissiveColor     0 0 0       #MFColor
      shininess         0.2   #MFFloat
      transparency            0     #MFFloat
}
```

MaterialBinding

Ovaj nod se mora koristiti da bi se na objekte primijenili materijali definisani u nodovima koji se nalaze iznad noda objekta.

Materijal se može postavljati na dijelove objekta, strane ili vrhove. Postavljanje na strane ili vrhove je moguće samo kod onih objekata koji imaju strane i vrhove. Uz to postavljanje može biti određeno po indeksu materijala za svaku stranu, dio ili vrh, ali to radi samo kod objekata koji dozvoljavaju indeksiranje. Kao primjer ovoga može se navesti IndexedFaceSet nod jer on ima polje koje se zove materialIndex.

Kada lista tekućih materijala specificira višestruke vrijednosti za višestruka polja, npr. tri vrijednosti za diffuseColor, dvije za emissiveColor i jednu za shininess, vrijednosti će se obilaziti onako kako su postavljene na objekat. Period ovog kruga će biti broj vrijednosti u polju koje ima najviše vrijednosti (u ovom slučaju diffuseColor). Svi materijali će se ciklično proći i kada polje ostane bez vrijednosti, čekaće se na kraj tekućeg kruga na njegovoj prvoj vrijednosti.

Npr.

Ciklus	0	1	2	3
diffuseColor	0	1	2	0

emissiveColor	0	1	1	0
shininess	0	0	0	0

Možemo vidjeti da je ciklus 3 identičan kao ciklus 0. U slučaj kada je potreban samo jedan materijal, prve vrijednosti definisane u svakom polju (broj indeksa 0) se smatraju osnovnim materijalom.

Načini postavljanja su

DEFAULT Koristi default postavljanje.
OVERALL Osnovni materijal je postavljen na cijeli objekat.
PER_PART Materijali kruže po dijelovima objekta. Dijelovi su definisani u definiciji noda.
PER_PART_INDEXED Materijali se postavljaju na svaki dio prema indeksu, kao što je navedeno u nodu objekta. To jeste, nod objekta sadrži polje kao što je materialIndex kod IndexedFaceSet-a, koje određuje koji će materijal biti postavljen na koji dio.

PER_FACE Materijal se postavlja na svaku stranu objekta, a kruženje se dešava na način koji je ranije opisan.

PER_FACE_INDEXED Materijal je također postavljen na svaku stranu, ovaj puta onako kako je određeno indeksima koji se nalaze u nodu oblika.

PER_VERTEX Materijali su postavljeni na svaki vrh objekta. Kao što smo rekli, ovo se primjenjuje samo na objekte koji imaju vrhove.

PER_VERTEX_INDEXED Isto kao i gore, osim što su materijali određeni po indeksu umjesto da se ciklusi vrše automatski.

MaterialBinding nod se definiše sa:

```
MaterialBinding {
      value DEFAULT     #SFEnum
}
```

Normal

Korištenje ovog noda omogućava nam da definišemo skup 3D vektora normala koje će koristiti bilo koji sljedeći objekti bazirani na vrhovima u grafu scene. Normale se koriste za kontrolu količine i ugla refleksivnosti površine. Nod ne crta ništa na sceni, neho uglavnom utiče na izgled objekata. Ako normale nisu navedene, one će se generisati automatski za one objekte koji ih zahtijevaju kao npr. IndexedFaceSet, IndexedLineSet i PointSet.

Normal nod se definiše sa

```
Normal {
      vector      0 0 1 #MFVec3f
}
```

NormalBinding

Kao i MaterialBinding, ovaj nod određuje ponašanje tekućih normala kada su one postavljene na objekte koji slijede u grafu scene. Također kao MaterialBinding, različiti nodovi objekata će tretirati ova ponašanja različito, zavisno od , između ostalog, poretka njihovih dijelova.

Kada specificiramo ponašanje koje zahtijeva normale, treba biti siguran da postoji odgovarajući broj vektora, inače će doći do greške u renderovanju. Kao i MaterialBinding, ovo se odnosi samo na objekte koji imaju strane i vrhove.
Postoje sljedeća ponašanja u 1.0 specifikaciji VRML-a:

DEFAULT Koristi default postavljanje

OVERALL Cijeli objekat ima istu normalu.

PER_PART Jedna normala za svaki dio objekta.

PER_PART_INDEXED Jedna normala za svaki dio objekta, indeksirano.

PER_FACE Jedna normala za svaku stranu objekta.

PER_FACE_INDEXED Jedna normala za svaku stranu objekta, indeksirano.

PER_VERTEX Jedna normala za svaki vrh objekta.

PER_VERTEX_INDEXED Jedna normala za svaki vrh objekta, indeksirano.

NormalBinding nod se definiše sa:

```
NormalBinding {
      value DEFAULT     #SFEnum
}
```

Separator

Na kraju objasnimo VRML grupne nodove. Separator nod je u VRML scenama jedina kontrolna struktura koja omogućava izolaciju jednih dijelova koda od drugih. Kada se pojavi Separator tekuće stanje scene se spasi u memoriju prije čitanja i renderovanja njegove djece. Kada se dođe do kraja Separatora, graf scene se vrati iz memorije i prethodno stanje zamijeni sve promjene u osobinama koje su uvela djeca Separatora.

Uz to, Separator nodovi imaju osobinu koja se zove **render culling**. Ovo je snažna alatka koja omogućava djeci noda da budu preskočena tokom interpretacije scene, i bazira se na računanju graničnog boxa Separatora i poređenju sa tekućom zapreminom pogleda. Ovo se može isključiti, pa se ponašanje kontroliše pomoću VRML-ovog viewer-a/renderera.

Separator nod se definiše sa:

```
Separator {
```

```
      renderCulling     AUTO
# SFEnum AUTO omogućava gledaocu da odluči da li će se raditi culling
ili ne. Ostale vrijednosti koje su definisane su ON i OFF.
}
```

Group

Najjednostavniji od grupnih nodova, predstavlja osnovnu klasu za ostale grupne nodove. Grupa sadrži uređenu listi nodova djece. Ova lista se prolazi linearno, ne može se izvršiti render culling i ne izvršava push i pop stanja scene. Može se posmatrati kao logička asocijacija nodova koji su povezani na neki način.

Group nod se definiše sa:

```
Group {
}
```

Switch

Snažna kontrolna struktura za manipuliranje VRML scenama sa skriptovima ili programima, jer ovaj nod može biti projektovan da se prođu jedno, nijedno ili sva njegova djeca. Ako ga kontroliše neki eksterni izvor, može se koristiti za variranje izgleda scene ili osobina objekata.

Polje koje se zove whichChild određuje indeks djeteta koje će se proći. Djeca su pobrojana sekvencijalno, počevši od nule. Vrijednost -1 je default i označava da se ne treba proći nijedno dijete. Vrijednost -3 znači da se prolaze sva djeca i tada se switch ponaša kao Group.

Switch nod se definiše sa:

```
Switch {
      whichChild  -1    #SFLong
}
```

IndexedLineSet

Kao i IndexedFaceSet, IndexedLineSet nod predstavlja način opisivanja objekta sa odgovarajućim brojem poligona. Za razliku od IndexedLineSet-a ovi poligoni nemaju čvrste stranice. Umjesto toga one su skupovi linija. Poligoni se konstruišu iz vrhova koji se nalaze u tekućem skupu 3D koordinata. Polje coordIndex definiše poredak koordinata sa indeksom -1 koje predstavlja kraj tekućeg poligona.

Sljedeći primjer ilustrira upotrebu indeksiranih koordinata

```
Coordinate3 { 0 0 0 , 0 1 0, 1 1 0, 1 0 0,
      0 0 -1, 0 1 -1, 1 1 -1, 1 0 -1}
# osam tačaka koje će se koristiti za opisivanje kocke
# sada smještene u indekse od 0 do 7

IndexedLineSet {
      coordIndex [ 0, 1, 2, 3, -1 # prednji dio objekta
            0, 1, 5, 4, -1 # lijeva strana objekta
```

```
            4, 5, 6, 7, -1 # zadnji dio objekta
            3, 2, 6, 7, -1 # desna strana objekta
            0, 4, 7, 3, -1 ] # dno objekta
}
```

Na nod se primjenjuje tekuća kumulativna transformacija. Definisana ponašanja za postavljanje materijala i normala su:

PER_PART postavlja materijal ili normalu na svaki segment linije

PER_FACE postavlje materijal ili normale na svaku poliliniju ili na sve linije koje čine poligonalni objekat

PER_VERTEX postavlja materijal ili normalu na svaki vrh

DEFAULT ponašanje je OVERALL za materijale i PER_VERTEX_INDEXED za normale.

Teksture se primjenjuju na linije na isti način kao i na IndexedFaceSet.

IndexedLineSet nod se definiše sa

```
IndexedLineSet {
      coordIndex 0 #MFLong
      materialIndex -1 #MFLong
      normalIndex -1 #MFlong
      textureCoordIndex -1 #MFLong
}
```

PointSet

Pomoću ovog noda možemo predstaviti skup tačaka lociran kao podskup tekućih koordinata. Ovaj podskup je definisan sa indeksom početne koordinate i brojem tačaka u skupu. Kad je zadata ova informacija, VRML definiše prvu tačku na koordinatama početnog indeksa i zatim kreira tačku na svakoj koordinati u sekvenci dok ne budu kreirane sve tačke.

Tekuća transformacija utiče na osobine noda kao i kod svih nodova oblika. Jedan materijal se po defaultu postavlja na cijeli skup. Ako je MaterialBinding postavljen na PER_PART, PER_FACE ili PER_VERTEX, jedan materijal se postavlja na svaku tačku sukcesivno.

Postavljanje normala se vrši po defaultu na PER_VERTEX, sa po jednom normalom za vrh svake tačke.

PointSet nod se definiše sa:

```
PointSet {
      startIndex 0 # SFLong - određuje početnu koordinatu. Ako je
postavljanje PER_VERTEX, ovo polje određuje početnu normalu i vrh
      numPoints -1 # SFLong - broj tačaka u skupu. Vrijednost -1
indicira da se koriste sve preostale definisane koordinate.
}
```

Nodovi karakteristika

Sada ćemo se pozabaviti nodovima koji utiču na način na koji će objekti biti crtani.

FontStyle

Format i stil AsciiText nodova može biti određen pomoću FontStyle noda. Kao kod HTML-a, VRML definiše specifične stilove. Ostavljeno je browseru da prevede npr. ‘SERIF' u Times Roman ili Bookman. Veličina se mjeri u jedinicama mjere na sceni, obično metrima i koristi se za računanje visine teksta i spacinga.
Dozvoljene vrijednosti polja za FontStyle su:

family

SERIF (odgovara fontu Times Roman)

SANS (odgovara fontu Helvetica)

TYPEWRITER (sličan fontu Courier)

style

NONE - nema modifikacija na family

BOLD - family postaje Bold

ITALIC - family postaje Italic

Default format je:

```
FontStyle š
      size  10    #SFFloat
      family      SERIF #SFEnum
      style NONE  #SFBitMask
ć
```

World Info

Ovaj nod sadrži generalne informacije o virtuelnom svijetu, kao što je naziv svijeta. Ova informacija se prikazuje u title baru prozora browsera, kao TITLE tag u HTML-u.

Ovaj nod također može sadržati i info string sa ostalim informacijama o fajlu (ključne riječi i sl.)

```
WorldInfo {
   title "Floppy's VRML97 Tutorial Example 1"
   info ["(C) Copyright 1999 Vapour Technology"
         "guide@vapourtech.com"]
}
```

8.4 Tranformacije

Sljedeća grupa nodova utiče na vizualnu orijentaciju ili prezentaciju objekata koji slijede. Svaki nod osobine koji vrši neku vrstu transformacije je kumulativan, tj. efekat se dodaje na efekat svih prethodnih nodova osobina da bi se kreiralo stanje transformacije. Ovo je efikasan način da se transformacije razbiju u korake gdje se crtaju određeni objekti ili vr[e određene akcije.

Rotation

Ako je svaki objekat na sceni tačno poravnat sa osama efekat neće biti realističan. Rotation nod određuje osu i iznos rotacije (u radijanima). Svaka pojava Rotation noda u grafu scene se akumulira u tekuće stanje tranformacije i utiče na nodove koji ga slijede.

Rotation nod se definiše sa:

```
Rotation {
      rotation    0 0 1 0     #SFRotation
}
```

Scale

Upotreba ovog noda omogućava preciznu kontrolu nad skaliranjem sljedećih oblika duž tri ose. 3D skaliranje je izraženo kao vektor kao vektor sa floating point brojem za svaku osu, tako da se može izvršiti neuniformno skaliranje.

Scale nod se definiše sa

```
Scale {

      scaleFactor 1 1 1 #Sfvec3f
}
```

Translation

Termin translacija se koristi u 3D grafici za opisivanje tranformacije na objektu koja se vrši paralelno sa koordinatnim osama tj. ne sadrži rotaciju ili skaliranje. Drugim riječima, to je linearna transformacija pomjeranja sa 3D vektorom koji opisuje dužinu za koju se objekat miče duž osa.

Translation nod se definiše sa:

```
Translation {
      translation 0 0 0 SFVec3f
}
```

Transform

Kod svih prostih tranformacija koje smo spominjali dosada nod je bio definisan da izvršava jednu akciju. Ako u isto vrijeme želimo izvršiti više različitih transformacija,

možemo ili koristiti pojave pojedinačnih nodova ili grupisati sve naše transformacije u jedan nod koji se zove Transform.

Ovaj nod može sadržati jednu ili više sljedećih transformacija: Translation, Scale ili Rotation. Uz to, Tranform dodaje center polje koje mijenja koordinatni početak tekućeg stanja tranformacije i dijeli Scale u dva odvojena polja, scaleFactor za iznos skaliranja i scaleOrientation za orijentaciju skaliranja. Ova modifikacija omogućava više kontrole od običnog Scale noda.

Transform nod se definiše sa:

```
Transform {
      translation 0 0 0       #SFVec3f
      rotation    0 0 1 0     #SFRotation
      scaleFactor 1 1 1       #SFVec3f
      scaleOrientation  0 0 0 #SFRotation
      center            0 0 0             #SFVec3f
}
```

Za detaljniju ilustraciju Transform noda pogledajmo sljedeći primjer:

Ova definicija Transform noda :

```
Transform {
      translation T1
      rotation    R1
      scaleFactor S
      scaleOrientation  R2
      center T2
}
```

je ekvivalentna sekvenci

```
Translation { translation T1 }

Translation { translation T2 }

Rotation { rotation R1 }

Rotation { rotation R2 }

Scale { scaleFactor S }

Rotation { rotation -R2 }

Translation { translation -T2 }
```

TransformSeparator

Sa porastom važnosti transformacija razumljivo je da je uveden grupni nod koji je dizajniran za njih. Kao i Separator nod, TransformSeparator spašava graf scene prije interpretiranja njegove djece. Međutim, to spašavanje (push operacija) utiče samo na tekuću transformaciju, a ostatak stanja ostaje nepromijenjen. Na ovaj način element kao što je kamera ili izvor svjetlosti se može pozicionirati a da njegova transformacija ne utiče na ostatak scene.

TransformSeparator nod se definiše sa:

```
TransformSeparator {
}
```

MatrixTransform

Na kraju, MatrixTransform nod vrši klasičnu 3D geometrijsku transformaciju koristeći matricu četiri puta četiri. Neka je zadata matrica

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

Unutrašnih devet brojeva, počevši od gornjeg lijevog ugla su matrica rotacije i definišu koliko se objekat rotira oko koje ose:

1 0 0

0 1 0

0 0 1

Vanjskih sedam brojeva su matrica translacije i opisuju kojiko se objekti miču duž osa

0
0
0
0 0 0 1

Ovo se može sumirati kao

[matrica originalnih pozicija objekata] * [matrica promjena u pozicijama objekata] = [matrica novih pozicija objekata]

MatrixTransform nod se definiše sa:

```
MatrixTransform {
      matrix 1 0 0 0    #SFMatrix
             0 1 0 0
             0 0 1 0
             0 0 0 1
}
```

8.5 Nodovi tekstura

Sljedeća klasa nodova osobina su oni koji definišu teksture koje se omotavaju oko površina nodova objekata. Bez tekstura VRML objekti bi bili kolekcija ravnih obojenih strana i linija sa dobrim sjenčenjem. Teksture se inače upotrebljavaju

prilikom 3D renderovanja da simuliraju realni objekat i daju dodatnu kompleksnost površini. Ovo se postiže primjenom dvodimenzionalne slike (fotografije ili crteža) na površinu.

Texture2

Sirovi materijal za mapu teksture se specificira u Texture2 nodu. Svaki sljedeći objekat koji zahtijeva teksturu će koristiti sliku i parametre koji su specificirani u tom nodu. Slika se definiše ili pokazivanjem na lokalnu datoteku pomoću URL-a ili ugrađivanjem te slike direktno u nod. Ako se kao filename navede prazan string, tekstur će biti isključena.

Definisane su dvije specijalne ose za pravce u kojima se tekstura primjenjuje na objekat. Te ose se nazivaju S i T. S je horizontalno orijentisana osa, a T je vertikalno orijentisana, ali ova definicija je podložna varijacijama jer se ove ose ne moraju poklapati sa Kartezijanskim osama. Texture2 nod dozvoljava kontrolu nad omotavanjem teksture jer se ona primjenjuje i na S i na T osu.

Definisani su sljedeći način omotavanja teksture:

REPEAT kada je tekstura primijenjena u svojim punim dimenzijama i počine od svog početka.

CLAMP sprječava omotavanje preko veličine teksture.

Teksture2 nod se definiše sa

```
Texture2 {
      filename    ''          #SFString
      image       0 0 0       #SFImage
      wrapS       REPEAT      #SFEnum
      wrapT       REPEAT      #SFEnum
}
```

Texture2Transform

Da bi se mogao odrediti način primjene tekstura na površinu, Texture2Transform nod sadrži polja koja određuju translaciju, rotaciju, faktor skaliranja i centar.

Kumulativno, ova polja definišu 2D transformaciju koja se primjenjuje na teksturu svih sljedećih površina.

Texture2Transform nod se definiše sa

```
Texture2Transform {
      translation 0 0   #SFVec2f
      rotation    0     #SFFloat
      scaleFactor 1 1   #SFVec2f
      center            0 0   #SFVec2f
}
```

TextureCoordinate2

Da bismo koristili teksture u indeksiranim nodovima kao što je PointSet, IndexedLineSet i IndexedFaceSet, mora se kreirati mapa koja povezuje određenu tačku na teksturi sa odgovarajućim skupom indeksa. Ovo mapiranje se vrši pomoću TextureCoordinate2 objekata.

Par koordinata se sastoji od dva broja koja se kreću u opsegu između 0 i 1, gdje je 0 početak ose, a 1 najdalja tačka koju doseže teksturana toj osi. S koordinata je navedena prvo, a zatim slijedi T koordinata.

TextureCoordinate2 nod se definiše sa

```
TextureCoordinate2 {
      point 0 0   #MFVec2f
}
```

ShapeHints

Iako nije direktno povezan sa teksturom, ShapeHints utiče na izgled površina na koje su primijenjene teksture. ShapeHints može definisati IndexedFaceSet-ove da sadrže uređene vrhove, čvrste ili konveksne strane.

Dodavanje ShapeHints noda sceni omogućava neke optimizacije u renderovanju koje mogu skratiti vrijeme renderovanja. Npr. korisnik može implementirati različita skraćenja koja se baziraju na vidljivosti zadnjih strana ili promjeni karakteristika osnovnog osvjetljenja.

ShapeHints također utiče na način generisanja normala za stranice. Ako normale nisu zadate u IndexedFaceSet nodu, VRML zahtijeva da se one generišu zbog računanja refleksija svjetla na objektu. ShapeHints sadrži polje koje se zove creaseAngle i koje pomaže generisanju normala određivanjemkoje ivice treba da budu zaobljene ili uglačane, a koje treba da budu facetirane ili oštre. Ovo određivanje se vrši upoređivanjem vrijednosti polja creaseAngle sa sa uglom koji čine dva susjedna poligona. Ako je ugao između dvije poligonalne strane manji od vrijednosti creaseAngle, prelaz između tih strana će biti glatko osjenčen. Default vrijednost za creaseAngle je 0.5 radijana, odnosno oko 30 stepeni.

Na kraju, ShapeHints može određivati poredak strana objekta, koji će se koristiti umjesto default poretka definisanog u samom nodu. Opcije za ovo uređivanje su sljedeće:

UNKNOWN_ORDERING Poredak strana ostaje default.

CLOCKWISE Vrhovi strana su uređeni u smjeru kazaljke na satu počevši od vrha koji je najudaljeniji od koordinatnog početka.

COUNTERCLOCKWISE Vrhovi strana su uređeni u smjeru obrnutom od kazaljke na satu također počevši od vrha koji je najudaljeniji od koordinatnog početka.

Dva preostala polja ShapeType i faceType mogu se definisati kao UNKNOWN_SHAPE_TYPE ili mogu uzeti vrijednosti SOLID i CONVEX respektivno.

ShapeHints nod se definiše sa

```
ShapeHints {
      vertexOrdering    UNKNOWN_ORDERING  #SFEnum
      shapeType   UNKNOWN_SHAPE_TYPE      #SFEnum
      faceType    CONVEX      #SFEnum
      creaseAngle 0.5   #SFFloat
}
```

8.6 Svjetla i kamera

Svi elementi VRML-a koji su dosada razmatrani mogu se kombinirati na beskonačan broj načina i kreirati sobe, predjeli, objekti i ostale scene. Međutim, nijedan od njih nam ništa ne znači ako nemamo način da ga vidimo i što je još važnije, omogućimo korisniku da varira različite poglede na scenu. Zbog toga su definisana svjetla i kamera. VRML definiše dva noda za kameru i tri noda za svjetlo.

Sada ćemo uvesti pojam tačke pogleda, koji se u računarskoj grafici obično naziva kamera ili oko kamere.

OrtographicCamera

Ortografska ili paralelna projekcija je projekcija koja translira 3D scenu ili objekat u 2D ravan bez obzira na perspektivu (drugim riječima, veličina objekata ostaje ista).

Takvu projekciju ima graf scene kada se gleda kroz ortografsku kameru. Zapremina pogleda kamere je definisana kao pravougaonik i objekti ne gube na veličini sa udaljenošću.

U VRML-u, ako u sceni nije definisana kamera, default pozicija za ovaj nod je (0 0 1) (tj. jedan metar od površine 2D projekcije), gledajući duž negativne z-ose. OrtographicCamera ima polja koja se mogu koristiti za prilagođavanje ove inicijalne pozicije i orijentacije, kao i visine zapremine pogleda.

Uz to, nodovi transformacija utiču na nodove kamere, pa se translacija, rotacija ili skaliranje mogu staviti prije definicije kamere da bi dodatno uticali na njene inicijalne osobine.

Postavljanjem kamere na scenu projektant scene preuzima kontrolu gledaočevog prvog pogleda na scenu. Sa ove početne tačke gledaočev softver obično omogućava korisniku da modificira kameru i kreće se po sceni.

OrtographicCamera nod se definiše sa:

```
OrtographicCamera {
      position    0 0 1 #SFVec3f
      orientation 0 0 1 0     #SFRotation
      focalDistance     5     #SFFloat
```

```
      height      2     #SFFloat
}
```

PerspectiveCamera

Perspektivna projekcija se razlikuje od ortografske projekcije - kada se 3D scena translira u 2D ravan, vodi se računa o perspektivi. Drugim riječima, objekti se smanjuju sa svojom udaljenošću od kamere. PerspectiveCamera nod opisuje takav pogled i zbog toga njegova zapremina pogleda izgleda kao odsječena prava piramida.

Default lokacija kamere je ista kao i kod ortografske kamere (0 0 1) i ona gleda duž negativne z-ose. Polje koje se zove heightAngle može se koristiti za mijenjanje ukupnog vertikalnog ugla zapremine pogleda.

Ostale default vrijednosti su iste kao i kod ortografske kamere.

PerspectiveCamera nod se definiše sa

```
PerspectiveCamera {
      position    0 0 1 #SFVec3f
      orientation 0 0 1       0     #SFRotation
      focalDistance     5     #SFFloat
      heightAngle 0.785398    #SFFloat
}
```

U VRML-u kamere služe kao zamjena naše oči, ali ona ne može funkcionirati u mraku. Zbog toga postoje nodovi svjetla.

PointLight

Kao najjednostavniji od izvora svjetlosti PointLight definiše svjetlo koje se prostire u svim pravcima iz fiksne 3D lokacije. Ovo svjetlo ravnomjerno osvjetljava scenu u svim pravcima. Različita polja definišu lokaciju, boju i intenzitet izvora svjetlosti.

Nodovi svjetla su pod uticajem kumulativnog stanja transformacije i mogu uticati na objekte čije ih pojave slijede. Izvor svjetla ugrađen u Separator nod ne utiče na izgled objekata definisanih izvan tog separatora.

PointLight nod se definiše sa

```
PointLight {
      on    TRUE        #SFBool - vrijednost FALSE  će ugasiti
svjetlo
      intensity   1     #SFFloat    - 0 - bez intenziteta, 1 puni
intenzitet
      color 1 1 1 #SFColor
      location    0 0 1 #SFVec3f
}
```

DirectionalLight

DirectionalLight nod nudi korisniku precizniju kontrolu nad objektima na koje utiče. Umjesto jednakog osvjetljenja u svim pravcima, DirectionalLight osvjetljava pravac koji je paralelan 3D vektoru definisanom u polju direction.

Zrake idu paralelno i ne rasipaju se kao u realnom svijetu. Osvjetljenje koje ovo svjetlo proizvodi je istog intenziteta od izvora sve do beskonačnosti. Također, pošto ono postoji kao vektor i ne varira sa dužinom, ne može se reći da izvor ima neki koordinatni početak.

Ponašanje ovog svjetla unutar Separatora i njegov odnos sa tekućim stanjem transformacije je isti kao kod PointLight.

DirectionalLight nod se definiše sa:

```
DirectionalLight {
      on    TRUE        #SFBool
      intensity   1     #SFFloat
      color 1 1 1 #SFColor
      direction   0 0 -1      #SFVec3f
}
```

SpotLight

Ova vrsta svjetla omogućava više kontrole i realizma. Spotlight ima izvor lociran u fiksnim 3D koordinatama i osvjetljenje koje se prostire u obliku kupe duž 3D vektora.

Uz to, intenzitet osvjetljenja slabi eksponencijalno sa udaljenošću od centra kupe. Postoje polja za specificiranje stepena slabljenja i ugla kupe.

Ostale default vrijednosti i ponašanja SpotLight-a su iste kao kod PointLight i DirectionalLight.

SpotLight nod se definiše sa

```
SpotLight {
      on    TRUE        #SFBool
      intensity   1     #SFFloat
      color       1 1 1 #SFVec3f
      location    0 0 1 #SFVec3f
      direction   0 0 -1      #SFVec3f
      dropOffRate 0     #SFFloat
      couOffAngle 0.785398    #SFFloat - određuje ugao između vektora
pravca i ivice kupe u radijanima
}
```

8.7 Specifični aspekti VRML-a u odnosu na Internet

VRML je jednostavno file format za dostavljanje i upotrebu 3D scena i objekata preko Interneta. Očigledno, postoje zahtjevi za jezikom za modeliranje objekata. U 1.0 specifikaciji postoje neke naznake ove adaptacije u obliku dva noda dizajnirana za kooperaciju sa World Wide Web-om i jedan za 3D renderovanje.

WWWAnchor

Ovo je grupni nod koji ima sličnu ulogu kao HTML link. Kada se izabere neko od njegove djece, ovaj nod pokreće punjenje nove VRML scene. Implementacija ovog ‘biranja' zavisi od browsera. Kao i u HTML-u, nod samo predlaže koja akcija se može poduzeti, ali ne i kako se ona izvodi, pa čak ni kada se izvodi.

Nova scena koja treba biti napunjena specificira se pomoću polja name koje sadrži URL koji pokazuje na fajl. Ako polje name sadrži prazan string, nakon biranja jednog od njegove djece ne izvršava se ništa.

WWWAnchor je grupni nod u istom smislu kao i Separator, jer izvodi push i pop operacije tekućeg stanja scene da bi da sačuvao dok se interpretiraju njegova djeca.

Kada se fajl otvori preko URL-a, sa njime se mogu proslijediti i koordinate tačke na objektu. Ovo se postiže postavljanjem polja map na POINT umjesto njegove default vrijednosti NONE. Koordinate se šalju dodavanjem ‘?x,y,z' na kraj URL-a.

Npr, ako polje name ima vrijednost

http://www.outer.net/vrml/sample.wrl

i korisnik bira dijete WWWAnchor grupe na koordinati (4, 2.5, -4), onda će URL imati oblik

http://www.outer.net/vrml/sample.wrl?4,2.5,-4

Ova mogućnost se može koristiti za filtriranje rezultujućeg VRML-a kroz skript koji može promijeniti sadržaj fajla u zavisnosti od vrijednosti koordinata.

Na kraju, postoji mehanizam ugrađen u WWWAnchor koji omogućava da se cijeloj grupi da ime sa nekim značenjem. Ovo je unapređenje URL-a, jer pruža korisniku informaciju kuda vode djeca te grupe.

WWWAnchor nod se definiše sa:

```
WWWAnchor {

      name        ''    #SFString
      description ''    #SFString - dodatni tekst koji objašnjava
destinaciju grupe
      map   NONE        #SFEnum - da bismo proslijedili odabrane
koordinate treba postaviti ovo polje na POINT
      # djeca
}
```

WWWInline

Ovaj nod je najvažniji faktor koji odlučuje o uspjehu VRML-a na tržištu. Većina potencijalnih korisnika rade na personalnim kompjuterima preko analognih modemskih konekcija i mogućnost VRML-a da se prilagodi raspoloživoj hardverskoj

konfiguraciji je veoma važna. WWWInline nod omogućava da individualni nodovi iz VRML-ovog grafa scene budu smješteni kao serija fajlova na jednoj ili više lokacija.

Djeca WWWInline noda se dobiju pomoću specificiranog URL-a kada zatrebaju korisniku. Ovo znači da softver može donositi inteligentne odluke vezane za što efikasniji prenos podataka.

Uz to, ako je dijete Inline-a nod objekta, njegov granični box može biti specificiran pomoću polja bboxSize i bboxCenter. Ovo daje korisniku dodatnu kontrolu nad efikasnošću izvršenja VRML-a.

Ove sposobnosti VRML-a mogu dovesti do formiranja WWW biblioteke nodova, tako da se scene mogu kreirati i renderovati sa nodovima koji se uopće ne nalaze na lokalnom sistemu.

Sintaksa za ovaj nod je:

```
WWWInline {
      name        ''    #SFString - prazan string - nema akcije
      bboxSize    0 0 0       #SFVec3f
      bboxCenter  0 0 0       #SFVec3f
}
```

LOD (Level of Detail)

LOD se odnosi na skup kriterija koji se koriste za optimizaciju. U osnovi, on omogućava programu za gledanje da bira pogled na objekat u zavisnosti od udaljenosti korinika od tog objekta.

Ako gledamo sobu sa slikom na zidu, ona može izgledati kao jednostavni jednobojni kvadrat ako stojimo blizu suprotnog zida. Primicanjem bliže kvadrat se pretvara u tri kvadrata koja se nalaze jedan u drugom i formiraju okvir slike. Ako se još približimo, to postaje okvir sa drvenom teksturom i bitmapiranom slikom niske rezolucije. Stojeći tačno ispred, vidimo 24-bitno zapisanu sliku u okviru od mahagonija sa pozlatom.

Ovo se izvršava na sljedeći način: Centar LOD grupe se koristi za računanje udaljenosti posmatrača od grupe objekata. Rezultat se upoređuje sa nizom udaljenosti. Ako je manji od prve vrijednosti u nizu, bira se prvo dijete LOD grupe i renderuje. Ako je vrijednost između prvoe i druge vrijednosti u nizu, bira se drugo dijete itd.

Odavde zaključujemo da djeca počinju od najdetaljnije i završavaju sa najmanje detaljnom verzijom objekta. Također, ako niz ima n elemenata, treba postojati n+1 djece u grupi. Ako ih je manje, zadnje će se koristiti sve do kraja niza.

LOD koncept je posebno snažan u kombinaciji sa WWWInlining nodom. Sam po sebi, ubrzava renderovanje izbjegavanjem nepotrebnih detalja. U kombinaciji sa Inlining-om, djeca LOD-a mogu postojati na udaljenom serveru dok korisnik ne stigne do mjesta gdje je potrebno njihovo pozivanje i prikazivanje. Nadalje, odluka o tome koji nivo detalja će se prikazati može biti donesena i u zavisnosti od faktora kao što su brzina Internet konekcije i snaga korisnikovog kompjutera.

LOD nod se definiše sa:

```
LOD {
      range []    #MFFloat
      center      0 0 0       #SFVec3f
}
```

Instancing

Nod može biti dijete više od jedne grupe. Ovo se zove instancing (korištenje iste pojave noda više puta) i vrši se pomoću ključne riječi USE.

Ključna riječ DEF ujedno i definiše imenovani nod i kreira njegovu prvu pojavu. Ključna riječ USE ukazuje da će se najskorije definisana pojava ponovo koristiti. Ako nekoliko nodova imaju isto ime, onda se uzima zadnji DEF u toku parsiranja. Upotreba DEF/USE je ograničena na jedan fajl.

Nodovi se ne mogu dijeliti između fajlova.

Npr, renderovanje ove scene će rezultirati crtanjem tri sfere, dvije sa imenom Joe i druga manja sfera će biti nacrtana dvaput.

```
Separator {
      DEF Joe Sphere {}
      Translation { translation 2 0 0 }
      Separator {
            DEF Joe Sphere { radius .2}
            Translation { translation 2 0 0 }
      }
      USE Joe     #ovdje će se korstiti sfera sa radiusom .2
}
```

Mogućnosti proširenja

Dodaci na VRML su podržani kroz samoopisujuće nodove. Nodovi koji nisu dio VRML-a moraju prvo imati opis svojih polja, tako da sve implementacije VRML-a mogu parsirati i ignorisati te dodatke.

Ovaj opis se zapisuje odmah nakon otvaranja vitičaste zagrade za nod i sastoji se od ključne riječi ‘fields' iza koje slijedi lista tipova i imena polja koja taj nod koristi, zatvorena u uglate zagrade, a polja odvojenih zarezima. Npr, da Cube nije standardni VRML nod bio bi zapisan ovako:

```
Cube {
      fields [SFFloat width, SFFloat height, SFFloat depth ]
      width 10 height 4 depth 3
}
```

Specificiranje polja nodova koji jesu dio standardnog VRML-a nije greška; VRML parseri moraju tiho ignorisati te specifikacije.

isA relacije

Nodi tip noda može također biti nadskup postojećeg noda koji je dio standarda. U ovom slučaju, ako se implementacija novog tipa noda ne može pronaći, novi tip noda se tretira kao postojeći nod na kome se on bazira. Da bi se ovo podržalo, novi tipovi noda mogu definisati polje isA koje sadrži imena tipova čiji su oni nadskup. Npr. novi tip materijala koji se zove Extended Materijal i dodaje indeks refrakcije osobinama materijala biće zapisan kao:

```
ExtendedMaterial {
      fields      [ MFString isA, MFFloat indexOfRefraction,
            MFColor ambientColor, MFColor diffuseColor,
            MFColor specularColor, MFColor emissiveColor,
            MFColor shininess, MFFloat transparency]
      isA ['Material']
      indexOfRefraction .34
      diffuseColor      .8    .54   1
}
```

Višestruke isA relacije se mogu specificirati po potrebi; implementacija je takva da se uzima prva spomenuta relacija za koju postoji odgovarajuća definicija. isA relacije predstavljaju korisnu tehniku za proširivanje definicije postojećih tipova noda i dodavanje posebnih osobina za specifične upotrebe. To je često lakše nego definisati potpuno novi tip noda.

8.8 Specifičnosti VRML-a 2.0

VRML 2.0 unio je promjene u strukturi fajla i dodata je mogućnost animacije.

Osnovni oblici

Shape nod sadrži dva polja za opis izgleda i oblika objekta - appearance i geometry.

```
Shape {
   appearance Appearance {
      material Material {
      }
   }
   geometry Box {
   }
}
```

Upotreba objekta

Ako imamo mnogo identičnih objekata, možemo skratiti pisanje koda tako što ćemo definisati objekat i koristiti te definicije.

Možemo definisati objekat FBOX i koristiti tu definiciju kad god nam je potrebno da kreiramo box.

```
DEF FBOX Shape {
   appearance Appearance {
      material Material {
      }
   }
```

```
   geometry Box {
   }
}

USE FBOX
```

Osim oblika objekta možemo definisati i njegov izgled koji ćemo kasnije koristiti.

```
Shape {
   appearance DEF APP1 Appearance {
      material Material {
      }
   }
   geometry Box {
   }
}

Shape {
   appearance USE APP1
   geometry Box {
   }
}
```

Drugi način ponovne upotrebe VRML koda je Inline nod. On uzima podatke iz eksternog fajla i insertira ih u naš fajl. Tako, ako smo imali model stolice u fajlu *chair.wrl* možemo ga insertirati u našu scenu sa

```
Inline {
   url "chair.wrl"
}
```

Primjer: tut12.wrl

Koordinatni sistem

U VRML-u se radi sa klasičnim 3d koordinatnim sistemom, koji funkcionira po pravilu desne ruke. Sve udaljenosti se mjere u metrima.

Transformacije

VRML podržava tri vrste transformacija objekata - translaciju, rotaciju i skaliranje. Ove transformacije vrše se pomoću Transform noda i primjenjuju na njegovu djecu.

```
Transform {
   translation 1 1 1
   rotation 0 1 0 0.78
   scale 2 1 2
   children [
      USE FBOX
   ]
}
```

Primjer: tut13.wrl

Appearance nod

Appearance nod opisuje izgled objekta. Ovaj nod može imati polja material ili texture. Material polje sadrži Material nod.

Ovaj nod može sadržavati sljedeća polja:

- **diffuseColor -** boja objekta
- **specularColor** - boja odsjaja na sjajnim objektima
- **emissiveColor** - boja svjetlosti koju emituje objekat (glow)
- **ambientIntensity** - količina ambijentnog svjetla koju reflektuje objekat
- **shininess** - sjajnost objekta
- **transparency** - providnost objekta

npr.

```
Shape {
   appearance Appearance {
      material Material {
         emissiveColor 0 0.8 0
         transparency 0.5
      }
   }
   geometry Box {
   }
}
```

Ako objekat ima teksturu, ona se definiše pomoću texture polja koje sadrži jedan od tri texture noda (ImageTexture, MovieTexture ili PixelTexture)

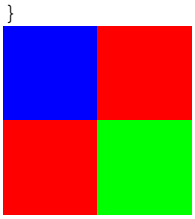
ImageTexture sadrži osnovnu mapu objekta koja može biti u JPG ili PNG formatu

```
Appearance {
   texture ImageTexture {
      url "brick.jpg"
   }
}
```

MovieTexture sadrži pokretnu teksturu u MPEG formatu i njene parametre (brzinu, da li se vrti u petlji itd.)

PixelTeksture omogućava da definišemo vlastitu teksturu kao bitmapu u VRML fajlu.

```
DEF PIXMAP Appearance {
   texture PixelTexture {
      image 2 2 3 0xFF0000 0x00FF00 0x0000FF 0xFF0000
   }
}
```



Primjer: tut14.wrl

Osnovni geometrijski oblici

VRML podržava nekoliko nodova za definisanje osnovnih geometrijskih oblika. To su nodovi Box, Cylinder, Sphere, Cone i Text.

```
geometry Box {
   size 5.5 3.75 1.0
}

geometry Cylinder {
   radius 0.5
   height 10
   top FALSE
}

geometry Cone {
   bottomRadius 5
   height 10
   side TRUE
   bottom FALSE
}

geometry Sphere {
   radius 10,000,000
}

geometry Text {
   string ["Hello", "World"]
   fontStyle USE HELLOFONT
   maxExtent 5
   length [3, 3]
}
```

Primjer: tut15.wrl

Anchor nod

Anchor nod služi kao hiperlink u virtuelnom svijetu. Aktivira se kada se klikne na neko od njegove djece i otvara fajl iz polja url.

```
Anchor {
   children [
      USE HEAD
   ]
   description "Back to the Tutorial"
   url "tut16.html"
}
```

Eventi

Većina nodova sadrži evente. Postoje dva tipa evenata: eventIn i eventOut.

EventOut predstavlja «odlazeći» event koji generiše neku informaciju kao što je promjena vrijednosti ili vrijeme klika mišem.

EventIn je «dolazeći» event koji prihvata informaciju od okruženja noda i radi nešto sa njom.

Polja koja sadrže evente zovu se «exposed». Ova polja imaju dva definisana eventa:

- set_fieldname - postavljenje vrijednosti
- fieldname_changed - generiše event na promjenu vrijednosti polja

Route

Povezivanje evenata vrši se pomoći ROUTE naredbe.

Npr.
ROUTE SENSOR.touchTime TO SOUND.startTime

Ovdje je pokretanje zvuka povezano sa klikom mišem.

Proto

Definiše prototip objekta u poljima koja su varijabilna. Tim poljima zatim pomoću IS dodjeljujemo različite vrijednosti.

```
PROTO VBox [
   field SFColor boxColour 1 0 0
]
{
   Shape {
      appearance Appearance {
         material Material {
            diffuseColor IS boxColour
         }
      }
      geometry Box {
      }
   }
}
```

crveni box:

```
VBox { }
```

zeleni box:

```
VBox {
   boxColour 0 1 0
}
```

Animacija

Pomoću ROUTE-a se uspostavljaju veze izmedju eventIn i eventOut polja (evenata). Jedan event pokreće drugi ili više evenata.

Eventi su kao poruke koje uvezuju virtuelni svijet. Event se sastoji iz dva dijela:

- sama poruka
- vremenski pečat

Pomoću vremenskih pečata sistem utvdjuje šta se desilo prije, a šta poslije. Ovo predstavlja interni podatak browsera.

Inicijalni eventi mogu biti generisani od stane **sensor** nodova i **script** nodova.

Sensor nodovi

Sensori okruženja mogu biti TimeSensor, VisibilitySensor, ProximitySensor i Collision. Ovi sensori ne prihvataju ulaz direktno od korisnika, nego detektuju evente u okruženju kao što su prolaženje vremena, pozicija korisnika itd.

TimeSensor je u suštini apstraktni tajmer na čije evente se mogu dešavati pojedine promjene u okruženju.

```
TimeSensor {
   exposedField   SFTime      cycleInterval     1
   exposedField   SFBool      enabled           TRUE
   exposedField   SFBool      loop              FALSE
   exposedField   SFTime      startTime         0
   exposedField   SFTime      stopTime          0
   eventOut       SFTime      cycleTime
   eventOut       SFFloat     fraction_changed
   eventOut       SFBool      isActive
   eventOut       SFTime      time
}
```

VisibilitySensor je nevidljivi box koji šalje event kada uđe u polje pogleda korisnika ili izađe iz njega. Može se koristiti za zaustavljanje animacija kada nisu u polju pogleda korisnika.

```
VisibilitySensor {
   exposedField   SFVec3f     center            0 0 0
   exposedField   SFBool      enabled           TRUE
   exposedField   SFVec3f     size              0 0 0
   eventOut       SFTime      enterTime
   eventOut       SFTime      exitTime
   eventOut       SFBool      isActive
}
```

ProximitySensor je vrlo sličan Visibility sensoru, s tim što detektuje i promjenu pozicije korisnika unutar svog opsega. To omogućava plejanje nekog zvuka dok je korisnik u granicama određenog prostora.

```
ProximitySensor {
   exposedField   SFVec3f     center            0 0 0
   exposedField   SFVec3f     size              0 0 0
   exposedField   SFBool      enabled           TRUE
   eventOut       SFBool      isActive
   eventOut       SFVec3f     position_changed
   eventOut       SFRotation  orientation_changed
   eventOut       SFTime      enterTime
   eventOut       SFTime      exitTime
}
```

Primjer: tut32c.wrl

Collision

Omogućava ili onemogućava koliziju korisnika sa geometrijom scene. Detektuje koliziju.

```
Collision {
   eventIn        MFNode      addChildren
   eventIn        MFNode      removeChildren
   exposedField   MFNode      children          []
   exposedField   SFBool      collide           TRUE
   field          SFVec3f     bboxCenter        0 0 0
   field          SFVec3f     bboxSize          -1 -1 -1
   field          SFNode      proxy             NULL
   eventOut       SFTime      collideTime
}
```

Ostali sensori

Ovi sensori mogu primiti ulazni podatak od korisnika. Oni detektuju klikove mišem, povlačenja i druge operacije. Možemo ih koristiti za pomjeranje objekata, startovanje animacija itd.

TouchSensor detektuje interakciju miša sa geometrijom.

```
TouchSensor {
   exposedField   SFBool      enabled           TRUE
   eventOut       SFVec3f     hitNormal_changed
   eventOut       SFVec3f     hitPoint_changed
   eventOut       SFVec2f     hitTexCoord_changed
   eventOut       SFBool      isActive
   eventOut       SFBool      isOver
   eventOut       SFTime      touchTime
}
```

SphereSensor možemo koristiti da rotiramo objekat mišem oko centra sensora.

```
SphereSensor {
   exposedField   SFBool      autoOffset        TRUE
   exposedField   SFBool      enabled           TRUE
   exposedField   SFRotation  offset            0 1 0 0
   eventOut       SFBool      isActive
   eventOut       SFRotation  rotation_changed
   eventOut       SFVec3f     trackPoint_changed
}
```

CylinderSensor je sličan SphereSensoru osim što je kretanje objekta ograničeno samo oko lokalne y ose senzora.

```
CylinderSensor {
   exposedField   SFBool      autoOffset        TRUE
   exposedField   SFFloat     diskAngle         0.262
   exposedField   SFBool      enabled           TRUE
   exposedField   SFFloat     maxAngle          -1
   exposedField   SFFloat     minAngle          0
   exposedField   SFFloat     offset            0
   eventOut       SFBool      isActive
```

```
   eventOut       SFRotation  rotation_changed
   eventOut       SFVec3f     trackPoint_changed
}
```

PlaneSensor služi za povlačenje objekta po ravni u dvije dimenzije.

```
PlaneSensor {
   exposedField   SFBool      autoOffset        TRUE
   exposedField   SFBool      enabled           TRUE
   exposedField   SFVec2f     maxPosition       -1 -1
   exposedField   SFVec2f     minPosition       0 0
   exposedField   SFVec3f     offset            0 0 0
   eventOut       SFBool      isActive
   eventOut       SFVec3f     trackPoint_changed
   eventOut       SFVec3f     translation_changed
}
```

Interpolatori

Ovi nodovi mijenjaju određenu vrijednost u vremenu. Prema vrijednosti koju mijenjaju mogu biti

- ColorInterpolator
- CoordinateInterpolator
- NormalInterpolator
- OrientationInterpolator
- PositionInterpolator
- ScalarInterpolator

Npr.

```
ColorInterpolator {
   eventIn        SFFloat  set_fraction
   exposedField   MFFloat  key                []
   exposedField   MFColor  keyValue           []
   eventOut       SFColor  value_changed
}
```

Polje key definiše vremena ključnih frejmova, a polje key value vrijednosti boje u tim ključnim frejmovima.

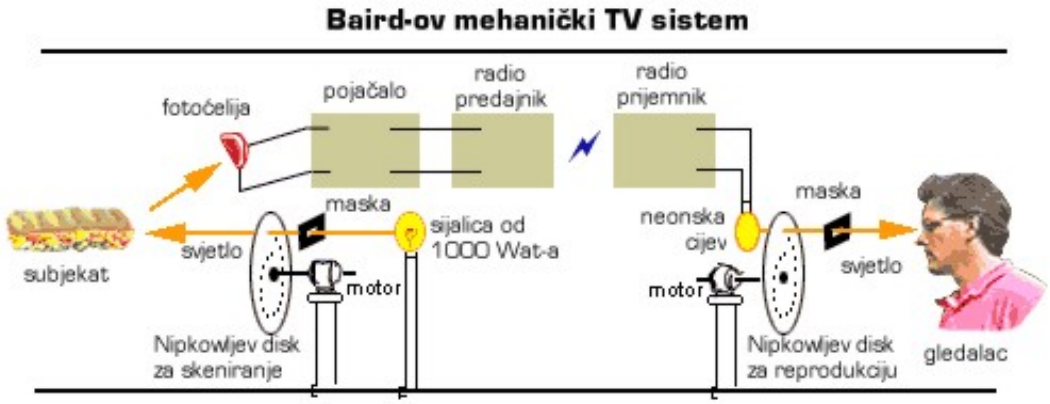
Primjer: tut34.wrl

9. Osnovni koncepti televizije

9.1. Razvoj televizije

9.1.1. Mehanička televizija

Paul Nipkow je 1884. godine izumio jedan disk sa rupama koje idu spiralno u njegov centar. Ovaj pronalazak je uobličio razvoj televizije. Inžinjeri John Logie Baird i Charles Francis Jenkins su, izme|u ostalih, koristili ovaj Nipkow-ljev disk za kreiranje prvih sistema za skeniranje, prenos i prijem slike u 1920-tim. Oni su kreirali cijele TV sisteme na bazi mehaničkog skeniranja i prijema slike. Bez katodne cijevi.



9.1.2 Elektronska televizija

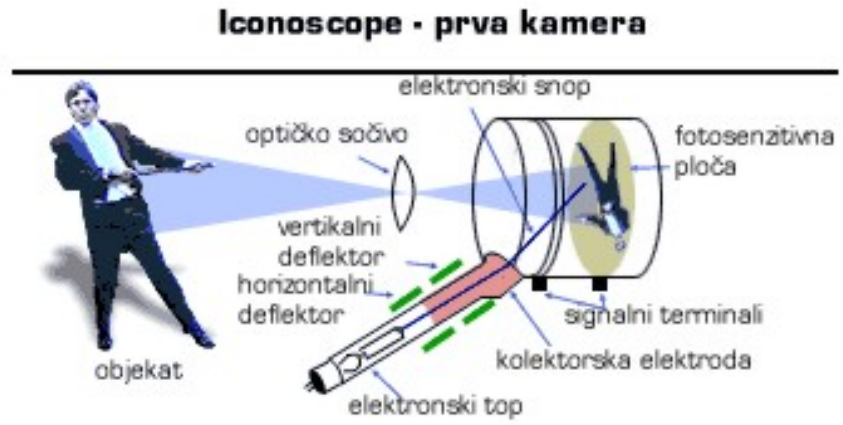
Elektronski tv sistemi su zaostali za mehaničkim nekoliko godina, uglavnom zato što je mehanička televizija bila jeftinija za proizvodnju i nije koristila složene komponente. Bilo je veoma teško naći finansijsku potporu za razvoj elektronske televizije kada je mehanička televizija radila tako dobro.

Nakon toga su Vladimir Kosmo Zworykin i Philo T. Farnsworth napravili neke značajne proboje i elektronska televizija je uhvatila korak.

Vladimir Zworykin je pronašao finansijsku potporu od Davida Sarnoff-a iz RCA koji je predvidio da se elektronska televizija može bolje isplatiti.

Farnsworth i Zworykin su obojica, radeći odvojeno, napravili veliki napredak prema komercijalnoj televiziji i televizorima koje svalo može priuštiti. Do 1935. obadvojica su emitovali koristeći elektronske sisteme. Ali, Baird Television je bila prva 1928. sa mehaničkim sistemom.

U to vrijeme veoma mali broj ljudi je imao tv prijemnik preko koga je mogao gledati mutnu sliku na dvo ili tro-inčnim ekranima. Budućnost televizije nije izgledala sjajno. ali je konkurencija bila velika.



Do 1939. RCA i Zworykin su bili spremni za emitovanje programa i počeli su sa prenosom Svjetskog sajma u New York-u (World-s Fair)

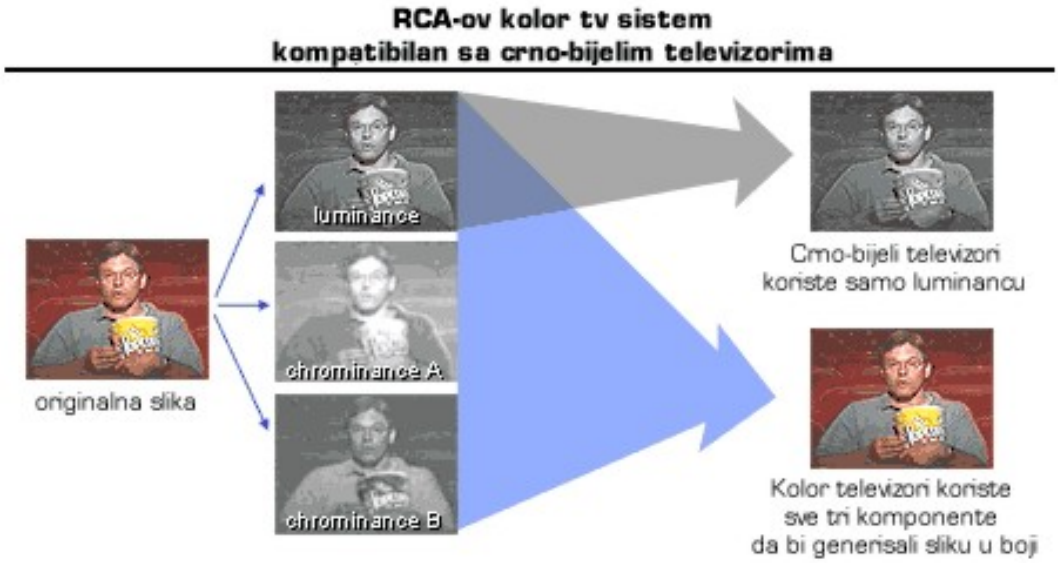
Stvari su se odvijale brzo i 1941. National television Standards Commitee je odlučio da je vrijeme da se stvore standardi za TV emitovanje u SAD. Pet mjeseci kasnije, 22 nacionalne TV stanice su usvojile te standarde.

Nakon II svjetskog rata nastalo je "Zlatno doba televizije". Nažalost, bila je crno-bijela.

9.1.3 Televizija u boji

CBS je razvio TV sistem u koloru nekoliko godina prije svog konkurenta, RCA. Me|utim, ovaj sistem nije bio kompatibilan sa crno-bijelim TV prijemnicima.

RCA je zatim razvio kolor sistem koji je radio i na crno-bijelim prijemnicima. Ovaj sistem je usvojio NTSC 1953.



9.2. Komponentni i kompozitni video

Kada se televizija tek pojavila, prikazivala je crno bijele slike. Tehnički termin za video signal koji je crno bijel je luminantni (luminanace) signal.

Jednu dekadu kasnije, naučnici su otkrili da se signal može emitovati u koloru. Problem je bio sljedeći: većina gledalaca je imala crno-bijele tv prijemnike. Izum kolor signala morao je biti takav da ga mogu emitovati i crno-bijeli tv prijemnici. Zbog toga je kolor signal jednim svojim dijelom morao biti luminantni. Zato su naučnici dodali tv signalu krominantni (chrominance) dio. Kolor tv prijemnik je mogao kombinovati oba dijela signala i prikazivati sliku u koloru, dok su crno-bijeli tv prijemnici koristili samo luminantni dio signala.

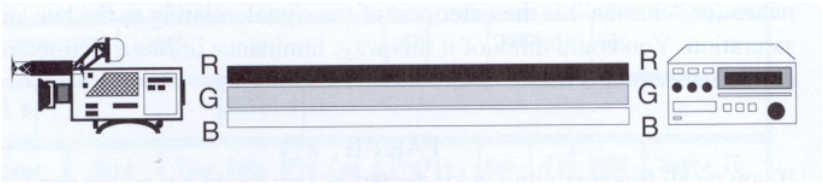
Danas se elektronski video signal sastoji od dvije vrste signala: luminantnog i krominantnog. Luminantni dio je svjetloća signala koja se sastoji od stepena nijansi od crne do bijele. Krominanaca ili kroma je kolor dio signala i odnosi se na boju i zasićenje (hue i saturation). Možemo to posmatrati i ovako: luminanca crta sliku, a kroma je boji.

Najmanji dio video slike je piksel. Ako se dovoljno približimo televizoru možemo vidjeti piksele. Kod kolor televizora svaka fosforna tačka je osvijetljena sa tri elektronska topa unutar katodne cijevi. Jedan top izbacuje crvene zrake, drugi zelene, a treći plave. Kombinacijom te tri komponente može se pikselu dati bilo koja boja.

Na ovaj način se mogu kreirati i crno-bijele nijanse, tako da se i luminantni signal može predstaviti na ovaj način.

Komponentni video je vrsta video signala gdje se RGB informacija, bilo da je u analognom ili u digitalnom formatu, čuva u odvojenim kanalima, koristeći odvojene kablove i odvojeno interno procesiranje za svaku komponentu boje.

Ovo prouzrokuje dvije stvari. Dobra strana je što crveni, zeleni i plavi signal ostaju veoma čisti. Slika izgleda dobro i nema mnogo gubitaka u presnimavanju. Loša strana je što je ovakva oprema skuplja, jer su potrebi posebni kablovi, trake, magnetoskopi i sl.

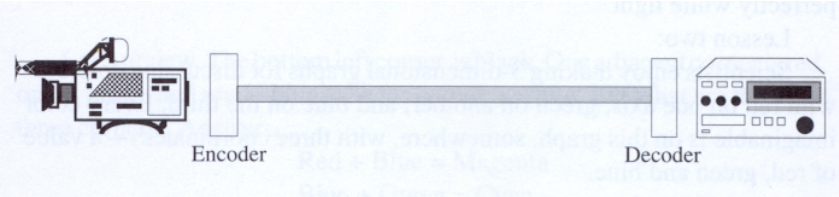


Komponentni signal

Kompozitni video je mnogo jednostavniji. Kompozitni signal miješa zajedno sve tri komponente boje. Metod po kojem se to radi baziran je na standardima (NTSC, PAL i SECAM). U našim krajevima koristi se PAL standard.

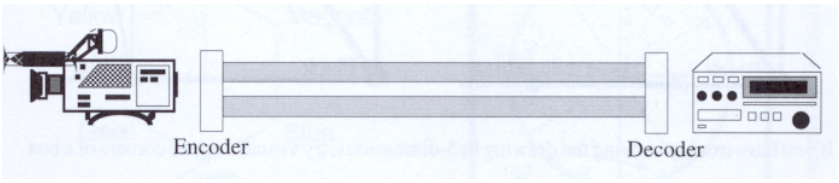
TV prijemnici dekodiraju i dekompresuju PAL video da bi se prikazao na monitoru. Kompozitni video je jeftin i jednostavan. Me|utim, kod presnimavanja se gubi na kvalitetu, zbog više kodiranja i dekodiranja signala. Skoro sva profesionalna oprema je kompozitna.

Komponentni video, zbog komplikovanosti tehnologije, koristi se samo ondje gdje je potrebno više puta presnimavati materijal, kao npr. kod specijalnih efekata.



Kompozitni signal

Postoji i treća opcija. Ako ne želimo da se mučimo sa tri kabla (komponentni signal) ili da imamo problema sa kvalitetom u presnimavanju (kompozitni signal), možemo koristiti dva kabla. Tako možemo odvojiti luminantni signal (kao Y) i krominantni signal (kao C). Ovako se formira S-VHS i Hi-8 format signala. Oni nisu sasvim komponentni, ali nisu ni pomiješani kao kompozitni. Ovaj signal se zove **Y/C video ili pseudo-komponentni**.



Y/C signal

9.3 Komponentni i kompozitni video sistemi

Prema imenu video produkta možemo odrediti da li je on komponentni, kompozitni ili Y/C video. Svaki video, čak i digitalni, mora biti kodiran na jedan od ovih načina. Slijedi lista najčešćih video formata:

Komponentni:

- D1 (19 mm digitalni)
- D5 (19 mm digitalni)
- Beta SP (1/2” analogni)
- MII (1/2” analogni)
- Betacam (1/2” analogni)
- Digital Betacam (1/2” digitalni)
- DCT (19 mm digitalni)
- (4:2:2 = digitalni komponentni)
- (RGB = analogni komponentni)

Y/C (ili pseudo-komponentni)

- S-VHS (1/2” analogni)

- Hi-8 (8 mm analogni)

Kompozitni:

- D2 (19 mm digitalni)
- D3 (19 mm digitalni)
- 1” tip C (1” analogni)
- ¾” U-matic (3/4” analogni)
- ¾” SP (3/4” analogni)
- 8 mm (8 mm analogni)
- VHS (1/2” analogni)
- (NTSC, PAL = analogni kompozitni)

Iako su D1, D2 i D3 trake sve formata 19 mm, kasete su različite i ne mogu se me|usobno mijenjati. Ista situacija je i sa ½” formatima.

Kada se video montira, kodiranje i dekodiranje izme|u kompozitnog i komponentnog formata može degradirati signal. Nažalost, zbog cijene komponentne opreme, mnoge produkcije koriste jedan dio komponentne opreme, kao što su D1 magnetoskopi, a izme|u njih se nalaze komponentni switcheri. Sve prednosti komponentnog signala mogu doći do izražaja samo ako je cijeli lanac produkcije komponentan.

Kvalitet video trake

Mada Y/C formati teoretski mogu proizvesti bolji signal od kompozitnih formata, trake koje se koriste za snimanje su značajno manjih gustina magnetskog oksida nego one koje se koriste za profesionalnu upotrebu. Ta gustina oksida u kombinaciji sa brzinom trake, definiše tzv. maksimalni signal bandwith koji može biti snimljen.

Tako|e primijetimo da širina trake (1/2”, ¾”, 1”) nema direktne veze sa kvalitetom video trake. Prve video trake bile su široke 2” i danas više ne postoje. Samo dio trake koji pokrivaju glave za snimanje sa svakim skeniranjem (za svako polje) u kombinaciji sa gustinom oksidnih čestica u tom dijelu utiče na video kvalitet.

^estice oksida na video traci su na neki način slične neravninama na hrapavom papiru. Hrapavi papir ima mali broj neravnina po inču, kao jeftine trake. Finiji papir ima više neravnina po inču, kao high density ili profesionalne trake. Prema tome, što je veća gustina, bolji je kvalitet.

9.4 Neki osnovni pojmovi televizijske terminologije

Key

Key je sabiranje dvije ili više slika. Jedna slika je video element koji ima rupe kroz koje se vide druge slike. Nekada se umjesto pozadine koja je u jednoj od tri osnovne boje (crvenoj, zelenoj ili plavoj) ubacuje slika. Ovaj način key-a je zastario i zove se chroma key.

Black

Black ili “crno” je oznaka za “prazan video signal”. On se koristi kao pauza u montaži prije ili poslije video clip-a. Može se dobiti direktno sa magnetoskopa ili video miksete ili kao crni background sa računara.

Frejm

U PAL standardu sekunda je podijeljena na 25 frejmova. Znači, frejm je najmanja jedinica od koje se sastoji video signal (slika). Tehnički, slika jednog frejma se može još podijeliti na dvije poluslike (field-ove) što ima veze sa tehnikom zapisa slike na magnetoskopu.

Time code

Svaka traka sa zapisanim video signalom sadrži time code. To je kod koji se sastoji od vremena izraženog u satima, minutama, sekundama i frejmovima. Prije montaže nova kaseta ili traka se “kodiraju” tj. usnimava im se time code. Nakon toga se dobije traka sa crnim video signalom i kodom na osnovu koga se mogu orijentisati ure|aji za montažu. Taj kod je osnovica za snalaženje na traci, bilježenje gdje se nalazi koji video clip i bilo koju drugu operaciju sa snimljenim materijalom.

Time code izgleda ovako

00	**:**	**01**	**:**	**15**	**:**	**10**
sati		minute		sekunde		frejmovi

9.5 Lanac TV produkcije

Televizijska emisija nastaje kao proizvod koji prolazi odre|eni lanac produkcije.

Prva faza je, naravno, pisanje scenarija i knjige snimanja. Zatim se organiziraju snimanja u enterijeru ili eksterijeru, priprema scenografija, kostimi i rasvjeta. Slijedi snimanje u studiju ili na nekoj drugoj lokaciji.

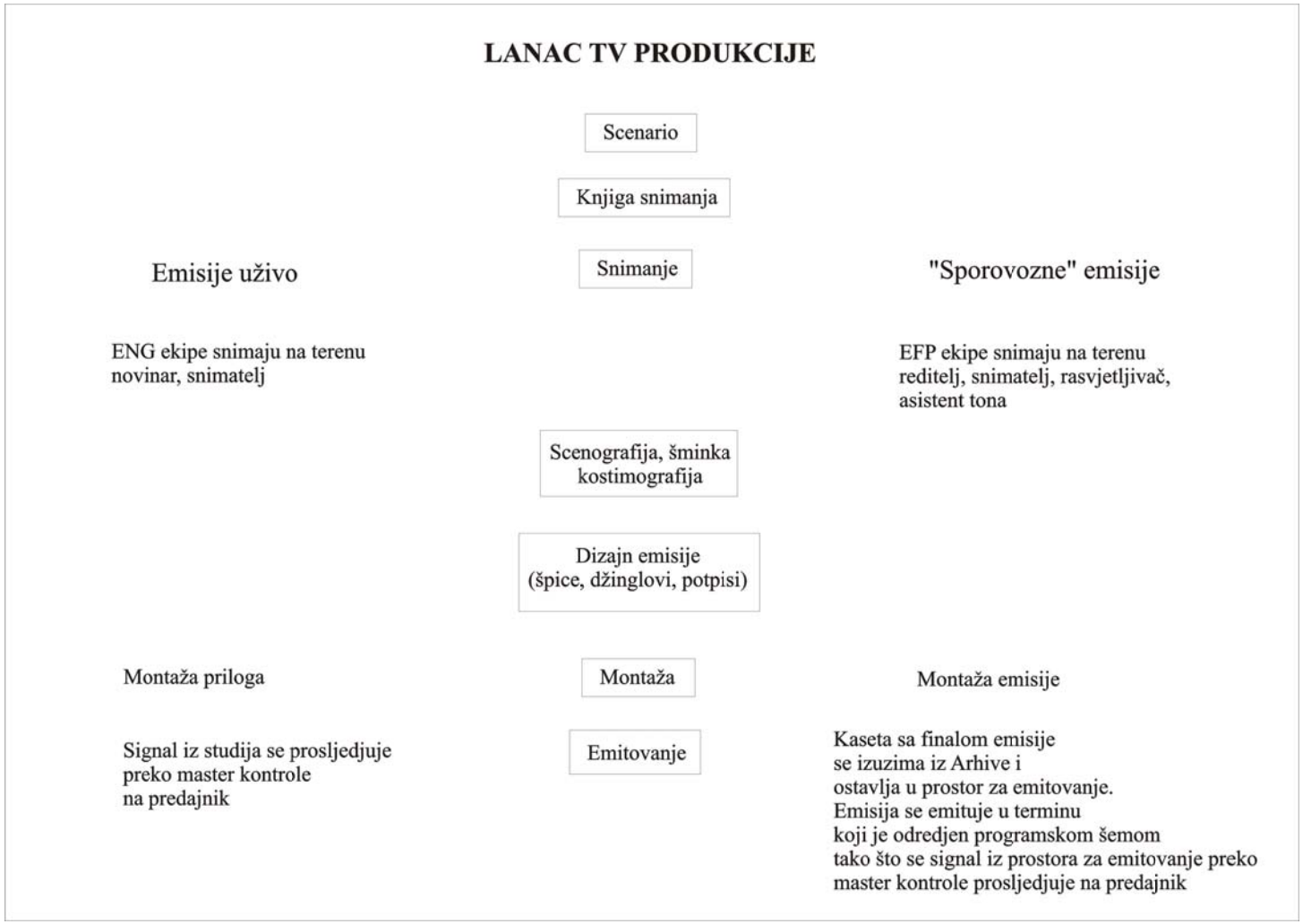
Ukoliko se snima u studiju koriste se tzv. studijske kamere, a njihovi signali se miksaju sa ostalim signalima preko audio i video miksete koje se nalaze u tonskoj i video režiji.

Snimanja “na terenu” vrše se mobilnim kamerama, a rade ih tzv. ENG ekipe koje se sastoje od snimatelja, asistenta tona i rasvjetljivača.

Nakon snimanja materijal ide u montažu. [pice i elementi dizajna se dodaju u montaži, a mogu biti ura|ene kao posebni video clipovi ili se uvoziti u program za montažu i key-ati (potpis) ili čak raditi direktno u programu za on-line montažu.

Nakon montaže televizijska emisija je spremna za emitovanje. Sa magnetoskopa za emitovanje signal sa video trake ide preko master kontrole i režije dnevnog programa na predajnik i prikazuje se preko antenskih prijemnika na kućnim televizorima.

LANAC TV PRODUKCIJE



10. Forme TV dizajna

Televizijski dizajn ili dizajn televizijskog programa se ispoljava kroz više formi. To mogu biti statične forme kao što su telop ili potpis, i pokretne forme kao što su animacija, špica ili video clip.

Statične forme se sastoje od jedne bitmape te zbog toga ne mogu sadržavati nikakav pokret.

Pokretne forme se sastoje od frejmova i imaju određeno trajanje. Za grafiku u televizijskom dizajnu karakteristično je da su trajanja kraća (najviše oko 1 minut).

1. Potpis

Jedan od najjednostavnijih i najstarijih oblika grafike u TV dizajnu je **potpis**. Potpisom se zovu bilo kakva slova koja se pojavljuju preko slike. Npr. kada se emituje izjava neke osobe, u donjem dijelu ekrana se pojavljuje njeno ime i prezime i eventualno funkcija koju obavlja.

U doba početaka filma potpisi ili titlovi su se ubacivali između kadrova na crnoj pozadini i slikani su kamerom.

Na određenom stepenu razvoja televizije uvedeni su tzv. **karakter generatori**. To su bile sprave koje su specijalno napravljene za generisanje potpisa. Oni su sadržavali određeni broj fontova, podržavali neke jednostavne operacije nad slovima i eventualno promjenu boje slova. Također su mogli emitovati slova, osim statično, i u dva osnovna pokreta**: rol** i **krol**.

Rol je u televizijskoj termnologiji pokret više redova slova po vertikali, najčešće odozdo nagore.

Krol je pokret jednog reda slova po horizontali, najčešće zdesna nalijevo.

Ovi pokreti se još uvijek koriste u odjavnim špicama, a krol i prilikom emitovanja nekih obavještenja za gledaoce.

U današnje vrijeme se još uvijek u živom programu koriste karakter generatori za potpise, ali u produkciji oni su dio programa za nelinearnu montažu i podržavaju razne efekte koji se mogu dati slovima, uključujući i pokret.

Tako danas potpisi mogu u sebi sadržavati neku animaciju, a skoro uvijek iza slova postoji pozadina urađena u skladu sa kompletnim dizajnom emisije.

Potpis u živom programu

Grafička pozadina za potpis u emisiji Žarišta TV BiH

2. Telop

Telop je također u današnje vrijeme skoro prevaziđena forma. On potiče iz vremena kada nije bilo grafike u elektronskom smislu na televizijskom ekranu, nego se ona crtala na papirima i slikala tzv. kamerom čitačem.

Telopi su korišteni za najavu određenih emisija u programu ili kao ilustracija unutar informativnih i drugih emisija.
Pojavom elektronske grafike nastali su telopi koji su se sastojali iz slova sa karakter generatora postavljenih na određenu jednobojnu pozadinu koja se generisala sa video miksete.

U današnje vrijeme rijetko se koriste telopi zbog svoje statičnosti. Njihova upotreba se zadržala u informativnim emisijama gdje se koriste za ilustraciju vijesti za koje nema usnimljenih kadrova koji bi je pokrili. Oni su tada grafički složeni i gledaocu pobliže objašnjavaju informaciju koju sluša.

Npr. ako se desio zemljotres u Nikaragvi, telop može sadržavati kartu mjesta događanja, grafički iskombinovanu sliku ruševine, sliku izvještača sa lica događaja i eventualno neki tekst sa naslovom vijesti ili imenom izvještača.

Upotreba ovakvih ilustracija u današnje vrijeme je neizostavna i one dostižu veoma visok stepen savršenstva grafičkog dizajna. U terminologiji ih još zovu “stils” što je nastalo od engleske riječi still image – statična slika.

Stil

3. Špice

Svaka emisija ima najavnu i odjavnu špicu.

Najavna špica je jedna od najsloženijih formi u televizijskom dizajnu. Ona je kruna dizajna televizijske emisije. Njeno trajanje zavisi od trajanja same emisije, pa su špice polusatnih emisija obično od 10 do 15 sekundi, emisija koje traju 1 sat od 20 do 30 sekundi itd. Trajanje najavne špice rijetko prelazi četrdesetak sekundi.

Sadržajno, najavna špica emisije nema nekih posebnih pravila, osim da se u zadnjoj trećini ili četvrtini njenog trajanja najčešće pojavljuje naslov emisije. Obično najavna špica uvodi gledaoca u sadržaj emisije ili barem u oblast kojom se emisija bavi.

Najavne špice se rade u svim tehnikama televizijskog dizajna, a u zadnje vrijeme najpopularniji je compositing.

Odjavna špica u pravilu sadržava popis imena članova ekipe koja je učestvovala u procesu stvaranja televizijske emisije. Pojedine emisije kao što su dnevnik, vijesti ili neke kraće informativne emisije u odjavnoj špici sadrže samo ime urednika i realizatora ili čak samo naziv televizije i datum. Druge emisije imaju potpisanu čitavu ekipu.

Odjavna špica se najčešće kreće po ekranu u rolu ili krolu. Ona može imati neku grafički dizajniranu pozadinu koja se uklapa u dizajn čitave emisije. U tom slučaju, kao i kod potpisa sa pozadinom, veoma je važan kontrast između slova i pozadine, da bi se postigla njihova dobra čitljivost.

4. Džinglovi

Džinglom se zove kraća forma u televizijskom dizajnu koja se emituje kao spona ili prelaz između dijelova televizijske emisije. Ukoliko emisija ima rubrike, džingl može sadržavati nazive tih rubrika. Također, koriste se i univerzalni džinglovi sa imenom emisije ili nekim elementom iz dizajna emisije.

Trajanje džingla je između 3 i 10 sekundi. Najčešće su muzički akcentirani.

5. Dizajn emisije

Dizajn emisije obuhvata sve grafičke forme koje sadrži jedna emisija. Najavna špica, potpisi, telopi, džinglovi i odjavna špica treba da budu dijelovi istog dizajna. U taj dizajn obično se uklapa i scenografija i kostimografija u emisiji.

Televizijski dizajner u saradnji sa scenografom i kostimografom kreira dizajn emisije. U idealnom slučaju u televiziji postoji art-direktor koji rukovodi kompletnim dizajnom televizijskog programa i koordinira rad televizijskih dizajnera sa radom scenografa, kostimografa, šminkera, te tako utiče na kompletan vizuelni utisak koji gledalac ima prilikom gledanja tv programa.

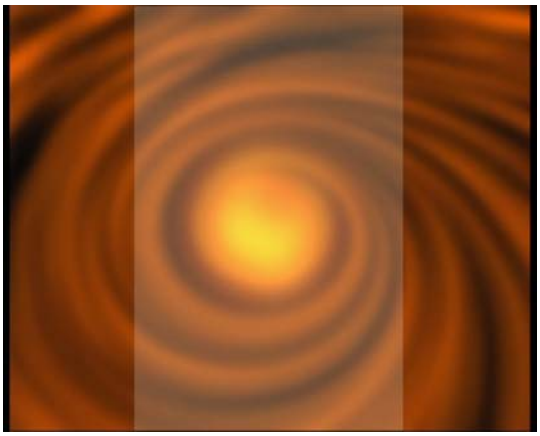
6. Video clip

Video clip je generalna forma u nestatičnom televizijskom dizajnu. U video clip spada i najavna špica i džingl i odjavna špica, ali i sve druge forme pokretne slike. Video clip je reklama, muzički spot, kao i čitava emisija.

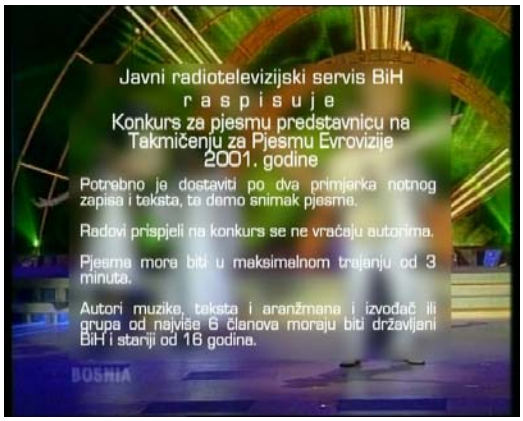
Sa stanovišta računarske grafike, video clip je sve što se može zapisati u formatu zapisa pokretne slike (npr. MPEG, AVI i sl.)

Programi za animaciju i compositing kao izlaz imaju video clip.

Frejm iz najavne špice za emisiju Žarišta TV BiH



Pozadina za odjavnu špicu emisije Žarišta tv BiH



Frejm iz video clip-a koji najavljuje konkurs za Takmičenje za Pjesmu Eurovizije na TV BiH

11. Principi 3D modeliranja

U realnom svijetu sve se sastoji od atoma koji zajedno formiraju molekule. U svijetu kompjutera nemamo takav element za građenje objekata kao što je atom. Kompjuterska memorija može sadržavati samo elektronske kodove koji predstavljaju brojeve ili simbole. To su elementi za građenje u svijetu kompjutera. Ovi brojevi i simboli mogu biti organizovani da predstavljaju druge strukture kao što su Kartezijanske koordinate, jednakosti, zapremine prostora, nagibe, osobine površine i fizičke atribute kao što su napon, mekoća i gustoća i zajedno čine alatke za modeliranje različitih objekata. Ne postoji univerzalan način modeliranja koji omogućava da napravimo objekte s kojima se svakodnevno susrećemo.

J. Vince u "3D Computer Animation" ustanovljava da se kompjutersko modeliranje sastoji od širokog spektra različitih tehnika. Npr. jednostavna kocka sadrži šest ravnih strana, osam vrhova i dvanaest ivica. Ovakav objekat može biti predstavljen na nekoliko načina a najjednostavniji je definiranje pomoću Kartezijanskih koordinata njegovih vrhova. Ovi podaci se mogu koristiti za konstruisanje ivica, koje onda formiraju strane. Na kraju strane mogu biti uređene da formiraju površine kocke. Ovakav model je poznat pod imenom granična reprezentacija. Ali da li se ista tehnika može koristiti za modeliranje sfere? Odgovor je da, ali bi sfera morala biti napravljena od velikog broja ravnih površina što bi otežalo rad programu za sjenčenje.

Drugi pristup modeliranju sfere bazira se na jednakosti

$$x^2+y^2+z^2 = r^2$$

gdje je r radius, a (x, y, z) bilo koja tačka na površini sfere. Ovaj implicitni metod definiranja površine zahtijeva da se otkriju vrijednosti x, y i z koje zadovoljavaju vrijednost r. Ovaj metod ne samo da identificira tačke na površini, nego i klasificira tačke unutar i izvan sfere pa se koristi za volumetrijske načine modeliranja.

Razmotrimo primjer vatre. Konstrukcija plamenova iz koordinata, ivica i ravnih površina nije praktična, a digitalizacija može biti opasna. Geometrija ne nudi rješenje u obliku jednakosti, pa moramo pronaći drugu tehniku. Sistemi dijelova i fraktali nude korisne solucije ovog problema.

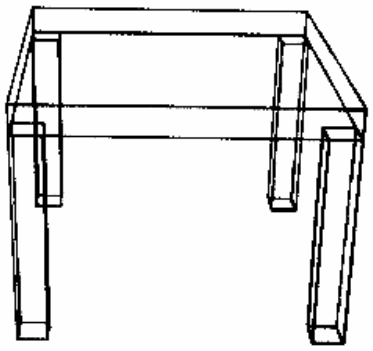
Na kraju, razmotrimo problem čajnika. On ima tijelo, dršku i nosač i zadnja dva elementa dodiruju tijelo tako da formiraju kompleksne krive presjeka. Konstrukcija tih krivih može biti izuzetno teška, međutim tehnika sakrivenih površina ih kreira automatski. Da bi se ovaj korisni efekat postigao moramo biti sigurni da drška i nosač zaista presijecaju tijelo.

Ako pregledamo unutrašnjost čajnika vidjećemo da ova suvišna geometrija prodire u tijelo. Također ćemo otkriti da je čajnik potpuno beskoristan za služenje čaja jer nema

rupe u tijelu kuda bi čaj prošao kroz nosač. Da li je to važno? Možemo reći da je to samo kompjuterski model. Ali u CAD aplikacijama dizajner može željeti da proračuna težinu čajnika, njegovo težište, površinu i možda njegov momenat inercije. Zato je važno da fizički opis objekta bude odgovarajući.

11.1. Žičani model

Jedan od najjednostavnijih metoda kreiranja 3D objekata je njihovo konstruisanje iz skupa linija koje predstavljaju ravne ivice koje identificiraju njegove glavne geometrijske osobine. Takav metod je poznat ako žičani model jer izgleda kao da je objekat konstruisan od žice. Slika 1 prikazuje sto koji je napravljen u žičanom modelu.



slika 1

Kako su ovi objekti napravljeni od nekoordiniranog skupa ivica, to može uzrokovati teškoće programima u otklanjanju ivica koje su maskirane drugim površinama. To je zato jer numerički podaci ne sadrže informaciju koja pridružuje ivice njihovim površinama. Zato se žičani model ne razmatra kao validan način modeliranja, nego se taj termin koristi za opisivanje transparentnog pogleda na model konstruisan od ivica.

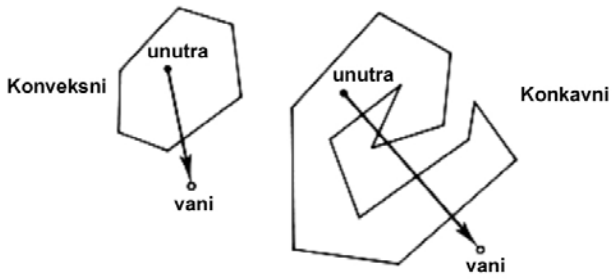
Ideja definiranja površine objekta preko njegovih ivica još uvijek postoji, ali treba postojati i odgovarajuća struktura podataka koja bi opisala kako su ivice povezane da formiraju elemente ograničenih površina, što vodi ka graničnoj reprezentaciji.

11.2. Granična reprezentacija

Granična reprezentacija se odnosi na takav način modeliranja koji konstruiše 3D objekat iz opisa površina, a ne zapremine. Npr. kocka koja je ranije opisana predstavlja opis površine pomoću koordinata vrhova koji definiraju stranice, a one formiraju poligone površina. Ovo je očito granična reprezentacija jer postoji eksplicitna geometrija za definiranje bilo koje tačke na površini kocke. Međutim, nemamo slične podatke za definiranje tačaka unutar i izvan kocke. Nadalje, iako je jednostavno izračunati zapreminu kocke, ona se ne može direktno izračunati iz ovih podataka o granicama. Potrebna je određena reorganizacija tih podataka.

11.2.1. Planarni poligoni

Poligon definiše klasu figure konstruisanu iz lanca ravnih ivica. Ovo uključuje oblike čije su granice konveksne, konkavne ili se presijecaju i mogu ali ne moraju biti planarni. Zbog ovog širokog opsega konstrukcija poligonalno modeliranje se uglavnom radi na planarnim poligonima koji su ili konkavni ili konveksni, mada konkavni poligoni komplikuju stvari. Npr. kod konveksnog poligona svaka tačka unutar granice treba da presiječe granicu samo jednom da bi se našla izvan dok konkavne granice mogu uzrokovati veliki broj čudnih presjeka.

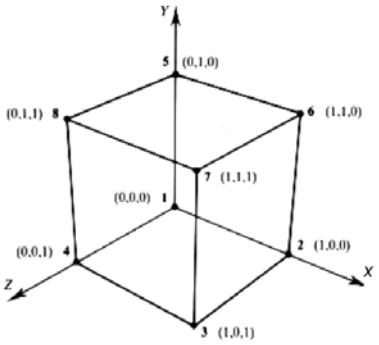


slika 2

Slika 2 ilustrira ovaj efekat. Nadalje, bilo koje dvije susjedne ivice u konveksnom poligonu mogu se koristiti kao vektori da bi se izveo konzistentan ortogonalni vektor na površinu, dok konkavni poligon može preokrenuti pravac vektora zavisno od ugla koji zatvaraju ivice.

Ovi popratni efekti kod konkavnih poligona mogu biti veoma nepovoljni kada se razvija algoritam za sjenčenje pa se ili izbjegava njihovo dozvoljavanje kod modeliranja ili se poligon triangulira tj reducira na mrežu trouglova koji mogu biti samo konveksni i uvijek su planarni. Nadalje ćemo pretpostaviti da su svi poligoni planarni i konveksni.

Pravljenje objekata pomoću poligona je relativno jednostavno i zahtijeva samo da se identificiraju važne ivice i konstruiše familija strana koje pokrivaju površinu. U kompjuterskoj animaciji pri modeliranju uvijek moramo voditi računa o tome kako će objekat biti animiran. Da bismo ovo objasnili konstruišimo kocku.



p	vrhovi			
1	1	2	6	5
2	2	3	7	6
3	3	4	8	7
4	4	1	5	8
5	5	6	7	8
6	1	4	3	2

Poligoni

v	x	y	z
1	0	0	0
2	1	0	0
3	1	0	1
4	0	0	1
5	0	1	0
6	1	1	0
7	1	1	1
8	0	1	1

Vrhovi

slika 3

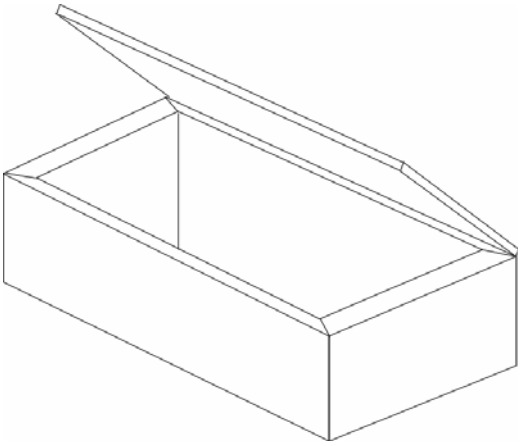
Slika 3 pokazuje njenu poziciju relativno u odnosu na neki koordinatni sistem i zbog jednostavnosti pretpostavimo da je svaka strana duga jednu jedinicu dužine.

Kocka može biti konstruisana na različite načine i jedan od njih bi bio da smjestimo koordinate osam vrhova u tabelu kako je prikazano na slici 3. Druga tabela bi referencirala ove vrhove da formiraju šest graničnih poligona. Program onda može konstruisati bilo koji poligon izvlačenjem četiri vrha iz tabele poligona.

Kocka će biti animirana tako što će se spljoštiti na vrhu broj 7 sa koordinatama (1, 1, 1). Ovaj vrh bi mogao lagano padati prema dolje, recimo na (1, 0.5, 1) mijenjanjem y koordinate vrha od 1 do 0.5 u određenom periodu vremena. Program za renderovanje koji zapravo kreira sliku u boji bi prema podacima iz dvije tabele konstruisao poligone kocke u svakom frejmu animacije.

Ako se reducira y koordinata vrha 7, gornji poligon kocke više nije planaran, nego je uvrnut i veoma je moguće da će renderer proračunati osvjetljenje na pogrešan način. Zato, ako zaista želimo da se kocka spljošti preko vrha 7, onda vrhovi 6 i 8 moraju također biti pomaknuti da bi se očuvala planarnost gornjeg poligona.

U realnosti način na koji kocka mijenja svoju geometriju zavisi od njenog sastava. Promjene na kocki koja je napravljena od spužve su drukčije od promjena na kocki od gline. Također kocka napravljena od aluminijskih ploča reaguje drukčije od kocke koja je napravljena od papira.
Druga vježba za animaciju može biti podizanje gornje ploče kocke kao da je fiksirana za jednu od ivica. Sa predloženim načinom modeliranja ovo predstavlja ozbiljan problem, jer ne samo da gornje vrhove dijele četiri stranice, nego i stranice kocke nemaju debljine i zato njihova unutrašnjost ne postoji. Konstruisanje čvrste kocke zahtijeva modeliranje kao što je prikazano na slici 4.

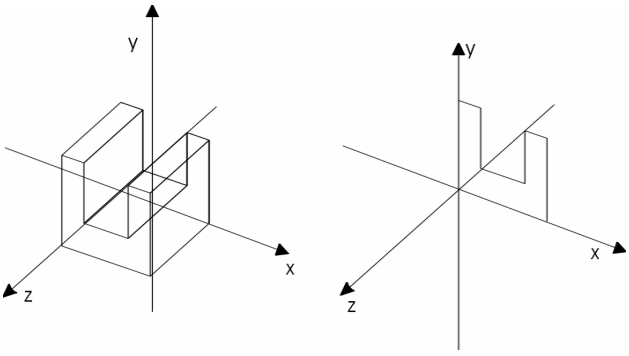


slika 4

Možemo vidjeti da zahtjevi animacije moraju biti anticipirani u fazi modeliranja i da renderer mora imati dovoljno geometrije da bi mogao uraditi svoj posao, a dio tog posla je da ima pristup ivicama koje formiraju granične poligone. U takvim slučajevima objekti su definirani u obliku hijerarhije poligona napravljenih od ivica, koje su opet napravljene od vrhova i predstavljeni su pomoću tri liste podataka o poligonima, ivicama i vrhovima.

11.2.2. Ekstrudiranje

Iako je metod poligonalne površine pogodan za konstruisanje objekata on može postati nepraktičan za kompleksne objekte. Da bi se to prevazišlo, uvedena su neka skraćenja kao pomoć u pravljenju generičkih formi. Prva takva tehnika je ekstrudiranje. Ono automatski konstruiše graničnu reprezentaciju istezanjem presječnog oblika duž njegove ose i kreira njegovu ekstruziju duž zadane dužine.



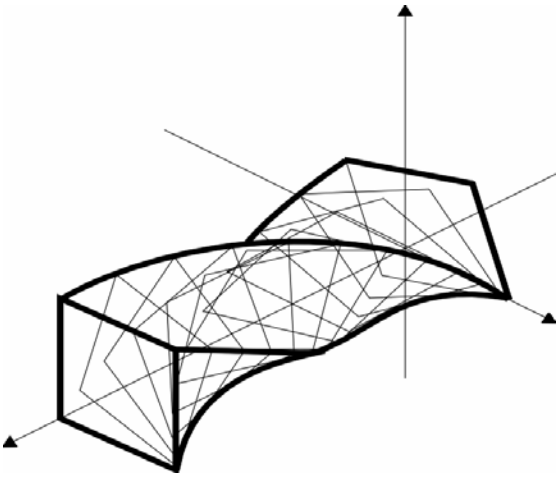
slika 5

Slika 5 ilustrira presjek i 3D površinu kreiranu pomicanjem presjeka u pravcu z-ose.

Konstruisanje površine može se vršiti automatski pomoću kompjuterskog programa koji zahtijeva samo podatke o koordinatama presječne površine i dužinu istezanja. Tada postoji dovoljno informacija o svakom vrhu na ekstrudiranoj površini. U procesu pravljenja objekta program može raditi u skladu sa konvencijom konstruisanja poligona čiji su vrhovi orijentirani u pravcu kazaljke na satu ili obrnuto. Ako se ovaj pravac ne slaže sa

orijentacijom koju podrazumijeva renderer, može doći do problema sa sjenčenjem unutrašnjosti ili spoljašnjosti objekta. Zbog toga neki komercijalni sistemi za modeliranje imaju komandu koja omogućava preokretanje pravca granica poligona.

Dalji razvoj ovog metoda je konstrukcija familije poligona površine na nekoliko stepena ekstruzije, gdje je ne svakom stepenu presječna površina rotirana. Ovo omogućava modeliranje uvrnutih oblika. Slika 6 ilustrira primjer ovog procesa.

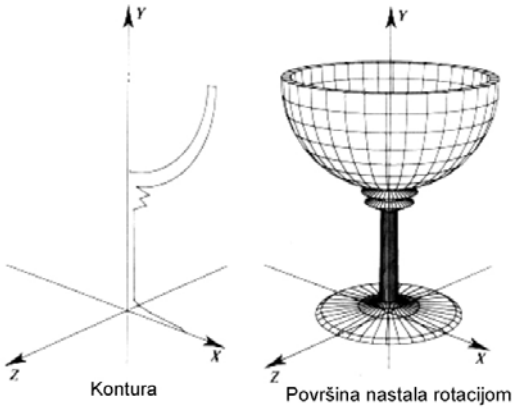


slika 6

Nekada se tokom ovog procesa može mijenjati i veličina presječne površine ili se čak njen oblik pretvarati u neki drugi. Idealno, program za ekstrudiranje treba oomogućiti korisniku da definira početni i krejnji oblik presječne površine i ugao rotacije koji se na nju primjenjuje u svakom stepenu procesa ekstrudiranja. Nadalje, umjesto ravne dužine ekstrudiranja, procedura treba dozvoliti da bilo koja 3D kontura formira centralnu liniju finalnog objekta.

11.2.3. Razvijene površine

Razvijene površine nam pomažu da konstruišemo objekte kao što su čaše, sfere, boce ili neke šahovske figure. Ovi objekti posjeduju simetriju oko centralne ose. Opet se zahtijeva glavna 2D kontura koja se ovaj put rotira oko jedne od 3D osa da bi se dobila razvijena površina.



slika 7

Slika 7 pokazuje kako se iz 2D konture može formirati čaša za vino. Primijetimo da je finalna površina još uvijek konstruisana iz planarnih poligona čiji broj zavisi od broja vrhova u originalnoj konturi i veličine ugla za koji se ona zaokreće u svakom stepenu konstrukcije. Korak ugla postaje veoma koristan metod u kreiranju više objekata od jedne konture.

Kao i kod ekstrudiranja, razvijene površine se mogu generisati automatski, s tim što korisnik specificira konturu, broj inkremenata koji će se napraviti prilikom rotacije od 360 stepeni i osu oko koje će se vršiti rotacija. Ovakav program će proizvoditi samo simetrične objekte, međutim, uz malo modifikacija može se koristiti i za kreiranje asimetričnih objekata. Za početak zamislimo površinu koja se dobije tako što se tokom rotacije konture oko referentne ose sama osa pomjera duž nekog zatvorenog puta. Slično, zamislimo razbijene oblike koji se mogu kreirati modificiranjen glavne 2D konture dok se ona rotira.

11.2.4. Površine slobodnog oblika

Brojne tehnike se koriste za modeliranje objekata u kompjuterskoj grafici. Koja tehnika će se koristiti zavisi od objekta koje se modelira, kao i od toga kakav finalni rezultat želimo da postignemo. Poligonalni modeli su brzi i laki za prikazivanje, ali često ne predstavljaju dobro glatke površine. Oni se uglavnom koriste kod kompjuterskih igara radi brzine prikaza. Ostale tehnike mogu prikazati glatke površine ali su sporije za prikazivanje. Jedna od takvih tehnika je upotreba parametarskih krivih i površina.

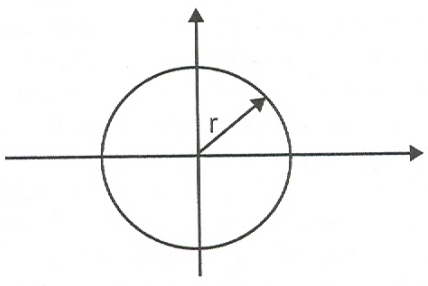
Parametarske krive

Implicitna funkcija za krivu u 3D prostoru je tipa
$f(x, y) = 0$

Npr. krug radijusa r centriran u 0 je definisan jednakošću

$x^2 + y^2 + r^2 = 0$

Sve tačke (x, y) koje zadovoljavaju gornju jednakost se nalaze na krugu.



Linija se može smatrati specijalnim slučajem krive. Linija sa dvije krajnje tačke, jedna na (x1, y1) i druga na (x2, y2) ima implicitnu jednakost

$y - m(x - x1) - y1 = 0$

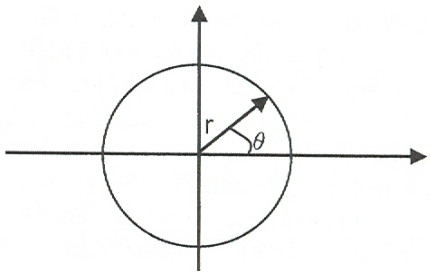
gdje je m=(y2-y1)(x2-x1) i naziva se nagibom linije

Sve tačke (x, y) koje zadovoljavaju gornju jednakost su pozicionirane na liniji. Ista kriva-linija može se predstaviti parametarskom funkcijom. Parametarske funkcije definišu svaku koordinatu po parametru koji izaberemo.

Naš krug se može predstaviti parametrom θ na sljedeći način

$x = r\cos\theta$, $y = r\sin\theta$

Kako θ varira od 0 do 360 mi proizvodimo tačke duž kruga.



Naša linija se takođe može definisati pomću parametra t kao

$x = (x2-x1)*t + x1$
$y = (y2-y1)*t + y1$

gdje t varira od 0 (definirajući krajnju tačku (x1,y1)) do 1 (definirajući krajnju tačku (x2.y2)). Može se pokazati da u 3D prostoru možemo definisati liniju po parametru t kao

$x = a_x t + d_x$
$y = a_y t + d_y$
$z = a_z t + d_z$

Za t=0 imamo krajnju tačku (d_x,d_y,d_z) i za t=1 imamo krajnju tačku $(a_x + d_x, a_z + d_z, a_z + d_z)$. Ove dvije krajnje tačke su dovoljne za definisanje cijele linije. Drugim riječima, one definišu granice linije i zovu se još kontrolne tačke linije.

U kompjuterskoj grafici često aproksimiramo krivu pomoću linearnih segmenata. Sgmenti se zovu linearni jer su definisani jednakošću koja je linearna po parametru t (t na prvi stepen). Linearna aproksimacija sfere se može napraviti pomoću poligona.

Linearna aproksimacija krive (ili površine) obično zahtijeva veliki broj vertexa i podataka i normalama da bi se aproksimirala glatka površina. Ovo je veoma teško za manipulaciju.

Pierre Bezier je prvi razvio skup parametraskih kubnih jednakosti koje predstavljaju krive i površine koristeći mali skup kontrolnih tačaka. Kasnije su razvijene mnoge druge funkcije koje proizvode bolje rezultate na račun efikasnosti. Treba imati na umu

da ove funkcije i dalje aproksimiraju krivu ili površinu, ali sa manje memorije i lakšom manipulacijom.

Kubične krive

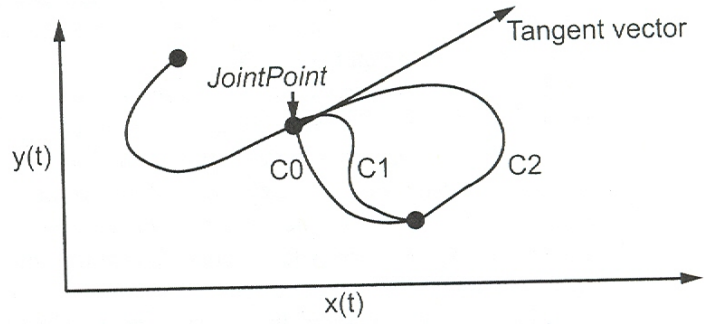
Krive se mogu aproksimirati skupom segmenata. Vidjeli smo kako se koriste segmenti koji su po prirodi linerani. Međutim oni ne moraju biti linearni. Oni mogu biti kvadratični (parametarski polinomi gdje je t dignuto na drugi stepen), kubični (t na treći stepen) ili više. U praksi, kubični polinomi su najlakši za korištenje i kontrolu, pa su veoma popularni u kompjuterskoj grafici.

Ako je svaki segment krive Q(t) = (x(t), y(t), z(t)), kubične polinomske jednakosti koje definišu (x,y,z) tačke na krivoj izgledaju ovako. Kubični polinomi imaju 4 koeficijenta (a, b, c i d) i zato segment krive ima 4 granice (kontrolne tačke) koje ga definišu

$x(t) = a_x t^3 + b_x t^2 + c_x t + d$
$y(t) = a_y t^3 + b_y t^2 + c_y t + d$
$z(t) = a_z t^3 + b_z t^2 + c_z t + d$

$(0<=t<=1)$

Različite vrste krivih (i površina) su klasificirane prema vrijednostima koeficijenata u polinomskim jednakostima. One određuju kako kriva interpolira kontrolne tačke. Neke od često korištenih krvih su Bezier, B-Spline, Hermite itd. Prve dvije ćemo obajsniti u nastavku.



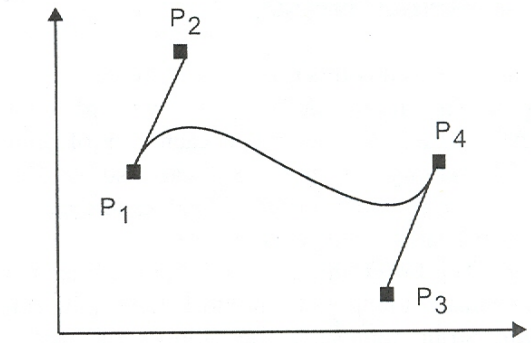
segmenti krive se susreću u joint point

Kako je kriva aproksimirana sa više od jednog krivog segmenta, važno je razumjeti kako se segmenti krive ponašaju u tački u kojoj se susreću, tzv. joint point. Ako se dva kriva segmenta krive susreću, onda oni imaju C0 kontinuitet. Da bi kriva bila kontinualna, ona mora imati kontinuitet najmanje C0. Ako su uz to vektori tangenti dva segmenta krive (prvi izvod, takođe poznat kao brzina), jednaki na joint point, za krivu se kaže da ima C1 kontinuitet. Ako je zakrivljenost dva segmenta krive takva (drugi izvod, još poznata kao ubrzanje krive) također jednaka u joint point, onda se za krivu kaže da je C2 kontinualna kao što je prikazano na slici. C1 kontinuitet je neophodan ako želimo biti sigurni u glatkost krive. C2 kontinuitet garantuje glatki prelaz između segmenata.

Slines i posebno NURBS imaju C1 i C2 kontinuitet na svojim sastavnim tačkama i često se preferiraju nad drugim tipovima krive.

Bezier-ove krive

Kubični (Bezier-ov) segment krive ima 4 kontrolne tačke: dvije od njih, P1 i P4 definiraju krajeve krive, a druge dvije, P2 i P3, utiču na oblik krive kontrolišući vektore tangenti krajeva.



Bezierova kriva definisana sa 4 kontrolne tačke

Kao što je prikazano na slici, početna i krajnja tangenta su određene vektorima P1P2 i P3P4. Kažemo da Bezierova kriva interpolira (prolazi kroz) prvu i zadnju kontrolnu tačku i aproksimira druge dvije.

Dva Bezier-ova segmenta krive mogu se konstriusati da imaju C0 i C1 kontinuitet ako možemo osigurati da dijele sastavnu tačku (joint point) P4 i da su vektori tangenti u P4 jednaki ili da je P5-P4 = P4-P3 kao što je prikazano na slici.

Možemo aproksimirati cijele krive pomoću Bezierovih segmenata na ovaj način. Za kompleksnije krive, matematika koja osigurava C1 kontinuitet segmenata duž krive prilagođavanjem kontrolnih tačaka postaje komplikovana. Bolje rješenje nude splines.

B-Splines

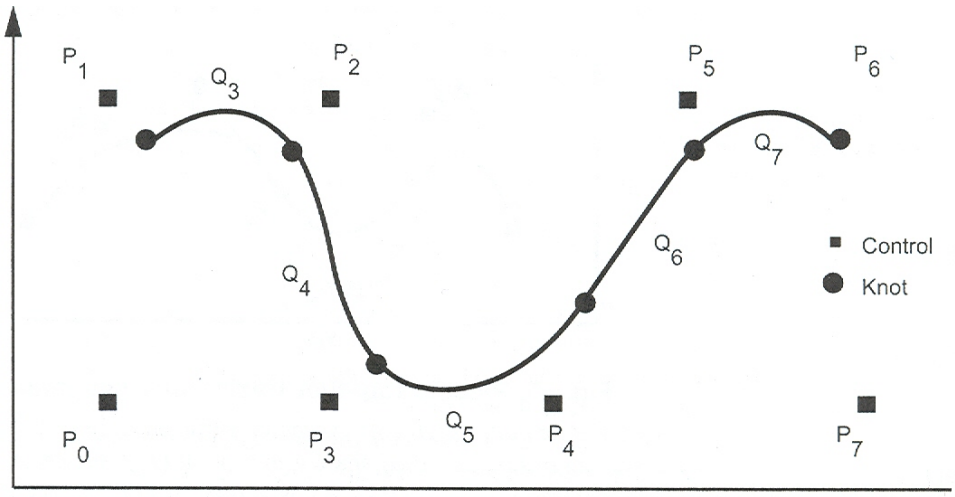
Spline krive potiču od fleksibilnih traka koje se koriste za kreiranje glatkih krivih u tradicionalnom tehničkom crtanju. Nalik Bezier-ovim krivim, one se formiraju matematički iz aproksimacija kubičnih polinomskih funkcija.

B-splines su jedan tip spline-a koji je možda najpopularniji za primjenu u kompjuterskoj grafici. Kontrolne tačke za cijelu B-spline krivu su definisane u konjunkciji. One definišu C2 kontinulane krive, ali individualni segmenti ne moraju prolaziti kroz kontrolne tačke. Susjedni segmenti dijele kontrolne tačke, pa je su ovaj način nametnuti uslovi kontiniuiteta. Zbog ovog razloga, kada razmatramo spline, mi razmatramo cijelu krivu (koja se sastoji od svojih segmenata) umjesto njene individualne segmente koji se moraju spojiti. Kubične B-spline su definisane pomoću serije od m=n+1 kontrolnih tačaka $P_0,P_1...P_n$.

Svaki segment splinea Qi 3<=i<=n

je definisan pomoću 4 kontrolne tačke P_{i-3}, P_{i-2}, P_{i-1}, P_i.

Npr, na slici je prikazan spline sa m=8 kontrolnih tačaka. Individualni segmenti krive su Q_3, Q_4, Q_5, Q_6 i Q_7. Q_3 je definisan sa 4 kontrolne tačke, $P_0 - P_3$, Q_4 tačkama $P_1 - P_4$ itd.



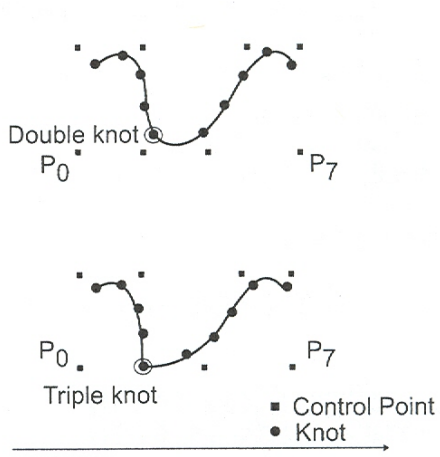
Uniformni neracionalni B-spline

Vidimo da svaka kontrolna tačka utiče na 4 segmenta. Npr. na gornjoj slici, tačka P_4 utiče na segmente Q_3, Q_4, Q_5 i Q_6. Pomjeranje kontrolne tačke će uticati na ova 4 segmenta, ali ne i na cijelu krivu. Ovo je veoma korisna osobina B-spline-a.

Tačke spajanja (joint points) između segmenata nazivaju se **knots**. Knot između segmenta i i segmenta i+1 se označava sa k_i. Inicijalna tačka prvog segmenta i krajnja tačka zadnjeg segmenta se također zovu knots, tako da imamu ukupno n-1 knotova za spline koji razmatramo. Kada su knotovi uniformno postavljeni, kao što je prikazano na slici, spline se zove **uniformni neracionalni spline**. Nažalost, taško je definisati i kontrolisati spline jer segmenti ne interpoliraju kontrolne tačke.

Neuniformni neracionalni B-spline se definiše sa n+5 knotova. Knotovi ne moraju biti uniformno razmaknuti i određeni su od strane korisnika. Prednost je da možemo prisiliti krivu da interplira određene kontrolne tačke. Neuniformni B-spline koristi pojam sekvence vrijednosti knotova: nesilazna sekvenca vrijednosti knotova koja definiše njihova mjesta na krivoj. Npr, ako pretpostavimo da je gornja kriva bila neuniformni neracionalni B-spline, njena knot sekvenca bila bi (0,1,2,3,4,5,6,7,8,9,10,11). (n+5=12 knotova)

Ako su usšješne knot vrijednosti jednake u sekvenci, to se zove **multiple knot**. Multiple knotovi uzrokuju da kriva aproksimira pridruženu kontrolnu tačku bliže. U stvari, 3 uspješne knot vrijednosti prisiljavaju krivu da interpolira kontrolnu tačku, omogućavajući da je oblik krive lakši za definisanje. Definisanje multiple knotova ne vodi gubitku kontinuiteta, osima na pridruženoj kontrolnoj tački. Ako modificiramo našu krivu gore pretpostavljajući različitu knot sekvencu, rezultati izgledaju ovako:



Neuniformni neracionalni B-spline. Multiple knots, dupli knot sa knot sekvencom (0,1,2,3,4,4,5,6,7,8,9,10) uzrokuje C1 kontinuitet, dok trostruki knot sa knot sekvencom (0,1,2,3,4,4,4,5,6,7,8,9) rezultira u samo C0 kontinuitetu

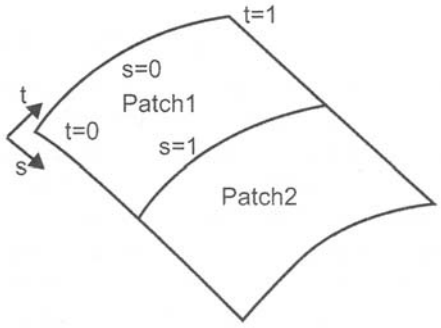
Nurbs

B-splines (ili bilo koje druge neracionalne krive) mogu se definisati homogenim koordinatama dodavanjem W(t)=1 kao četvrtog elementa u parametarskoj jednakosti; Q(t)=(X(t), Y(t), Z(t), W(t)). (Kao i obično, prebacivanje u homogene koordinate u 3D prostoru uključuje dijeljenje sa W(t)). Ovaj proces se zove racionalizacija krive. Prednost racionalnih krivih je da su one invarijantne pod roracijom, skaliranjem, translacijom i perspektivnim transformacijama. Ovo znači da treba da primijenimo transformacije samo na kontrolne tačke i onda reevaluiramo jednakost krive da bi smo generisali transformisanu krivu. Neuniformni racionalni B-spline se još zove NURB i često se koristi u kompjuterskoj grafici.

Parametarske bikubične površine

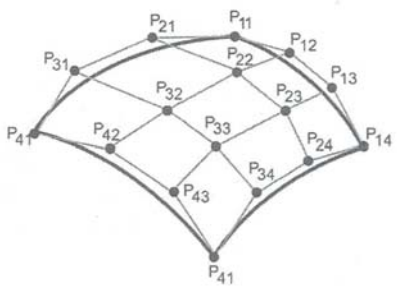
Parametarske bikubične površine su generalizacija kubičnih krivih. Površina je izdijeljena u segmente parametarskih površina ili patcheve, koji su slijepljeni zajedno da formiraju cijelu površinu.

Parametarska jednakost za svaki patch je definisana preko dva parametarska domena – s i t, definišući površinu umjesto krive.



C0 i C1 kontinuitet kao granica površine

Svaki patch je definisan blendingom kontrolnih tačaka (4*4=16 kontrolnih tačaka je potrebno za definisanje patch-a). Zapravo, možemo zamisliti kontrolne tačke kao da definišu krive na specifičnim intervalima domena koji su onda spojeni u mesh i definišu površinu.

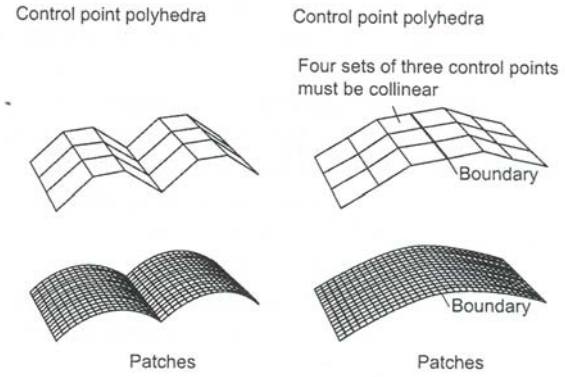


Kontrolne tačke za patch

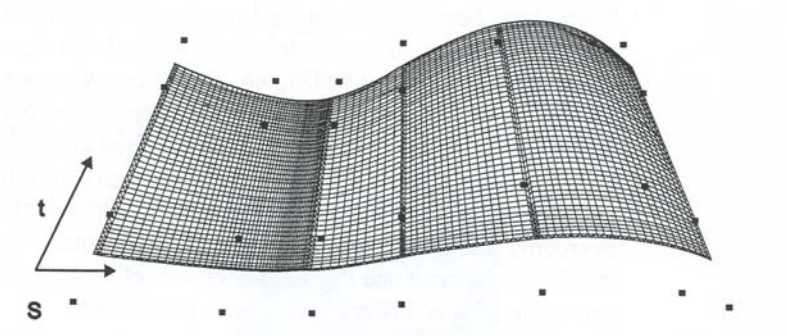
Ograničenja kontinuiteta za površine su ista kao za krive. C0 kontinuitet implicira da se dva patch-a sreću na graničnoj krivoj. C1 kontinuitet implicira da dva patch-a imaju glatki prelaz na graničnoj krivoj itd.

Osnovna funkcija koja se koristi za Bezier-ove krive može se proširiti duž dva parametarska domena da definiše Bezier-ove komadiće površine. Ove površine nisu mnogo popularne jer imaju poteškoća u spajanju kontinualnih površina.

NURBS površine mogu se definisati kao logičko proširenje jednakosti za NURBS krive, kao što smo vidjeli ranije. Ove površine imaju iste osobine kao i NURBS krive – C2 kontinuitet, mogućnost uvođenja diskontinuiteta kroz poziciju knotova i invarijantnost na transformacije. One su jako rasprostranjene u definisanju vanjskih krivina površine.



C0 i C1 kontinuitet na granicama dva komadića površine



NURBS površina

11.3. Zapreminska reprezentacija

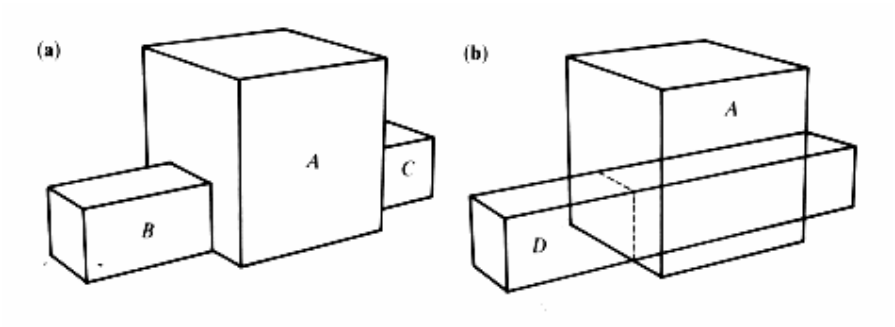
Kao što je ranije spomenuto, zapreminski načini modeliranja konstruišu objekte iz skupa 3D osnovnih zapremina umjesto da kreiraju graničnu površinu koja zatvara željenu zapreminu. Dva glavna pristupa su konstruktivna čvrsta geometrija (CSG - constructive solid geometry) i šeme prostorne podjele. Druga tehnika ima ograničenu primjenu u komercijalnoj kompjuterskoj animaciji, iako u zadnje vrijeme postaje važna u naučnoj i medicinskoj vizualizaciji.

Posljedica ovih alternativnih šema je da one utiču na algoritme potrebne za generisanje finalne slike i opseg efekata za animaciju koje je moguće kreirati. Prvo razmotrimo ideje vezane za CSG.

11.3.1. Konstruktivna čvrsta geometrija

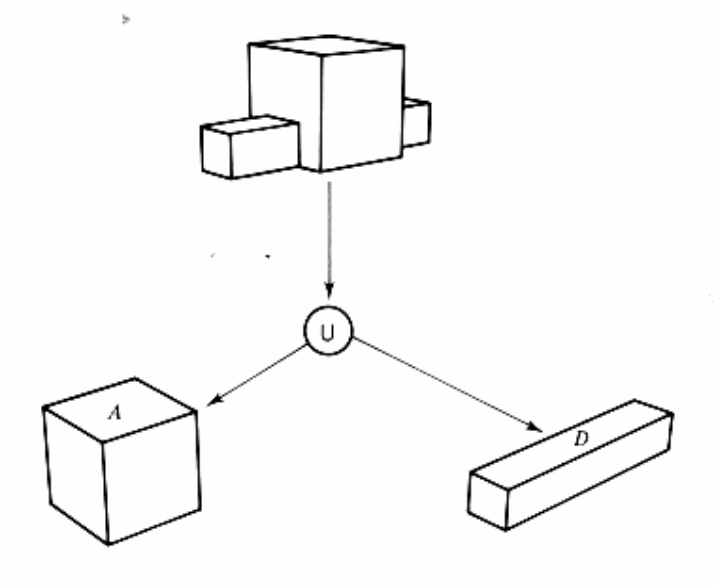
Konstruktivna čvrsta geometrija ili set-theoretic modeliranje pokušava da napravi objekat od malog skupa 3D formi kao što su box, sfera, cinidar, konus, torus i heliks. Ovo može izgledati kao nemoguć zadatak što u nekim primjenama i jeste, kao npr. u modeliranju ljudskog lica. Zato se ova tehnika koristi u modeliranju scena koje imaju inherentnu geometrijsku formu. Npr. neke mehaničke komponente ili arhitektonske strukture se mogu napraviti pomoću CSG-a.

Jedna od neobičnih osobina ove tehnike je sposobnost sabiranja i oduzimanja zapremina, kao i identificiranje zapremine prostora koju dijele dvije primitive koje se sijeku. Počnimo sa razmatranjem operatora unije koji kombinira dva objekta.



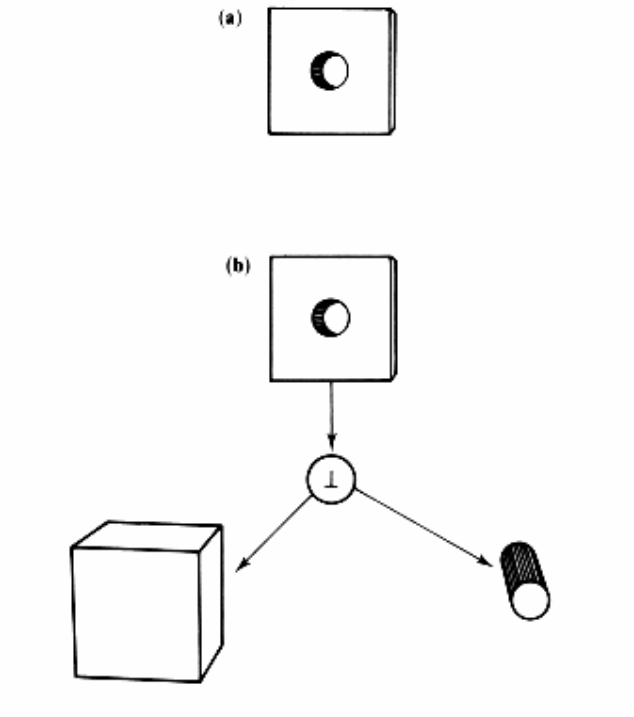
slika 15

Npr. recimo da je potrebno napraviti objekat sa slike 15a. Možemo vidjeti da se on sastoji od tri blok elementa A, B i C, ali ako su B i C identični, onda se može zamisliti da se objekat sastoji od dva bloka A i D koji se presijecaju, kao što je prikazano na slici 15b. Koristeći notaciju CSG-a, finalni objekat se konstruiše kao unija A i D kao što je pokazano dijagramom stabla na slici 16.



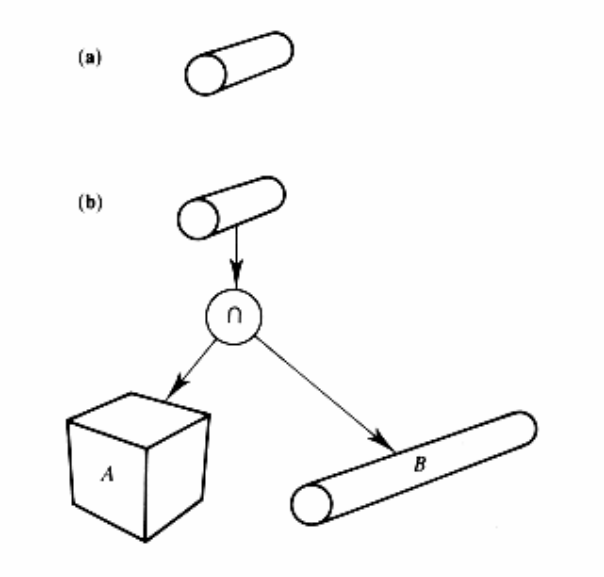
slika 16

Kao drugi primjer razmotrimo objekat pokazan na slici 17a - to je blok koji sadrži rupu. On može biti definiran koristeći operaciju razlike kojoa oduzima jednu zapreminu od druge i pokazana je na slici 17b zajedno sa dijagramom njenog stabla.



slika 17

Na kraju, operacija presjeka identificira zapreminu koju dijele dva objekta koja se sijeku. Slika 18a ilustrira objekat koji može biti određen zajedničkim prostorom koji dijele dva objekta A i B na slici 18b. Pomoću unije, razlike i presjeka mogu se napraviti i mnogo kompleksnije strukture.



slika 18

Gornja šema izgleda snažna i jeste, ali da bi radila potreban je mehanizam koji bi je implementirao u kompjuterskom okruženju, gdje stupaju na scenu gometrija i matematika.

Iako postoji jednačina sfere, nema ekvivalentne jednačine za box, međutim moguće je izvesti jednu jednačinu iz druge. Razmotrimo jednačinu ravni:

$ax + by + cz + d = 0$

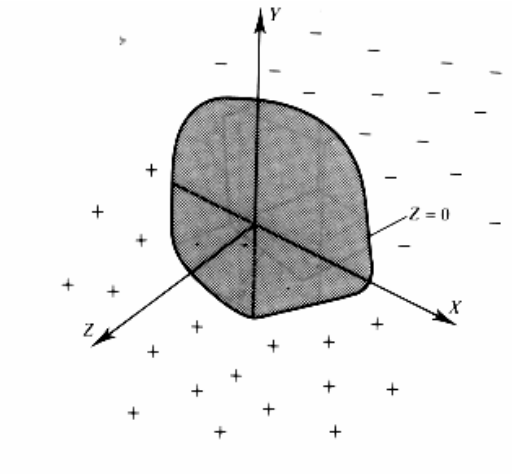
Ako se za a, b i c uzmu određene vrijednosti kao što su

$a = 0, b = 0, c = 1$ i $d = 0$,

onda jednačina postaje

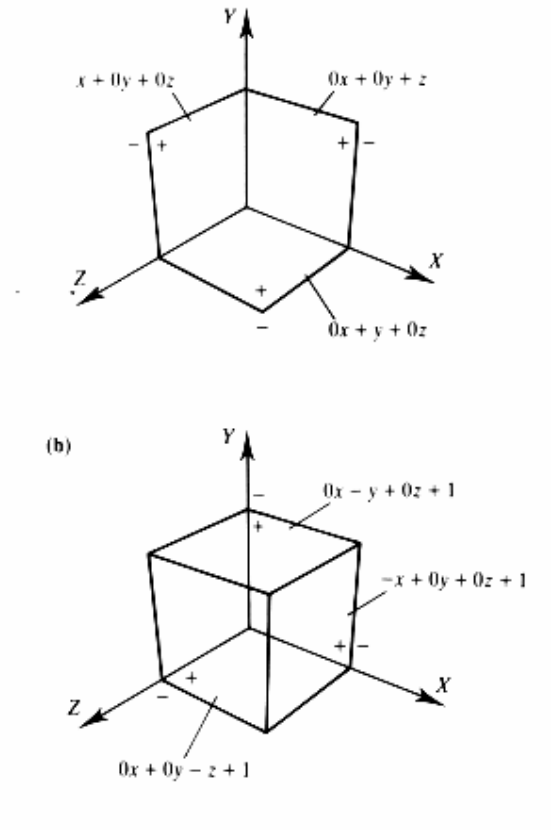
$0x + 0y + z = 0$

Postoji beskonačan broj tačaka koje zadovoljavaju ovu jednačinu i bez obzira na vrijednosti x i y, z mora biti 0. Geometrijski, ova jednačina definira ravnu površinu paralelnu x i y osama, gdje je $z = 0$, kao što je prikazano na slici 19.



slika 19

Ako razmotrimo izraz na lijevoj strani jednačine, još jednom ćemo primijetiti da bez obzira na vrijednosti x i y, ako je vrijednost z pozitivna, izraz će biti pozitivan, a ako je vrijednost z negativna, izraz će biti negativan. Mijenjajući znak jednakosti za različite vrijednosti koordinata, otkrivamo da je prostor podijeljen na dva dijela pomoću ravni, odvojen gdje je jednakost zadovoljena. Slika 20 ilustruje taj efekat dijeljenja, odakle postaju dva poluprostora. CSG koristi dijeljenje prostora za konstruisanje svojih zapreminskih primitiva.



slika 20

Kao primjer, jedinična kocka se može konstruisati iz presjeka šest ravnih površina. Slika 20a ilustrira izraze izvedene iz jednačina ravni za prve tri ravni, a slika 20b za

druge tri ravni. Primijetimo da su izrazi uređeni tako da je njihov pozitivni poluprostor unutar zapremine kocke. Bilo koja tačka (x, y, z) biće unutar kocke ili na njenoj površini, ako svih šest izraza ima vrijednost koja je pozitivna ili nula:

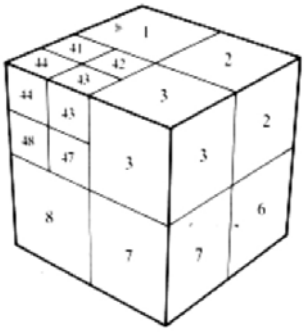
$0x + 0y + z >= 0$
$0x + y + 0z >= 0$
$x + 0y + 0z >= 0$
$0x - y + 0z + 1 >= 0$
$-x + 0y + 0z + 1 >= 0$
$0x + 0y - z + 1 >= 0$

Zamijenimo npr. tačku (1, 1, 1) (koja je vrh kocke) u gornjim izrazima. Njihovo razvijanje proizvodi : 1, 1, 1, 0, 0, 0 koji su 0 ili pozitivni, što znači da tačka nije unutar kocke. Međutim, tačka (2, 1, 1) koja je izvan kocke, proizvodi vrijednosti: 1, 1, 2, 0, -1, 0 koje sadrže negativnu vrijednost, potvrđujući da je tačka vani. Renderer sada može upotrijebiti ovaj rezultat da identificira tačke koje leže na površini objekta.

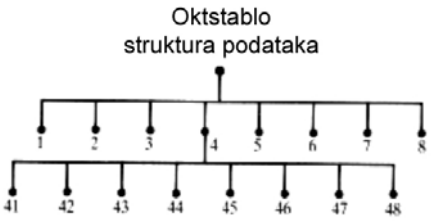
Druga snažna osobina CSG načina modeliranja odnosi se na mogućnost ispitivanja unutrašnjosti objekta (pretpostavljajući da je on već modeliran). Ovo je postignuto presijecanjem objekta sa ravni koja ga dijeli na dva dijela, i tako je moguće vidjeti detalj koji postoji samo na jednoj strani površine. Ova osobina omogućava kreiranje animacija koje uključuju kretanje po unutrašnjosti objekta.

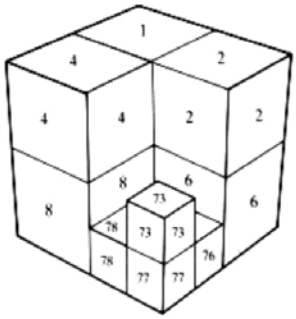
11.3.2. Prostorna podjela

Algoritmi modeliranja pomoću prostorne podjele pokušavaju konstruisati 3D objekte iz familije ugniježdenih box zapremina ili zapremine prostora podijeljene u voksele. Npr. kocka prostora prikazana na slici 21 može se podijeliti u osam manjih kocki koje zatim mogu biti podijeljene u drugih osam bez ikakve realne granice. Ovakva prostorna podjela se može predstaviti dijagramom stabla kako je prikazano na istoj ilustraciji, koji također identificira zapremine sa brojevima. Dijagram stabla, koji je poznat kao oktstablo (octree), se koristi da opiše geometriju objekta bilježenjem onih prostornih podjela koje su totalno sadržane u objektu ili dalje podijeljene.

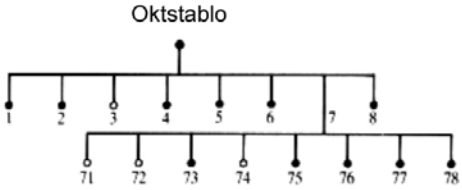


slika 21





slika 22



Slika 22 prikazuje kako se jednostavni blok objekat može predstaviti pomoću oktstabla. Što detalji postaju finiji, stablo postaje dublje, što proizvodi važna pitanja vezana za kompleksnost i smještanje u memoriju. Međutim, ovaj način modeliranja je relativno nov i predmet je daljeg istraživanja.

Druga šema prostorne podjele pretpostavlja da je zapremina prostora podijeljena u određen broj zapreminskih elemenata koji se zovu **vokseli**, 3D ekvivalent za piksele. Ove zapremine se identične veličine i mogu biti u jednom od dva stanja - ili su puni ili su prazni, ne postoji međustanje.

Voksel šeme su se pokazale veoma korisnim u 3D reprezentaciji struktura koje su skenirane uz pomoć CT skenera koji se koristi u meducinskoj dijagnostici. Svakom vokselu je dodijeljena vrijednost koja predstavlja tip materijala koji on kodira (npr. kost, meso ili masnoća) i uz pomoć specijalnih programa za sjenčenje mogu se generisati slike koje prikazuju interni pogled na zapreminu. Napravljena je veoma snažna alatka za vizualizaciju za hirurge kada su neki vokseli napravljeni transparentnim, a neki neprovidnim; ovo omogućava da se vidi samo struktura kostiju. Animirane sekvence skupova voksel podataka su veoma korisne u medicinskom obrazovanju jer pokazuju unutrašnji izgled ljudskog tijela bez vršenja operacije.

11.3.3. Proceduralno modeliranje

Dosada smo istraživali standardne tehnike za predstavljanje objekata koji su dio našeg svakodnevnog života, kao što su automobili, stolovi, zgrade, čajnici i leteći logotipi. Međutim, postoji mnogo drugih stvari koje će prouzrokovati probleme ako ih pokušamo modelirati pomoću poligona, komadića ili voksela. Npr. kako možemo modelirati snijeg ili kišu? Ili valove koji se razbijaju o pješčanu plažu? I šta sa planinama, drvećem, kosom, vatrom, maglom i elekrtičnim poljima? Lista je beskrajna i pronalaženje šeme za modeliranje ovakvih objekata trajaće još mnogo godina. Međutim, postoji nekoliko šema koje treba ispitati i primijeniti u nekim specifičnim oblastima.

Proceduralno modeliranje uvodi skup tehnika koje uključuju kompjuterske procedure da bi izračunale geometriju nekog oblika. Npr. ako želimo modelirati planinu, očigledno je nemoguće opisati svaki prevoj, brdo i dolinu. Nadalje, uopće ne dolazi u obzir pozicioniranje svake stijene ili kamena. Ali, ako prebacimo ove odluke o dizajnu proceduri, onda nam ona može dati rješenje koje zadovoljava naše zahtjeve. Procedura može biti tako uređena da prihvata određene parametre koji utiču na njeno ponašanje u toku modeliranja.

11.3.3.1. Fraktali

Jedan od prvih koncepata koje ćemo istraživati je dimenzija. Svi znamo i prihvatamo da naš fizički svijet ima tri dimenzije, ali šta ovaj parametar zapravo znači? Jedan odgovor potiče iz činjenice da se stvari koje postoje u načem univerzumu mogu geometrijski opisati sa tri koordinate koje definišu poziciju u prostoru. Ovakav prostor se zove Euklidov, ako su zakoni za mjerenje udaljenosti, površina i zapremina linearni. Ovo neće biti slučaj kada su u pitanju veoma velike udaljenosti, ali prostor koji nas neposredno okružuje je trodimenzionalan i linearan.

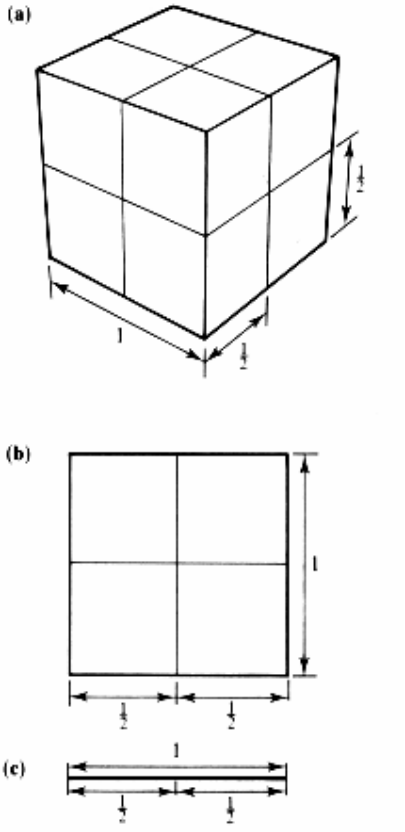
Drugi važan koncept u svijetu fraktala je **samosličnost**. Ovo je osobina koja pripada nekom oblicima koji, kada su uvećani, ponavljaju isti detalj koji sadrži originalni oblik. Pahuljica snijega je dobar primjer; njene kristalne ivice se sastoje od kristala koji imaju sličan oblik kao što je i ona sama. Međutim, ovi kristali se prestaju pojavljivati ako se gledaju pod jačim uvećanjem i pojavljuje se novi tip detalja.

U apstraktnom svijetu matematike virtualno je sve moguće. Matematičari su tokom vremena konstruisali veliki broj imaginarnih formi koje se mogu sada razumjeti kroz

razvoj fraktala. Prije nego što razmotrimo neke od tih apstraktnih koncepata, potražimo još neke artefakte koji posjeduju osobinu samosličnosti.

Kocka ima tu osobinu jer se može konstruisati od osam sličnih kocki skaliranih faktorom polovine. Slika 23a ilustrira ovu relaciju. Sada trebamo formulu koja će pokazati odnos dimenzija (3) sa brojem samosličnih dijelova (8) i faktorom skaliranja (1/2), ali prije toga razmotrimo jedno- i dvo-dimenzionalne slučajeve.

Kvadrat je također samosličan jer se može opisati sa kvadratima. Slika 23b pokazuje da je kvadrat ekvivalentan sa četiri samoslična kvadrata koja su skalirana faktorm polovine. Još jednom želimo da pronađemo relaciju između dimenzija (2), broja sličnih dijelova (4) i faktora skaliranja (1/2).



slika 23

Na kraju, segment prave linije je samosličan i ekvivalentan sa dva samoslična segmenta skalirana faktorom polovine, kao što se vidi na slici 23c. Opet želimo otkriti relaciju između dimenzija (1), broja samosličnih dijelova (2) i faktora skaliranja (1/2).

Tabela 2 sadrži informacije koje smo saznali iz ova tri primjera.

Dimenzija	Samoslični dijelovi	Faktor skaliranja
D	N	S
1	2	½
2	4	½
3	8	½

tabela2

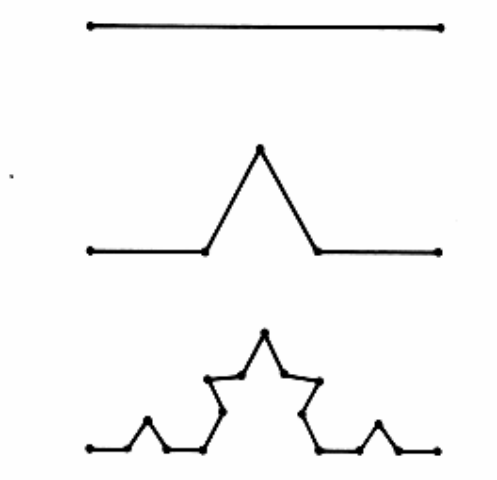
Pravilo koje povezuje ova tri primjer jeste:

$N = 1/S^D$

gdje je N broj samosličnih dijelova, S faktor skaliranja i D dimenzija. Rearanžiranje ove formule omogućava definiciju dimenzije u zavisnosti od samosličnih dijelova i faktora skaliranja:

$D = \log(N)\log(1/S)$

Ovo se može dokazati uvrštavanjem N=8 i S=1/2, odakle dobijamo D=3. Sada imamo koristan način za definiranje dimenzije koji nas uvodi u svijet fraktala.



slika 24

Prvi primjer fraktala je otkriven u matematičkoj krivoj pahuljice (von Kochovoj krivoj) koja se može formirati zamjenom segmenata linije sa četiri slična segmenta linije koji su skalirani na odgovarajući način kao što se vidi na slici 24. Faktor skaliranja S zamijenjenih segmenata linije je 1/3, a njih ima 4. Znači, S=1/3 i N=4 pa je

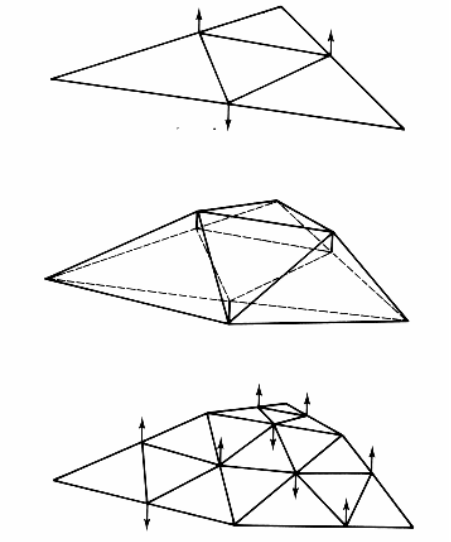
$D=\log(4)\log(1/3)=1.2619...$

Ovakva vrijednost za dimenziju u početku izgleda iznenađujuće, ali sjetimo se da to više nije povezano sa nečim što fiziČlki postoji u našem univerzumu, to je matematička konstrukcija i termin ‘dimenzija' se ne koristi u njenom originalnom značenju. Da bi se izbjegla ova konfuzija uveden je termin “fraktal” ili “dimenzija sličnosti”. Uprkos tome, iako je kriva pahuljice još uvijek kriva sa topološkom dimenzijom 1, ona ima važnu osobinu da njena fraktalna dimenzija 1.2619... govori da je to entitet koji ima i “linijske” i “površinske” kvalitete - to je fraktal. Zapravo, postoji klasa prostornih krivih koje imaju topološku dimenziju jedan, a fraktalnu dimenziju dva, jer zauzimaju sav raspoloživi prostor u ravni.

Ne postoji matematička funkcija koja daje x i y koordinate pahuljice, one se moraju izvesti pomoću algoritma koji računa kompjuterski program. Fraktali se generišu

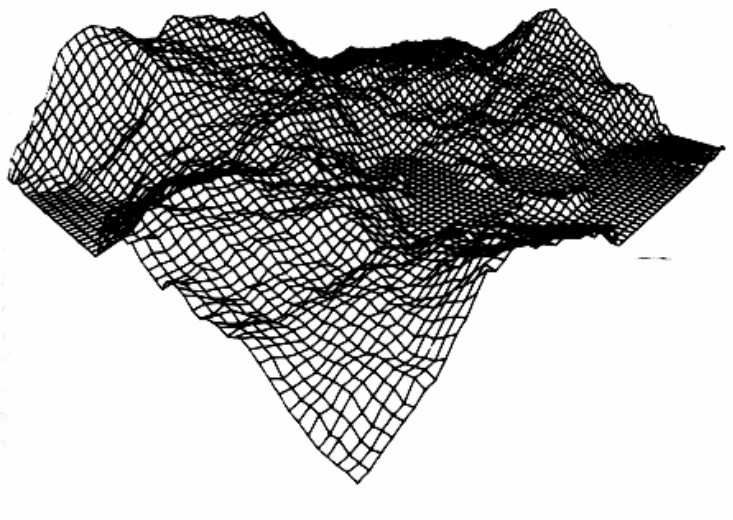
pomoću procedura i relativno jednostavan program je sposoban da crta krivu pahuljice u različitim nivoima detalja.

Imajući u vidu da su sami fraktali imaginarne konstrukcije matematičara i da imaju beskonačan rekurzivni nivo detalja, jedini način kako kompjuter može njima manipulisati je aproksimacija. Štaviše, ako fraktalne forme postoje u prirodi, one nikada nemaju onu preciznost u samosličnosti kao što ima njihova teoretska forma, pa se zato zovu "statistički samoslične".



slika 25

Iako je kriva pahuljice interesantna, ona ima ograničenu primjenu u kompjuterskoj animaciji. Koja je korist od fraktala? Oni postaju izuzetni korisni kada se krećemo u tri dimenzije i uvedemo slučajne brojeve. Npr. razmotrimo trougao sa slike 25a; on može biti zamijenjen sa četiri manja trougla izvedena iz srednjih tačaka njegovih stranica. Oni su također pomaknuti za slučajan iznos što odražava fraktalnu dimenziju koja je potrebna da bi bila proporcionalna dužinama strane. Može se napraviti površina koja je nabrana prema potrebi ponavljanjem dijeljenja trouglova u još manje trouglove. Slike 25b i 25c ovo ilustruju.



slika 26

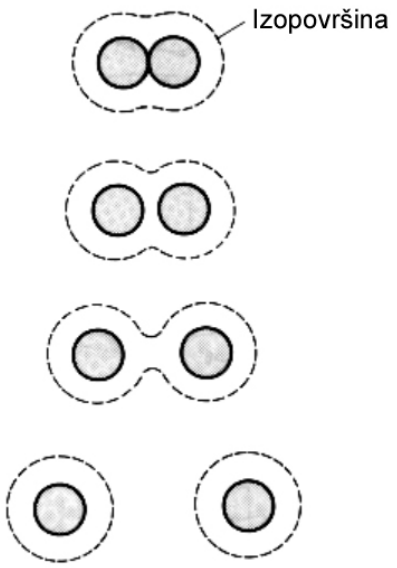
Slika 26 ilustruje fraktalnu površinu koja ima određeni skup visina do neke zajedničke vrijednosti i kreira neku pseudo-planinu ili morsku površinu. Još jednom, veoma je jednostavno napraviti program koji može kreirati fraktalni pejzaž koji se bazira na nekim inicijalnim karakteristikama visine i dimenzijom sličnosti koja kontroliše detaljnost površine.

Drugi način upotrebe fraktala u kompjuterskoj grafici jeste kreiranje nizova brojeva generisanih procesom podjele, koji se zatim koriste kao mape tekstura izmaglice i oblaka. Moderni komercijalni sistemi za animaciju obično posjeduju alatke za modeliranje fraktalnih terena i formiranje fraktalnih mapa tekstura.

11.3.3.2. Soft objekti

Soft objekti se mogu najbolje opisati pomoću fizičkog primjera. Zamislimo da je električno nabijena sfera obješena na plafon u sredini sobe i da se ne može micati i da mi imamo metar koji mjeri jačinu električnog polja koje ona isijava. Ovaj metar se može koristiti da nacrta jačinu električnog polja očitavanjem iz različitih pozicija. Jedan od načina za izvođenje ove operacije je mjerenje na tačkama sa konstantnom udaljenošću, možda u obliku obične kubične rešetke. Iz ove 3D matrice očitavanja možemo nacrtati površinu konstantne električne jačine - ovakva ‘izopovršina' će izgledati kao skup koncentričnih sfera sa jačim očitavanjem u centru i slabijim kako se odmičemo od sfere.

Ako posmatramo drugu nabijenu sferu, koja dodiruje prvu, i ponovimo isti eksperiment mjerenja, otkrićemo različitu familiju izopovršina koje se sastoje od dvije odvojene sfere koje se sijeku. Međutim, na tački presjeka umjesto precizne granice između sfera primijetićemo “soft” spoj gdje se jedan električni potencijal dodaje na drugi. Ako se jedna od sfera lagano pomjeri relativno u odnosu na drugu, originalna izopovršina će izgledati kao da se stiska da bi održala kontinuitet površine. Ali, u određenoj tački površina će se podijeliti u dvije odvojene izopovršine koje okružuju dvije sfere iako će one još uvijek odražavati neku formu privlačenja i postati rastegnute duž linije koja povezuje njihove centre. Slika 27 pokazuje ovaj eksperiment kao 2D presjeke.



slika 27

Soft objekti ne zahtijevaju električne naboje; oni su implicitno definisani pomoću jednakosti koje opisuju skalarno polje. Takve jednakosto mogu definisati sferu, cilindar, elipsoid, torus i ravan, koji se, kombinirani na različitim udaljenostima, mogu koristiti za konstruisanje različitih struktura.

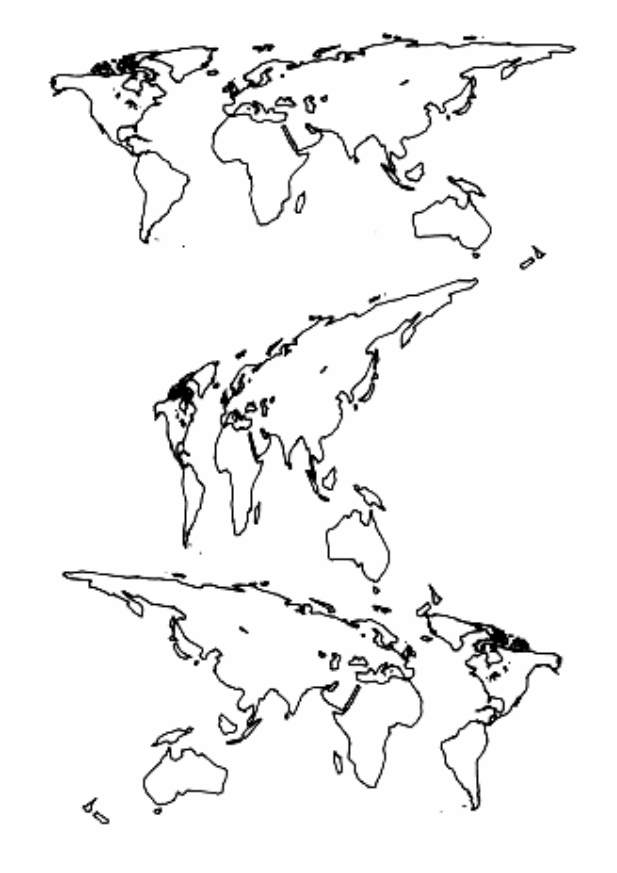
Kako je izopovršina kreirana od aditivnih i subtraktivnoh efekata skalarnih polja, ista procedura se može koristiti za manipuliranje drugim atributima, kao što je boja. Npr. recimo ako su crvena, plava i zelena sfera bile postavljene na vrhove trougla, izgled soft objekta bi inicijalno bio tri odvojene sfere (ako pretpostavimo da su dovoljno udaljene). Kako bi se one primicale centru trougla, tri izopovršine ne samo da bi bile poremećene prisustvom drugih objekata, nego bi prilikom tog poremećaja apsorbirale boju susjeda. Nivo influence koji se odražava na izopovršini bi se kretao od tri odvojene površine do jedne soft elastične površine koja sadrži tri objekta, sa bojom koja je raspoređena u skladu sa njihovim fizičkim odnosom.

Ovaj tip modeliranja omogućava animiranje fluida, posebno kada se simuliraju mokre ljepljive površine, jer se one mogu modelirati pomoću sistema sfera postavljenih tako da kreiraju jednu kontinuiranu neravnu izopovršinu.

11.3.3.3. Proceduralna manipulacija

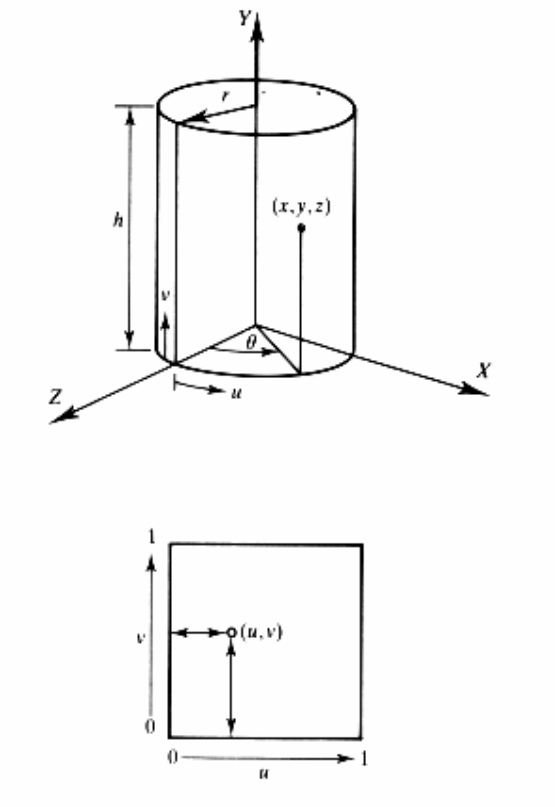
Proceduralne tehnike se također mogu koristiti za manipuliranje podacima o koordinatama za kreiranje 3D objekata kao npr. transformiranje 2D skupova podataka u 3D strukture.

Slika 28 pokazuje mapu svijeta koja je prilikom digitalizacije postala fajl sa xy koordinatama, ali se dodavanjem z koordinate gdje je z=0 brzo pretvara u 3D planarne površine kojima se može manipulirati kao bilo kojim drugim 3D objektom.



slika 28

Druga manipulativna tehnika je omotavanje mape oko nekog oblika kao što je cilindar ili sfera. U slučaju cilindra, treba da definiramo 2D sliku kao mapu adresiranu koordinatam u i v, i zatim da nađemo maping funkcije koje će ih transformirati u tačke (x, y, z) na cilindru.



slika 29

Ako su zadati uslovi kao na slici 29 gdje je:

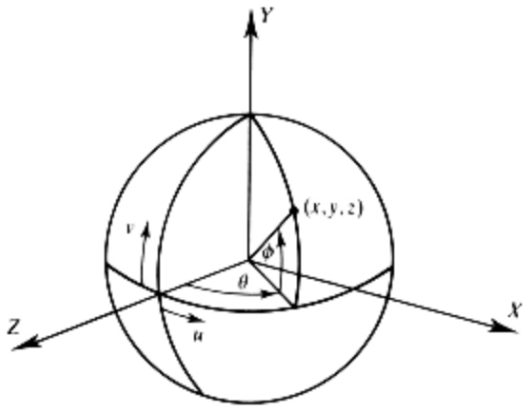
r - radius cilindra
h - visina cilindra

0<=u<=1
0<=v<=1
onda je ugao θdat sa

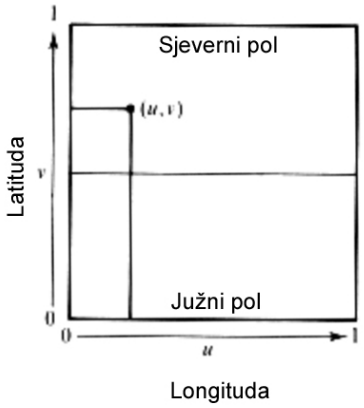
$\theta = 2\pi\upsilon$

i vrijednosti za x, y i z su

x=rsin(u), y=vh, z=rcos(u)



slika 30



Druga korisna funkcija mapira tačku (u,v) u tačku (x,y,z) na sferi. Geometrija za ovu operaciju je prikazana na slici 30 gdje je sfera sa radiusom r locirana sa centrom u koordinatnom početku. Još jednom imamo 2D mapu adresiranu koordinatama u i v, gdje je u longituda, a v latituda. Longitudinalni ugao θje dat sa

$\theta = 2\pi\upsilon$

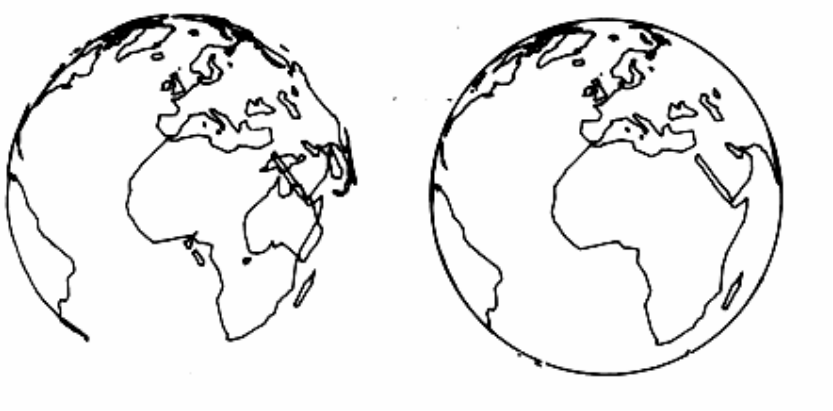
a ugao latitude φ je dat sa

$\varphi = \pi(v - 0.5)$

pa su onda vrijednosti x, y i z

$x = r\sin(\theta)\cos(\varphi)$, $y=r\sin(\varphi)$, $z=r\cos(\theta)\cos(\varphi)$

Ako ove funkcije sada primijenimo na mapu svijeta prikazanu na slici 28, dobivamo 3D objekat sa slike 39. Lijeva slika je prikazana transparentno, dok je desna slika zatvorena krugom i uklonjene su sakrivene linije. Ne smijemo zaboraviti da je ova slika samo kolekcija linija i zato se ne može renderovati. Međutim, 2D mapa se može transformisati u mrežu trouglova i zatim na nju primijeniti sferično omotavanje. To bi stvorilo 3D graničnu površinu koja se može renderovati.

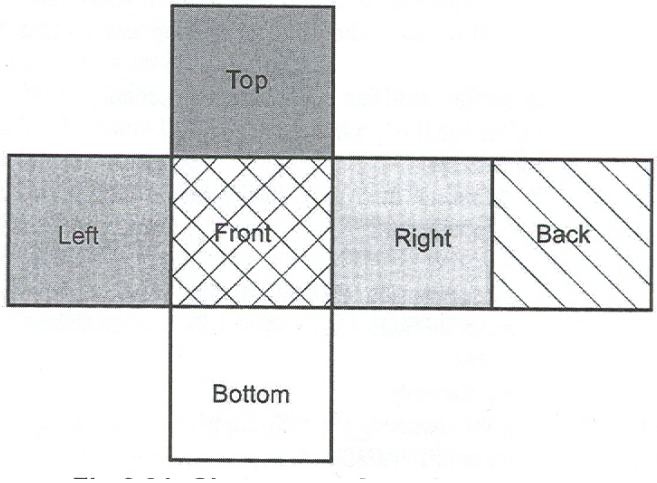


slika 31

Environment mapping

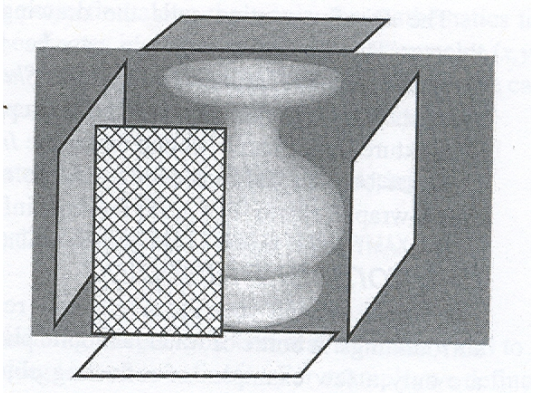
Ako pogledamo oko sebe možemo primijetiti da mnogo objekata reflektuje svoje okruženje. Flaša vode, mobilni telefon, CD cover, okvir slike itd, su samo neki od primjera reflektujućih objekata koji se ne mogu naći ni u jednoj 3D sceni. Da bi se 3D svijet učinio realističnijim, objekti treba da pokažu refleksije okruženja. Ova refleksija se postiže metodom mapiranja tekstura koji se zove environment mapping.

Cilj environment mapping-a je da renderuje objekat kao da je on refleksivan, tako da boje na njegovoj površini reflektuju okruženje u kome se nalazi. Drugim riječima, ako gledamo perfektno ispoliran, perfektno refleksivan srebreni objekat u sobi, na njemu se vide refleksije zidova, poda i drugih objekata oko njega. Objekti čije refleksije vidimo zavise od pozicije sa koje gledamo i uglova površina objekta. Naravno, objekti obično nisu potpuno refleksivni, tako da se boja refleksije modulira bojom objekta.



Pravi environment mapping se može postići upotrebom raytracing-a, ali on nije uvijek neophodan. Češće se koriste određeni trikovi za postizanje reflektivnog efekta. Vrlo često se koristi cube environment mapping metod. Sa objekta koji ima refleksivne površine prvo se generiše 6 slika okruženja u svakom pravcu (front, back, up, down, left, right). Ovo možemo zamisliti kao postavljanje kamere u centar objekta i pravljenje fotografija u određenim pravcima. Na osnovu vektora normale svakog

vertexa objekta bira se odgovarajuća slika. Ta slika se onda projicira koristeći planarni mapping kao što je prikazano na slici.

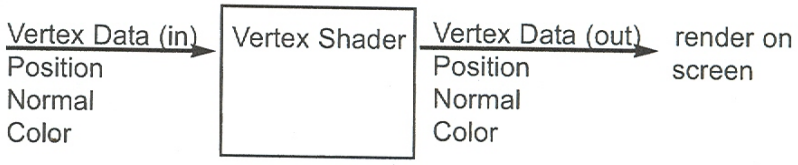


Vertex shaders

Koncept shader-a nije nov u grafičkoj industriji. Pixar ga koristi već godinama u filmovima kao što je Toy Story ili Incredibles. Međutim, vertex shading efekti su do sada bili toliko komputaciono kompleksni da su se mogli proizvesti samo offline koristeći render farme. Sa novom generacijom kompjutera mogu de razviti real-time vertex shaderi da udahnu život i karakter u likove i okruženja, kao što je magla koja uranja u dolinu ili vijuga preko brda, ili realistična facijalna animacija kao što su bore koje se pojave na licu kada se lik osmijehne.

Vertex shader je funkcija za grafičko procesiranja koja se koristi da doda specijalne efekte objektima u 3D okruženju vršenjem matematičkih operacija na podacima o vertexima. Podaci o vertexima su podaci o tačkama duž površine, na nakoj određenoj preciznosti (ne samo vertexi koji definiraju poligone). Svaki podatak o vertexu se definiše pomoću varijabli. Npr. vertex je uvijek definisan svojom pozicijom u 3D okruženju pomoću svojih x, y i z koordinata. Vertexi također mogu biti definisani pomoću vektora normale, boje i karaktristika texture i svjetla. Vertex shaderi ne mijenjaju tip podataka, nego njihove vrijednosti.

Vertex shader se može zamisliti kao magični box. Podaci o vertexima ulaze, a izlaze modifikovani. Šta se dešava unutar box-a zavisi od toga kako je programer razvio shader.



11.3.3.4. Strukture podataka

Programski jezici nude mnoštvo mehanizama za organiziranje struktura podataka kao što su: multidimenzionalne tabele, liste, prstenovi, kvad - i oktstabla. U aplikacijama kompjuterske grafike ove se strukture podataka koriste za smještanje podataka o objektima, izvorima svjetla i kamerama. Parametri kamere su relativno laki za

smještanje, jer oni obuhvataju samo podatke o kontrolisanju njene pozicije i orijentacije, i možda njenu fokusnu dužinu. Podaci o izvorima svjetla mogu se smjestiti u dvodimenzionalnu tabelu, gdje se za svaki tip svjetla (point, direktno i spot) mogu pridružiti podaci o intenzitetu, boji, poziciji, pravcu, faktoru slabljenja i uglu konusa. Kako ovo uključuje relativno malu količinu podataka, dovoljno je deklarisati niz maksimalne dužine pedeset, koji će bit više nego dovoljan za ovu svrhu.

Smještanje objekata je, međutim, drugi problem jer ne postoji unificiran sistem za opisivanje njihove 3D geometrije. Objekti konstruisani od planarnih poligona treba da referenciraju svoje vrhove, ivice, normale površine, koeficijente refleksije, providnost, grubost i koeficijent sjaja. Objekti napravljeni od komadića površine, međutim, će također trebati da imaju podatke o površini, ali umjesto poligona, oni treba da sadrže kontrolne vrhove, knot vektore, parametre napetosti i kosine i moguće vektore nagiba i uvrtanja. Proceduralni objekti kao što su fraktali i sistemi dijelova ne samo da zahtijevaju program za njihovo pravljenje, nego i strukturu podataka koja bi prihvatila različite verzije ovih modela u zavisnosti od kontrolnih parametara.
Objekti konstruisani od zglobnih elemenata kao npr. ruka, nekad trebaju čuvati podatke o osama oko kojih se elementi rotiraju i podatke o stepenu rotacije oko tih osa.

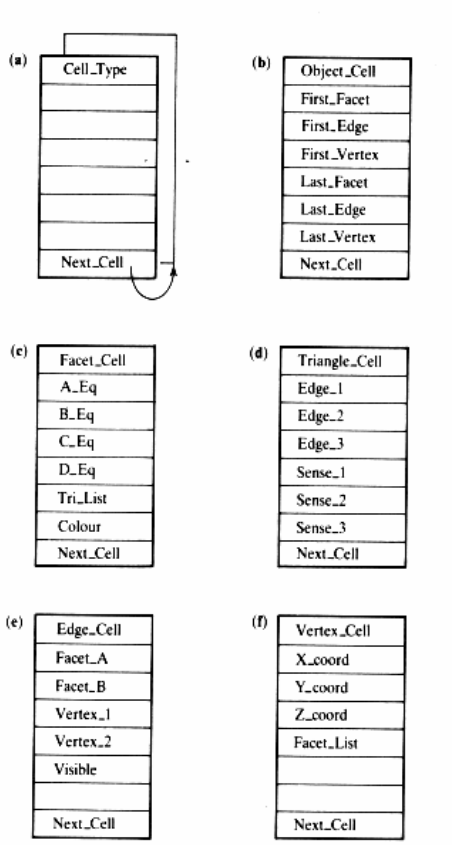
U sistemima konstruktivne čvrste geometrije koriste se implicitne jednakosti za opisivaje geometrijskih primitiva i zahtijevaju potpuno drugačiju strukturu podataka od sistema baziranih na poligonima.

Široki opseg tipova podataka se dalje komplikuje zbog dinamičke prirode objekata, jer se primitive često koriste za gradnju kompleksnijih objekata. Tokom sekvence animacije konstruktor može željeti da ukloni neke elemente iz konstrukcije.

Na žalost, ne postoji univerzalno rješenje ovih problema jer oni često zavise od mogućnosti programskog jezika. Fortran, Pascal, C, C++ i Lisp nude programerima individualna okruženja gdje su mogućnosti kao što su dinamički nizovi i garbage collection ili ostavljeni potpuno u ruke programeru ili podržani sistemskim alatkama. Međutim, treba spomenuti da su hijerarhijske strukture podataka od vitalnog značaja u podržavanju mnogih koncepata koji se koriste u kompjuterskoj grafici.

Već smo vidjeli da se vrh, ivica i podaci o poligonu mogu smjestiti u obliku tabele, koja se lako može mapirati u niz. Međutim, ista struktura podataka neće podržavati objekte kao što su oktahedri, tetrahedri i cilindri bez određenih modifikacija. Glavni problem predstavlja pitanje koliko ivica može poligon imati i neki sistemi za modeliranje uspostavljaju ograničenje za maksimum. Možemo napraviti sve od trouglova, što pojednostavljuje stvar, ali ovo uvodi dodatne ivice koje nemaju značaja u opisu granice objekta i mogu, ali ne moraju, uzrokovati smetnje.

slika 32



Kao ilustraciju, razmotrimo hipotetičku strukturu podataka koja, iako nekompletna, služi kao koristan način da spomenemo neka pitanja vezana za smještanje grafičkih podataka. Za početak pretpostavimo da imamo pristup memorijskoj ćeliji koja može smjestiti osam vrijednosti (slika 32a). Prvi entry smješta kod pod nazivom Cell_Type koji opisuje primjenu ćelije. Npr. jedan tip ćelije se može koristiti za smještanje liste podataka o stranama neke površine. Drugi tip može sadržati informaciju o boji pojedinih strana itd. Ove kodove također mogu koristiti procedure koje provjeravaju kojem se tipu ćelije pristupa. Može se kreirati entry pod nazivom Next_Cell koji se koristi kao pokazivač na sljedeću ćeliju, pa se na taj način kreira dinamička lista beskonačne dužine.

Kada procedura pristupi objektu vrijednost pokazivača identificira Object_Cell (slika 32b). On sadrži šest pokazivača: tri pokazuju na liste strana, ivica i vrhova i sljedeća tri na zadnje ćelije u tim listama. Prva tri pokazivača omogućavaju da objekat bude konstruisan na bazi strana, ivica ili vrhova, a druga tri omogućavaju brz pristup

strukturi podataka kada se smještaju novi elementi. Konstruisanje pomoću strana će se koristiti prilikom uklanjanja stražnjih strana i renderovanja, konstrukcija pomoću ivica će se koristiti kod odrezivanja, a konstrukcija pomoću vrhova kod skaliranja, rotiranja ili translacije objekta. Next_Cell pokazivač može pokazivati na drugu ćeliju koja sadrži parametre kao što je boja, sjaj, hrapavost i providnost.

Facet_Cell (slika 32c) sadrži četiri parametra izvedena iz jednačine ravni koja opisuje stranu; ovo će se koristiti za uklanjanje stražnjih strana, izračunavanje osvjetljenja, mapiranje tekstura i ray tracing. On također sadrži pokazivač na listu trouglova koji formiraju stranu. Colour entry pokazuje na posebnu ćeliju koja sadrži informaciju o boji objekta.

Triangle_Cell (slika 32d) sadrži tri pokazivača na ivice koje formiraju trougao i tri binarne vrijednosti koje indiciraju orijentaciju ivice. One su potrebne ako strane objekta dijele ivice i trouglovi moraju biti formirani u konzistentnom smjeru kazaljke na satu ili obrnuto.

Edge_Cell (slika 32e) pokazuje na dvije strane koje pripadaju ivici i dva vrha koja je formiraju. Parametar Visible kontrološe da li je strana formirana procesom triangulacije i da li ona ikada može biti viđena na silueti objekta.

Na kraju, Vertex_Cell (slika 32f) smješta x, y i z koordinate vrha i pokazivač na listu strana koji identificira strane koje dijele taj vrh. Ovo je potrebno radi izračunavanja normala za proces glatkog sjenčenja.

Postoje mnoge druge osobine u ovoj strukturi podataka, ali ovaj primjer demonstrira količinu detalja koja je potrebna za organiziranje podataka koje koriste programi za kompjuterske animacije.

12. Kompjuterska animacija

Kompjuterska animacija se može posmatrati kao virtuelni televizijski studio u kome se snima film. U tome studiju postoje likovi (glumci), scenografija, svjetlo i kamere. Osnova stvaranja u svakom filmskom izrazu je **scenario**. To je priča filma. U njemu su razrađeni mjesto odvijanja radnje, njen tok, dijalozi i dramaturgija.

Kod kompjuterske animacije scenario je preobražen u **storyboard**. Storyboard je animacija izražena u slikama koje u obliku skica prikazuju ključna dešavanja. Na taj način se opisuje priča, uvode likovi i definiše tajming animacije.

Zatim se modeliraju **objekti** (likovi i scenografija). Principi ovog modeliranja su opisani u poglavlju Načini modeliranja.

Svakom objektu se dodjeljuje **materijal** koji se kreira u posebnom editoru. On sadrži parametre koji se odnose na refleksiju svjetla, transparenciju, glatkoću, sjajnost itd. Implementacija ovih karakteristika se realizira kroz različite **tehnike sjenčenja**.

Primjena materijala na objekat zove se **mapiranje**. Ono se definiše u skladu sa oblikom objekta i postavljanjem mape. Na objekat se može primijeniti i bit mapa što je opisano u poglavlju Proceduralna manipulacija.

Slijedi postavljanje svjetla. Načini osvjetljavanja scene i njihova implementacija su opisani u nastavku.

Nakon toga se postavlja kamera i definišu pokreti aktera animacije.

Proces generisanja slike iz ovog virtuelnog tv studija se zove **rendering**. Ovaj proces radi poseban program koji uzima u obzir koordinate objekata i ostalih elemenata scene u trenutku vremena, njihove materijale sa svojim parametrima, položaj kamere i vidljivost pojedinih objekata pa na osnovu toga generiše sliku u traženoj rezoluciji, formatu i kvaliteti.

Najprije je potrebno identifikovati koje komponente površina (obično poligoni) modela su vidljive iz trenutne tačke pogleda. Ovaj proces uključuje back-face culling, kao i identificiranje površina zakljonjenih površinama ispred njih. Kada su određene vidljive površine, možemo im jednostavno dodijeliti boju i obojiti ih. U većini slučajeva, medjutim, ne želimo da površine budu obojene samo jednom bojom. One trebaju izgledati u skladu sa količinom svjetla koje primaju, kao u realnom svijetu. Da bismo mogli simulirati ovaj efekat moramo definisati osobine materijala površine, ne samo njegovu boju, nego i način kako reaguje na svjetlost. Ovaj proces se zove **sjenčenje**. Potrebno je definisati izvore svjetla da osvijetle scenu, što nam omogućava da je vidimo – proces nazvan osvjetljenje. Kada su sjenčenje i osvjetljenje postavljeni, algoritmi za sjenčenje se koriste za finalno rendanje slike.

Postoje posebne **tehnike sakrivenih površina** na osnovu kojih renderer određuje koji su dijelovi objekata vidljivi u trenutku vremena.

12.1. Načini osvjetljenja

Prema Vince-u, načini osvjetljenja se odnose na različite tipove izvora svjetlosti koji se mogu simulirati pomoću kompjuterske grafike; to je pokušaj da se modeliraju različiti tipovi izvora svjetlosti koji postoje u realnom svijetu. Sunce je najvažniji izvor svjetlosti u našim životima, jer ono ne samo da osvjetljava naš svijet, nego ga i grije.

U našim kućama osvjetljenje uključuje svjetla na plafonu, zidovima, reflektore, svijetleće cijevi i ploče. Kroz prozore dopire prirodno svjetlo izvana. Zbog toga, ako renderer treba biti sposoban da sračuna nivoe svjetla na površinama naših objekata, on će zahtijevati podatke koji se odnose na razne tipove osvjetljenja.

Simuliranje realnosti je često suviše kompleksno, pa je ono obično kompromis između vremena renderovanja i realizma koji želimo postići. Zbog toga možemo očekivati da kompjuterski generisane slike često ne izgledaju sasvim prirodno.

Tačkasti izvori svjetla

Tačkasti izvor svjetla je najjednostavniji za modeliranje jer treba specificirati samo njegovu poziciju u prostoru i njegov intenzitet i boju. Njegova pozicija ne predstavlja problem jer samo treba izabrati pogodno mjesto u prostoru da bi se osvijetlili važni dijelovi scene. Kada jedan izvor svjetla nije dovoljan, mogu se uvesti dodatni izvori svjetla. Međutim, potreban je način kodiranja njegove boje i intenziteta.

U ranijem izlaganju smo vidjeli da se boje čuvaju u obliku njihovih triju komponenti: crvene, zelene i plave. To bi značilo da renderer mora kreirati tri intenziteta boje za svaki piksel - po jedan za svaku primarnu boju. Ovo takođe znači da se izvori svjetla mogu zamisliti kao da se sastoje od tri obojena svjetla na istom mjestu, sa intenzitetima koji su prilagođeni da kreiraju odgovarajuću boju. Generalno, izvor svjetla zahtijeva šest parametara, x, y i z koordinate njegove lokacije i tri broja koji definišu intenzitete primarnih boja.

Usmjereni izvori svjetla

Sunce, koje je udaljeno oko 94 miliona milja od Zemlje je usmjereni izvor svjetla. Njegova velika udaljenost znači da su zrake svjetla koje padaju na mali komadić Zemljine površine virtualno paralelne i istog intenziteta. To znači da se usmjereni izvori svjetla mogu specificirati pomoću boje, intenziteta i pravca koji je kodiran u obliku jediničnog vektora. Ugao između tog vektora i normale na površinu zatim koristi renderer da odredi koja količina svjetlosti pada na određenu površinu.

Spot izvori svjetla

Spot svjetlo simulira reflektor koji kreira kontrolisan snop svjetla u obliku konusa. Parametri koji opisuju ovo svjetlo su pozicija, intenzitet, boja i ugao, ali se dodatni

realizam može postići uvođenjem funkcija koja umanjuje intenzitet svjetla prema granici konusa i kreira mekanu ivicu.

Ostale osobine izvora svjetlosti

Da bi se postigao dodatni realizam u osvjetljavanju scene uvode se još neke osobine izvora svjetlosti kao što su smanjenje intenziteta sa udaljenošću, koje se izračunava pomoću odgovarajuće funkcije, balansiranje nivoa osvjetljenja, koje dolazi do izražaja ako imamo više svjetala koja osvjetljavaju isti objekat, zatim izračunavanje sjena i uvođenje negativnog svjetla. Negativno svjetlo je specifično za kompjuterske izvore svjetla i postiže se davanjem negativne vrijednosti intenzitetu svjetla. Tada svjetlo, umjesto da dodaje osvjetljenje određenoj površini, ono ga oduzima i kreira tamne dijelove scene.

12.2. Načini refleksije

Refleksija se koristi za opisivanje kako se svjetlo reflektuje od površinu. Postoje tri osnovna tipa refleksije svjetlosti i to su **ambijentna, difuzna i spekularna.**

Boja objekta se određuje prema prirodi svjetlosnog zračenja i odnosu upijenog i reflektovanog zračenja. Svjetlo koje dolazi od Sunca je jednaka mješavina svih vidljivh frekvencija, iako zavisi od doba dana i vremenskih uvjeta. Ova izbalansirana mješavina uzrokuje osjećaj da je to svjetlo bijelo, pa kada govorimo o bijelom papiru, mislimo na površinu koja reflektuje zrake svjetlosti svih frekvencija. Kada kažemo da je lopta crvena, to znači da kada je osvijetljena bijelom svjelošću, ona apsorbira većinu frekvencija i reflektira one koje stvaraju osjećaj crvenog u našem mozgu.

Ambijentno svjetlo

Ambijentno svjetlo simulira stepen konstantnog svjetla koje postoji zahvaljujući višestrukim refleksijama koje uzrokuju različiti izvori svjetlosti iz okruženja u kome se nalaze objekti. Ono nema izvora jer predstavlja svjetlo koje dolazi iz svih pravaca. Međutim, da bi se kontrolisao njegov uticaj na različite objekte, površinama je dodijeljen parametar koji određuje koji procenat ambijentnog svjetla se reflektuje nazad prema posmatraču.

U računanju sveukupnog svjetla koje posmatrač vidi, ambijentna komponenta je predstavljena izrazom:

I_aK_a

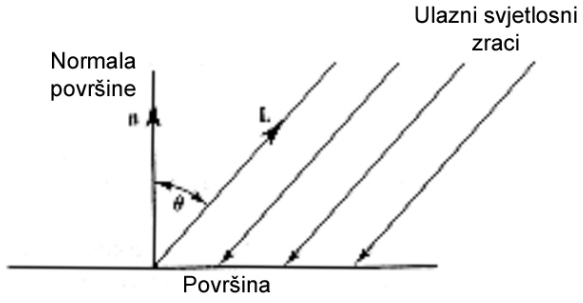
gdje je I_a slučajni ambijentni intenzitet, a K_a parametar koji je pridružen površini da kontroliše proporciju svjetlosti koju ona reflektuje. Ovaj izraz se mora izvesti za crvenu, zelenu i plavu komponentu spektra.

Difuzna refleksija

Difuzno svjetlo kreiraju površine čija neravnost prouzrokuje da se svjetlo jednako reflektuje u svim pravcima. To znači da, kada se posmatraju takve površine, one izgledaju da imaju konstantan intenzitet bez obzira na poziciju posmatrača, ali im se

svjetloća mijenja kako se mijenja njihova pozicija u odnosu na izvor svjetlosti. Da bismo simulirali ovakvo ponašanje u kompjuterskoj grafici nije nam važno gdje se nalazi posmatrač, nego moramo znati ugao između izvora svjetla i površine.

Razmotrimo slučaj kada je usmjereni izvor svjetlosti neposredno iznad površine. Svjetlo će se reflektovati nazad u okruženje u svim pravcima. Međutim, ako je površina tako postavljena da je ugao između nje i zraka svjetlosti 45, onda je efektivna iluminacija reducirana za faktor 0.707. Ako se, u graničnom slučaju, ugao poveća na 90, teoretski površina neće uopće primati bilo kakvo svjetlo.



Slika 1

Slika 1 ilustruje ovu situaciju. Prilikom računanja difuzne komponente renderer mora uvesti kosinusnu funkciju za slabljenje intenziteta svjetlosti (Lambertov zakon).

Difuzna komponenta se onda računa sa

$I_i K_d(Lon)$

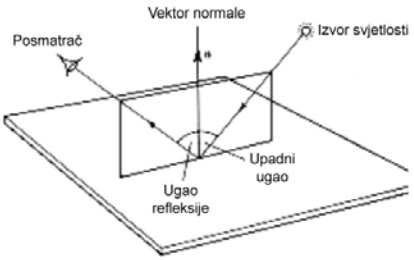
gdje je I_i intenzitet izvora svjetlosti, K_d frakcionalna konstanta pridružena površini za kontrolu koliko se difuznog svjetla reflektuje, L vektor usmjeren prema izvoru svjetlosti i n normala na površinu.

Ovaj izraz se također računa po jednom za svaku od tri komponente boje.

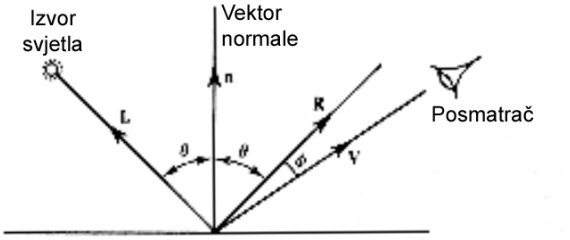
Spekularna refleksija

Glatka ili ispolirana površina proizvodi spekularne refleksije, koje uključuju perfektne refleksije koje vidimo u prozorima ili ogledalu i refleksije izvora svjetlosti koje vidimo na drugim površinama koje nisu tako glatke. Za računanje refleksije okruženja koja se vidi na površinama razvijena je posebna tehnika koja se zove ray tracing.
Posmatrajući ogledalo ili neku drugu ispoliranu površinu možemo zaključiti da je važna pozicija posmatrača. Zapravo, postoje dva zakona od kojih jedan kaže da je

upadni ugao jednak uglu reflektovane zrake i da upadna zraka, reflektovana zraka i normala na površinu u tački refleksije leže u istoj ravni. Slika 3 ilustrira ove zakone.



slika 2

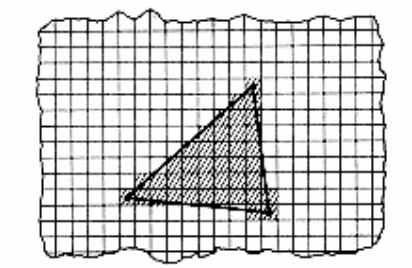


slika.3

Kako su u kompjuterskoj grafici izvori svjetla teoretske tačke, bilo bi nerealno da renderer pravi refleksije svih izvora svjetlosti kao pojedinačne svijetle tačke, posebno jer fizički izvori svjetla imaju konačnu veličinu i refleksije postaju zamrljane zbog različitih stepena grubosti površine. Renderer zapravo treba nastojati da simulira ovakvo ponašanje. Bliska aproksimacija ovog efekta može se kreirati tako što ćemo omogućiti posmatraču da vidi spekularnu refleksiju čak i kada ona nije locirana na optimalnoj poziciji. Zapravo, kako ilustruje slika 4, ako su upadni ugao i ugao refleksije □, i ugao greške □, onda se spekularna komponenta može izračunati pomoću sljedećeg izraza:

$I_i K_s \cos^g \phi$

gdje je I_i intenzitet svjetla, K_s spekularni koeficijent nezavisan od boje, □ ugao greške i g konstanta koja određuje sjajnost površine. Vrijednost g=10 stvara efekat grube plastike, dok vrijednost 150 stvara veoma mali odbljesak tipičan za veoma ispolirane površine.



slika 4

Da bismo izračunali ϕ, θ mora biti poznato, pa se izraz može pisati kao

$I_i K_s(R \circ V)^g$

gdje se kosinus zamjenjuje proizvodom vektora R i V; R je jedinični vektor koji predstavlja reflektovanu zraku, a V jedinični vektor usmjeren prema posmatraču. Nažalost, računanje vektora R nije tako jednostavno, pa se R V računa indirektno pomoću jediničnog vektora svjetla L na sljedeći način:

$V \circ L = \cos 2\theta + \phi =$
$= \cos 2\theta \cos\phi - \sin 2\theta \sin\phi =$
$= \cos\phi \cos^2\theta - \sin^2\theta - 2\sin\theta \cos\theta \sin\phi$
i
$n \circ V = \cos\theta + \phi$
$= \cos\theta \cos\phi - \sin\theta \sin\phi$

Kombiniranjem gornje dvije jednakosti dobijamo

$R \circ V = 2(n \circ L)(n \circ V) - V \circ L$

pa se originalni izraz za spekularnu refleksiju može pisati kao

$I_i K_s(2(n \circ L)(n \circ V) - V \circ L)^g$

Izraz za ukupnu refleksiju

Prosti model refleksije sadrži tri komponente za ambijentnu, difuznu i spekularnu refleksiju, od kojih svaka ima tri kolor uzorka. Ukupno svjetlo koje reflektuje površina može se izraziti kao suma ove tri komponente

$I = I_a K_a + [I_i K_d(L \circ n) + I_i K_s(R \circ V)^g]$

i odnosi se samo na jedan izvor svjetlosti, tako da se izraz u zagradama mora računati za svaki izvor svjetlosti posebno.

Virtualna priroda ovih izvora svjetlosti omogućava određeno kršenje zakona fizike. Npr. izvor svjetlosti može biti postavljen tako da osvjetljava samo određeni skup objekata. Animator također može postaviti izvor svjetlosti unutar objekta. Parametri svjetla se mogu mijenjati u vremenu tokom animacije, ali isprobavanje svih ovih mogućnosti oduzima mnogo vremena zbog renderovanja.

12.3. Tehnike sjenčenja

Pomoću načina refleksije se može izračunati nivo svjetlosti na određenoj površini. Razvojem ovih ideja može se kreirati sjenčeni izgled 3D objekata, s tim što moramo znati da realizam sjenčenja zavisi od načina osvjetljenja objekta.

Iako postoje različiti načini smještanja osjenčene slike, pretpostavićemo da postoji 24-bitni frame store gdje se svaka primarna boja zapisuje sa preciznošću od 8 bita.

Kako programi za sjenčenje smještaju svoje intenzitete boja u frame store, oni u sliku uvode dva tipa greške. Prvi je uzrokovan čuvanjem intenziteta boje u obliku integera (greška kvantizacije) i drugi, važniji, je uzrokovan time što je slika konstruisana iz mnoštva tačaka.

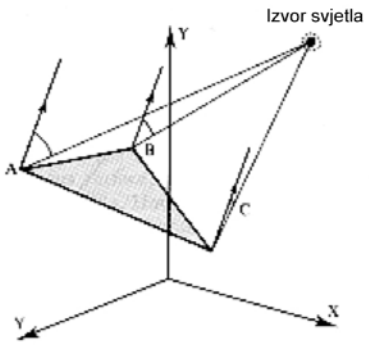
Ovaj problem se može uočiti na slici 4 koja pokazuje ivicu poligona zajedno sa tačkama njegovih piksela. Vidimo da se na mjestima gdje tačka samo dodiruje poligon taj piksel prikazuje u istoj boji kao i piksel koji je potpuno pokriven poligonom. Zato je ivica poligona neravna i pojavljuje se greška u prikazu tog poligona. Ova pojava se naziva **"stepenasti aliasing".**

Zbog toga su preduzeta istraživanja za pronalaženje efikasnih **anti-aliasing procedura** koje, razumljivo, usporavaju vrijeme renderovanja.

Algoritmi za anti-aliasing postavljaju ivične pixele na različite intenzitete, tako da da se oni stapaju u glatku sliku. Ranije objašnjeni midpoint-line algoritam sada postavlja pixele različitog intenziteta u zavisnosti od njihove udaljenosti od idealne linije, umjesto prostog aktiviranja pojedinih pixela.

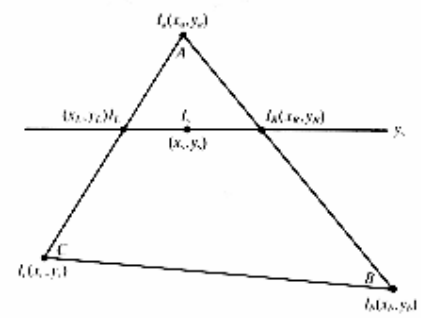
Gouraud sjenčenje

Henri Gouraud je 1971. prvi značajnije doprinio algoritmima za sjenčenje uvođenjem tehnike linearne interpolacije za povezivanje boja preko planarne površine. On je predložio da se oko prevari tako što se boja površine dobija iz nekoliko uzoraka. Ovaj algoritam je lako razumjeti ako se pogleda slika 5 koja prikazuje 3D trougao osvijetljen sa tačkastim izvorom svjetlosti. Ovo znači da svaki od vrhova A, B i C ima različit intenzitet boje jer zrake koje proizvodi izvor svjetlosti zatvaraju različite uglove sa normalama na površinu. U slučaju da je izvor svjetlosti beskonačno udaljen, zrake bi bile paralelne, i uglovi identični.



slika 5

Ako su intenziteti svjetlosti na vrhovima trougla Ia, Ib i Ic, onda možemo pretpostaviti da se intenzitet svjelosti duž ivice koja povezuje vrhove A i B mijenja linearno od I_a do I_b. To znači da se u prostoru slike može napraviti interpolacija pomoću x i y koordinata vrhova (slika 6).



slika 6

Npr, ako tri vrha A, B i C u prostoru slike imaju koordinate

$A(x_a,y_a)$, $B(x_b,y_b)$, $C(x_c,y_c)$

onda je intenzitet I_L u tački (x_L,y_s) dat linearnom interpolacijom pomoću y_a, y_c i y_s

$$I_L = \frac{I_a(y_s - y_c) + I_c(y_a - y_c)}{(y_a - y_c)}$$

Slično, za desni ugao intenzitet I_R u tački (x_R, y_s) je dat linearnom interpolacijom koristeći y_a, y_b i y_s

$$I_R = \frac{I_a(y_s - y_b) + I_b(y_a - y_s)}{(y_a - y_b)}$$

Sada imamo intenzitet I_L lijeve strane trougla i I_R na desnoj strani trougla, pa se intenzitet I_s može dobiti linearnom interpolacijom pomoću x_R, x_L i x_s

$$I_s = \frac{I_L(x_R - x_s) + I_R(x_s - x_L)}{(x_R - x_L)}$$

Sada imamo intenzitet svjetla za poziciju $I_s(x_s,y_s)$ na nekom rasteru, pa procesiranjem trougla na bazi scan konverzije možemo ga osjenčiti na Gouraud način.

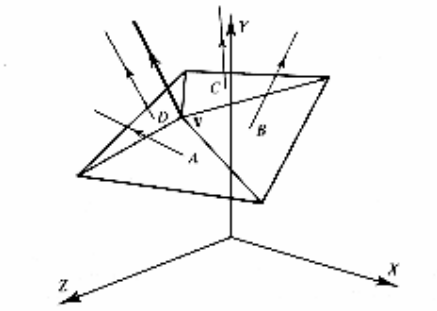
Već smo ustanovili da se slike u boji mogu kreirati podjelom slike na tri primarne aditivne komponente boje, crvenu, zelenu i plavu, koje se mogu smjestiti individualno u 24-bitni frame store. Da bismo proizveli sliku u boji koristeći Gouraud sjenčenje, algoritam mora snabdjeti interpolirane intenzitete piksela plavom, zelenom i crvenom komponentom svjetla. Ako se poligoni koji čine objekat osjenče ovom tehnikom, finalna slika će izgledati kao da je konstruisana od facetirane (izbrušene) površine, pa se ovo sjenčenje često naziva "facetirano" ili "ravno sjenčenje".

Phong sjenčenje

Dok Gouraud sjenčenje pretpostavlja da je u pitanju površina koja ima difuznu refleksiju, Phong sjenčenje simulira ispoliranu površinu dodavanjem spekularnih efekata difuznoj komponenti.

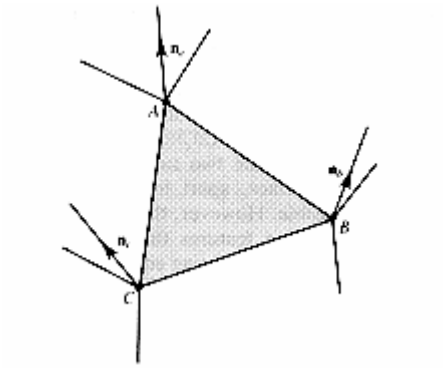
Da bi izračunali spekulrnu komponentu, Phong algoritmi koriste geometrijske relacije između posmatrača, objekta i izvora svjetlosti. Spekularni odsjaj se može primijetiti samo pod odgovarajućim uglom. Za spekularne kalkulacije najvažnija je normala površine na slučajnu zraku svjetlosti kao što je prikazano na slici 3, i, kao što pokazuje slika 4, posmatrač još uvijek može vidjeti smanjeni odbljesak, čak i kada ugao posmatranja nije tačno jednak uglu refleksije. Kod ravno sjenčenih planarnih površina normala na površinu je konstantna, dok Phong sjenčenje razvija dodatnu spekularnu komponentu koja se mora dodati ambijentnoj i difuznoj komponenti.

Da bismo izračunali ovu spekularnu komponentu, moramo izvesti vektor normale za svaki piksel koji se renderuje, što uključuje znatne geometrijske izračune, ali Phong pristup nudi kompromis interpolirajući normale u prostoru slike koristeći srednje normale na vrhovima poligona.



slika 7

Kao primjer razmotrimo poligon na slici 7 gdje su normale n_a, n_b i n_c srednje normale izračunate iz normala na površinu koje dijele zajednički vrh. Sada možemo linearno interpolirati n_a i n_c da dobijemo n_L koristeći vrijednosti y_a, y_c i y_s (slika 8)



slika 8

$$n_L = \frac{n_a(y_s - y_c) + n_c(y_a - y_s)}{(y_a - y_c)}$$

Sličan izraz može biti dat za

$$n_R = \frac{n_a(y_s - y_b) + n_b(y_a - y_s)}{(y_a - y_b)}$$

Sada, kao kod Gouraud sjenčenja, možemo linearno interpolirati između normala n_L i n_R da dobijemo n_s:

$$n_s = \frac{n_L(x_R - x_s) + n_R(x_s - x_L)}{(x_R - x_L)}$$

Sada imamo pseudonormalu površine n_s koja se može zamijeniti u spekularnu komponentu refleksije

$I_i K_s (2(n_s \text{ o } L)(n_s \text{ o } V) - (V \text{ o } L))^g$

Iako Phong sjenčenje proizvodi vrlo prihvatljive slike, one još uvijek nisu realistične, jer ni Gouraud ni Phong sjenčenje ne uzimaju u obzir optičke efekte kao što su sjene, višestruke refleksije, povezivanje boja i površine ogledala. Ovi efekti se postižu drugim načinima.

12.4. Tehnike sakrivenih površina

Algoritmi sakrivenih površina omogućavaju da prilikom renderovanja scene objekti koji su bliže kameri sakrivaju one udaljenije. U realnom svijetu ovaj fenomen je prirodan jer naše oko prima svjetlosne talase koji su u interakciji sa transparentnim i neprozirnim objektima. U našem kompjuterskom okruženju, objekti su opisani pomoću mnoštva tehnika za modeliranje i u osnovi se čuvaju kao skupovi brojeva. Prostorne relacije se također čuvaju numerički u obliku koordinata i zato maskiranje jednog objekta drugim treba izvesti iz tih podataka i saopćiti procesu renderovanja.

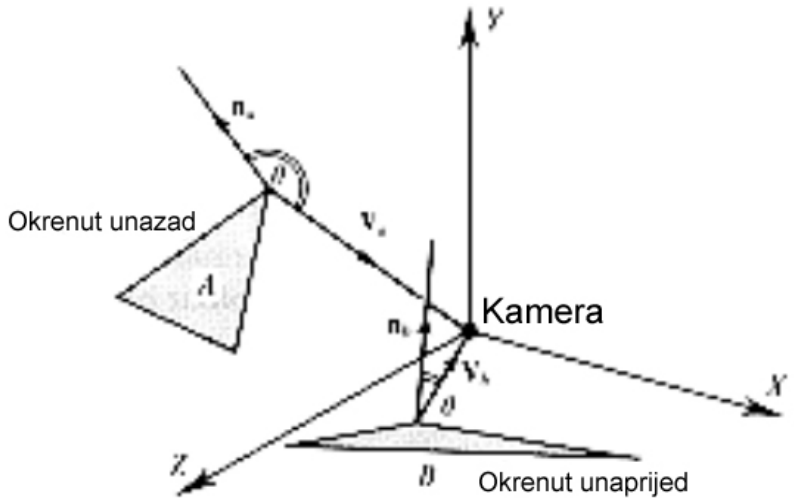
Mnogi objekti u realnom svijetu imaju čvrstu strukturu koja onemogućava da prodiru jedan u drugi; neki, kao što su tečnosti i gasovi, dozvoljavaju objektima da prolaze kroz njih; neki su elastični i mogu prilagođavati svoj oblik u kontaktu s drugim objektima. Ovi efekti se ne mogu inkorporirati u naše jednostavne numeričke modele; oni moraju biti dizajnirani kao posebne procedure koje prilagodile naš animirani svijet zakonima fizike.

Numerička priroda kompjuterskih modela također prouzrokuje druge probleme: zamislimo npr. slučaj dva ili više objekata koji postoje u istoj zapremini prostora. U realnosti, jedan objekat normalno zamjenjuje drugi kada je pomjeren u isti prostor, iako tečnosti i gasovi, zbog svoje molekularne strukture, mogu dijeliti isti prostor, dok se u kompjuteru objekat translira na drugu poziciju u WCS modificiranjem vrijednosti njegovih koordinata. Ako ova modifikacija prouzrokuje njegovo pomjeranje u drugi objekat, to će se riješiti pomoću algoritma za ukljanjanje sakrivenih površina. Na neke tehnike on se ne može primijeniti, dok druge omogućavaju kreiranje efekata koji su u realnom svijetu fizički nemogući što predstavlja još jedan izazov za animatora.

12.4.1. Uklanjanje stražnjih strana – back face culling

Jednostavan način za uklanjanje sakrivenih površina je ignoriranje onih površina koje "gledaju" suprotno od kamere. Npr. kod konveksnih objekata kao što je kocka, sfera, cilindar, tetraedar i sl. uklanjanje stražnjih strana kreira prirodan čvrst izgled. Međutim, ako je jedan od ovih objekata smješten

ispred drugog, oni će imati izgled žičanog modela. Bez obzira, uklanjanjem stražnjih strana scena se znatno pojednostavljuje i skraćuje proces renderovanja.



slika 9

Uslov stražnjih strana se može testirati za bilo koji planarni poligon računanjem proizvoda njegove normale površine sa vektorom pogleda kao što je prikazano na slici 9. Poligon B "gleda" unaprijed i zato je vidljiv jer vektori n_b i V_b zatvaraju ugao manji od 90; međutim, poligon A "gleda" unazad jer je ugao između vektora n_a i V_a veći od 90.

Većina algoritama za uklanjanje sakrivenih površina upotrebljava ovaj oblik pretprocesiranja poligona.

12.4.2. Slikarov algoritam

Ako se skup 3D poligona čuva u uređenoj sekvenci i zatim jedan po jedan renderuje u frame store, ne može se očekivati da finalna slika prikazuje pravi izgled 3D scene. Vjerovatno će se dogoditi da se poligoni koji su bliže kameri rendaju u frame store da bi kasnije bili parcijalno maskirani udaljenijim poligonima. Ovaj neželjeni efekat se može izbjeći ako se poligoni tako sortiraju da se prvo renderuje najudaljeniji, a na kroju najbliži. Ovo podsjeća na slikanje pejzaža kada slikar prvo naslika nebo, zatim planine, onda polja i drveće i na kraju ukrašava scenu sa objektima u prvom planu.

Sortiranje poligona u ovu dubinsku sekvencu nije uvijek tako jednostavno, posebno kada npr. imamo situaciju da duguljasti poligon prolazi kroz centar kružnog prstena. Jedan kraj poligona može biti ispred prstena, a drugi iza, pa proces sortiranja ne može razriješiti pitanje prioriteta. Ovaj problem se rješava razbijanjem takvih poligona u manje elemente. Slikarov algoritam nije baš popularan jer ima druge nedostatke kao što je nemogućnost podržavanja unutarnjih objekata i anti-aliasinga.

12.4.3. Scan-line algoritam

Scan-line algoritam se u osnovi bavi računanjem piksela sadržanih u jednoj liniji slike u frame store-u i primjenom algoritma na svaki raster, što može biti nekoliko stotina puta da bi se izrenderovala scena. Generalno, proces počinje od gornje linije i ide liniju po liniju do dna. Neki sistemi mogu izvršiti ovaj ciklus za nekoliko milisekundi, što omogućava animaciju u realnom vremenu.

Da bi izračunao intenzitete piksela u svakoj liniji, algoritam inicijalno sortira vidljive objekte u vertikalnu sekvencu da bi mogao voditi računa o tome koji objekat presijeca određenu scan-liniju. On onda analizira površine koje presijecaju tekuću liniju i razrješava problem dubine razmatrajući male dijelove poligona relevantne za taj raster. Na ovaj način on može podržati unutarnje objekte i implementirati anti-aliasing strategije.

12.4.4.Z-bafer

Z-bafer algoritam omogućava veoma elegantan mehanizam uklanjanja sakrivenih površina, ali nažalost on ima druge nedostatke kao što su nemogućnost podržavanja transparentnih površina i anti-aliasinga. Bez obzira, on je veoma koristan zbog svoje jednostavnosti i često je implementiran u hardveru.

Z bafer je zapravo memorijski element za smještanje informacije o dubini za svaki piksel u frame store-u i inicijaliziran je na neki veoma veliki broj. Kada se renderuje prvi poligon, dubina poligona se računa za svaki piksel koji je zahvaćen i ako su te vrijednosti manje nego inicijalni broj, onda se njima prepisuje z-bafer. Frame store se puni odgovarajućim intenzitetima boje. Kako se rendaju sljedeći poligoni u uređenoj sekvenci, njihove dubine se uspoređuju sa tekućim i ako su bliže od njih, one također prepisuju z-bafer i frame store. Ako su dubine piksela veće nego tekuće vrijednosti, to će značiti da je poligon udaljeniji nego prethodna površina i zbog toga nevidljiv; zbog

toga ti pikseli neće biti ažurirani i površine će ostati sakrivene. Nastavljajući na ovaj način, z-bafer algoritam može prihvatiti poligone u bilo kojoj sekvenci i podržati presjek objekata na nivou piksela.

Nažalost, kako ne postoji mehanizam za predstavljanje providnosti površine, ovaj algoritam ne može podržati transparenciju. Slično, on ne može uvesti anti-aliasing u scenu jer ne postoji način da se predstavi lista onih poligona koji dodiruju pojedine piksele.

12.4.5. A-bafer

A-bafer algoritam (Carpenter, 1984) je razvio Lucasfilm Ltd i nazvao ga po njegovoj sposbnosti da renderuje sliku sa anti-aliasingom koristeći tehniku akumulacije srednjih područja. Algoritam se bazira na konceptu bit maski pridruženih pikselima koje su uspostavljene za tekuću liniju rastera; one bilježe koliko poligon pokriva neku tačku rastera i kontrolišu finalnu boju piksela koji sadrži jedan ili više poligona koji se parcijalno maskiraju.

Algoritam radi tako što prvo identificira one površine koje pokrivaju ili dodiruju tekući raster i uzimanjem jedne od njih u bilo kojoj sekvenci, vodi računa da li ona potpuno ili parcijalno maskira piksele. [to površina više pokriva piksel, to je veći njen uticaj na finalnu boju tog piksela. Smještanjem u bafer dubine površine za taj piksel, bilo koja površina koja se kasnije renderuje može biti upoređena sa tom dubinom i ako je dalja, može biti potpuno ili parcijalno maskirana bližim površinama. Ovo je jednostavan mehanizam za uklanjanje sakrivenih površina.

12.4.6. ZZ-bafer

ZZ-bafer (Salesin, 1989) je dobio ime od z-bafera koji uzima z-dubinska mjerenja za rješavanje uklanjanja sakrivenih površina. On također dijeli scenu u matricu četverougaonih ćelija, gdje svaka ćelija ima sposbnost da čuva listu poligona koji dodiruju piksel u njenom domenu i koliko ga dodiruju. Lista također bilježi i providnost poligona i zbog toga može podržavati mnogo nivoa transparentnih površina. Kako su poligoni uneseni u uređenoj sekvenci, razvijaju se liste za svaku ćeliju i one mogu rasti i smanjivati se. Npr. recimo da određena ćelija čuva listu transparentnih površina i da je sljedeći renderovani poligon potpuno prekriva. To znači da tekuća lista više nije ni od kakve koristi i ona se može ponovo uspostaviti za ovaj zadnji poligon. Liste mogu eventualno biti renderovane sa procesiranjem svakog poligona i, zahvaljujući prostornoj nezavisnosti ćelija, mogu biti date multiprocesorskom okruženju.

12.4.7. Jednostruki scan-line dubinski bafer

Ovo je hibridni algoritam baziran na scan-line pristupu renderovanju gdje se finalna slika renderuje jedan po jedan raster. Dok z-bafer algoritam zahtijeva dubinski bafer za svaki piksel na displeju, ovaj pristup zahtijeva dubinski bafer samo za tekući raster. Ovo znači dosta spašavanja u memoriju i dozvoljava dubinskom baferu da ima veću rezoluciju. Npr. svaki piksel može biti podijeljen u 16 podpiksela i čuvati dinamička lista za poligon koji ih zahvata. Ova hibridna strategija omogućava podržavanje transparencije i anti-aliasinga.

Postoje još dvije tehnike osvjetljenja, a to su ray tracing i radiosity. Ray tracing renderuje sliku piksel po piksel. Za svaki piksel zamišljena zraka svjetla koja pada na tu tačku slijedi se nazad u prostor objekata da bi se ustanovilo njeno polazište. Ovo omogućava tačno opisivanje spekularnog fenomena, ali se samo pretpostavlja ponašanje kod višestrukih difuznih refleksija.

Sjene dobijene pomoću ray-tracinga su mnogo oštrije i kvalitetnije od običnih sjena, ali je zato proces renderinga duži.

Za računanje ray-tracing-a koristi se struktura podataka koja se zove **quadtree**.

Quadtree predstavlja scenu sa tačke gledišta svjetla. Korijenski nod quadtree strukture sadrži listu svih objekata koje svjetlo obuhvata. Ako je previše objekata vidljivo, nod generiše četiri druga noda, od kojih svaki predstavlja četvrtinu pogleda i sadrži listu objekata u tom dijelu pogleda. Ovaj proces se dalje nastavlja, dok svaki nod ne bude imao samo mali broj objekata ili dok se ne dostigne granica dubine quadtree-ja. Ova granica se može postaviti za svako svjetlo.

Svaka zraka svjetla treba da prođe test presjeka sa objektima u samo jednom nodu listu quadtree-ja. Ovo ubrzava proces ray-tracing-a. Generalno, povećanje maksimalne dubine quadtree-ja ubrzava ray-tracing na račun zauzetosti memorije.

Radiosity je globalna tehnika osvjetljenja koja računa zračenje za male komadiće površine. Ono opisuje uslove difuzne refleksije koja postoji u okruženju. Ono je neovisno od pogleda i omogućava veoma brzo

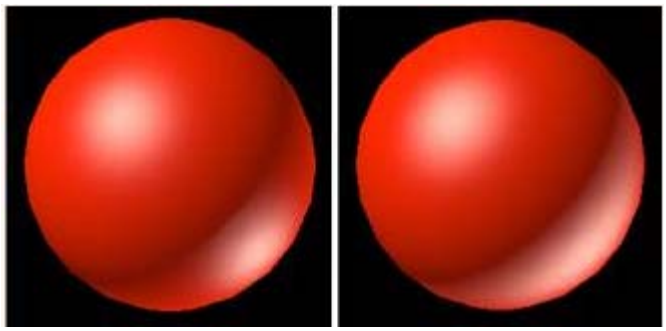
renderovanje scene (ponekad u realnom vremenu) linearnom interpolacijom intenziteta svjetla na komadićima.

12.5. Vrste svjetla u 3DMax-u 4.2.

- Ambijentno svjetlo
- Omni svjetlo – tačkasto svjetlo koje odgovara 6 target spot svjetala postavljenoh ukrug i okrenutih prema vani
- Target spot svjetlo – usmjerno svjetlo u obliku konusa
- Free spot svjetlo – usmjereno svjetlo u obliku konusa bez target-a
- Target direct – direktno svjetlo koje osvjetljava scenu na način kao što Sunce osvjetljava Zemljinu površinu. Koristi se da simulita Sunčevu svjetlost. Direktna svjetla imaju oblik kružne ili četverougaone prizme
- Free direct – svjetlo koje simulira Sunčev sistem i ima parametre za određivanje položaja svjetla u vremenu i prostoru

12.6. Vrste sjenčenja u 3DMAx- u 4.2.

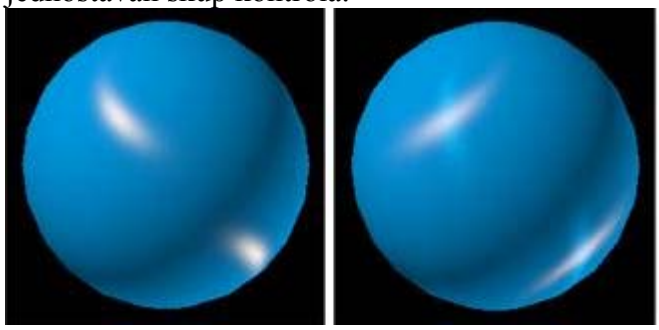
Za standardne materijale na raspolaganju je 7 načina sjenčenja:



Blinn sjenčenje kreira glatke površine sa nešto sjaja i predstavlja univerzalni način sjenčenja

Phong sjenčenje ima iste osobine kao i blinn, ali može bolje podržati sjajnost.

Metal sjenčenje proizvodi metalik izgled objekta, **Oren-Nayar-Blinn** kreira dobre mat površine kao što su platno ili terakot, slično kao Blinn, dok **Strauss** sjenčenje može koristiti i za metalik i nemetalik površine i ima jednostavan skup kontrola.



Anisotropic sjenčenje se koristi za površine sa neizotrpočnim odsjajima (koji nisu kružni). Dobro je za modeliranje kose, stakla ili metala.

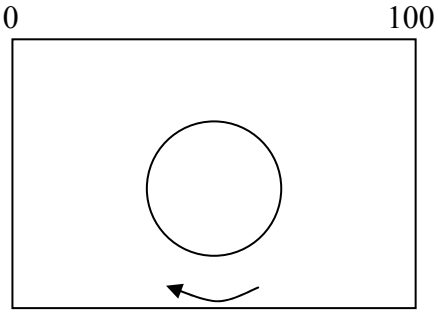
Multi-Layer sjenčenje proizvodi kompleksnije odsjaje jer sadrži dva anizotrpočna lejera odsjaja.

12.5. Primjer animacije po osnovnim fazama

Pregled svih faza u kreiranju jednostavne kompjuterske animacije moguć je kroz primjer kreiranja globusa koji se okreće.

Faza1. Storyboard

Kako se radi o jednostavnoj animaciji storyboard bi se sastojao samo iz jedne slike koja sadrži kuglu i podataka o trajanju njenog okretanja. Neka se kugla okreće 4 sekunde i neka se za to vrijeme okrene jedamput.



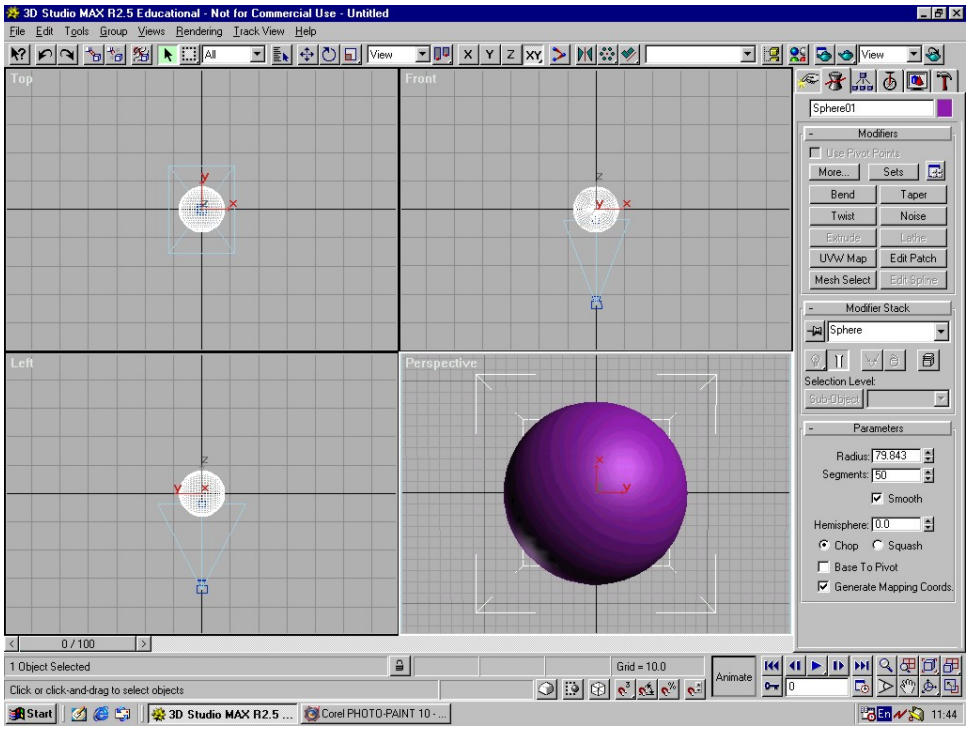
Zemljina kugla se okreće

Faza2. Kreiranje objekata

U ovoj fazi kreiramo kuglu. Prilikom kreiranja kugle važno je definisati dovoljan broj segmenata, da bi kugla izgledala dovljno glatko. U programu 3D Max iskustvo je pokazalo da je optimalno staviti oko 50 segmenata. Također treba uključiti opciju za generisanje maping koordinata, kako bi se materijal korektno postavio na objekat.

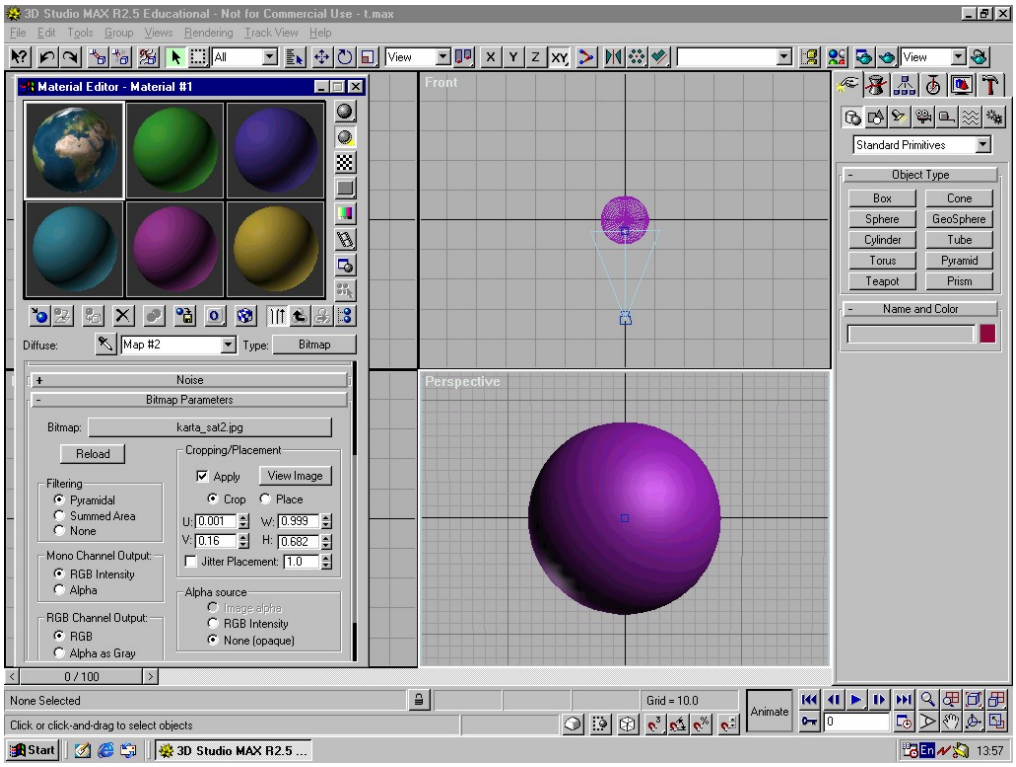
Faza3 Kreiranje kamere

Da bi se imao bolji pregled cjelokupne scene, dobro je odmah kreirati kameru. U našem primjeru kreiraćemo Target kameru sa objektivom 35mm.



Faza4. Kreiranje materijala

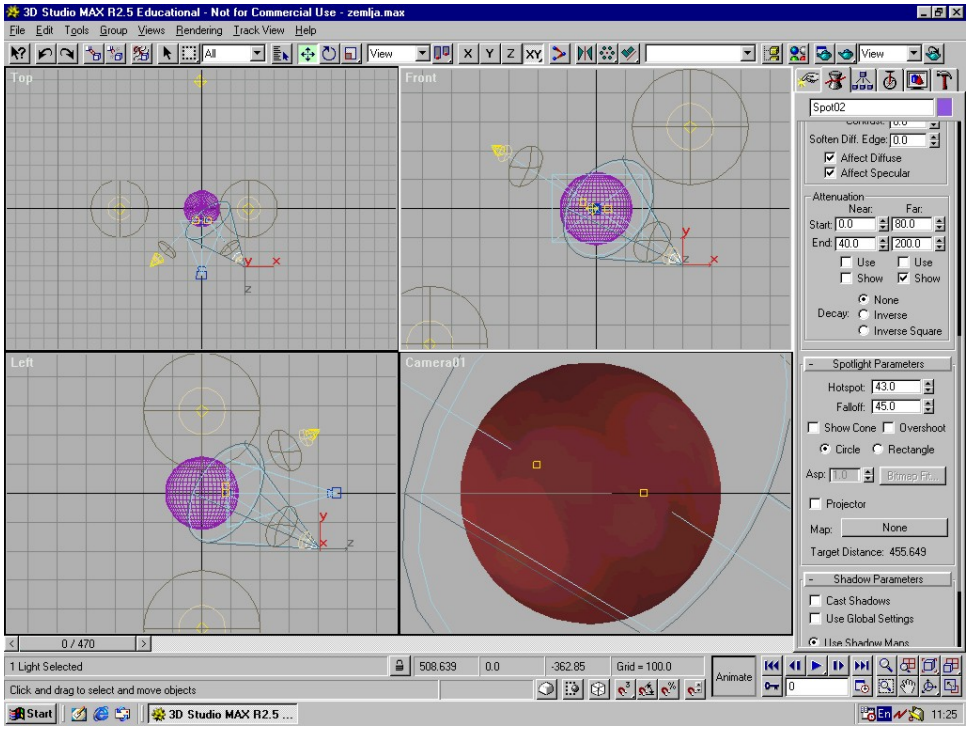
Pretpostavljamo da već imamo kreiranu bit mapu sa kartom svijeta. Materijali se prave u Materijal editoru. Naš materijal će kao osnovu imati bit mapu sa odgovarajućim parametrima sjajnosti i transparencije koja je potrebna u našoj sceni.



Nakon renderovanja naša scena izgleda ovako

Faza5: Postavka svjetla

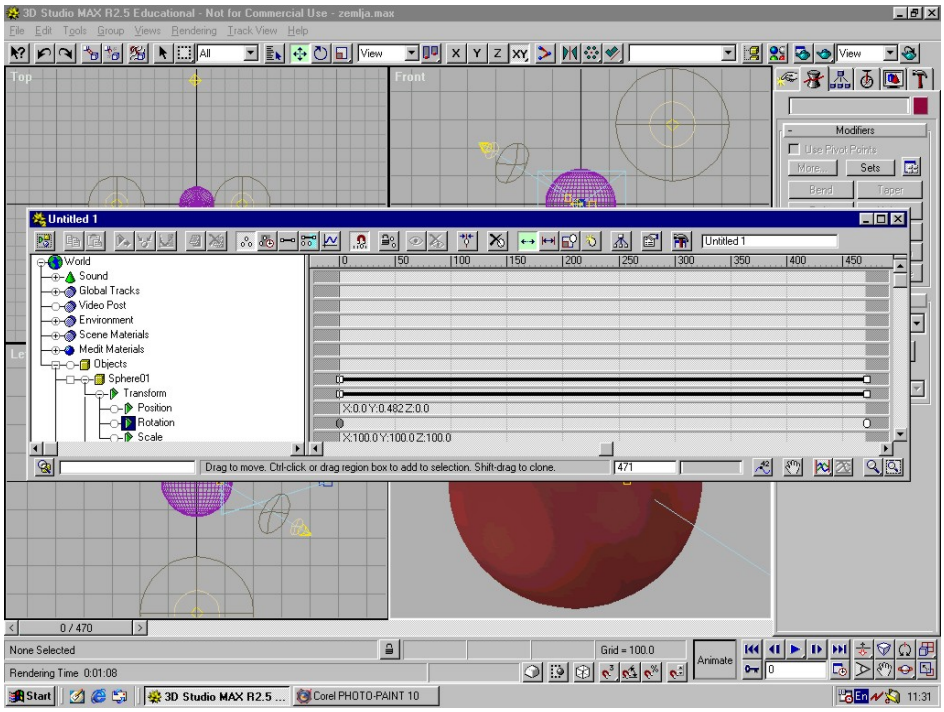
Ova scena je osvijetljena sa 3D Max default svjetlom. Zato treba postaviti svjetlo. Postavka svjetla nije trivijalna i u tv i filmskim produkcijama postoje majstori svjetla koji se time bave. Mi ćemo se poslužiti isprobanim metodom sa jednim kontra svjetlom, dva bočna spot svjetla i dva omni svjetla.



Sada naša kugla izgleda ovako:

Faza6: Animacija

Nakon što smo postavili svjetlo možemo pokrenuti kuglu. Prvo se definiše ukupno trajanje animacije (u našem slučaju 100 frejmova – 4 sekunde). Zatim se odredi položaj kugle (rotacija) u nultom i 100-tom frejmu. Ovi frejmovi se zovu ključni frejmovi za našu animaciju. **Ključni frejm (key frame)** je onaj frejm u kome se definiše neko dešavanje na animiranom objektu (pokret, rotacija, promjena oblika, promjena nekog parametra objekta). Sve međufaze između ključnih frejmova program sam izračunava.



Ovdje dolazi do izražaja prednost računara u odnosu na klasični način animacije kod crtanih filmova, gdje se svaki frejm posebno crtao.

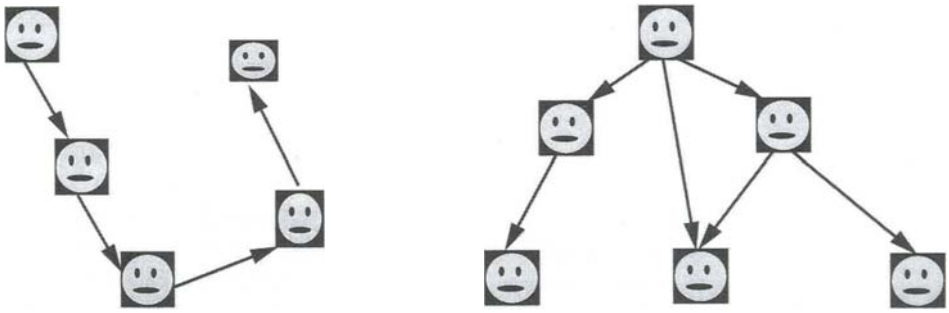
Na kraju se izrenderuje finalna animacija u neki od formata zapisa slike, koji zavisi od načina snimanja na magnetoskop ili prikaza animacije. Ako je u pitanju animacija za web stranicu, obično se koristi neki format zapisa pokretne slike (avi, MPEG), a ako se radi o tv produkciji format zavisi od a/d konvertera pomoću kojeg se snima finalna slika na magnetoskop.

13. OpenGL grafičko programiranje

Kada su u pitanju kompjuterske igre, potrebna nam je real-time grafika, tj. program mora biti interaktivan i imati trenutan odgovor na ulaz. Na pritisak dugmenta, grafika se mijenja. Na povlačenje miša lik se pomiče. U ovom poglavlju naučićemo neke tehnike za interakciju sa korisnikom u realnom vremenu.
Za početak ćemo istražiti šta je zapravo kompjuterska igra i prodiskutovati proces njenog dizajniranja. Na osnovu tih diskusija može se implementirati prava igra.

Šta je kompjuterska igra?

Ako želimo dizajnirati igru, moramo definisati šta znači riječ igra. Igra se može smatrati reprezentacijom fantazije koju igrač treba da iskusi. Priča je način za reprezentaciju fantazije kroz uzrok-efekt relacije sugerirane kroz sekvencu činjenica i detalja. Igre pokušavaju da predstave fantaziju pomoću drveta sekvenci. To omogućava igraču da kreira njegovu vlastitu priču birajući između grana drveta.



priča igra

slika 1. odnos priče i igre

Dizajner igre radi na proizvodnji igre da privuče igrače. Zašto ljudi igraju igre? Šta ih motivira? Šta čini igre zabavnim? Odgovori na ova pitanja su krucijalni za dobar dizajn igre. Iako se igre igraju kroz različite medije, mi smo prvenstveno zainteresirani za igre na kompjuteru.
Postoje mnoge motivacije za ljude da igraju igre: učenje, fantazija. Ovi faktori motiviraju ljude, a drugi faktori utiču na izbor pojedine igre. Zato je pri dizajniranju igre važno razumjeti motivaciju ciljne publike. Različite igre motiviraju različite vrste publike. Intelektualna publika može se privući

igrom koja predstavlja vježbu za intelekt, pa će oni preferirati one igre koje nude mentalni izazov. Ako je ciljna publika motivirana akcijom i bitkom, onda to treba biti integrirano u igru.
Dobra grafika, boja, animacija i zvuk su također faktori koji su bitni za igrače. Oni podržavaju fantaziju i pružaju dokaz o realističnosti igre.

Dizajn igre

Postoje standardni koraci u profesionalnom dizajnu igara. Ovi koraci prisiljavaju dizajnera da razmišlja o važnim stvarima u igri prije početka njene implementacije. Ovo je veoma važan proces, jer ako ciljevi igre nisu jasno definisani, ona vjerovatno neće zadovoljiti ciljnu publiku.

1. Publika

Prvi korak u procesu dizajna igre je identificiranje ciljne publike. U našem primjeru odabraćemo djecu kao ciljnu publiku. Npr. dječake od 8 do 12 godina.
Kada je publika odabrana, potrebno je istražiti šta ta grupa voli, a šta ne voli. Recimo da smo utvrdili da dječaci vole da ciljaju i pucaju u objekte visokog ranga.
Neka naša igra bude pucačka. Cilj igre će biti da omogućimo korisniku da nacilja i puca na objekte. Igra će također poboljšati motoriku igrača i koordinaciju ruka-oko.

2. Okruženje

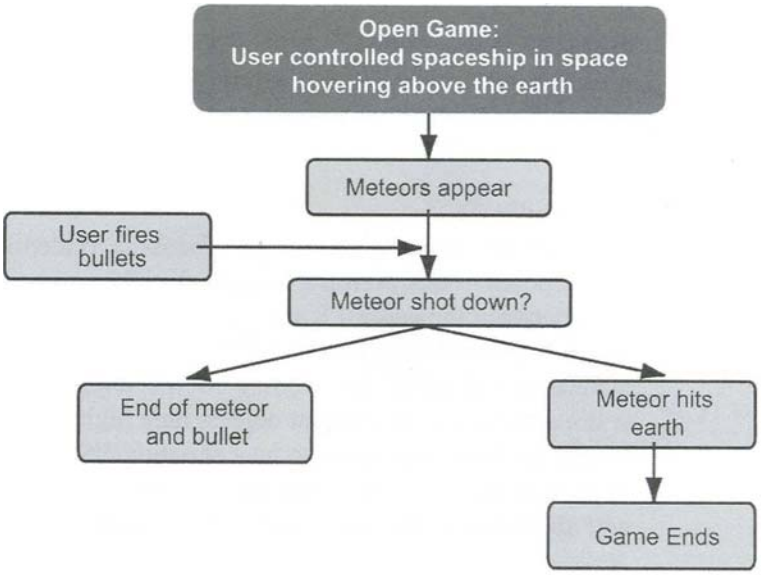
Sljedeći korak je da definišemo okruženje u kome se igra dešava. Već smo rekli da će igra biti pucačka, ali koji likovi igraju igru i gdje su oni smješteni? Definišimo da se igra dešava u svemiru. Potreban nam je lik koji će biti kontrolisan od strane igrača. Ovaj lik će omogućiti igraču da puca. Također nam treba lik na kojeg ćemo pucati. Zatim želimo da se ovi likovi uklope u izabrano okruženje.

Imajući sve ovo na umu, definišemo svemirski brod kontrolisan od strane korisnika kao pucača i meteore koji se slučajno pojavljuju kao mete. Da bi igra imala dobru priču, treba nam i treći lik koji je objekat koji korisnik nastoji da zaštiti. Silueta planete Zemlje može biti dobar izbor, jer će svako željeti da je zaštiti od meteora.

3. Interaktivni scenario

Scenario, ako je odgovarajući, sadrži dijaloge i liniju priče implementirane u igri.
Definišimo jednostavni scenario za našu igru. Igra počinje sa scenom svemira simuliranom pomću zvijezda i neba. Dio planete Zemlje je na dnu ekrana. Svemirski brod lebdi iznad površine Zemlje. Igrač može pomjerati brod lijevo i desno i pucati prema gore. Metori, generisani od strane igre, slučajno se pojavljuju na nebu. Igrač treba da pogodi meteore. Ako metak pogodi meteor, on je uništen. Ako meteor pogodi Zemlju, igra se završava sa eksplozijom.

Interaktivna logika igre može biti prikazana na sljedeći način:



slika 2. logika igre

4. Storyboard

Storyboard pomaže u procesu dizajna ilustriranjem igre u pokretu. Obično su storyboardi brze skice ključnih momenata tokom igre. Bilo kakve nepravilnosti u igri mogu se pokazati u ovom koraku.
Storyboard ilustrira samo glavne momente u priči. Za našu igru, mogu se ilustrirati sljedeći ključni momenti:

- uvodna scena: lebdenje svemirskog broda
- slučajna pojava meteora na sceni
- pozicioniranje svemirskog broda i pucanje na meteor
- kraj igre kada meteor pogodi Zemlju

5. Implementacija

Na kraju, u implementaciji se vrše finalni odabiri u razvoju igre. Neki od tih odabira su:

- Da li je igra 2D ili 3D?

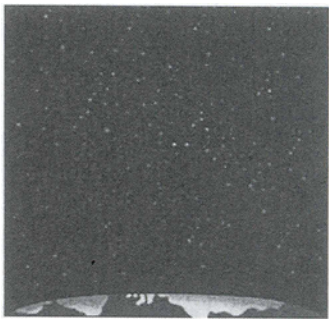
U ovom momentu naš izbor je da to bude 2D igra.

- Koja vrsta izgleda je prikazana u igri?

Igra treba da se dešava u svemiru. Ići ćemo na izgled sličan crtanom filmu.

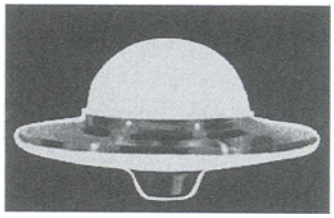
- Kako su implementirani likovi?

Nebo i zvijezde se mogu implementirati pomoću background slike. Dio te slike može biti i zemlja, pri čemu moramo znati njene koordinate radi detekcije kolizije.



slika 3. background

Svemirski brod će biti izrendan pomoću sljedeće slike



slika 4. svemirski brod

Meteori su okrugli oblici koji se slučajno pojavljuju na nebu i padaju u pravcu Zemlje. Meci su također okrugli oblici koji se ispaljuju prema gore kada korisnik pritisne desno dugme miša.

- Koji su ulazni metodi za korisnika?

U arcade igrama joystick omogućava korisniku da kontroliše igru. Joystick radi veoma slično mišu. On omogućava korisniku da pomjera likove desno, lijevo, gore i dolje i da puca pritiskom na dugme. Korisnik povlači miša sa pritisnutim lijevim dugmetom da bi pomicao svemirski brod. Klik na desno dugme omogućava ispaljivanje metaka.

- Koje alatke su potrebne za razvoj igre?

C++ i OpenGL. Razvićemo igru u C++ jer ovaj jezik odgovara procesu razvoj igara.

- Na kojoj platformi će se izvršavati igra?

Windows (ali može se kompajlirati i izvršavati na bilo kojoj platformi)

Implementacija igre

Algoritam igre baziran na interkativnom logičkom dijagramu prikazanom ranije je:

```
till game ends do {
        monitor user input to update spaceship location
        monitor user input to generate bullet
        randomly create meteors
for all meteors {
                        update meteor location
                        if meteor hits earth {
                                                Display explosion
                                                End Game;

                                                }
                        if meteor has gone out of screen
                                destroy meteor

                        else
                                draw meteor
 for all bullets {
                        update bullet location
                        if bullet hits meteor{
                                destroy meteor
                                destroy bullet
                } else if bullet is out of screen
```

```
                        destroy bullet

                else
                        draw bullet
            }

        }
        drawspaceship

}
```

Počnimo razvoj igre definisanjem likova.

Likovi

Objektno orijentisano programiranje se dobro uklapa u programiranje igara, jer funkcionalnost likova može biti implementirana putem klasa. Uzećemo jednostavne C++ klase za kreiranje likova u igri.

Meteor

Najprije razmotrimo objekat meteor. On ima lokaciju koja identificira njegove (x,y) koordinate na ekranu. Na osnovu te lokacije meteor se iscrtava. On mora imati mogućnost da ažurira lokaciju za svaku petlju koda, tzv. *tick* igre. Također se mora znati da li je on izvan ekrana (u tom slučaju ga uništavamo) ili je pogodio planetu Zemlju.
Klasa Meteor je definisana ovako:

```
class Meteor
        {
        public:
                Meteor(int ulD);
                Virtual ~ Meteor();
                void Draw();            //Draw the meteor
                void DrawCollision();   //Draw the meteor that has collided
                void Update();          //Update meteor location, for this
example simply                                  decrement it's y-
coordinate in order to move it down
                bool HitEarth();                //returns true if the meteor has
hit the Earth
                bool OutOfScreen();     //returns true if meteor is out of screen
                Glint ID;               //unique ID to identify the meteor
                Gfloat*GetLocation() {return location;}
                Private:
                GLFloat location [2];   //meteor's          location.location[0]=x-
```

```
,location[1]=y-                                    coordinate
        };
```

Niz location smješta tekuće (x,y) koordinate meteora. Za ovaj primjer pustićemo meteore da padaju ravno dolje duž y – ose.
Update funkcija pomalo dekrementira location[1] varijablu da bi meteor padao. Možemo eksperimentisati sa dodavanjem x vrijednosti meteorima.
Draw rutina crta bijelo obojenu tačku (glVertex) na tekućoj poziciji meteora, dok DrawCollision rutina iscrtava veću crvenu tačku da pokaže koliziju.
Funkcije HitEarth i OutOf Screen provjeravaju tekuću poziciju meteora da bi vidjeli da li on pogađa Zemlju ili je izašao iz granica world koordinata. Vraćaju vrijednost TRUE ako je test pozitivan.
Koristimo ID varijablu da pojedinačno identificiramo svaki meteor.
Funkcija createMeteor se koristi za slučajno kreiranje meteora tokom igre. Ona generiše slučajni broj između 0 i 1 i kreira meteor samo ako je taj broj manji od konstante. Ovo osigurava da nemamo previše meteora na ekranu.

```
void createMeteor() {
if (((float)rand()/RAND _MAX) < 0.99)
        return;
 SYSTEMTIME systime;
 GetSystem Time(&systime);
 int ID - systime.wMinute*60+systime.wSecond;
  Meteors[ID] - new Meteor(ID);    //create a meteor with
identifier=ID
}
```

Meteori se kreiraju samo ako je slučajni broj generisan između 0 i 1 manji od konstante 0.99. Smanjenje ove konstante će uzrokovati da više meteora bude kreirano. Svaki meteor je pojava sa posebnim ID, koji je integer. ID meteora se izvodi od minute i sekunde tekućeg vremena i jedinstven je sve dok igrač ne igra duže od jednog sata. Ovaj ID se koristi kao ključ za lociranje i inseritranje metora na mapi meteora:

```
typedef map<int, Meteor*> MeteorMap;
static MeteorMap Meteors;
```

Koristimo STL container, u ovom slučaju mapu, za smještanje meteora. STL mape su lak način smještanja objekata u niz indeksirano po ključu. U ovom slučaju, ID je ključ koji se korsiti za smještanje i dobavljanje objekata sa mape meteora. STL iteratori se koriste za kretanje po STL containeru.

Bullet objekat

Klasa koja podržava Bullet objekat je definisana slično kao Meteor. Ona ima dodatnu funkciju koja testira kolizije između metka i datog meteora.

```
class Bullet {
        public:

        Bullet(int ulD, int x, int y);    // Define a new bullet at location(x,y).

        virtual ~ Bullet();
        void Draw();                      //draw  Bullet
        void Update();                    //Update Bullet location
        bool OutOfScreen();               //returns true if Bullet is out of screen
        bool Coliide(Meteor *);           //returns true if Bullet collides with Meteor
        GUfoat ID;
        private:
        GLfloatt location[2];             //(x,y) location of bullet

  };
```

Draw rutina crta crvenu tačku (glVertex) na tekućoj lokaciji metka, dok DrawCollision crta veću crvenu tačku da bi prikazala koliziju. Update funkcija neznatno inkrementira location[1] varijablu, pomjerajući metak naviše.
Svi meci koji su kreirani se unose u BulletMap i lociraju po njihovom ID-u.

```
typedef      map<int,      Bullet*>
BulletMap;
static BulletMap Bullets;
```

Meci se kreiraju kada korisnik klikne desnim dugmetom miša. Uskoro ćemo vidjeti kako možemo nadzirati mouse events. Funkcija createBullet se poziva kada se klikne desnim dugmetom miša. Funkcija je definisana ovako:

```
// Create a bullet and add it to a Bulletmap
void createBullet(int x, int y) {
  SYSTEMTIME systime;
  GetSystem Time(&systime);
  int ID - systime.wMinute*60+systime.wSecond;
  Bullets[ID]- new Bullet(ID,x,y);
}
```

Svemirski brod

Razmotrimo na kraju objekat svemirski brod. Kao i meteor i metak, on ima location varijablu koja određuje njegove (x,y) koordinate.

```
class SpaceShip
{
public:
   SpaceShip();
   virtual – SpaceShip();
   void SetPixels(BITMAPINFO *ssinfo, GLubyte  *sspixels);   // set the
   pixels and
     bitmapinfo for spaceship

     void Draw();                                  //Draw the spaceship
     void Update(GLfloat x, GLfloat y);            // update location of spaceship
     GLfloat GetXLocation(){ return location[0];}
     GLfloat GetYLocationl(){return location[1];}
     void StartMove();                             //start moving the spaceship
     void StopMove();                              //stop moving the spaceship update
                                                   of spaceship will not do anything

           private:

           GLfloat location[2];
           GLubyte *rnypixels;                          BITMAPINFO *myirnginfo;
   //pixel info of spaceship
   //bitmap info of spaceship image

           bool b_updatesss;

   //are we updating location of spaceship

           };
```

Funkcija SetPixels se koristi da inicijalizira pixmap svemirskog broda. Rutina Draw crta pixmapu na tekućoj lokaciji objekta.
Svemirski brod se kreće kada korisnik drži desno dugme i pomjera miša. Kada korisnik pusti dugme, svemirski brod se prestane kretati. Boolean varijabla b_updates vodi računa o tome da li se svemirski brod kreće ili ne, tj. da li se treba ažurirati njegova lokacija.
Update funkcija se poziva samo kada b_updates ima vrijednost TRUE. Ona uzima kao parametar tekuću poziciju miša i crta svemirski brod na toj

lokaciji. U ovoj vježbi pomjeraćemo svemirski brod samo lijevo i desno, tako da je samo x-koordinata potrebna za kretanje.

Ulaz korisnika

OpenGL i GLUT biblioteka omogućavaju veliki broj funkcija za podršku eventima. Event-handling thread je posebna vrsta izvršavanja programa koji slušaju na evente, kao što je ulaz korisnika, tajmer itd. Ako je event detektovan, poziva se odgovarajuća „callback" funkcija. Callback funkcija zatim daje odgovarajući odgovor na event. GLUT funkcija koja se koristi za kreiranje event handling thread-ova je

```
glutMouseFunc(void(*func)(int button, int state, int x, int y))
```

Ova funkcija postavlja callback za miša za tekući prozor da bude u funkciji func. Kada korisnik pritisne i otpusti dugme miša u tom prozoru, svaki pritisak i otpuštanje generiše poziv ove funkcije. Callback funkcija prima informaciju o tome koje dugme je pritisnuto (GLUT_RIGHT_BUTTON ili GLUT_LEFT_BUTTON), stanje tog dugmeta (GLUT_UP or GLUT_DOWN), i x,y lokaciji miša u prozoru (u world koordinatnom sistemu). U našem primjeru, definišemo našu callback funkciju kao

```
void getMouse(int button, int state, int x, int y)
```

Kada se klikne desnim dugmetom, želimo da se ispali metak. Metak polazi iz centra svemirskog broda. Ako je svemirski brod bio pomjeran, također želimo da ga zaustavimo. Ako je dugme pritisnuto, želimo da počnemo pomjerati svemiski brod.

```
void getMouse(int button, int state, int x, int y){
//If right mouse is clicked up, create a bullet located at the center of the
spaceship image. Stop spaceship from moving
        if (button - - GLUT_RIGHT_BUTTON && state - - GLUT_UP) {
                createBullet(spaceship->GetXLocation() + 25,spaceship- >
GetYLocation()  + 25);
                spaceship- > StopMove(); }
                else {          // Start spaceship motion
               spaceship- >StartMove();


      }}
```

Komanda

glutMotionFunc (void (*funcll int X, int y))

se koristi za postavljanje motion callback funkcije. Ona se poziva kada se miš pomjera unutar prozora dok je jedno ili više dugmadi na njemu pritisnuto. Želimo da se lokacija svemirskog broda ažurira kada korisnik povlači miša okolo. Definišemo motion callback funkciju na sljedeći način

```
void getMouseMotionlint x, int yl{
        // update spaceship when mouse is moved.
        // spaceship motion is enabled only when button is clicked down
        spaceship->Update(x,  SCREENMAXY-y );
}
```

Primijetimo nešto neobično. Ne koristimo istu y koordinatu koju primamo od mouse klika. Naše world koordinate imaju origin u donjem lijevom uglu prozora. Y koordinata raste kako se krećemo prema gore po ekranu. Windows koristi (0,0) na gornjem lijevom uglu prozora, pa y raste kada se ide dolje. Lokacija mouse evenata je u koordinatnom sistemu prozora i treba biti reverzna da bi se korektno mapirala u naše world koordinate. Kod ove klase se nalazi u SpaceShip.cpp i SpaceShip.h.

Timer callbacks

Slično callback-ovima za mouse evente, GLUT ima mogućnost da registruje timer callbacks. Oni se okidaju u regularnim intervalima vremena (the ticks) igre. Ovaj mehanizam je izuzetno koristan u igrama, kada želimo da se logika igre rukovodi ovim intervalima.

GLUT funkcija

void glutTimerFunc (unsigned int msecs , void (*funcHint value), value);

postavlja timer callback da bude trigerovan u određenom broju milisekundi. Ona očekuje kao parametar pointer na eventhandling funkciju, kao i bilo koju integer vrijednost da joj se proslijedi. U init() kodu našeg programa, registrujemo timer callback kao

glutTimerFunc(100, timer, 0);

Ovo znači da će funkcija timer (prikazana u nastavku) biti pozvana nakon 100 milisekundi. Timer funkcija kreira meteore slučajno, poziva Display funkciju i na kraju se još jednom poziva da reregistruje timer callback, počinjući proces ispočetka.

```
void timer(int value) {
        createMeteorO;
         // Force a redisplay, Display function contains main game logic
        glutPostRedisplay();
          // Restart the timer
        glutTimerFunc(10, timer, 0);
}
```

Sastavljanje

Sada kad smo definisali sve dijelove, pogledajmo glavni program. Glavna Display() funkcija slijedi logiku prethodno navedenog algoritma. Ako meteor m pogodi Zemlju, pozivamo funkciju

```
void EndProgram(Meteor *m)
```

Ova funkcija prvo prikazuje meteor (tačka u crvenoj boji), zatim puni seriju slika da dočara eksplodiranje Zemlje i na kraju završava program. Inicijalizacioni pozivi koji postavljaju sve event handlere, startaju generator slučajnih brojeva i kreiraju svemirski brod, nalaze se u funkciji init:

```
void init(void){
        // Timer callback
        glutTimerFunc(100, timer, 0);
        // Define mouse callback function
        glutMouseFunc(getMouse);
        / Define Mouse Motion callback function g
        lutMotionFunc(getMouseMotion);
        // random number generator seeded by current time
        SYSTEMTIME systime;
        GetSystem Time(&systime);
        // Seed random number generator
        Srand(systime.wMinute*60 + systime.wSecond);
        //Create the spaceship
        createSpaceShip();
}
```

Ovaj program je definisan na najjednostavniji način, radi lake čitljivosti. To nije najefikasniji kod. Postoji veliki broj trikova koji se mogu uraditi da se ubrza program: iscrtavanje pomoću XOR-a, minimiziranje broja petlji itd.
Također se mogu dodati i zvučni efekti, animacija eksplozije planete, ljepši meteor objekat itd.

Literatura

[1] R. Brinkmann – The Art and Science of Digital Compositing, Academic Press 1999.
[2] Foley, Vam Damm – Computer Graphics, Principles and Practice, Addison-Wesley Publishers 1995
[3] P. Lynch, S. Horton, Web Style Guide, Copyright Lynch & Horton, 1997. Yale University
[4] A. Mundi, Principles of Graphics Design, Mundi Design Studios
[5] Ž. Rože – Filmska gramatika, Jugoslovenska kinoteka 1960.
[6] VRML97 Functional specification and VRML97 External Authoring Interface (EAI), ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2002
[7] J. Vince – 3D Computer Animation, Addison-Wesley Publishers 1992.
[8] 3ds max Tutorial, Discreet
[9] K. Murdock – 3ds max 4 Bible, Hungry Minds Inc, 2001.
[10] Đulijano Belić - Mala škola fotografije
[11] Shalini Govil-Pai – Principles of Computer Graphics, Theory and Practice using OpenGL and Maya, Springer 2004.